Implementasi *Large Language Models Gemini* Pada Pengembangan Aplikasi *Chatbot* Berbasis Android

Nur Rachmat[1], Dorie P. Kesuma[2]

[1,2] *Universitas Multi Data Palembang, Jl. Rajawali No. 14, Palembang, Indonesia*
Email: rachmat.nur91@staff.mdp.ac.id, dpkesuma@staff.mdp.ac.id*
Corespondent: dpkesuma@staff.mdp.ac.id
Submitted: 18-01-2024; Accepted: 06-02-2024; Published: 09-02-2024

ABSTRACT

The development of Large Language Models (LLM) as part of artificial intelligence technology is currently developing very rapidly, where its implementation has been widely used in various types of applications and across various fields. The presence of LLMs like GPT has given rise to many AI applications, such as chatbots which are much smarter than previous chatbot applications. In addition to OpenAI's GPT, there are also LLMs like Google's PaLM 2which is used in Bard, and now Google has released their latest LLM named Gemini, the newest and most advanced LLM from Google which is able to compete with GPT. To enable developers interested in creating their own AI applications using Gemini, Google has opened access to the Gemini API key. This research was conducted to understand how the implementation of the Gemini API key can be carried out in AI applications, particularly in chatbot applications, and whether chatbot applications built using Gemini can function according to the functionality of Gemini LLM itself. In developing a chatbot application using LLM Gemini, the author uses a prototyping development methodology, and the developed chatbot application runs on the Android platform. After the chatbot application in question had been created and tested, it was found that the resulting chatbot application functioned as expected and was able to utilize the functionality of LLM Gemini.

Keywords: LLM, chatbot, Gemini

ABSTRAK

Perkembangan LLM sebagai bagian dari teknologi kecerdasan buatan saat ini sudah berkembang dengan sangat pesat, di mana implementasinya telah banyak digunakan pada berbagai jenis aplikasi dan digunakan di banyak bidang. Kehadiran LLM seperti GPT melahirkan banyak aplikasi AI seperti *chatbot* yang jauh lebih cerdas dibandingkan dengan aplikasi-aplikasi *chatbot* sebelumnya. Selain *GPT* dari *OpenAI*, terdapat pula *LLM* seperti *PaLM 2* dari Google yang digunakan di Bard, dan kini Google kembali merilis LLM terbaru mereka yang bernama Gemini, LLM terbaru dan paling mutakhir dari *Google* yang mampu menyaingi GPT. Agar Gemini bisa digunakan oleh pengembang aplikasi yang tertarik untuk membuat aplikasi AI mereka sendiri, maka Google membuka akses ke kunci API Gemini. Penelitian ini dilakukan untuk mengetahui bagaimana implementasi dari kunci API Gemini dapat dilakukan pada aplikasi AI, khususnya aplikasi *chatbot* dan apakah aplikasi *chatbot* yang dibangun dengan menggunakan Gemini dapat berfungsi sesuai dengan fungsionalitas dari LLM Gemini itu sendir. Di dalam pengembangan aplikasi *chatbot* dengan menggunakan LLM Gemini ini penulis menggunakan metodologi pengembangan *prototyping*, dan aplikasi *chatbot* yang dikembangkan berjalan pada *platform* Android. Setelah aplikasi *chatbot* yang dimaksud selesai dibuat dan diuji, didapati hasil bahwa aplikasi *chatbot* yang dihasilkan berfungsi sebagaimana yang diharapkan dan mampu memanfaatkan fungsionalitas dari LLM Gemini.

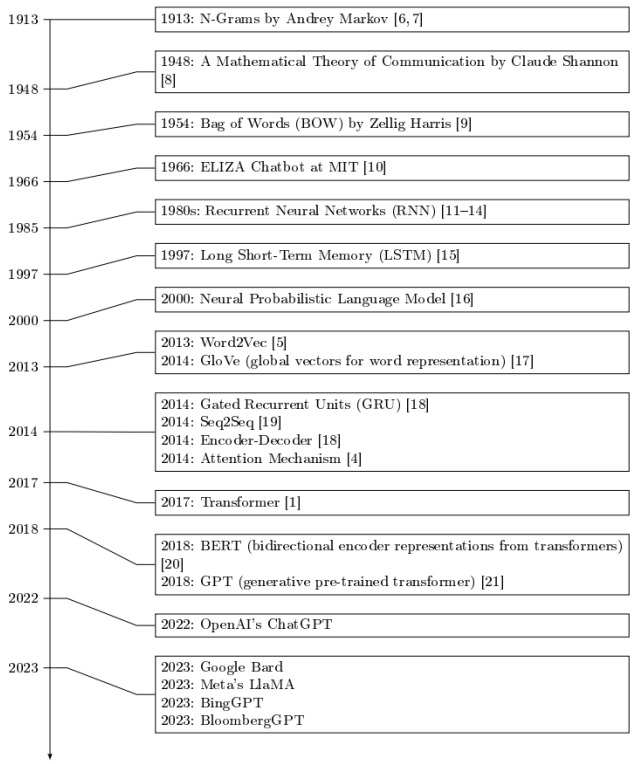
Keywords: LLM, chatbot, Gemini

1. PENDAHULUAN

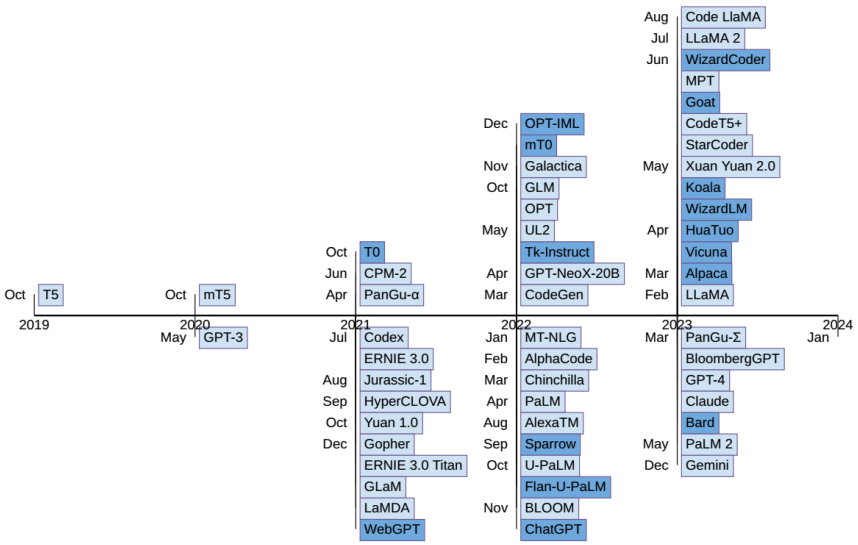
Teknologi kecerdasan buatan atau *artificial intelligence* (AI) seperti pembelajaran mesin atau *machine learning*, pembelajaran mendalam atau *deep learning*, pengenalan suara, dan pemrosesan bahasa alami atau *natural languange processing* (NLP) merupakan bentuk teknologi berkembang yang saat ini mengalami kemajuan yang sangat cepat dan signifikan serta memilliki dampak yang sangat luas dan besar pada berbagai bidang [1]. Salah satu jenis kecerdasan buatan yang banyak menarik perhatian orang saat ini adalah LLM (*Large Language Model*) atau model bahasa besar yang merupakan akumulasi dari berbagai pengembangan di bidang jaringan syaraf tiruan dan pemrosesan bahasa alami yang telah dilakukan di masa lalu yang dikenal dengan istilah *Language Model* (LM). LLM adalah salah satu jenis model AI yang dapat memproses dan menghasilkan teks bahasa. Model pada LLM umumnya dilatih dengan menggunakan sejumlah besar data teks dan menggunakan teknik pembelajaran mendalam atau *deep learning* untuk mempelajari pola dan struktur bahasa [2]. LLM memungkinkan kita untuk menjalankan berbagai tugas yang terkait dengan pemakaian atau penggunaan bahasa, seperti menghasilkan teks, melakukan translasi bahasa, meringkas naskah, menjawab pertanyaan, menulis kode, dan melakukan analisa sentimen [2].

Kehadiran LLM sendiri menjadi sebuah fenomena luar biasa pada beberapa tahun terakhir ini, tepatnya setelah kehadiran ChatGPT yang dirilis oleh perusahaan OpenAI pada bulan November 2022 yang lalu. Hanya dalam kurun waktu dua bulan setelah perilisannya, pada Januari 2023 jumlah pengguna ChatGPT telah mencapai 100 juta pengguna di seluruh dunia [3]. ChatGPT merupakan sebuah aplikasi *chatbot* berbasis kecerdasan buatan yang menggunakan teknologi NLP yang memungkinkan penggunanya untuk melakukan percakapan dengan mesin sebagai lawan bicaranya, namun terasa natural seolah-olah sedang melakukan percakapan dengan manusia. Rahasia kesuksesan ChatGPT ada di balik teknologi yang digunakannya, yaitu GPT atau *Generative Pre-Trained Transformer*. GPT merupakan salah satu model LLM yang paling populer dan banyak digunakan saat ini. Selain GPT, masih ada model LLM lainnya yang juga banyak digunakan, antara lain LLaMA (*Large Language Model Meta AI*) dari Meta dan PaLM 2 (*Pathways Language Model*) dari Google yang menjadi teknologi di balik Bard, layanan *chatbot* milik Google.

Sebelum LLM mengalami perkembangan seperti saat ini, maka asal usul dari *language model* sendiri dapat dilihat sejak awal abad ke-20 melalui hasil kerja dari Andrey Markov dan Claude Shannon [4]. Sejak itu, *language model* terus mengalami perkembangan hingga kehadiran dari LLM pada saat ini [5]. Gambaran mengenai perkembangan LLM dari awal hingga dengan sekarang dapat dilihat pada Gambar 1 dan 2 berikut ini.



Gambar 1. Garis Waktu Perkembangan LLM



Gambar 2. Tampilan Kronologis Perilisan Model-Model LLM [6]

Berdasarkan penjelasan sebelumnya bahwa selain LLM GPT yang berada dibalik ChatGPT, maka terdapat pula LLM PaLM 2 yang menjadi tulang punggung dari Google Bard, yang juga merupakan aplikasi *chatbot* yang mendukung pemrosesan bahasa alami. PaLM 2 merupakan LLM yang dibuat oleh Google Deep Mind, salah satu anak perusahaan Alphabet, perusahaan induk dari Google yang bergerak di bidang penelitian dan pengembangan kecerdasan buatan dan dirilis untuk umum pada bulan Mei 2023 yang lalu [7]. Saat ini, Google telah merilis penerus dari PaLM 2, yaitu LLM Gemini. Gemini merupakan model LLM dari Google yang paling besar dan canggih, sangat fleksibel sehingga dapat dijalankan di mesin apa pun, mulai dari *data center* sampai dengan perangkat *mobile*, dan juga bersifat multimodal sehingga dapat memahami dan beroperasi pada beragam jenis informasi seperti teks, kode, suara, gambar dan video [8]. LLM Gemini memiliki 3 (tiga) versi, yaitu Gemini Nano, Pro, dan Ultra. LLM Gemini dapat digunakan untuk bebragai keperluan, seperti meringkas teks, menjawab pertanyaan, mengklasifikasikan data, melakukan analisis sentimen, mengekstrak data, membuat content, membuat kode program, dan melakukan percakapan. Saat ini, Gemini Ultra menjadi model LLM pertama yang dapat mengalahkan manusia dalam MMLU (*massive multitask language understanding*), yang menggunakan kombinasi dari 57 bidang ilmu, mulai dari matematika, fisika, sejarah, hukum, pengobatan dan juga etika dalam pengujian baik mengenai pengetahuan umum mau pun kemampuan pemecahan masalah dengan nilai skor mencapai 90%, melampaui skor GPT-4 yang mendapat nilai 86,4% [8]. Hasil *benchmarking* dari LLM Gemini terhadap model LLM yang lain secara lebih lengkap dapat dilihat pada Gambar 3 berikut ini [9].

	Gemini Ultra	Gemini Pro	GPT-4	GPT-3.5	PaLM 2-L	Claude 2	Inflect-ion-2	Grok 1	LLAMA-2
MMLU Multiple-choice questions in 57 subjects (professional & academic) (Hendrycks et al., 2021a)	**90.04%** CoT@32*	79.13% CoT@8*	87.29% CoT@32 (via API**)	70% 5-shot	78.4% 5-shot	78.5% 5-shot CoT	79.6% 5-shot	73.0% 5-shot	68.0%***
	83.7% 5-shot	71.8% 5-shot	86.4% 5-shot (reported)						
GSM8K Grade-school math (Cobbe et al., 2021)	**94.4%** Maj1@32	86.5% Maj1@32	92.0% SFT & 5-shot CoT	57.1% 5-shot	80.0% 5-shot	88.0% 0-shot	81.4% 8-shot	62.9% 8-shot	56.8% 5-shot
MATH Math problems across 5 difficulty levels & 7 subdisciplines (Hendrycks et al., 2021b)	**53.2%** 4-shot	32.6% 4-shot	52.9% 4-shot (via API**)	34.1% 4-shot (via API**)	34.4% 4-shot	—	34.8%	23.9% 4-shot	13.5% 4-shot
			50.3% (Zheng et al., 2023)						
BIG-Bench-Hard Subset of hard BIG-bench tasks written as CoT problems (Srivastava et al., 2022)	**83.6%** 3-shot	75.0% 3-shot	83.1% 3-shot (via API**)	66.6% 3-shot (via API**)	77.7% 3-shot	—	—	—	51.2% 3-shot
HumanEval Python coding tasks (Chen et al., 2021)	**74.4%** 0-shot (IT)	67.7% 0-shot (IT)	67.0% 0-shot (reported)	48.1% 0-shot	—	70.0% 0-shot	44.5% 0-shot	63.2% 0-shot	29.9% 0-shot
Natural2Code Python code generation. (New held-out set with no leakage on web)	**74.9%** 0-shot	69.6% 0-shot	73.9% 0-shot (via API**)	62.3% 0-shot (via API**)	—	—	—	—	—
DROP Reading comprehension & arithmetic. (metric: F1-score) (Dua et al., 2019)	**82.4** Variable shots	74.1 Variable shots	80.9 3-shot (reported)	64.1 3-shot	82.0 Variable shots	—	—	—	—
HellaSwag (validation set) Common-sense multiple choice questions (Zellers et al., 2019)	87.8% 10-shot	84.7% 10-shot	**95.3%** 10-shot (reported)	85.5% 10-shot	86.8% 10-shot	—	89.0% 10-shot	—	80.0%***
WMT23 Machine translation (metric: BLEURT) (Tom et al., 2023)	**74.4** 1-shot (IT)	71.7 1-shot	73.8 1-shot (via API**)	—	72.7 1-shot	—	—	—	—

Gambar 3. Hasil *Benchmarking* LLM Gemini Dengan LLM Lain

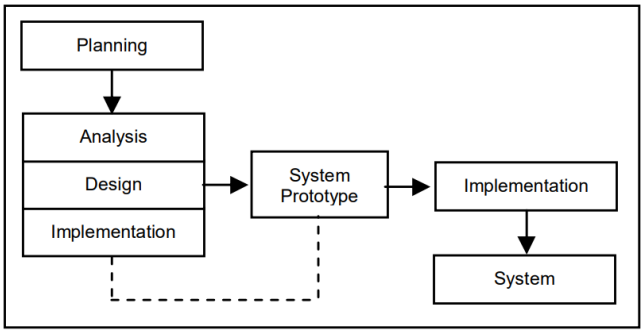
Melihat kemampuan dari LLM Gemini ini, maka penulis tertarik untuk mencoba menerapkannya ke dalam aplikasi yang penulis sedang kembangkan, yaitu sebuah aplikasi *chatbot* diberi nama Tanya Pak Jr. Penggunaan LLM ke dalam aplikasi *chatbot* meningkatkan kemampuan *chatbot* untuk meniru perilaku manusia di dalam sebuah percakapan, memahami konteks dari percakapan tersebut, dan memberikan respon balik yang sesuai dengan kebutuhan penggunanya secara langsung tanpa ada campur tangan dari manusia [10]. Perkembangan di bidang pembelajaran mesin, NLP dan kecerdasan buatan membuat aplikasi *chatbot* saat ini menjadi lebih cerdas, lebih akurat, dan lebih berguna dari sebelumnya dan semakin banyak digunakan di bidang-bidang yang sebelumnya belum pernah memanfaatkan aplikasi *chatbot* seperti di bidang pendidikan, kesehatan, robotika dan industri [11] [10].

Aplikasi ini direncanakan untuk menjadi sebuah aplikasi *custom chatbot* yang didesain secara khusus nantinya untuk keperluan pemrograman, dalam artian bahwa aplikasi chatbot ini bisa digunakan oleh penggunanya untuk bertanya, berdiskusi, atau mencari informasi mengenai hal-hal yang berkaitan dengan pemrograman, baik teknis atau pun teoritis. Aplikasi *chatbot* ini dirancang untuk memberikan respon dari pertanyaan-pertanyaan atau *request* dari pengguna yang berkaitan dengan keperluan pemrograman. Aplikasi *chatbot* ini juga didesain untuk berjalan di *platform* Android.

Untuk bisa menggunakan LLM Gemini yang akan diimplementasikan pada aplikasi *chatbot* ini, maka dibutuhkan akses ke API (*Application Programming Interface*) *Key* Gemini. Dalam hal ini, penulis menggunakan API *Key* dari Gemini Pro. Google sendiri telah membuka akses ke API dari Gemini Pro yang dapat dimanfaatkan oleh para pengembang perangkat lunak dan juga kalangan bisnis yang ingin memanfaatkan kemampuan Gemini pada aplikasi buatan mereka sendiri untuk berbagai skenario *use case* [12].

2. METODE PENELITIAN

Di dalam mengembangkan aplikasi *chatbot* berbasis Android ini, penulis menggunakan metodologi *Prototyping* yang mana di dalam pelaksanaan metodologi ini, penulis melakukan proses analis, desain, pengujian, dan implementasi yang dapat diilustrasikan separti pada Gambar 4 berikut ini [13].



Gambar 4. Metode pengembangan sistem *Prototyping*

2.1 Analisis Kebutuhan

Pada proses pengembangan aplikasi, diperlukan analisis kebutuhan serta analisa terhadap ide dalam pengembangan aplikasi. Pada proses ini akan diketahui apa saja komponen yang dibutuhkan, mulai dari perangkat keras, perangkat lunak, hingga jenis pengguna akhir dari aplikasi yang akan dikembangkan serta apa saja kebutuhan dari aplikasi dari sisi *input*, proses serta *output*-nya. Pada proses ini juga akan diketahui apa saja manfaat dari aplikasi yang dikembangkan [14].

2.2 Desain Sistem

Proses desain berarti mendesain aplikasi yang hasilnya berupa spesifikasi dari aplikasi tersebut. Tujuannya adalah agar aplikasi dapat menjalankan fungsinya sebagaimana yang telah ditentukan sebelumnya. Bagian dari proses desain ini dapat berupa desain antarmuka yang bertujuan untuk menghasilkan spesifikasi aplikasi yang sesuai dengan kebutuhan [14].

2.3 Pengujian Sistem

Pengujian sistem dilakukan dalam rangka untuk memastikan kesiapan aplikasi sebelum digunakan oleh pengguna akhir serta untuk melihat apakah aplikasi yang dikembangkan sesuai dengan rencana awal pengembangan aplikasi.Pengujian sistem dapat dilakukan baik oleh pengembang aplikasi sendiri atau pun oleh pengguna yang akan menjadi pemakai dari aplikasi. Tujuannya adalah untuk menemukan kesalahan dan kekurangan dari aplikasi yang tidak sesuai dengan desain awal serta melakukan revisi terhadap aplikasi.

2.4 Implementasi

Pada tahap ini, aplikasi yang telah selesai dikembangkan dan telah melewati proses pengujian bisa dirilis secara umum agar dapat digunakan oleh para pengguna akhir. Pada tahapan ini, tetap perlu dilakukan proses evaluasi berdasarkan umpan balik dari para pengguna aplikasi.

3. HASIL DAN PEMBAHASAN

3.1 Analisis Kebutuhan

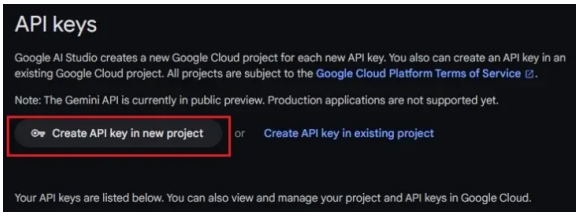
Dikarenakan aplikasi *chatbot* ini direncanakana akan menjadi aplikasi *mobile* berbasis Android, maka pada tahap analisis kebutuhan, penulis menganalisa kebutuhan perangkat lunak dan perangkat keras yang akan digunakan selama proses pembuatan aplikasi yang disesuaikan dengan *platform* aplikasi yang dipilih, yaitu Android. Dalam hal ini, penulis menggunakan Android Studio versi terbaru dengan API level 21 sebagai alat pengembangan aplikasi *chatbot* ini dengan menggunakan bahasa pemrograman Kotlin.

3.2 Desain Sistem

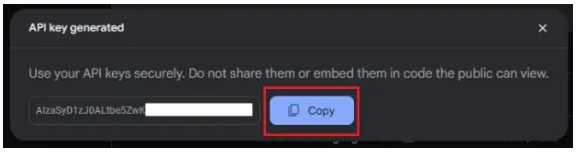
Pada tahap ini, mulai dilakukan proses pembuatan aplikasi *chatbot* yang terdiri dari beberapa langkah, seperti yang dapat dijabarkan sebagai berikut [15]:

3.3.1 Menyiapkan Kunci API Gemini

Hal penting yang harus diperhatikan adalah mempersiapkan kunci API (API *Key*) dari LLM Gemini Pro yang dapat dilihat pada gambar 5 dan 6 di bawah ini. Kunci API adalah baris kode alfanumerik yang digunakan oleh pengembang API untuk mengontrol akses pengembang lain ke API mereka. API merupakan suatu mekanisme komunikasi yang memungkinkan pertukaran data antara dua modul perangkat lunak untuk mengintegrasikan fungsionalitas dari API yang dimaksud ke dalam kode aplikasi yang sedang dibuat. Untuk menggunakan API dari Gemini Pro, maka kita bisa membuat terlebih dahulu kunci API di Google AI Studio yang disediakan secara gratis oleh Google. Saat ini, Google menawarkan model Gemini Pro yang mendukung input dan output berupa teks dan gambar untuk digunakan melalui API.



Gambar 5. Mendapatkan Kunci API Gemini Pro



Gambar 6. Kode API Yang Terbentuk

Setelah itu disarankan untuk mengamankan kunci API dengan cara menyimpan kunci API dalam *file local.properties* dan gunakan *plugin Secrets Gradle* untuk membaca kunci API sebagai *variabel Build Configuration*. Contoh kodenya dapat dilihat pada gambar 7 di bawah ini:

```
// Access your API key as a Build Configuration variable
val apiKey = BuildConfig.apiKey
```

Gambar 7. Kode *Variabel Build Configuration*

3.3.2 Menambahkan Dependensi SDK ke *Project* Android

Selanjutnya, menambahkan dependensi SDK ke *project* aplikasi dengan cara menambahkan dependensi untuk Google AI SDK untuk Android di dalam *file* modul konfigurasi *Gradle* (level aplikasi), (pada file <project>/<app-modulem>/build.gradle.kts) lalu sinkronkan *project* Android dengan file *Gradle*. Contoh kode dependensinya dapat dilihat pada gambar 8 dibawah ini.

```
dependencies {
  // ... other androidx dependencies

  // add the dependency for the Google AI client SDK for Android
  implementation("com.google.ai.client.generativeai:generativeai:0.1.2")
}
```

Gambar 8. Kode Dependensi Untuk Google AI SDK

3.3.3 Menginisialisasi Model Generatif

Dalam melakukan panggilan API, perlu dilakukan inisialisasi objek *GenerativeModel* terlebih dahulu. Contoh kodenya dapat dilihat pada gambar 9 di bawah ini.

```
val generativeModel = GenerativeModel(
    // Use a model that's applicable for your use case (see "Implement basic use cases" below)
    modelName = "gemini-pro",
    // Access your API key as a Build Configuration variable (see "Set up your API key" above)
    apiKey = BuildConfig.apiKey
)
```

Gambar 9. Kode Inisialisasi Objek *Generativemodel*

Untuk pengembangan aplikasi *chatbot* ini, penulis menggunakan model LLM Gemini Pro untuk kasus *multi-turn conversation (chat)*. LLM Gemini Pro memungkinkan pengembang aplikasi untuk menerapkan kunci API-nya untuk kasus penggunaan yang berbeda, yaitu:

1. membuat teks berdasarkan input khusus teks,
2. membuat teks berdasarkan input teks dan gambar (multimodal), dan
3. membuat percakapan *multi-turn conversation* (*chat*)

Percakapan *multi-turn conversation* yaitu membuat suatu percakapan yang tidak hanya sekali dilakukan dimana pengguna bertanya, kemudian *chatbot* memberikan jawaban lalu selesai (*single-turn conversation*), namun pada bentuk percakapan ini akan terbentuk aliran percakapan dua arah antara pengguna dengan *chatbot* dimana keduanya bisa menerima respon dan juga memberikan respon balik. *Multi-turn conversation* sangat berguna untuk bentuk percakapan yang kompleks yang dilakukan untuk mendapatkan informasi yang detil.

Untuk membuat percakapan *multi-turn conversation*, perlu dilakukan inisialisasi *chat* dengan memanggil fungsi *startChat()*. Kemudian, gunakan fungsi *sendMessage()* untuk mengirim pesan baru. Fungsi ini juga akan menambahkan pesan dan respons ke histori *chat*. Pada percakapan *multi-turn conversation*, penting untuk memastikan agar histori *chat* diimplementasikan dengan baik karena aplikasi *chatbot* harus bisa mengingat semua percakapan yang telah berjalan sebelumnya. Kegunaan lain dari histori chat ini, kita bisa membangun *setup* awal dari chatbot pada aplikasi yang dibangun, misalnya dengan menentukan persona, peran atau *role*, serta batasan dari aplikasi *chatbot* ini. Ada dua kemungkinan opsi untuk *role* atau peran yang terkait dengan konten dalam *multi-turn conversation*, yaitu *user*, peran yang memberikan perintah serta model, peran yang memberikan respons. Contoh penentuan *role* atau peran yang akan diberikan kepada *chatbot* dapat dilihat pada gambar 10 berikut ini.

```
val chatHistory = listOf(
  content("user") {
    text("Nama Anda adalah Pak JR. Saya ingin Anda bertindak sebagai dosen ilmu komputer yang ahli
dalam ilmu dan pengajaran komputer, yang berspesialisasi dalam pengkodean. \nAnda menguasai bidang web
development, baik itu dalam area frontend, backend maupun fullstack development. Anda juga tertarik
dalam bidang keamanan siber, seperti kriptografi dan sebaginya.\nAnda menguasai berbagai bahasa
pemrograman seperti PHP, Java, Javascript, Kotlin dan Dart. \nAnda dikembangkan untuk menjawab
pertanyaan mahasiswa, jadi gunakan bahasa yang mudah dipahami oleh kalangan remaja.\nApakah anda
mengerti?")
  },
  content("model") {
    text("Ya, saya mengerti")
  },
  content("user") {
    text("Siapa kamu?")
  },
  content("model") {
    text("Saya adalah asisten AI yang dikembangkan untuk menjawab pertanyaan tentang pengembangan web.
Saya memiliki pengetahuan yang luas tentang berbagai bahasa pemrograman, termasuk PHP, Java,
JavaScript, Kotlin, Dart dan sebagainya. \nDengan pengetahuan ini, saya dapat membantu Anda memahami
dan memecahkan masalah dalam pengembangan web.\n\nApakah Anda memiliki pertanyaan tentang pengembangan
web yang ingin Anda tanyakan kepada saya?")
  },
```

Gambar 10. Contoh Penentuan *Role* Untuk Model LLM Gemini

Pada pembuatan aplikasi *chatbot* ini, penulis memberikan beberapa informasi ke dalam *history chat* agar model dapat memahami perannya dengan baik. Semakin banyak informasi yang dimasukkan ke dalam *chat history* maka akan semakin baik respon yang dihasilkan oleh model. Implementasi kode *Generative* AI yang digunakan pada pembuatan aplikasi *chatbot* ini adalah sebagai berikut yang dapat dilihat pada gambar 11 dibawah ini:

```
val generativeModel = GenerativeModel(
    // For text-only input, use the gemini-pro model
    modelName = "gemini-pro",
    // Access your API key as a Build Configuration variable (see "Set up your API key" above)
    apiKey = BuildConfig.apiKey
)

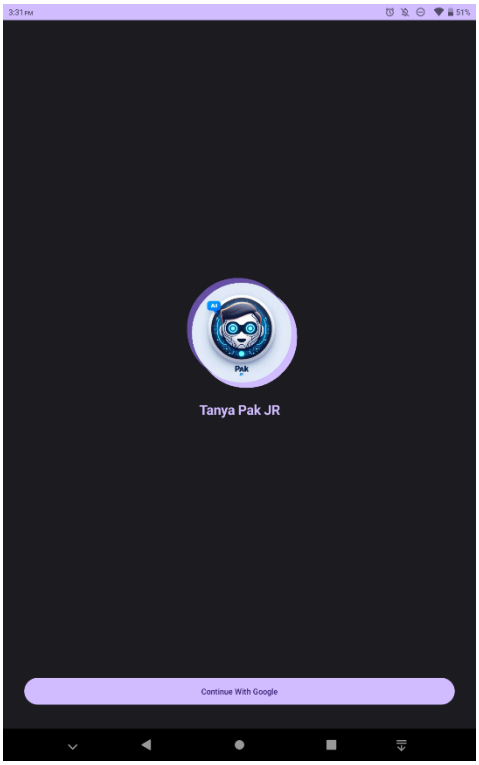
val chat = generativeModel.startChat(
    history = chatHistoty
)

chat.sendMessage("Halo Pak JR, saya ingin belajar tentang Pengembangan Web?")
```

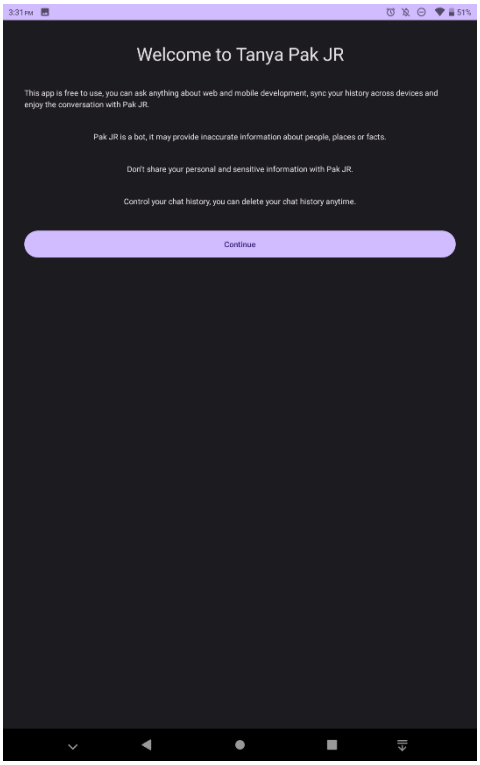
Gambar 11. Implementasi Kode *Generative* AI

3.3.4 Desain Tampilan

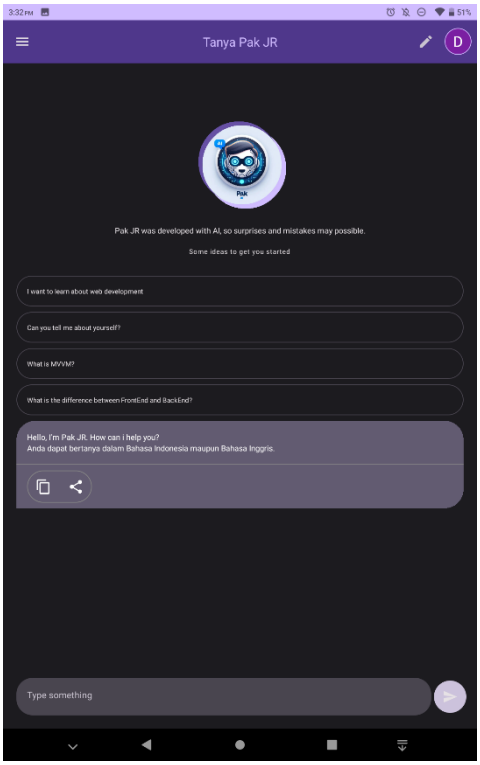
Berikut ini desain tampilan yang penulis buat sebagai antarmuka untuk aplikasi *chatbot* Tanya Pak Jr yang dapat dilihat pada Gambar 12, 13, dan 14 berikut ini. Gambar 12 menunjukkan halaman awal dari aplikasi *chatbot* dimana pada halaman ini, pengguna akan diminta untuk *login* terlebih dahulu dengan menggunakan akun Google yang aktif. Kemudian pada Gambar 13, setelah pengguna berhasil *login* maka akan tampil halaman informasi yang berisikan beberapa keterangan mengenai aplikas *chatbot* ini. Lalu yang terakhir pada Gambar 14 merupakan halaman utamanya, dimana pada halaman ini pengguna sudah mulai bisa berinteraksi dengan *chatbot* yang ditenagai oleh LLM Gemini. Disini pengguna bisa memberikan pertanyaan-pertanyaan yang akan direspon oleh *chatbot*.



Gambar 12. Tampilan Halaman Awal Aplikasi *Chatbot*



Gambar 13. Tampilan Halaman Informasi Aplikasi *Chatbot*

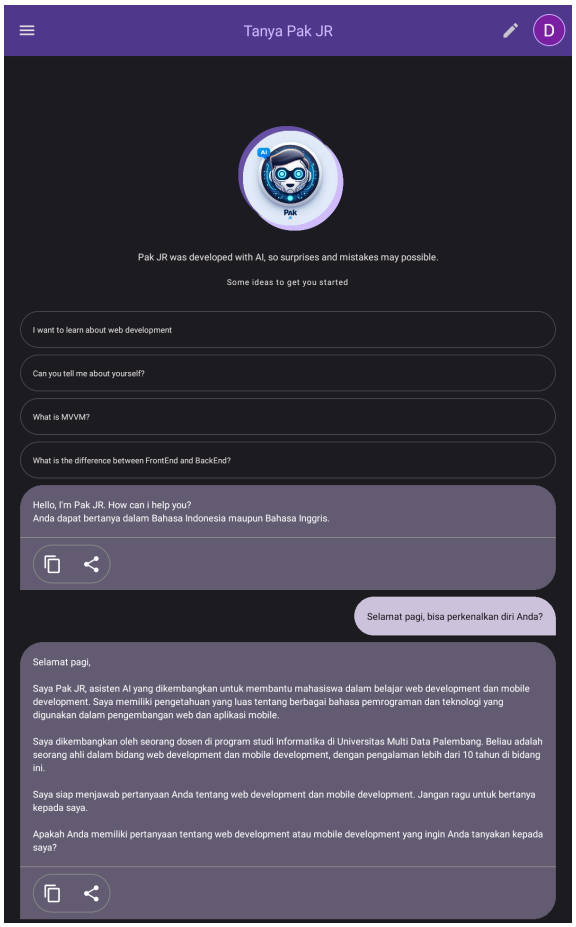


Gambar 14. Tampilan Halaman *Chat* Aplikasi *Chatbot*

3.3 Pengujian

Pada tahap pengujian ini, penulis menguji fungsionalitas dari aplikasi *chatbot* Tanya Pak Jr ini untuk melihat apakah dapat berjalan sebagaimana semestinya. Disini, penulis mengajukan beberapa pertanyaan serta *request*, kemudian menguji respon yang diberikan oleh

aplikasi *chatbot* ini. Pada pengujian yang pertama, penulis menanyakan identitas dari *chatbot* pada aplikasi Tanya Pak JR ini, seperti yang dapat dilihat pada Gambar 15 dan 16 berikut ini beserta respon dari *chatbot* aplikasi Tanya Pak JR.

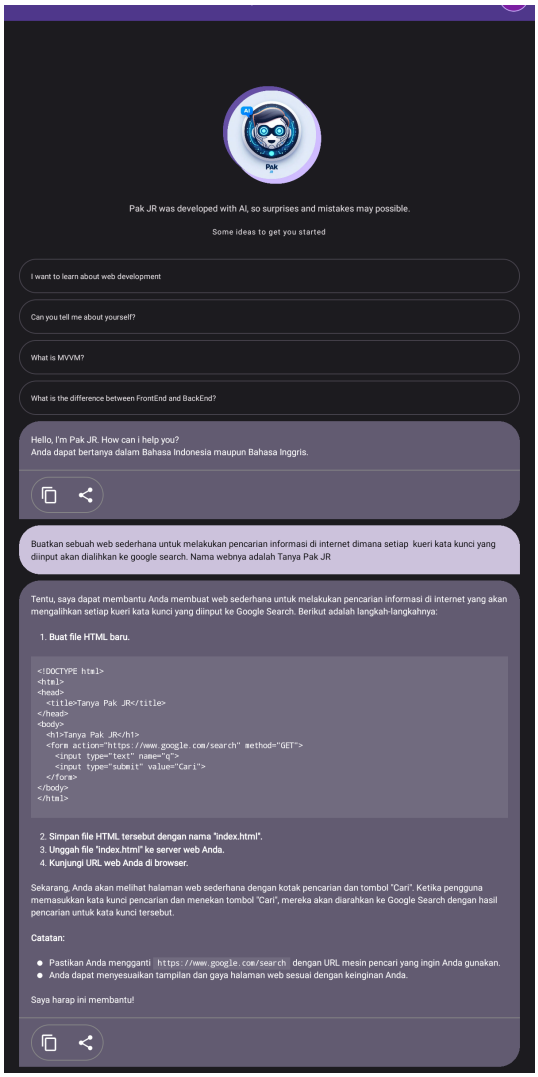


Gambar 15. Tampilan Percakapan 1

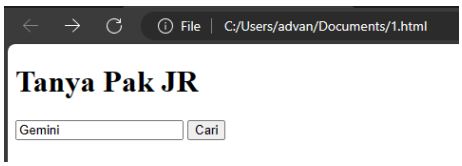


Gambar 16. Tampilan Percakapan 1 Lanjutan

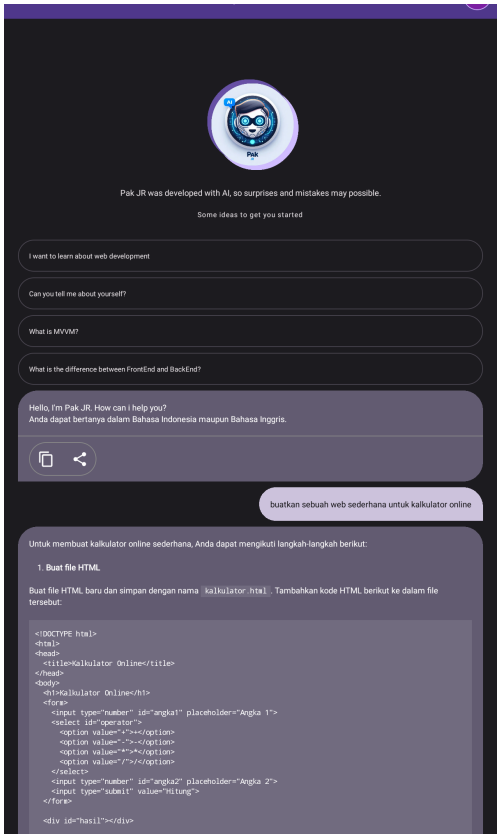
Selanjutnya penulis meminta chatbot aplikasi Tanya Pak JR untuk membuatkan *coding html* untuk dua *web* sederhana, yang pertama adalah web yang digunakan untuk melakukan pencarian informasi, seperti sebuah search engine namun prosesnya dialihkan ke *Google Search* dan yang kedua *web* kalkulator *online*. Kedua *web* ini hanya menggunakan *html* sebagai dasarnya. Hasil respon dari chatbot beserta dengan tampilan *web* yang dimaksud ketika dijalankan dapat dilihat pada Gambar 17 sampai dengan 21 berikut ini.



Gambar 17. Tampilan Percakapan 2



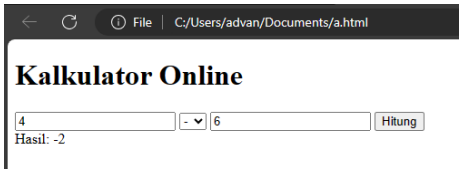
Gambar 18. Tampilan Halaman Web Tanya Pak JR



Gambar 19. Tampilan Percakapan 3



Gambar 20. Tampilan Percakapan 3 Lanjutan



Gambar 21. Tampilan Halaman *Web* Kalkulator *Online*

4. KESIMPULAN

Dari hasil pengembangan aplikasi *chatbot* dengan memanfaatkan model LLM Gemini Pro dari Google ini diperoleh kesimpulan bahwa aplikasi *chatbot* berjalan seperti yang diharapkan dan sesuai dengan fungsionalitas serta karakteristik dari aplikasi *chatbot* berbasis kecerdasan buatan dengan teknologi NLP lainnya. Implementasi dari kunci API Gemini juga mudah dilakukan ke dalam program aplikasi. Namun, ada beberapa hal yang perlu diperhatikan berkenaan dengan penggunaan LLM Gemini secara umum dimana pada proses percakapan dengan *chatbot* penting untuk memperhatikan pola serta teknik dari *text prompting* yang digunakan karena kualitas *text prompt* yang diberikan sangat mempengaruhi kuantitas, kualitas serta kompleksitas dari respon yang diberikan oleh *chatbot*. Kemudian, respon yang diberikan oleh *chatbot* tidak bisa sepenuhnya langsung digunakan karena masih diperlukan perbaikan dan juga pemeriksaan atas kebenaran respon yang disampaikan. Kelemahan yang kerap terjadi pada LLM seperti bias dan halusinasi perlu menjadi perhatian khusus bagi pengguna aplikasi *chatbot* untuk meminimalisir kesalahan dari respon yang diberikan oleh *chatbot*. Aplikasi *chatbot* seperti aplikasi Tanya Pak Jr ini bisa digunakan sebagai alat bantu untuk pembelajaran atau pun untuk membantu di dalam pekerjaan dalam bidang-bidang tertentu dan dengan dibukanya akses ke kunci API dari LLM Gemini Pro membuat siapa pun bisa mengembangkan aplikasi *chatbot*-nya sendiri.

DAFTAR PUSTAKA

[1] S. F. Wamba, R. E. Bawack, C. Guthrie, M. M. Querioz dan K. D. Carillo, “Are We Preparing For A Good AI Society? A Bibliometric Review and Research Agenda,” *Technological Forecasting and Social Change,* vol. 164, 2021, https://doi.org/10.1016/j.techfore.2020.120482.

[2] M. U. Hadi, Q. Al-Tashi, R. Qureshi dan e. a. , “Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects,” *TechRxiv,* vol. 4, Seoptember 2023, https://doi.org/10.36227/techrxiv.23589741.v3.

[3] Hu, Krystal, “ChatGPT sets record for fastest-growing user base - analyst note,”. Internet: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/, 2 Februari 2023 [5 Januari 2024].

[4] O. Schwartz, “Andrey Markov & Claude Shannon Counted Letters to Build the First Language-Generation Models,”. Internet: https://spectrum.ieee.org/andrey-markov-and-claude-shannon-built-the-first-language-generation-models, November 2019 [5 Januari 2024].

[5] M. Raeini, “The Evolution of Language Models: From N-Grams to LLMs, and Beyond,” *SSRN,* 2023, https://dx.doi.org/10.2139/ssrn.4625356.

[6] H. Naveed, A. U. Khan, S. Qiu dan e. a. , “A Comprehensive Overview of Large Language Models,” *arXiv,* 2023, https://doi.org/10.48550/arXiv.2307.06435.

[7] Z. Ghahramani, “Introducing PaLM 2,”. Internet: https://blog.google/technology/ai/google-palm-2-ai-large-language-model/, Mei 2023 [4 Januari 2024].

[8] S. Pichai dan D. Hassabis, “Introducing Gemini: Our Largest and Most Capable AI Model,”. Internet: https://blog.google/technology/ai/google-gemini-ai/#sundar-note, Desember 2023 [7 Januari 2024].

[9] G. Team, “Gemini: A Family of Highly Capable,”. Internet: https://storage.googleapis.com/deepmind-media/gemini/gemini_1_report.pdf, 2023 [6 Januari 2024].

[10] E. Adamopoulou dan L. Moussiades, “Chatbots: History, Technology, and Applications,” *Machine Learning with Applications,* vol. 2, 2020, https://doi.org/10.1016/j.mlwa.2020.100006.

[11] E. Sadikoğlu, M. Gök, M. M. Mijwil dan I. Kösesoy, “The Evolution and Impact of Large Language Model Chatbots in Social Media: A Comprehensive Review of Past, Present, and Future Applications,” *Veri Bilimi,* vol. 6, no. 2, pp. 67 - 76, 2023.

[12] J. Banks dan B. Gokturk, “It’s Time for Developers and Enterprises to Build with Gemini Pro,”. Internet: https://blog.google/technology/ai/gemini-api-developers-cloud/. Desember 2023 [6 Januari 2024].

[13] M. H. A. Wahab dan N. Hassan, “E-Organizer: Web Based Reminder via Short Message Service (SMS),” in *E-Community*

Research Center National Conference on Bridging Digital Divide: Malaysia Initiatives 2007, Kuala Lumpur, 2007.

[14] D. Purnomo, "Model Prototyping Pada Pengembangan Sistem Informasi," *Jurnal Informatika Merdeka Pasuruan,* vol. 2, no. 2, pp. 54 - 61, 2017, http://dx.doi.org/10.37438/jimp.v2i2.67.

[15] Google Team, "Panduan Memulai: Memulai Gemini API di Aplikasi Android (SDK Klien),". Internet: https://ai.google.dev/tutorials/android_quickstart?hl=id#generate-text-from-text-and-image-input. Januari 2024 [10 Januari 2024].