

BAB III

ANALISA MASALAH & PERANCANGAN APLIKASI

3.1 Tinjauan Umum Institusi

3.1.1 Sejarah Institusi

Yayasan Pembina Lembaga Pendidikan PGRI yang selama ini dikenal YPLP PGRI berpartisipasi aktif dalam pengembangan pendidikan di Indonesia melalui lembaga - lembaga pendidikan dimulai dari Sekolah Dasar (SD), Sekolah Menengah Pertama (SMP), Sekolah Menengah Atas (SMA) dan Sekolah Menengah Kejuruan (SMK) dan Perguruan Tinggi.

Program Akademi Manajemen Informatika Dan Komputer (AMIK) PGRI Tangerang didirikan pada tanggal 28 Januari 2001 dengan ijin operasional dari Mendiknas nomor 20/D/O/2001. Sehubungan dengan tuntutan sumber daya manusia, Yayasan Pembina Lembaga Pendidikan Perguruan Tinggi (YPLP-PT) PGRI telah maju lebih jauh lagi dengan mendirikan Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) PGRI Tangerang yang memperoleh ijin operasional 35/D/O/2001. Pada tahun 2008 Prodi Sistem Informasi dan Prodi Teknik Informatika Terakreditasi BAN-PT. Diketahui oleh Prof. Dr. H. Aris Gumilar, MM.

YPLP-PT PGRI mengemban misi melaksanakan amanat luhur yang terkandung didalam Pembukaan Undang-undang Dasar 1945, yaitu misi untuk mencerdaskan kehidupan bangsa Indonesia. Tuntutan zaman dan kebutuhan sumber daya manusia yang terampil dan profesional membuat YPLP-PT PGRI

Tangerang terus memperbaiki dan mengembangkan program-program pendidikan dengan biaya terjangkau yang masih sangat langka.

STMIK PGRI Tangerang terletak di Jl. Perintis Kemerdekaan II, Cikokol, Tangerang 15118 dan mempunyai nomor telepon (021) 5523052. Sekolah Tinggi ini menyediakan dua program studi Strata Satu (S1) yaitu Sistem Informasi dan Teknik Informatika.

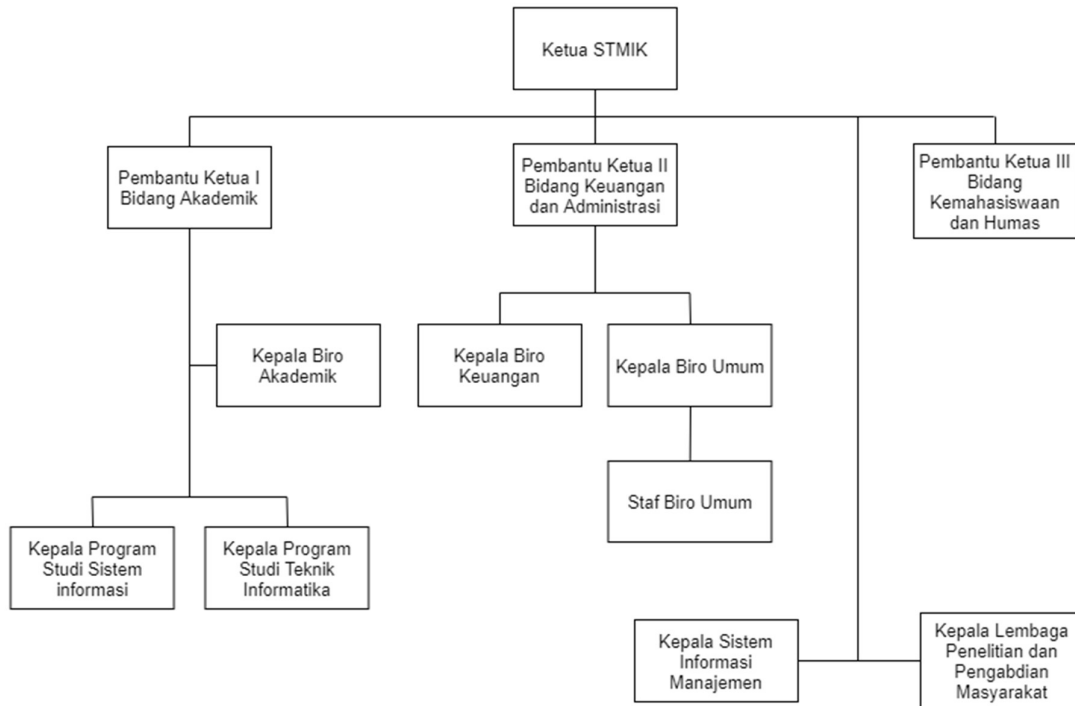
Visi

Terwujudnya STMIK PGRI Tangerang yang modern dan bermutu berbasis Teknologi Informasi yang selalu mengikuti perkembangan zaman dan bermutu dalam aspek layanan dan proses belajar mengajar sehingga berdaya saing di tingkat lokal dan nasional.

Misi

- a. Mencetak Sarjana Komputer yang memiliki integritas kepribadian, nasionalisme yang tinggi, pengembangan kepemimpinan, kemandirian, kerja sama dan penumbuhan rasa etika profesional serta mampu menciptakan dalam mengembangkan Teknologi Informasi.
- b. Meningkatkan kemampuan Transfer Sains dan *Art* di bidang Teknologi Informasi.
- c. Meningkatkan kemampuan *skill* di bidang Teknologi Informasi.

3.1.2 Struktur Organisasi



Gambar 3.1 Struktur Organisasi STMIK PGRI Tangerang

(sumber: Pembantu Ketua I Bidang Akademik STMIK PGRI Tangerang)

Berikut ini adalah daftar kepengurusan di STMIK PGRI Tangerang berdasarkan pada Surat Keputusan Nomor 17/STMIK PGRI/III/2013 tentang Pengangkatan Pengurus/Pembantu Ketua STMIK PGRI Tangerang Periode 2017-2021.

Ketua STMIK PGRI Tangerang	: Drs. H. Aep Gumiwa, M.M
Pembantu Ketua I Bidang Akademik	: Didi Kurnaedi, S.Kom, M.M, M.Kom
Pembantu Ketua II Bidang Keuangan dan Administrasi	: Drs. Agus Herwan, MM
Pembantu Ketua III Bidang Kemahasiswaan dan Humas	: Erna Oktora, M.Kom
Kepala Biro Akademik	: Arif Ginanjar, S.Sos, M.M

Kepala Biro Keuangan	:	Udis Supriyatna, A.Md
Kepala Biro Umum	:	H. Ama Sumarna, S.Kom
Staf Biro Umum	:	Herwanto, S.Kom Dini Pebrianti
Ketua Program Studi Sistem informasi	:	Muhammad Drajat, M.Kom
Ketua Program Studi Teknik Informatika	:	FX Agung Pramanto, S.Kom, M.Kom
Kepala Sistem Informasi Manajemen	:	Mawan Agus Nugroho, M.Kom
Kepala Lembaga Penelitian dan Pengabdian Masyarakat	:	Irfan Nasrullah, S.Kom, M.Kom

3.2 Identifikasi Kebutuhan Sistem

Agar aplikasi *chatbot* yang dibuat dapat merespon atau memberikan timbal balik terhadap masukan pengguna, maka diperlukan sebuah *knowledge* atau pengetahuan untuk *Bot*. Penulis menggolongkan 2 jenis pengetahuan ke dalam 2 tabel di *database* berdasarkan pada masukan yang diterima *Bot* dari pengguna. Kedua tabel tersebut adalah tabel *knowledge_base* dan tabel *command*.

Tabel *knowledge_base* memiliki 106 *record* dan 5 atribut. Atribut *pattern* merupakan pola-pola masukan pengguna yang menggunakan bahasa natural, dan atribut *template* berupa respon/jawaban yang akan diberikan *Bot*. Tabel di bawah ini menunjukkan data sampel dari tabel *knowledge_base*.

Tabel 3.1 Tabel *knowledge_base*

<i>id_kb</i>	<i>topic</i>	<i>pattern</i>	<i>template</i>	<i>flag</i>
9	kampus	fasilitas	Fasilitas yang tersedia di STMIK PGRI: 1. Ruang kelas ber-AC dan terpasang Infocus (Proyektor LCD)	1

			<p>2. Laboratorium Komputer</p> <p>3. Website portal dan internet</p> <p>4. Perpustakaan</p> <p>5. Lapangan Olahraga</p> <p>6. WiFi gratis</p> <p>7. Parkir gratis</p>	
80	kampus	fasilitas kampus stmik pgri tangerang fasilitasnya	<\$goto="fasilitas">	0
97	kampus	logo	<\$photo="file/logo_pgri.jpg">	1
98	kampus	logo gambar stmik pgri tangerang	<\$photo="file/logo_pgri.jpg">	0
73	kampus	akreditasi	<p>Kampus STMIK PGRI Tangerang terakreditasi BAN-PT dengan No. SK 161/SK/BAN-PT/Akred/PT/VII/2018.</p> <p>Program Studi Sistem Informasi terakreditasi C dengan No. SK 032/SK/BAN-PT/Akred/S/I/2015.</p> <p>Program Studi Teknik Informatika terakreditasi C dengan No. SK 275/SK/BAN-PT/Akred/S/VIII/2014.</p>	1
17	kampus	akreditasi terakreditasi akreditasinya kampus institusi stmik pgri tangerang	<\$goto="akreditasi">	0
100	kampus	sejarah	Pada awalnya, Akademi Manajemen Informatika dan Komputer (AMIK)	1

			PGRI didirikan pada tanggal 28 Januari 2001. Seiring perkembangan, AMIK PGRI berubah bentuk menjadi Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) PGRI pada tanggal 1 Maret 2004	
101	kampus	sejarah awal mula kampus stmik pgri tangerang berdiri	<\$goto="sejarah">	0
65	kampus - alamat	alamat	Kampus STMIK PGRI Tangerang beralamatkan di Jl. Perintis Kemerdekaan II, Cikokol, Tangerang 15118 https://goo.gl/maps/Gh2nPA8A62vRpc8t8	1
4	kampus - alamat	alamat lokasi letak terletak kampus stmik pgri tangerang	<\$goto="alamat">	0
103	kampus - kontak	kontak	Nomor Telepon (021) 552 3052 Fax (021) 5579 5777	1
5	kampus - kontak	kontak nomor telepon telpon fax stmik pgri tangerang	<\$goto="kontak">	0
68	kampus - dosen	dosen	Terdapat 15 dosen tetap yang mengajar di STMIK PGRI Tangerang, antara lain: - Agus Danuwidodo, S.Kom - Ahmad Fikri Sidharta, S.Kom - Didi Kurnaedi, M.M., M.Kom	1

			<ul style="list-style-type: none"> - Dody, S.Kom - Eko Dharmawan, S.Kom, M.M. - Erna Oktora, S.Kom, M.Kom - Fitri Nurmayanti - FX Agung Pramanto, S.Kom, M.Kom - Ika Dewi Lestari, S.Kom, M.Kom - Irfan Nasrullah, S.Kom, M.Kom - Lucia Puspitasari, S.Pd, M.Pd - Maindarto Sukowati, S.Kom, M.Si - Muhammad Drajat Julianto, M.Kom - Rakhmat Gunawan, S.E. - Ramlan, S.Kom, M.Kom 	
94	kampus - dosen	daftar list nama pengajar mengajar gelar bergelar jumlah total dosen tetap stmik pgri tangerang	<\$goto="dosen">	0
82	kampus - jurusan	jurusan	<p>STMIK PGRI Tangerang menyediakan 2 jurusan atau Program Studi jenjang Sarjana (S1) yaitu:</p> <ol style="list-style-type: none"> 1. Sistem Informasi 2. Teknik Informatika 	1
18	kampus - jurusan	penjurusan jurusan jurusannya program studi stmik pgri tangerang	<\$goto="jurusan">	0
3	mahasiswa - pendaftaran	biaya spp uang harga daftar mendaftar	<\$command="/biayadaftar">	0

		pendaftaran pendaftarannya mahasiswa baru		
20	mahasiswa - kelengkapan	fungsi manfaat kegunaan jas jaket almamater digunakan menggunakan dipakai memakai dikenakan mengenakan	Almamater hanya dikenakan pada saat tertentu saja seperti: 1. Ujian Tengah dan Akhir Semester. 2. Sebagai panitia dalam acara atau kegiatan kampus seperti seminar, workshop, ospek, dll. 3. Sebagai tanda pengenal mahasiswa dalam kegiatan atau tugas diluar kampus seperti: KKN (Kuliah Kerja Nyata).	0
27	organisasi	organisasi	Organisasi yang ada di STMIK PGRI Tangerang antara lain Badan Eksekutif Mahasiswa (BEM) dan Unit Kegiatan Mahasiswa (UKM).	1
12	perpustakaan	syarat persyaratan aturan peraturan pinjam minjam meminjam peminjaman buku perpustakaan perpus	Syarat/Peraturan peminjaman buku di perpustakaan STMIK PGRI sebagai berikut: - Peminjam adalah mahasiswa aktif STMIK PGRI, dosen serta staff STMIK PGRI dengan menunjukkan tanda pengenal (seperti Kartu Tanda Mahasiswa). - Lama peminjaman adalah 1 minggu. - Menjaga dan merawat buku yang dipinjam. - Mengembalikan buku tepat pada waktunya. Apabila terlambat, maka dikenakan denda sebesar Rp.2.000,- per hari.	0

Tabel *command* memiliki 15 *record* dan 4 atribut. Atribut *command* berisikan data masukan berupa *command* atau perintah pada *Telegram Messenger* (biasanya diawali dengan tanda garis miring “/”) serta atribut *response* berupa respon yang akan diberikan *Bot*.

Tabel 3.2 Tabel *command*

id	<i>command</i>	<i>response</i>	<i>flag</i>
1	!exist	Maaf, command yang kamu masukkan salah atau belum terdaftar.	1
2	!understand	Maaf, Vranciska belum mengerti perkataan kamu. Coba gunakan Bahasa Indonesia yang baik, benar dan sesuai dengan EYD. Vranciska akan bantu jawab pertanyaan kamu tentang STMIK PGRI Tangerang yaa :)	1
3	!dualanswer	Maaf saat ini Vranciska belum mempunyai jawaban yang *tepat* atas pertanyaan kamu. Coba diperjelas lagi maksud kamu atau gunakan kata kunci lain nya :)	1
4	!jelas	Mohon diperjelas lagi ya maksudnya supaya Vranciska tidak bingung.	1
5	!keyword	Beberapa kata kunci dibawah ini mungkin bisa membantu:	1
6	/start	Hai <name>, salam kenal aku Vranciska. Selamat datang di STMIK PGRI Bot! Disini Vranciska sebagai Virtual Assistant Chatbot akan membantu kamu memperoleh informasi-informasi umum berkenaan dengan STMIK PGRI Tangerang. Adakah yang ingin kamu tanyakan?	1
7	/help	Vranciska dibuat untuk membantu pengguna memperoleh informasi-informasi umum dalam lingkup STMIK PGRI Tangerang. Kamu dapat	1

		<p>langsung mengirimkan pertanyaan atau meminta informasi yang kamu butuhkan disini.</p> <p>Mohon gunakan Bahasa Indonesia yang baik dan benar sesuai dengan EYD. Selain itu kamu juga dapat langsung memilih menu informasi yang tersedia dimulai dengan mengetik atau menekan tombol garis miring [/].</p> <p>Apabila informasi yang Vranciska berikan belum cukup, kamu bisa bertanya langsung kepada Customer Service STMIK PGRI Tangerang dengan mengetik: `/tanya <spasi> isi pertanyaan` Atau memberikan saran dengan mengetik: `/saran <spasi> isi saran` dalam satu baris saja (tanpa enter)</p> <p>Contoh: `/tanya berapa biaya pendaftaran mahasiswa baru?`</p>	
8	/tanya	Pertanyaan kamu telah Vranciska terima dan diteruskan ke Customer Service STMIK PGRI Tangerang. Mohon ketersediaannya untuk menunggu balasan ya, terima kasih :)	2
9	/saran	Terima kasih atas saran yang diberikan :) Masukan kamu akan membantu kami dalam memberikan pelayanan yang lebih baik.	2
10	/pendaftaran	Informasi apa yang kamu butuhkan tentang pendaftaran?	3
11	/syaratdaftar	Berikut ini adalah syarat pendaftaran mahasiswa baru tahun akademik 2019/2020	3

		1. Mengisi Formulir Pendaftaran. 2. Fotocopy Ijazah yang dilegalisir 2 lembar. 3. Pas foto 3x4 sebanyak 3 lembar. 4. Membayar administrasi keuangan.	
12	/biayadaftar	Berikut adalah rincian biaya administrasi keuangan pendaftaran mahasiswa baru: <photo="file/biaya daftar.png">	3
13	/waktutempatdaftar	Pendaftaran mahasiswa baru dimulai dari 5 April 2019, Pukul 08.00 - 20.00 WIB di kampus STMIK PGRI Tangerang.	3
14	/jurusan	STMIK PGRI Tangerang menyediakan 2 jurusan atau Program Studi jenjang Sarjana (S1) yaitu 1. Sistem Informasi 2. Teknik Informatika.	1
15	/kontak	Nomor Telepon (021) 552 3052 Fax (021) 5579 5777	1

Untuk mendukung aplikasi *chatbot* yang telah dibuat, maka diperlukan sebuah *Web Admin*. Berikut ini adalah fungsi/kegunaan dari *Web Admin* yang akan dibuat:

1. Mengelola pengetahuan *Bot* seperti menambah, mengubah dan menghapus data pengetahuan.
2. Melihat *chat history* (riwayat percakapan) dan *feedback* (umpan balik) antara pengguna dengan *chatbot*.
3. Memberikan pesan balasan kepada pengguna atas *chat* atau *feedback* yang diterima oleh *chatbot*.

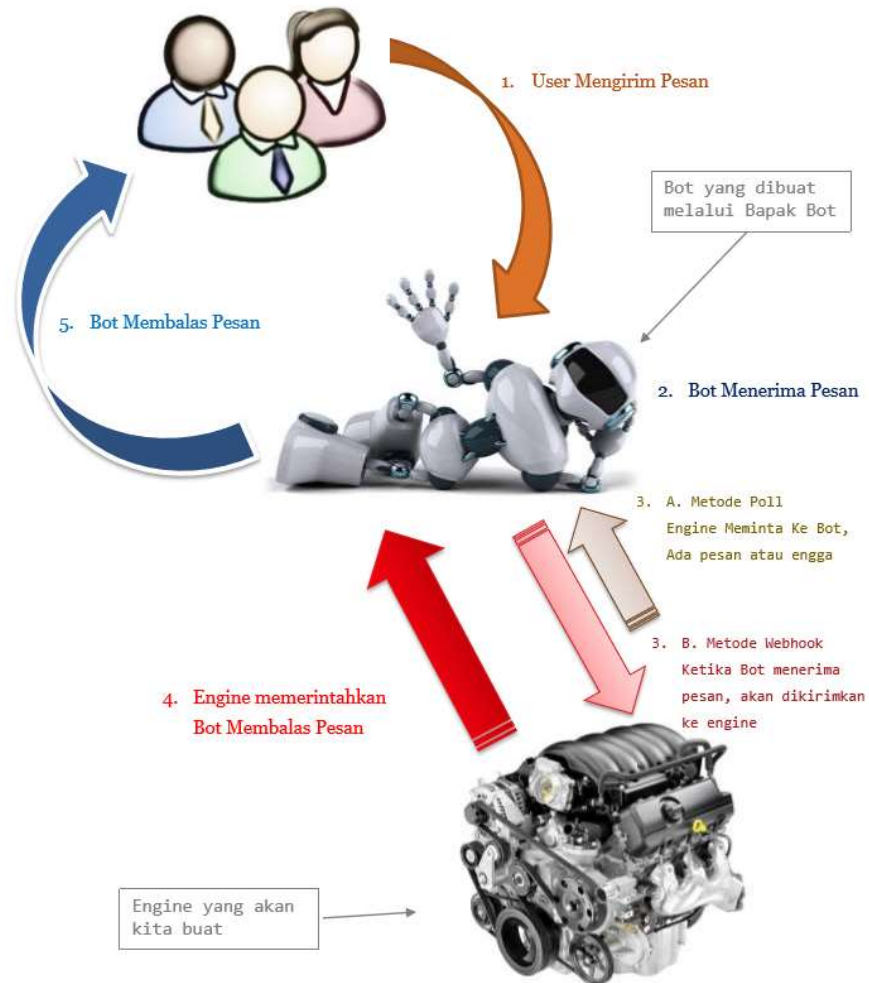
3.3 Metode yang Digunakan

Penulis membagi dua jenis metode yang digunakan dalam perancangan aplikasi *chatbot* pada media *Telegram Messenger* ini, yaitu metode untuk membuat *Bot Telegram* dan metode untuk pencocokan pola (*Fulltext Search Boolean Mode*).

3.3.1 Metode untuk Membuat *Bot Telegram*

Salah satu proses penting dalam pembuatan *Bot Telegram* adalah *updates* (pembaruan-pembaruan) pada *Bot*. *Bot* akan bekerja sesuai dengan pembaruan yang diterima, contohnya *Bot* mengirimkan pesan ke pengguna. *Bot* tidak akan bekerja (*idle*) apabila tidak ada pembaruan yang diterima.

(Telegram Team, Telegram Bot API 2013) mengungkapkan bahwa “*There are two mutually exclusive ways of receiving updates for your Bot — the getUpdates method on one hand and Webhooks on the other*”. Terdapat dua metode yang dapat digunakan untuk *Bot* dapat menerima pembaruan, yakni *getUpdates* (atau disebut juga metode *Long-Polling*) dan metode *Webhook*. Metode *Long-Polling* merupakan metode *default* dari *Telegram*. Gambar 3.1 akan menjelaskan mengenai cara kerja *Bot Telegram*.

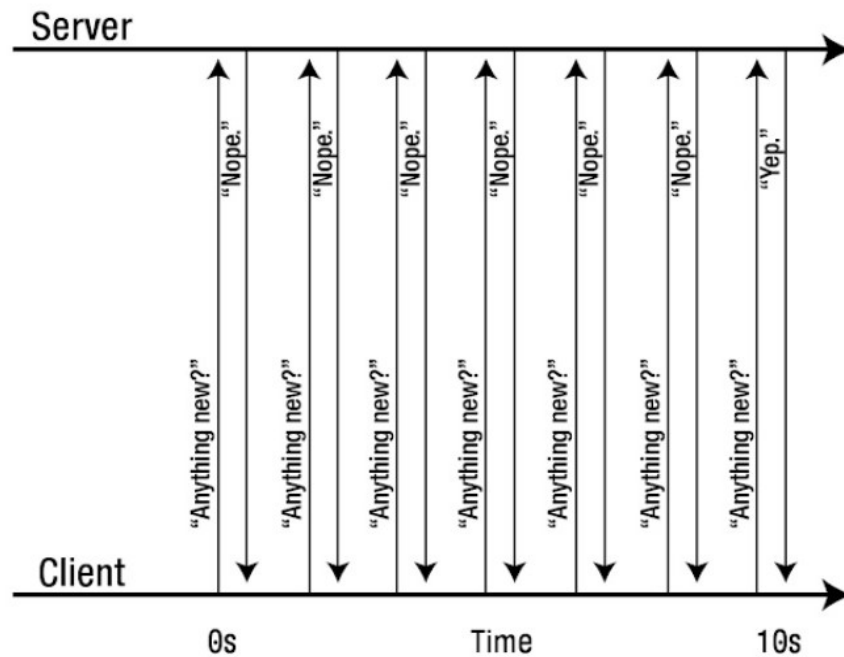


Gambar 3.2 Cara kerja *Bot Telegram*

(sumber: Edisi 3 - Membuat *Bot TELEGRAM* dari PHP - lite.pdf)

a. *Long-Polling*

Polling adalah mekanisme untuk mengambil data baru (pembaruan data). Teknik ini dilakukan dengan *client* meminta *server* untuk data baru secara terus menerus. *Polling* akan mengirimkan *HTTP request* (permintaan *HTTP*) untuk mengecek pembaruan atau informasi terbaru, kemudian *server* akan merespon apakah terdapat pembaruan atau tidak. Gambar 3.2 di bawah ini mengilustrasikan cara kerja metode *Polling* ini.



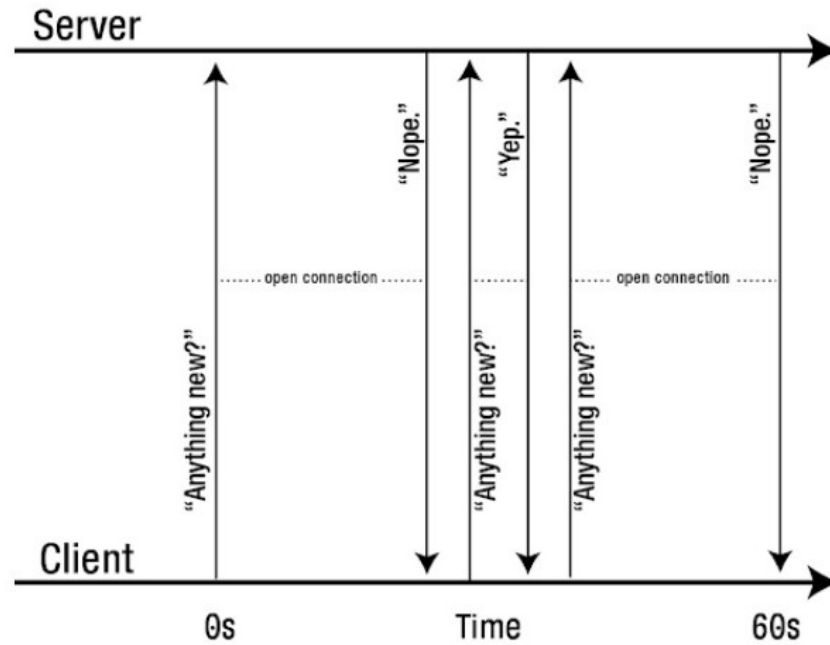
Gambar 3.3 Ilustrasi metode *Polling*

(sumber: *Realtime Web Apps: HTML5 WebSocket, Pusher, and the Web's Next Big Thing*. 2018:6)

Dari ilustrasi di atas, *client* meminta kepada *server* apakah ada pembaruan atau tidak secara terus menerus. Koneksi terbuka saat *client* melakukan *request* dan kemudian tertutup setelah *server* mengirimkan jawaban. Seperti halnya pada kehidupan nyata, komunikasi antara *client* dan *server* seperti di atas sangat tidak produktif. Hal tersebut akan menjadi masalah apabila beberapa *client* mengakses satu *server* yang sama dan menggunakan metode *Polling* berulang kali. Bila dalam *Polling* setiap *client* dilakukan *request* satu kali dalam satu detik, dan terdapat seratus ribu *client* melakukan hal yang sama dalam waktu yang bersamaan pula, maka terdapat enam juta *request* dalam satu menitnya. Hal ini tentunya dapat menyebabkan *server* menjadi sibuk dan tentu akan mempengaruhi

performanya dalam merespon dan memenuhi kebutuhan *client*. Kemudian pada perkembangan selanjutnya muncul metode *Long-Polling* sebagai solusi dalam mengatasi permasalahan tersebut.

Metode *Long-Polling* adalah metode *Polling* dengan interval waktu yang berkala. Pada *Long-Polling*, setiap *request* akan ditahan oleh *server* dan koneksi dibiarkan terbuka selama periode waktu tertentu. Lamanya waktu menunggu *server* dapat ditentukan oleh *client* atau *server* dapat menggunakan standar tetap. Apabila selama waktu yang telah ditentukan tidak ada pembaruan yang terjadi, *server* akan memberikan respon *timeout*. Sedangkan apabila terdapat pembaruan maka *server* akan langsung mengirimkan pembaruan tersebut. Setelah *client* menerima respon maka transaksi selesai dan koneksi antara *client* dan *server* ditutup. *Client* kemudian dapat membuat *request* baru kembali. Untuk lebih jelasnya mengenai metode *Long-Polling* ini dapat dilihat pada gambar 3.3 di bawah ini.



Gambar 3.4 Ilustrasi metode *Long-Polling*

(sumber: *Realtime Web Apps: HTML5 WebSocket, Pusher, and the Web's Next Big Thing*. 2018:7)

Menurut (Lengstorf, Leggetter dan Newman 2013, 7) salah satu permasalahan dengan metode ini adalah di antara *request Long-Polling* ada periode singkat dimana terdapat kemungkinan untuk data pada *client* menjadi tidak sinkron dengan data di *server* terjadi. Hanya ketika koneksi telah dibentuk kembali *client* dapat memeriksa untuk melihat apakah ada data baru (pembaruan) tersedia. Dampak negatif ini sangat bergantung pada data, terlebih jika datanya sangat sensitif terhadap waktu.

Untuk menggunakan metode *Long-Polling* ini memerlukan sebuah perangkat komputer sebagai *server* yang akan melakukan *request* ke *server Bot Telegram*. Metode *Long-Polling* ini cocok digunakan pada

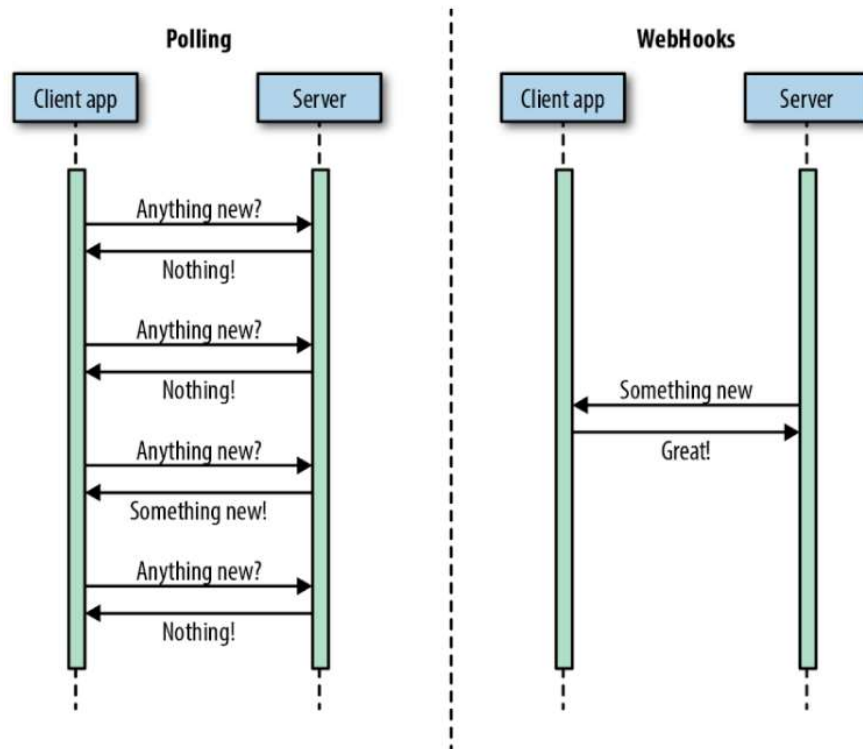
tahap pembuatan dan pengujian *Bot Telegram* sebelum di implementasikan. Metode ini juga dapat digunakan untuk pengembangan maupun pemeliharaan *Bot* yang telah dibuat sebelumnya.

b. *Webhook*

(Jin, Sahni dan Shevat 2018) mengungkapkan bahwa: “*A Webhook is just a URL that accepts an HTTP POST (or GET, PUT, DELETE). An API provider implementing Webhooks will simply POST a message to the configured URL when something happens. Unlike with request-response APIs, with Webhooks, you can receive updates in real time*”.

Webhook hanyalah sebuah URL yang menerima HTTP POST (atau GET, PUT, DELETE). Penyedia API yang menerapkan *webhook* hanya akan mengirim pesan ke URL yang dikonfigurasi ketika sesuatu terjadi. Tidak seperti API *request-response*, dengan *webhook*, anda dapat menerima pembaruan secara *real time*.

Webhook sangat bagus untuk berbagi data *real-time* dari satu *server* ke *server* lain dengan mudah. Jika pada metode *Long-Polling client* diharuskan membuat *request* pembaruan dan diikuti respon balasan, metode *webhook* hanya mengirimkan data ketika terjadi pembaruan. *Webhook* tidak membutuhkan koneksi yang terus terbuka. Pengirim membuat HTTP *request* ke penerima ketika ada pembaruan atau data baru. Gambar 3.4 di bawah ini mengilustrasikan perbedaan cara kerja *Polling* dan *Webhook*.



Gambar 3.5 Perbedaan *Polling* dan *Webhook*

(sumber: *Designing Web APIs: Building APIs That Developers Love*.

2018:20)

Sebelum dapat menggunakan metode *webhook*, pertama kali yang harus dilakukan adalah mengkonfigurasi *callback* URL. *Server* pengirim membuat *request* HTTP ke *server* penerima ketika ada pembaruan, dan mengirim data pembaruan tersebut. Dalam pembuatan *Bot Telegram*, API atau *server Bot Telegram* bertindak sebagai *server* pengirim. Dikarenakan metode *webhook* merupakan metode yang digunakan untuk komunikasi antar *server*, maka dalam pembuatan *Bot Telegram* diperlukan komputer *server* atau dapat juga menggunakan *hosting*. Persyaratan lainnya adalah *server* tersebut harus memiliki sambungan/koneksi HTTPS. Metode

webhook cocok digunakan jika untuk mengimplementasikan *Bot Telegram* yang telah dibuat.

Dari penjelasan di atas penulis akan mengambil kesimpulan mengenai metode *Long-Polling* dan metode *Webhook*. Metode *Long-Polling* adalah metode *default* dalam pembuatan *Bot Telegram*, dimana *client* meminta pembaruan ke *server* secara terus menerus dengan interval waktu yang berkala. Keunggulan metode ini adalah dapat menggunakan komputer apa saja sebagai *client* yang akan melakukan *request* ke API atau *server Bot Telegram*, dan cocok untuk tahap pembuatan dan pengujian sebelum *Bot Telegram* di implementasikan. Sedangkan kelemahan metode *Long-Polling* adalah proses *request-respon* yang tidak efektif dan dapat memakan sumber daya yang besar, serta adanya kemungkinan data antara *client* dan *server* menjadi tidak sinkron sehingga berimbas pada respon yang lama atau pertukaran data tidak *real-time*. Sedangkan keunggulan metode *Webhook* adalah pertukaran data yang *real-time* dan hanya perlu mengkonfigurasi *callback URL*, selanjutnya *server* pengirim akan mengirimkan data ke URL yang dikonfigurasi bila terdapat pembaruan. Dan kelemahan metode *webhook* adalah perlu adanya *server* yang memiliki koneksi HTTPS, atau menggunakan *hosting* bila tidak ada.

Untuk membuat aplikasi *chatbot* di *Telegram Messenger* ini, penulis akan menggunakan metode *Long-Polling* pada tahap desain dan penulisan kode program. Penggunaan metode *Long-Polling* akan memudahkan penulis dalam proses pembuatan, pengujian, dan menemukan kesalahan kode program atau *bug* yang ada pada *chatbot* yang dibuat. Pada tahap implementasi, aplikasi akan di *upload* ke *hosting* bersertifikat HTTPS yang penulis sewa dan menggunakan metode *Webhook*.

3.3.2 Pencocokan *Fulltext Search Boolean Mode*

Setelah *Bot Telegram* yang dibuat telah berjalan, dapat menerima dan mengirim pesan, langkah selanjutnya adalah menerapkan metode pencocokan pola dan respon agar *Bot* dapat memberikan respon jawaban yang tepat sesuai dengan pola masukan pengguna yang beragam. Gambar dibawah ini merupakan hasil menjalankan *query* pada *database* MySQL yang diurutkan berdasarkan nilai relevansi tertinggi dengan jumlah *record* pada tabel sebanyak 19 *record*.

```

mysql> SELECT id, pattern, MATCH(pattern)
-> AGAINST('lokasi bedugul' IN BOOLEAN MODE) AS relevansi
-> FROM tb_datawisata ORDER BY(relevansi) DESC;

```

id	pattern	relevansi
5	lokasi bedugul	1.7405182123184204
1	lokasi pura tanah lot	0.1053074449300766
2	lokasi pura besakih	0.1053074449300766
3	lokasi gunung batur	0.1053074449300766
6	lokasi sangeh	0.1053074449300766
7	lokasi pantai pandawa	0.1053074449300766
8	lokasi pura taman ayun	0.1053074449300766
9	lokasi monkey forest	0.1053074449300766
10	lokasi pantai kuta	0.1053074449300766
4	kuliner klungkung	0
11	objek wisata badung	0
12	objek wisata gianyar	0
13	objek wisata klungkung	0
14	objek wisata buleleng	0
15	objek wisata karangasem	0
16	objek wisata tabanan	0
17	objek wisata bangli	0
18	objek wisata jembrana	0
19	objek wisata denpasar	0

19 rows in set (0.00 sec)

Gambar 3.6 Nilai Relevansi Hasil Pencocokan *Fulltext Search Boolean Mode*

(sumber: Pencarian Informasi Wisata Daerah Bali menggunakan Teknologi *Chatbot*. 2017:151)

Metode *Fulltext Search Boolean Mode* melakukan pencarian terhadap kata 'lokasi bedugul' dan menghasilkan relevansi kata paling tinggi pada data dengan ID 5 karena mengandung kemiripan lebih banyak (kata yang muncul lebih banyak) dibandingkan data dengan ID 1, 2, 3, 6, 7, 8, 9, 10. Dibawah ini

akan dijabarkan perhitungan metode *Fulltext Search Boolean Mode* dalam menentukan nilai relevansi pencarian. Data yang diambil untuk dihitung adalah data dengan ID 5 ‘lokasi bedugul’ dan data dengan ID 6 ‘lokasi sangeh’.

Tabel 3.3 Term Frequency (TF) Kata

ID	Pattern	Kata	TF
5	lokasi bedugul	lokasi	1
		bedugul	1
6	lokasi sangeh	lokasi	1
		bedugul	0

Tabel 3.3 menunjukkan tingkat kemunculan kata pada dokumen. Pada data dengan ID 5, kata ‘lokasi’ muncul sebanyak 1 kali dan kata ‘bedugul’ muncul sebanyak 1 kali. Sedangkan pada data dengan ID 6, kata ‘lokasi’ muncul sebanyak 1 kali dan kata ‘bedugul’ tidak muncul sama sekali. Kemudian perhitungan dilanjutkan dengan mencari nilai *Inverse Document Frequency* (IDF) kata pencarian ‘lokasi bedugul’ pada dokumen.

Tabel 3.4 Inverse Document Frequency (IDF) Kata

Kata	Matching Record	Total Record	IDF
lokasi	9	19	$\log_{10}(19/9)$
bedugul	1		$\log_{10}(19/1)$

Tabel 3.4 menunjukkan nilai IDF yang didapat dari total *record* dibandingkan dengan jumlah *record* yang terkait dengan kata ‘lokasi’ dan ‘bedugul’. Perhitungan dilanjutkan dengan memasukkan nilai TF dan IDF pada persamaan $\{rank\} = \{TF\} * \{IDF\} * \{IDF\}$.

Tabel 3.5 Hasil Sementara Perhitungan Relevansi

ID	Kata	Rumus	Hasil Sementara
5	lokasi	$1 * \log_{10}(19/9) * \log_{10}(19/9)$	0.10530744850045075
	bedugul	$1 * \log_{10}(19/1) * \log_{10}(19/1)$	1.6352107719498268
6	lokasi	$1 * \log_{10}(19/9) * \log_{10}(19/9)$	0.10530744850045075
	bedugul	$0 * \log_{10}(19/1) * \log_{10}(19/1)$	0

Tabel 3.5 merupakan hasil sementara perhitungan nilai relevansi pada kata 'lokasi' dan 'bedugul', sehingga untuk mendapatkan nilai akhir dari relevansi pencarian kata 'lokasi bedugul' yaitu dengan menjumlah hasil sementara dari masing-masing kata.

Tabel 3.6 Hasil Akhir Perhitungan Relevansi

ID	Rumus	Hasil
5	$0.10530744850045075 + 1.6352107719498268$	1.74051822045027755
6	$0.10530744850045075 + 0$	0.10530744850045075

Tabel 3.6 merupakan hasil akhir perhitungan penerapan *Fulltext Search Boolean Mode* pada pencarian kata 'lokasi bedugul'. Hasil perhitungan manual pada data dengan ID 5 adalah 1.74051822045027755 dan hasil menjalankan *query* pada gambar 3.5 adalah 1.7405182123184204. Hasil dari perhitungan manual dan menjalankan *query* memperoleh hasil yang sama jika dilakukan pembulatan.

3.3.3 Penyaringan Format Pesan yang Diterima

Penulis mengategorikan dua jenis masukan/pesan yang dapat diterima oleh *Bot* yang akan dibuat ini yaitu pesan perintah dan pesan teks.

Pengkategorian ini dimaksudkan agar *Bot* mencari data respon/jawaban dari tabel yang sesuai dengan masukan yang diterima. Pesan perintah (*command*) akan mencari data respon dari tabel *command*, sedangkan pesan teks akan mencari data respon dari tabel *knowledge_base*. Untuk dapat membedakan kedua jenis pesan tersebut maka penulis menggunakan *regular expression*.

a. Pesan perintah (*command*)

Pesan berjenis perintah atau *command* yang dikirimkan pengguna diawali dengan tanda garis miring (/). Contohnya adalah “/start”, “/help”. Terdapat juga *command* yang dilanjutkan dengan teks biasa seperti contohnya “/tanya dimana alamat STMIK PGRI?”. Sintaks *regular expression* yang digunakan adalah sebagai berikut.

$$/^([\!\\\/\$])([[:word:]]+)(\s?)(.*)/$$

The screenshot shows a regular expression testing interface. The regular expression is `/^([\!\\\/\$])([[:word:]]+)(\s?)(.*)/`. The test string is `/start`. The tool reports 1 match. The match information table is as follows:

Match	Start	End	Text
Full match	0-6		/start
Group 1.	0-1		/
Group 2.	1-6		start
Group 3.	6-6		
Group 4.	6-6		

The explanation on the right details the components of the regex:

- `^`: asserts position at start of the string.
- `1st Capturing Group` `([\!\\\/\$])`: matches a single character from the list `[\!\\\/\$]`.
 - `\!` matches the character `!` literally (case sensitive).
 - `\/` matches the character `/` literally (case sensitive).
 - `\/` matches the character `\/` literally (case sensitive).
 - `\$` matches the character `$` literally (case sensitive).
- `2nd Capturing Group` `([[:word:]]+)`: matches a single character from the list `[[:word:]]+`.
 - `+` Quantifier — Matches between **one** and **unlimited** times, as many times as possible, giving back as needed (greedy).
 - `[:word:]` matches an alphanumeric character or `[_A-Za-z0-9_]` (also written as `\w`).
- `3rd Capturing Group` `(\s?)`: matches any whitespace character (equal to `[\r\n\t\f\v]`).
 - `?` Quantifier — Matches between **zero** and **one** times, as many times as possible, giving back as needed (greedy).
- `4th Capturing Group` `(.*)`: matches any character (except for line terminators).
 - `*` Quantifier — Matches between **zero** and **unlimited** times, as many times as possible, giving back as needed (greedy).

Gambar 3.7 Hasil dan Penjelasan *Match Regular Expression* dari String

“/start”

The screenshot displays a regular expression testing interface. On the left, the 'REGULAR EXPRESSION' field contains the pattern `/^([\w/\$])([[:word:]]+)(\s?)(.*)/`. Below it, the 'TEST STRING' field contains `/tanya dimana alamat STMIK PGRI?`. On the right, the 'EXPLANATION' section breaks down the pattern into four capturing groups: 1) `^([\w/\$])` for the slash, 2) `[[:word:]]+` for the word 'tanya', 3) `(\s?)` for the space, and 4) `(.*)` for the rest of the string. The 'MATCH INFORMATION' table at the bottom summarizes these matches.

Match	Start	End	Match
Full match	0-32		/tanya dimana alamat STMIK PGRI?
Group 1.	0-1		/
Group 2.	1-6		tanya
Group 3.	6-7		
Group 4.	7-32		dimana alamat STMIK PGRI?

Gambar 3.8 Hasil dan Penjelasan *Match Regular Expression* dari String “/tanya dimana alamat STMIK PGRI?”

b. Pesan teks

Pesan teks dapat terdiri satu kata atau lebih. Karena *Fulltext Search Boolean Mode* melakukan pencocokan terhadap tingkat kemunculan kata pada dokumen, maka ada kemungkinan mendapati hasil relevansi yang paling tinggi memiliki nilai yang sama. Dengan kata lain, terdapat lebih dari satu jawaban atas masukan yang diberikan. Hal ini biasanya terjadi pada masukan yang hanya terdiri dari satu kata, misalnya “biaya”. Biaya dapat merujuk pada *pattern* “biaya pendaftaran”, “biaya keterlambatan pengembalian buku”, atau “biaya pembuatan KTM”.

Oleh karena itu, penulis menggunakan atribut tambahan yaitu *flag* pada tabel *knowledge_base* sebagai tanda bahwa *pattern* pada *record* tersebut terdiri dari satu kata atau lebih. Nilai *flag=1* untuk *record* yang *pattern*-nya yang terdiri dari satu kata dan nilai *flag=0* untuk *record* yang

pattern-nya mengandung lebih dari satu kata. Selanjutnya dilakukan pencocokan menggunakan *Fulltext Search Boolean Mode* dengan filter *flag*.

Sintaks *regular expression* yang digunakan untuk menentukan teks yang diterima terdiri dari satu kata adalah sebagai berikut.

$$/^\backslash(\backslash\w+\backslash\w)(\backslash\W+)?\$/$$

The screenshot displays a regular expression tool interface. On the left, the 'REGULAR EXPRESSION' field contains the pattern `^\backslash(\backslash\w+\backslash\w)(\backslash\W+)?\$/` and the 'TEST STRING' field contains 'hallo'. A status bar indicates '1 match, 10 steps (~0ms)'. On the right, the 'EXPLANATION' section provides a detailed breakdown of the pattern:

- `^`: asserts position at start of the string
- `\backslash(\backslash\w+\backslash\w)`: 1st Capturing Group
 - `\backslash`: assert position at a word boundary: `(^\w|\w$|\W|\w|\W)`
 - `\w+`: matches any word character (equal to `[a-zA-Z0-9_]`)
 - `\w`: Quantifier — Matches between **one** and **unlimited** times, as many times as possible, giving back as needed (*greedy*)
 - `\backslash`: assert position at a word boundary: `(^\w|\w$|\W|\w|\W)`
- `(\backslash\W+)?`: 2nd Capturing Group
 - `?`: Quantifier — Matches between **zero** and **one** times, as many times as possible, giving back as needed (*greedy*)
 - `\W+`: matches any non-word character (equal to `[^a-zA-Z0-9_]`)
 - `\W`: Quantifier — Matches between **one** and **unlimited** times, as many times as possible, giving back as needed (*greedy*)
- `$`: asserts position at the end of the string, or before the line terminator right at the end of the string (if any)

The 'MATCH INFORMATION' section shows:

Match	Full match	Group 1.
Match 1	0-5 hallo	0-5 hallo

Gambar 3.9 Hasil dan Penjelasan *Match Regular Expression* dari String

“hallo”

The screenshot shows a regular expression testing interface. On the left, the 'REGULAR EXPRESSION' field contains `/^\b\w+\b)(\w+)?$/` and the 'TEST STRING' field contains 'hallo apa kabar?'. A status bar indicates 'no match, 15 steps (~0ms)'. On the right, the 'EXPLANATION' section details the components of the regex:

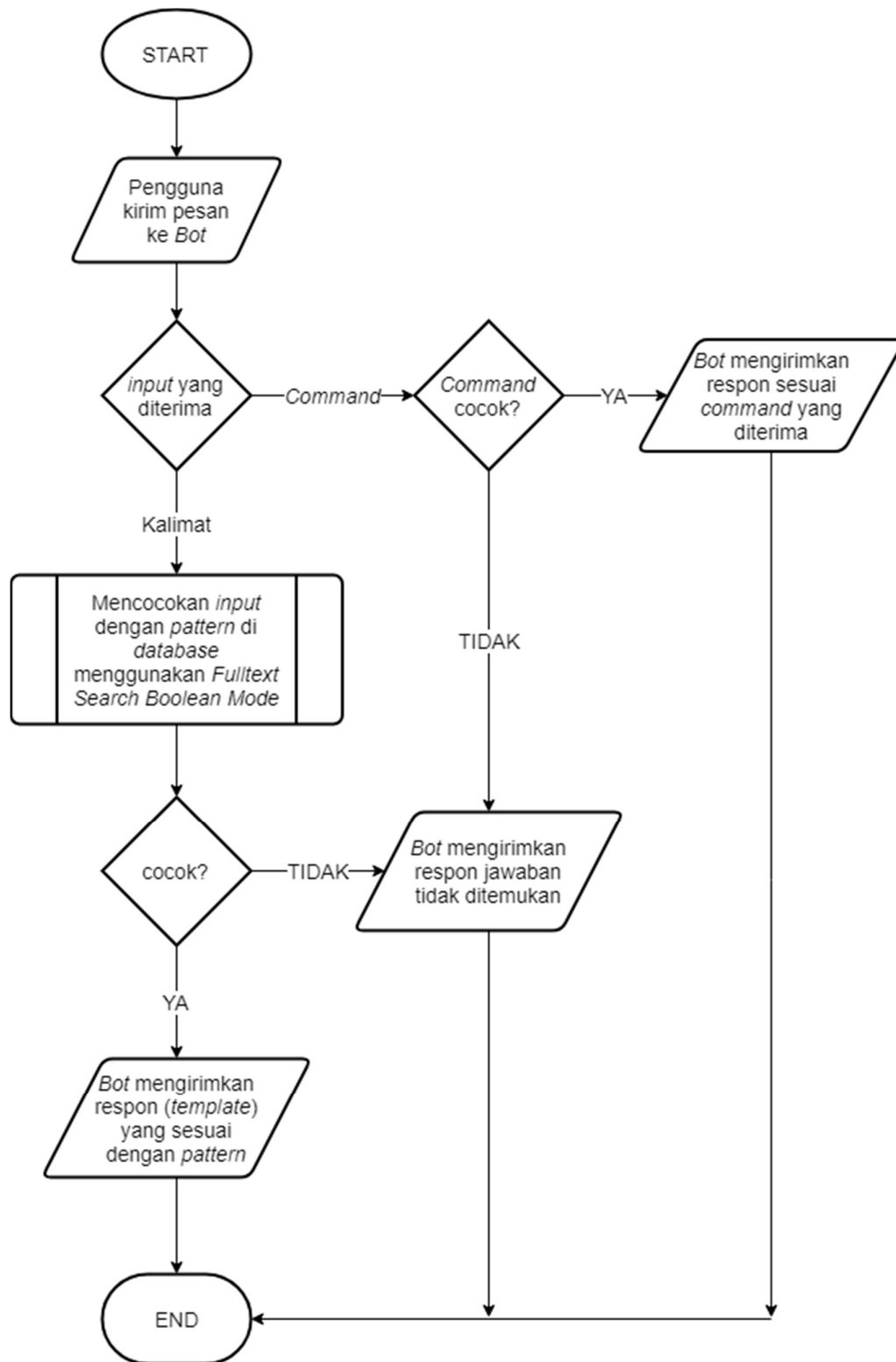
- `^` asserts position at start of the string
- 1st Capturing Group** `(\b\w+\b)`
 - `\b` assert position at a word boundary: `(^\w|\w$|\w!\w!\w!\w)`
 - `\w+` matches any word character (equal to `[a-zA-Z0-9_]`)
 - Quantifier** — Matches between **one** and **unlimited** times, as many times as possible, giving back as needed (*greedy*)
 - `\b` assert position at a word boundary: `(^\w|\w$|\w!\w!\w!\w)`
- 2nd Capturing Group** `(\w+)?`
 - Quantifier** — Matches between **zero** and **one** times, as many times as possible, giving back as needed (*greedy*)
 - `\w+` matches any non-word character (equal to `[^a-zA-Z0-9_]`)
 - Quantifier** — Matches between **one** and **unlimited** times, as many times as possible, giving back as needed (*greedy*)
- `$` asserts position at the end of the string, or before the line terminator right at the end of the string (if any)

The 'MATCH INFORMATION' section states: 'Your regular expression does not match the subject string.'

Gambar 3.10 Hasil dan Penjelasan *Match Regular Expression* dari String

“hallo apa kabar?”

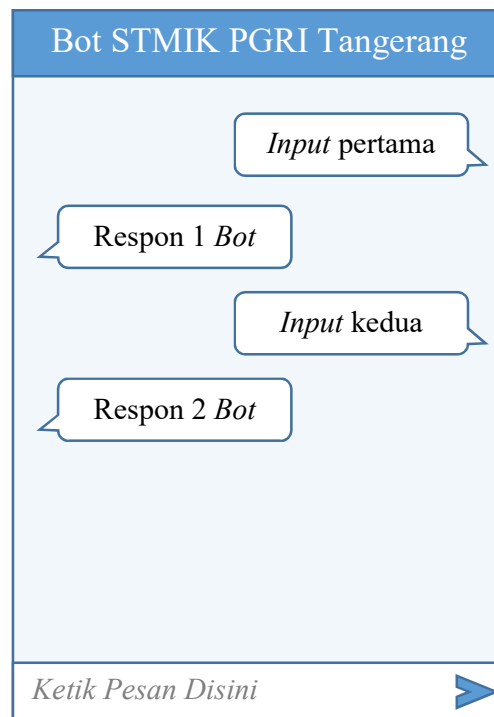
3.4 Perancangan *Flowchart*



Gambar 3.11 *Flowchart* Aplikasi *Chatbot* Pusat Informasi Mahasiswa

3.5 Perancangan Layar, Menu, *Database*

3.5.1 Perancangan Layar *Chatbot*



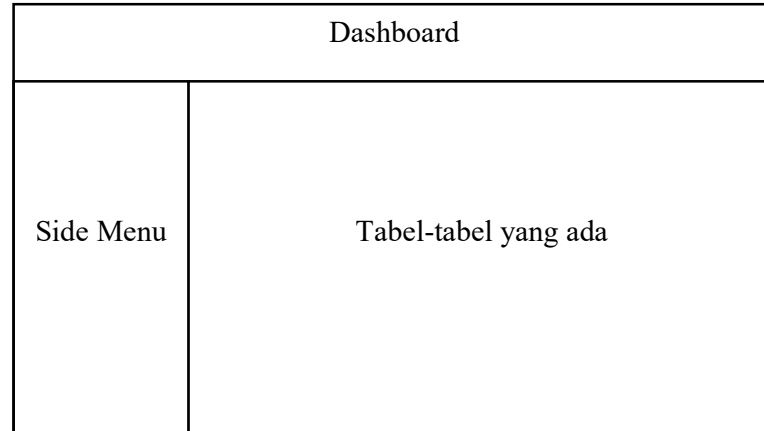
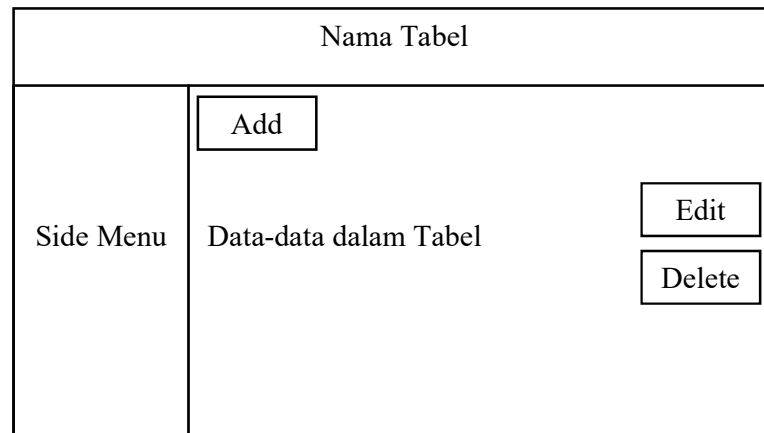
Gambar 3.12 Rancangan Layar *Chatbot*

3.5.2 Perancangan Layar *Web Admin*

a. Layar *Login*

The diagram shows a login form with a title "Login" centered at the top. Below the title, there are two input fields: "Username" and "Password". Below the "Password" field, there is a "Login" button.

Gambar 3.13 Rancangan Layar *Login*

b. Layar *Dashboard***Gambar 3.14 Rancangan Layar *Dashboard***c. Layar *View Data Tabel***Gambar 3.15 Rancangan *View Data Tabel***

d. Layar *Add/Edit* Data Tabel

Nama Tabel	
Side Menu	Col 1 <input type="text"/>
	Col 2 <input type="text"/>
	Col 4 <input type="text"/>
	<input type="button" value="Add/Edit"/>

Gambar 3.16 Rancangan Layar *Add/Edit* Data Tabel3.5.3 Perancangan Menu *Chatbot*


Bot STMIK PGRI Tangerang

Input pertama

Respon 1 *Bot*

Input kedua

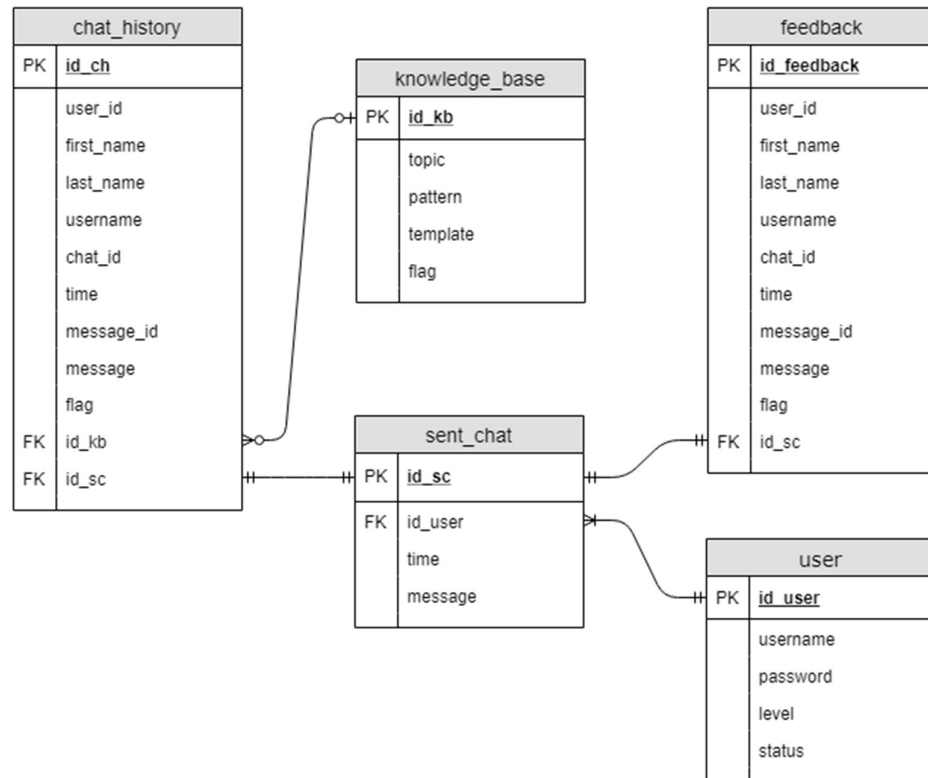
Respon 2 *Bot*

/menu 1
/menu 2
/menu 3
/ 

Gambar 3.17 Rancangan Menu *Chatbot*

3.5.4 Perancangan *Database*

Berikut ini adalah skema relasional *database* pada aplikasi yang akan dibuat.



Gambar 3.18 Skema Relasional *Database*

Terdapat juga satu buah tabel yang tidak memiliki relasi dengan tabel lain, yaitu tabel *command*. Sehingga terdapat total sebanyak 6 buah tabel yang akan digunakan pada perancangan aplikasi ini, antara lain tabel *command*, tabel *knowledge_base*, tabel *chat_history*, tabel *feedback*, tabel *user*, dan tabel *sent_chat*.

Untuk memperjelas perancangan *database* yang dibuat dalam perancangan aplikasi ini, maka dibuatlah spesifikasi *file* sebagai berikut:

a. Tabel *command*

Nama tabel : *command*

Isi : Data *command* / perintah *Bot*

Primary Key : *id*

No	Nama <i>Field</i>	Tipe	Panjang	Keterangan
1	<i>id</i>	<i>int</i>	11	<i>Primary Key</i>
2	<i>command</i>	<i>varchar</i>	20	<i>Command</i> pada <i>Bot</i>
3	<i>response</i>	<i>text</i>	-	Respon/jawaban
4	<i>flag</i>	<i>tinyint</i>	1	Tanda jenis <i>command</i>

b. Tabel *knowledge_base*

Nama Tabel : *knowledge_base*

Isi : Data pengetahuan *Bot*

Primary Key : *id_kb*

No	Nama <i>Field</i>	Tipe	Panjang	Keterangan
1	<i>id_kb</i>	<i>int</i>	11	<i>Primary Key</i>
2	<i>topic</i>	<i>varchar</i>	100	Topik/kategori pengetahuan
3	<i>pattern</i>	<i>pattern</i>	255	Pola masukkan pengguna
4	<i>template</i>	<i>text</i>	-	Respon/jawaban
5	<i>flag</i>	<i>tinyint</i>	1	Tanda jumlah kata pada <i>Field pattern</i>

c. Tabel *chat_history*

Nama Tabel : *chat_history*

Isi : Data *history* pesan yang diterima *Bot*

Primary Key : *id_ch*

No	Nama <i>Field</i>	Tipe	Panjang	Keterangan
1	id_ch	int	11	<i>Primary Key</i>
2	user_id	int	11	Id user <i>Telegram</i>
3	first_name	varchar	50	Nama depan
4	last_name	varchar	50	Nama belakang
5	username	varchar	50	<i>Username Telegram</i>
6	chat_id	int	11	Id <i>chat Telegram</i>
7	time	datetime	-	Tanggal dan waktu pesan
8	message_id	int	11	Id pesan <i>Telegram</i>
9	message	text	-	Isi pesan
10	flag	tinyint	1	Tanda pesan telah terjawab oleh <i>Bot</i> sesuai data pada <i>database</i> yang ada atau tidak
11	id_kb	int	11	<i>Foreign Key</i> , data pesan yang terkirim oleh <i>Bot</i> ke pengguna, diambil dari tabel <i>knowledge_base</i>
12	id_sc	int	11	<i>Foreign Key</i> , data pesan yang terkirim oleh admin ke pengguna, diambil dari tabel <i>sent_chat</i> .

d. Tabel *feedback*

Nama Tabel : feedback

Isi : Data pertanyaan/saran pengguna

Primary Key : id_feedback

No	Nama <i>Field</i>	Tipe	Panjang	Keterangan
1	id_feedback	int	11	<i>Primary Key</i>
2	user_id	int	11	Id user Telegram
3	first_name	varchar	50	Nama depan
4	last_name	varchar	50	Nama belakang
5	username	varchar	50	<i>Username Telegram</i>
6	chat_id	int	11	Id chat Telegram
7	time	datetime	-	Tanggal dan waktu pesan
8	message_id	int	11	Id pesan Telegram
9	message	text	-	Isi pesan
10	flag	tinyint	1	Tanda jenis pesan (pertanyaan atau saran)
11	id_sc	int	11	<i>Foreign Key</i> , data pesan yang terkirim oleh admin ke pengguna, diambil dari tabel <i>sent_chat</i> .

e. Tabel *sent_chat*

Nama Tabel : sent_chat

Isi : Data pesan yang dikirim oleh admin web

Primary Key : id_sc

No	Nama <i>Field</i>	Tipe	Panjang	Keterangan
1	id_sc	int	11	<i>Primary Key</i>
2	id_user	int	3	<i>Foreign Key</i> , id pengguna/admin dari tabel <i>user</i>
3	time	datetime	-	Tanggal dan waktu
4	message	text	-	Isi pesan

f. Tabel *user*

Nama Tabel : user

Isi : Data *login* pengguna *web admin*

Primary key : id_user

No	Nama <i>Field</i>	Tipe	Panjang	Keterangan
1	id_user	int	3	<i>Primary Key</i>
2	username	varchar	20	<i>Username</i>
3	password	varchar	255	<i>Bcrypt Password Hashing</i>
4	level	tinyint	1	Hak akses
5	status	tinyint	1	Status <i>user</i>