PENINGKATAN EFEKTIVITAS KUALITAS *CHATBOT* JALA TECH MELALUI IMPLEMENTASI CHAT GPT, AUTO-GPT, DAN LANGCHAIN

Disusun Oleh:

N am a : Rizan Qardafil
NIM : 20523244

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENINGKATAN EFEKTIVITAS KUALITAS *CHATBOT* JALA TECH MELALUI IMPLEMENTASI CHAT GPT, AUTO-GPT, DAN LANGCHAIN

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N am a : Rizan Qardafil

NIM : 20523244

Yogyakarta, 06 Oktober 2023

Pembimbing,

(Ari Sujarwo, S.Kom., M.IT)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENINGKATAN EFEKTIVITAS KUALITAS *CHATBOT* JALA TECH MELALUI IMPLEMENTASI CHAT GPT, AUTO-GPT, DAN LANGCHAIN

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan didepan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 16 Januari 2024

Tim Penguji

Ari Sujarwo, S.Kom, M.I.T.

Anggota 1

Andhika Giri Persada, S.Kom., M.Eng.

Anggota 2

Moh. Idris, S.Kom., M.Kom.

Mengetahui,
Ketua Program Studi Informatika– Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini:

Nama : Rizan Qardafil

NIM : 20523244

Tugas akhir dengan judul:

PENINGKATAN EFEKTIVITAS KUALITAS *CHATBOT* JALA TECH MELALUI IMPLEMENTASI CHAT GPT, AUTO-GPT, DAN LANGCHAIN

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya penulis sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 06 Oktober 2023

(Rizan Qardafil)

HALAMAN PERSEMBAHAN

Dengan rasa syukur yang sebesar-besarnya, penulis panjatkan puji syukur kehadirat Allah SWT atas nikmat-Nya yang telah membantu penulis berhasil menyelesaikan laporan akhir yang berjudul "PENINGKATAN EFEKTIVITAS KUALITAS CHATBOT JALA TECH MELALUI IMPLEMENTASI CHAT GPT, AUTO-GPT, DAN LANGCHAIN", terlepas dari segala keterbatasan yang ada. Penulis juga ingin menyampaikan rasa syukur penulis kepada Allah SWT yang telah memberikan penulis kesempatan untuk mendapat dukungan dan bantuan dari pihak-pihak yang selalu mendampingi penulis, memberi semangat dan mendoakan penulis agar tugas akhir ini dapat terselesaikan dengan baik dan tepat waktu. Penulis mendedikasikan laporan ini untuk:

1. Kedua Orang tua penulis yaitu Bapak Darmawan dan Ibu Rosmanidar.
2. Keluarga penulis lain juga telah memberikan dukungan kepada penulis selama ini.
3. Bapak Ari Sujarwo, S.Kom., MIT. selaku dosen pembimbing.
4. Program Studi Informatika FTI UII yang telah mewadahi penulis selama 3 tahun terakhir dalam menuntut ilmu dan mengembangkan diri.

Satu fase menuju dewasa akan segera dilewati satu dari sekian hari yang dinanti selama 3 tahun ini. Hari-hari ke depan akan penuh dengan petualangan baru. Semoga kelak yang telah diasah dapat menjadi bekal untuk menantang diri menjadi yang lebih baik di masa yang akan datang.

HALAMAN MOTO

"Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya."

- Al-Baqarah: 286

"Dan bersabarlah. Sesungguhnya Allah beserta orang-orang yang sabar."

- Al-Anfal: 46

"Bersabarlah kamu dan kuatkanlah kesabaranmu dan tetaplah bersiaga-siaga dan bertaqwalah kepada Allah supaya kamu menang."

- Ali-Imran: 200

"The only way to do great work is to love what you do."

- Steve Jobs

"Patience is a key element of success."

- Bill Gates

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah, Segala puji dan syukur penulis panjatkan kepada Allah SWT yang selalu melimpahkan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir ini. Serta serta shalawat kepada Nabi Muhammad SAW yang telah memberikan syafaat dan membawa islam dari zaman jahiliyah menuju zaman penuh pengetahuan yakni Islam, iman, dan ihsan.

Laporan ini disusun untuk memenuhi persyaratan kelulusan pada jalur magang di Fakultas Teknologi Industri Jurusan Informatika Universitas Islam Indonesia. Dalam penyusunan laporan ini, tidak lepas dari arahan dan bimbingan berbagai pihak. Penulis mengucapkan rasa hormat dan terima kasih kepada semua pihak yang telah membantu. Dengan segala hormat, penulis mengucapkan terima kasih kepada:

1. Puji syukur penulis panjatkan kepada Allah SWT yang telah memberikan kehidupan, keselamatan, kesehatan jasmani dan rohani, serta kesempatan untuk menyelesaikan praktek kerja magang ini dengan baik.
2. Penghargaan yang sebesar-besarnya penulis sampaikan kepada kedua orang tua penulis, yang telah memberikan dukungan, doa, dan bantuan yang sangat berharga, baik secara moril maupun materiil, selama penulis menjalani kerja magang.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Jurusan Informatika dan Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika - Program Sarjana.
4. Bapak Ari Sujarwo, S.Kom., MIT., selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk membimbing dan mengarahkan penulis sejak pelaksanaan kegiatan magang hingga selesai disusunnya laporan ini.
5. Farid Inawan, *principal engineer* Jala Tech, yang telah menerima penulis dengan baik selama masa kerja magang.
6. Kepada seseorang yang tak kalah penting kehadirannya. Rifa Husniyyah. Terima kasih telah menjadi bagian dari perjalanan hidup penulis. Berkontribusi banyak dalam penulisan tugas akhir ini, baik tenaga, waktu, maupun materi kepada penulis. Telah menjadi rumah, pendamping dalam segala hal yang menemani, mendukung ataupun menghibur dalam kesedihan, mendengar keluh kesah, memberi semangat pantang menyerah. Semoga Allah SWT selalu memberi keberkahan dalam segala hal yang kita lalui.

7. Seluruh karyawan Jala Tech yang berdedikasi yang telah menyambut penulis dengan hangat, memberikan bimbingan, dan memberikan bantuan selama penulis melakukan kerja magang di perusahaan tersebut.
8. Seluruh keluarga besar Program Studi Teknik Informatika Universitas Islam Indonesia atas dukungan dan bantuan yang diberikan selama masa perkuliahan, kerja magang, dan penyelesaian tugas akhir ini.
9. Terima kasih juga penulis ucapkan kepada sahabat-sahabat tercinta yang tidak hanya menjadi sahabat tetapi juga sumber ilmu, cerita, dan pengalaman yang sangat berharga selama kita bersama di perkuliahan dan di luar perkuliahan.
10. Azizi Shafa Asadel, Aldean Tegar, Natasia Adeline, Aninditha Cahyadi, JKT48 dan Bala-Bala Esports lainnya atas semua karya yang memberikan semangat serta menjadi *mood booster* bagi penulis selama penyusunan laporan akhir ini.

Magang di Jala Tech telah membuat penulis mengenal banyak konsep dan teknologi baru, terutama di bidang teknologi dan Pemrosesan Bahasa Alami. Pengalaman ini diperoleh dengan berpartisipasi secara aktif dalam industri teknologi informasi, bekerja bersama tim profesional berpengalaman yang memberikan bimbingan dalam memahami berbagai konsep dan mengembangkan kode untuk mengatasi tantangan dunia nyata.

Sebagai penutup, penulis menyadari bahwa laporan ini mungkin tidak sempurna. Oleh karena itu, kritik dan saran yang membangun dari para pembaca sangat diharapkan untuk perbaikan di masa mendatang. Penulis berharap laporan ini dapat memberikan kontribusi pada bidang ilmu pengetahuan dan teknologi yang lebih luas, mendorong kemajuan dalam pengetahuan dan praktik.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 06 Oktober 2023

(Rizan Qardafil)

SARI

Jala Tech merupakan perusahaan teknologi yang mengkhususkan diri dalam membantu petani udang meningkatkan kualitas tambak mereka melalui penggunaan aplikasi berbasis web. Salah satu fitur utama dalam aplikasi Jala adalah Jala *Chatbot*. Awalnya, *chatbot* ini diintegrasikan dengan menggunakan Chat GPT. Namun, dengan perkembangan teknologi yang sangat cepat, muncul inovasi baru dalam dunia Chat GPT, seperti *Function Call*, Auto-GPT, dan LangChain. Oleh karena itu, perlu dilakukan integrasi ulang pada fitur *chatbot* untuk menggantikan model integrasi yang awalnya hanya menggunakan Chat GPT dengan model yang lebih canggih, yaitu Chat GPT, Auto-GPT, dan LangChain. Hasil dari penelitian ini menunjukkan peningkatan yang sangat baik dalam kualitas informasi yang dihasilkan oleh fitur *chatbot* yang telah menerapkan integrasi dengan Chat GPT, Auto-GPT, dan LangChain. Dengan menggunakan Chat GPT, Auto-GPT, dan LangChain, integrasi dengan Chat GPT memungkinkan *chatbot* untuk terhubung dengan API dari layanan Jala lainnya. Selain itu, integrasi dengan Auto-GPT memberikan kemampuan *chatbot* untuk mengambil data eksternal secara berkelanjutan dan melakukan pelatihan mandiri. Di sisi lain, integrasi dengan LangChain dapat meningkatkan efisiensi penarikan data dari *database*. Oleh karena itu, dari sebelumnya hanya memiliki kemampuan untuk memberikan jawaban kepada pengguna, sekarang *chatbot* dapat melakukan pemrograman sendiri, menciptakan tambak, dan secara otomatis mengisi informasi terkait yang diperoleh dari perangkat bantu manajemen tambak serta informasi yang diberikan oleh pengguna.

Kata kunci: *Chatbot*, Chat GPT, Auto-GPT, LangChain.

GLOSARIUM

API	Perantara komunikasi antara dua aplikasi utama.
Backend	Pengembang aplikasi yang fokus pada server dan database.
Backlog	Daftar tugas atau spesifikasi kebutuhan sistem.
Chatbot	Akun otomatis yang menerima input pengguna dan memberikan respons yang sesuai.
Debugging	Proses mengidentifikasi dan memperbaiki kesalahan dalam pemrograman.
Design Pattern	Struktur arsitektur yang digunakan untuk menyelesaikan masalah yang sering muncul.
Invoke	Melakukan panggilan atau pemanggilan terhadap suatu fungsi.
Request	Permintaan dalam bentuk protokol HTTP pada API yang menghasilkan respons tertentu.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

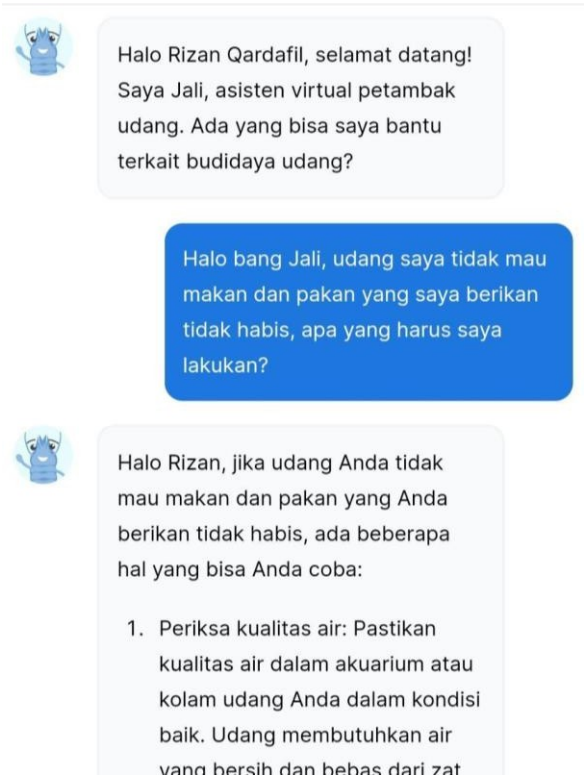
Era revolusi industri 4.0, yang ditandai oleh automasi dan pertukaran data digital, telah memiliki dampak yang signifikan pada berbagai sektor, termasuk dalam pertanian seperti budidaya udang. Revolusi ini melibatkan berbagai tren teknologi seperti automasi, pertukaran data, komputasi awan, *internet of things*, dan *artificial intelligence*, dengan tujuan untuk mempermudah pekerjaan manusia. Teknologi informasi menjadi kunci utama bagi perusahaan dalam meningkatkan efisiensi proses bisnis (Frestilia, 2013).

Jala Tech adalah sebuah perusahaan yang memiliki fokus di industri akuakultur, terutama dalam sektor budidaya udang. Perusahaan ini memiliki tekad yang sangat kuat untuk memberikan dukungan kepada petani udang dengan tujuan meningkatkan produktivitas dan menciptakan budidaya udang yang berkelanjutan melalui penerapan teknologi tinggi. Produk utama yang dihasilkan oleh Jala Tech, yang dikenal dengan nama Jala Baruno, telah dirancang untuk mengukur berbagai parameter penting dalam kolam udang. Selain itu, Jala Tech juga mengembangkan produk digital seperti aplikasi berbasis web dan *mobile* yang berperan dalam pemantauan dan pengelolaan data tambak dan kolam udang secara *online* (Jala Tech, 2023).

Dalam menghadapi perkembangan teknologi yang sangat cepat, Jala Tech berusaha untuk menyesuaikan diri dengan kemajuan tersebut dalam upaya meningkatkan efektivitas *chatbot*. Hal ini menjadi sangat penting untuk mempermudah pengguna dalam mengakses informasi, membantu dalam pengelolaan dan pencatatan data pada aplikasi Jala. Melalui implementasi teknologi Auto-GPT, Chat GPT, dan LangChain, telah terjadi peningkatan signifikan dalam produktivitas *chatbot* yang dapat melakukan tugas secara mandiri. Ini memungkinkan pengguna untuk mendapatkan informasi dengan akurasi yang tinggi, meningkatkan efisiensi mekanisme *chatbot*, serta membuat sinkronisasi antara berbagai layanan dalam aplikasi menjadi lebih mudah.

Pada awalnya, Jala Tech memanfaatkan implementasi *chatbot* dengan menggunakan Chat GPT sebagai solusi untuk memberikan informasi kepada pengguna. Sistem *chatbot* ini bertujuan untuk memberikan layanan interaktif dan responsif dalam menyediakan informasi yang dibutuhkan oleh pengguna, sehingga interaksi pengguna menjadi lebih efisien dan efektif. Seiring dengan perkembangan teknologi dan tuntutan pasar, Jala Tech semakin peka terhadap potensi-potensi yang dapat mengoptimalkan kinerja *chatbot*. Menyadari perlunya peningkatan

dalam hal fungsionalitas, Jala Tech kemudian mengambil langkah proaktif dengan berinisiatif untuk mengembangkan *chatbot* dengan pendekatan yang lebih holistik dan terintegrasi. Tidak hanya sekedar mengandalkan satu teknologi, tetapi dengan mengimplementasikan tiga teknologi berbeda secara simultan. Hal ini bertujuan untuk memberikan dimensi bar dalam efektivitas dan kinerja *chatbot* Jala Tech, serta menjawab tantangan tuntutan pengguna yang semakin kompleks dan beragam.



Gambar 1.1 Model percakapan interaktif *chatbot* Jala

Penggunaan model percakapan interaktif dalam pembuatan *chatbot* menciptakan antarmuka yang lebih mendekati cara manusia berkomunikasi. Gambar 1.1 menunjukkan interaksi *chatbot* yang berbasis tanya-jawab, yang memberikan pengalaman berkomunikasi yang lebih alami dan intuitif bagi pengguna. *Natural Language Processing* (NLP) memainkan peran penting dalam meningkatkan kemampuan *chatbot* untuk memahami bahasa manusia, sehingga *chatbot* dapat memberikan jawaban yang relevan dan solusi teknologi yang canggih dan efisien bagi pengelola tambak udang.

Laporan ini mencakup semua kegiatan yang terjadi selama masa magang di Jala Tech dan merupakan bagian dari laporan akhir untuk menyelesaikan program magang pada semester pertama tahun ajaran 2023/2024.

1.2 Ruang Lingkup Magang

Pelaksanaan magang berlangsung di Jala Tech selama tiga bulan, mulai dari Juni 2023 hingga September 2023. Jala Tech merupakan sebuah perusahaan teknologi yang berfokus pada memberdayakan petani udang dengan teknologi *Internet of Things* (IoT) dan sistem manajemen pertanian berbasis data. Perusahaan ini menyediakan *website* dan *mobile* sebagai solusi komprehensif bagi para petani tambak udang.

Selama magang, penulis terlibat dalam berbagai aktivitas yang mendukung pengembangan dan implementasi teknologi di Jala Tech. Beberapa aktivitas tersebut antara lain:

a. Mempelajari tentang Auto-GPT, Chat GPT, LangChain, dan Git dengan merujuk kepada dokumentasi yang disediakan oleh OpenAI di bawah bimbingan dari *principal engineer* Jala Tech.
b. Melakukan pengembangan *chatbot*, yang merupakan salah satu fitur dari aplikasi Jala.
c. Membuat model *chatbot* dengan memanfaatkan teknologi Auto-GPT, Chat GPT, dan LangChain.
d. Membuat API yang terhubung dengan layanan Jala Tech, memungkinkan integrasi dengan berbagai fitur dan layanan yang ada.
e. Mengimplementasikan fitur agar model dapat melakukan pelatihan dan penelitian mandiri, memperluas kemampuan *chatbot* dalam memahami dan merespons pengguna dengan lebih baik.
f. Mengimplementasikan model *chatbot* dengan mengintegrasikannya dengan *database server*, untuk menyimpan dan mengakses data dengan efisien.

1.3 Tujuan

Tujuan dari implementasi Auto-GPT, Chat GPT, dan LangChain pada *chatbot* Jala untuk peningkatan efektivitas *chatbot* adalah sebagai berikut:

a. Membuat *chatbot* yang responsif dan efisien.
b. Membuat *chatbot* dapat melakukan training mandiri dengan sumber daya eksternal.
c. Membuat *chatbot* terintegrasi dengan layanan Jala lainnya.
d. Membuat *chatbot* dapat mengeksekusi perintah.
e. Meningkatkan kualitas tanggapan *chatbot* terhadap berbagai pertanyaan dan permintaan pengguna.

1.4 Manfaat

Manfaat dari implementasi Chat GPT, Auto-GPT, dan LangChain pada pengembangan *chatbot* aplikasi Jala dengan menerapkan model adalah sebagai berikut:

a. Kemampuan Eksekusi Perintah: Dengan model Chat GPT, *chatbot* dapat secara efisien mengeksekusi perintah yang diberikan oleh pengguna, memudahkan akses informasi dan layanan yang diinginkan.
b. Integrasi dengan Layanan Jala Lainnya: Penerapan Chat GPT memungkinkan *chatbot* terintegrasi dengan layanan Jala lainnya, memberikan pengalaman pengguna yang holistik dan komprehensif.
c. Training Mandiri: *chatbot* dengan model Auto-GPT memiliki kemampuan untuk melakukan training mandiri, sehingga terus meningkatkan kualitas tanggapan dan layanannya.
d. Layanan Responsif dan Efisien: Model Auto-GPT memungkinkan *chatbot* memberikan tanggapan yang relevan dan kontekstual, memberikan pengalaman pengguna yang lebih baik.
e. Mengoptimalkan Proses: Dengan penerapan LangChain, *chatbot* dapat mengoptimalkan proses dalam mencatat data tambak udang, meningkatkan efisiensi dan efektivitas pengelolaan data.

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan laporan akhir ini adalah sebagai berikut:

a. BAB I – PENDAHULUAN

   Bab ini akan membahas latar belakang aplikasi Jala, terutama dalam konteks masalah terkini yang mengenai fitur *chatbot*. Peningkatan yang diperlukan adalah dalam kemampuan *chatbot* dalam mengeksekusi perintah pengguna dan kemampuannya untuk berinteraksi secara alami. Selain itu, bab ini juga akan mencakup ruang lingkup magang, tujuan, manfaat, dan sistematika penulisan.

b. BAB II – DASAR TEORI

   Dalam bab ini, akan dibahas teori-teori yang menjadi dasar dari penulisan laporan. Teori yang akan dibahas mencakup aplikasi Jala *chatbot*, Auto-GPT, Chat GPT, dan LangChain.

c. BAB III – PELAKSANAAN MAGANG

Dalam bab ini akan diberikan penjelasan yang mendalam mengenai pengalaman magang yang telah dijalani di Jala Tech. Pembahasan akan mencakup berbagai aspek, termasuk penerapan metodologi *scrum* dalam pengembangan fitur *chatbot* dengan tujuan untuk meningkatkan efektivitas. Selain itu, akan dibahas juga bagaimana teknologi Chat-GPT, Auto-GPT, dan LangChain dimanfaatkan dalam pengembangan ini.

d. BAB IV – REFLEKSI PELAKSANAAN MAGANG

Dalam bab ini akan diberikan refleksi dari pengalaman magang penulis yang berlangsung selama tiga bulan di Jala Tech.

e. BAB V – KESIMPULAN DAN SARAN

Dalam bab ini mencakup rangkuman mengenai peningkatan efektivitas *chatbot* melalui pemanfaatan Auto-GPT, Chat GPT, dan LangChain dalam fitur *chatbot* aplikasi Jala. Selain itu, dalam bab ini juga akan memberikan rekomendasi terkait pengembangan aplikasi Jala di masa depan.

f. DAFTAR PUSTAKA

Bagian ini akan mencantumkan sumber-sumber referensi yang digunakan dalam penulisan laporan.

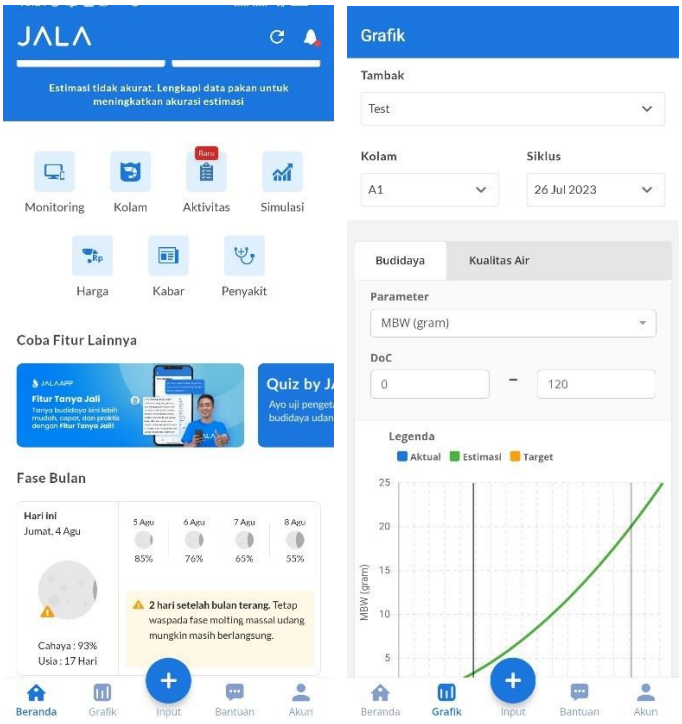
g. LAMPIRAN

Lampiran memuat berbagai gambar pendukung yang berkaitan dengan pengalaman magang di Jala Tech.

BAB II
KAJIAN LITERATUR

2.1 Aplikasi Jala

Aplikasi Jala merupakan aplikasi *mobile* yang khusus dikembangkan untuk membantu petambak udang meningkatkan produktivitas dan mengelola tambak udang (Jala Tech, 2023). Aplikasi ini terintegrasi dengan perangkat khusus yang digunakan untuk mengukur kualitas air, sehingga petambak udang dapat memantau langsung kondisi air di tambaknya melalui perangkat seluler. Dengan fitur ini, pengelolaan peternakan dapat dilakukan lebih efisien dan data *real-time* memungkinkan pengambilan keputusan lebih akurat. Untuk melihat antarmuka visual aplikasi Jala, lihat Gambar 2.1.



Gambar 2.1 Tampilan dashboard aplikasi Jala

Di samping perannya utama dalam pemantauan kualitas air di tambak, aplikasi Jala juga dilengkapi dengan berbagai fitur tambahan. Ini termasuk fitur manajemen keuangan, manajemen stok, perdagangan udang, akses ke informasi harga udang, portal berita seputar udang, dan buku digital mengenai penyakit udang. Namun, laporan ini akan menitikberatkan pada salah satu fitur khusus, yaitu fitur bantuan atau *chatbot*.

Aplikasi Jala menggunakan teknologi React Native dalam pengembangan *frontend* (Jala Tech, 2023). Sebelum dimulainya magang, integrasi model di aplikasi Jala, terutama pada fitur

chatbot telah menggunakan Chat GPT. Integrasi dalam *chatbot* ini melibatkan Chat GPT, Auto-GPT, dan LangChain melalui proses yang disebut "*Function Call*".

2.2 Chatbot

Sebuah program perangkat lunak yang dikenal sebagai *chatbot* telah dirancang untuk berinteraksi dengan manusia dan menciptakan ilusi memiliki kecerdasan buatan (Shawar & Atwell, 2004). *Chatbot* merupakan hasil kerjasama berbagai disiplin ilmu seperti *natural language processing*, *machine learning*, rekayasa perangkat lunak, dan *artificial intelligence*. *Chatbot* diciptakan dengan tujuan meniru percakapan manusia, baik melalui metode berbasis aturan atau kecerdasan buatan, serta memiliki kemampuan berkomunikasi melalui teks tertulis atau bicara. *Chatbot* mampu memanfaatkan pembelajaran mesin dan kecerdasan buatan untuk mengenali pola percakapan, sehingga dapat memberikan respons yang sesuai terhadap pertanyaan tertulis atau lisan, serta memberikan layanan atau informasi yang dibutuhkan. Selain itu, *chatbot* dapat terhubung dengan berbagai sumber data yang tersedia untuk memberikan informasi atau layanan sesuai dengan permintaan pengguna, seperti memberikan prakiraan cuaca, berita terbaru, atau membantu dalam proses pemesanan kamar hotel. *Natural Language Processing* juga dilengkapi dengan pengetahuan kontekstual untuk memberikan jawaban yang lebih mendalam terhadap pertanyaan (A. Iswandi, 2018).

Chatbot terdiri dari tiga komponen mendasar. Pertama, ini mencakup antarmuka yang memungkinkan *chatbot* berkomunikasi dengan pengguna. Kedua, menggabungkan kecerdasan buatan untuk meningkatkan kemampuannya memahami dan belajar dari masukan pengguna dari waktu ke waktu. Terakhir, ini melibatkan sistem informasi yang memfasilitasi integrasi antara *chatbot* dan aplikasi lain.

Chatbot dapat diklasifikasikan ke dalam dua kategori utama: *chatbot* berbasis aliran dan *chatbot* yang digerakkan oleh AI. Sederhananya, perbedaannya terletak pada bagaimana *chatbot* ini mengelola permintaan pengguna. *Chatbot* berbasis alur memiliki kemampuan untuk menangani berbagai macam permintaan dengan memanfaatkan NLP (*Natural Language Processing*) dani NLU (*Natural Language Understanding*) untuk memahami konteks dan bahasa. Hal ini memungkinkan mereka untuk memproses dan menanggapi permintaan seolah-olah ada percakapan alami yang terjadi antara pengguna dan *chatbot* (Lavena, 2019).

Saat membangun *chatbot*, elemen penting yang harus diperhatikan adalah konsep "*Flow*". *Flow* merupakan urutan langkah terstruktur yang diikuti oleh *chatbot* dalam mengelola

percakapan. Konsep *flow* ini mencakup tiga komponen penting: *intent*, *state*, dan *action* (Hadiwijaya, 2018).

a. Intent

Intent berfungsi sebagai kamus yang memungkinkan *chatbot* untuk mengidentifikasi dan memahami tujuan atau makna di balik pesan yang dimasukkan.

b. State

State berfungsi sebagai tempat penyimpanan bagi *chatbot* untuk menyimpan dan menunjukkan tahapan yang telah diselesaikannya selama percakapan.

c. Action

Action digunakan untuk mengkonfigurasi respons yang akan dihasilkan dan ditampilkan oleh *chatbot*.

Selain komponen inti, yaitu *flow*, *chatbot* dibangun menggunakan tiga kombinasi penting yang bekerja sama untuk menciptakan *chatbot* (Guzman & Ines, 2016), kombinasi tersebut yakni:

a. User Interface

User Interface atau antarmuka pengguna *chatbot* ini bertindak sebagai jembatan antara *chatbot* dan pengguna, memfasilitasi interaksi mereka melalui aplikasi perpesanan. Sangat penting bahwa antarmuka pengguna meningkatkan pengalaman pengguna secara keseluruhan saat berinteraksi dengan *chatbot*.

b. *Artificial intelligence* (Kecerdasan Buatan)

Kecerdasan Buatan (AI), yang sering disebut sebagai "artifisial", menjadi dasar pengembangan *chatbot*. AI memberdayakan *chatbot* untuk memahami dan menafsirkan interaksi pengguna, sehingga memungkinkan mereka untuk mengatasi masalah berdasarkan aturan yang telah ditetapkan sebelumnya.

c. Integrasi

Integrasi memainkan peran penting dalam fungsionalitas *chatbot*. Sangat penting untuk meningkatkan kemampuan *chatbot* dengan memasukkan fitur tambahan. Selain itu, mengintegrasikan *chatbot* dengan aplikasi lain dapat menawarkan informasi dan data tambahan.

Dalam ranah kebutuhan bisnis, *chatbot* memainkan peran penting dalam mengatasi tantangan komunikasi dengan pelanggan, sehingga meningkatkan kualitas layanan dan pengalaman komunikasi. *Chatbot* terbukti sangat efisien ketika menangani masalah yang tepat, spesifik, dan dapat diprediksi (Guzman & Ines, 2016). Dengan pesatnya perkembangan dunia

bisnis, *chatbot* menawarkan solusi alternatif yang berharga yang menyederhanakan akses dan interaksi pelanggan dengan bisnis.

2.3 Artificial Intelligence

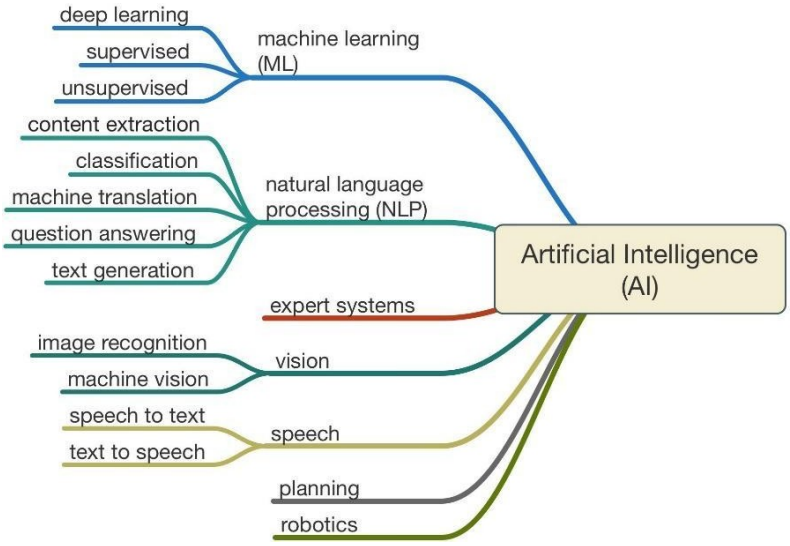
Artificial intelligence (AI) adalah sistem yang diciptakan untuk mengeksekusi tugas-tugas yang menyerupai kemampuan manusia, termasuk proses-proses seperti pemikiran manusia, pembelajaran, pengambilan keputusan, dan pemecahan masalah (Sihombing & Wirapraja, 2018). Menurut definisi dari (Rich, 2019), AI melibatkan desain komputer untuk melaksanakan fungsi-fungsi yang saat ini dijalankan oleh manusia. Ketika digunakan untuk tujuan yang positif, AI memiliki potensi untuk mempermudah berbagai aktivitas manusia. Sesuai dengan pandangan (Kamble & Shah, 2018), cakupan kecerdasan buatan mencakup pemahaman bahasa, sistem pembelajaran adaptif, penyelesaian masalah, persepsi, pemodelan, robotika, dan permainan.

Salah satu tantangan mendasar dalam bidang kecerdasan buatan (AI) adalah mengajarkan komputer untuk menunjukkan serangkaian karakteristik khusus. Ini mencakup memberikan komputer kemampuan untuk memperoleh pengetahuan, terlibat dalam proses kognitif, mengatasi masalah, memahami, belajar, merencanakan, serta kemampuan untuk mengelola dan memanipulasi objek. Bidang AI terus berkembang dengan dinamis dalam ranah Ilmu Komputer. Hingga saat ini, banyak penelitian yang fokus pada perkembangan AI, yang mencakup topik-topik seperti jaringan saraf (*neural network*), komputasi evolusioner (*evolutionary computing*), pembelajaran mesin (*machine learning*), pemrosesan bahasa alami (*natural language processing*), pemrograman otomatis, robotika, dan pemrograman berorientasi objek (*object-oriented programming*) (Yunanto & Herumurti, 2016).

Menurut (Russel & Norvig, 2010) mereka mengkategorikan *artificial intelligence* ke dalam empat dimensi yang berbeda. Dimensi pertama, "*thinking humanly*," berkaitan dengan cara teknologi AI meniru proses berpikir manusia. Dimensi kedua, "*acting humanly*," melibatkan kapasitas AI untuk melakukan tugas-tugas dengan cara seperti manusia. Dimensi ketiga, "*thinking rationally*," menandakan pendekatan pemecahan masalah logis yang digunakan oleh AI. Terakhir, dimensi keempat, "*action rationally*," mengacu pada penggunaan metode sistematis dalam sistem AI untuk mencapai tujuan tertentu.

Adanya *artificial intelligence* mempunyai peran penting dalam menyederhanakan upaya pemecahan masalah manusia. Menurut Dahria (2008), manfaat dari *artificial intelligence* antara lain:

a. Kecerdasan buatan tetap konstan, asalkan sistem dan programnya tidak berubah, menawarkan stabilitas yang bertahan lama.
b. Lebih mudah didistribusikan karena pengetahuan yang ada di dalam sistem komputer tetap dapat digunakan meskipun berpindah ke sistem komputer yang berbeda.
c. Terbukti lebih hemat biaya karena kecerdasan buatan secara konsisten melakukan tugas-tugas secara efisien dalam waktu yang lama.
d. Dokumentasi dapat dilakukan karena penggunaan komputer dapat memonitor setiap aktivitas sistem dengan cermat.



Gambar 2.2 Bagian-bagian pada *artificial intelligence*

Sumber: https://sis.binus.ac.id/2020/12/03/penjelasan-singkat-tentang-ai-2/ (Di akses 04 Agustus 2023)

2.4 Natural Language Processing

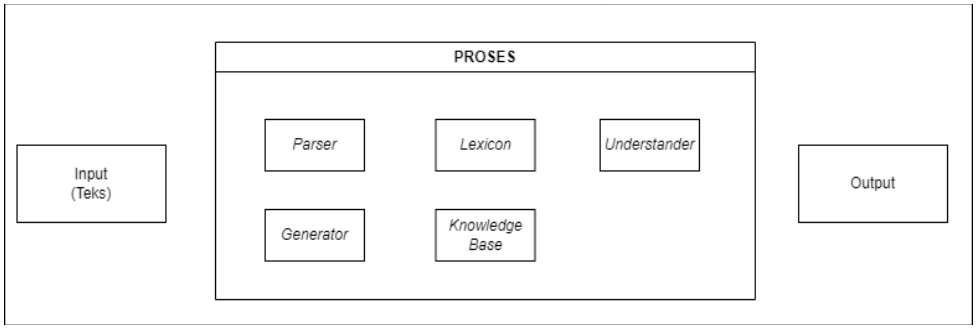
Natural Language Processing (NLP) adalah metode dalam pengembangan perangkat lunak yang memungkinkan komputer untuk memahami bahasa manusia dan memfasilitasi interaksi antara manusia dengan mesin (I. Iswandi et al., 2013). Fokus utama dari NLP adalah memberikan respons yang tepat berdasarkan pemahaman terhadap makna yang terkandung dalam bahasa manusia (Alamanda et al., 2016). NLP telah diterapkan dalam berbagai aspek kehidupan manusia karena kemampuannya sebagai antarmuka yang mudah digunakan, sehingga pengguna tidak perlu memiliki pengetahuan mendalam mengenai perintah-perintah yang kompleks dalam komputer.

Saat menangani masalah, NLP menggunakan dua teknik utama untuk analisis dan interpretasi input, diikuti dengan tindakan yang tepat. Teknik-teknik ini meliputi pelacakan

kata kunci dan analisis sintaksis dan semantik (Suparman, 1991). Analisis kata kunci melibatkan identifikasi frasa atau kata kunci tertentu dalam kalimat masukan, yang memungkinkan program untuk menghasilkan respons yang telah ditentukan ketika kata kunci ini terdeteksi. Di sisi lain, analisis sintaksis dan semantik menangani tantangan dalam menangani berbagai variasi bahasa. Teknik ini mengambil pendekatan langsung dengan memahami struktur dan makna yang tepat dari kalimat masukan.

Dalam beberapa tahun terakhir, kemajuan teknologi NLP sangat pesat, ditandai dengan munculnya teknologi NLP yang canggih seperti Auto-GPT, Chat GPT, dan LangChain. Auto-GPT adalah sistem kecerdasan buatan (AI) otonom yang bersifat *open-source* yang dilengkapi dengan akses internet, manajemen memori yang efisien untuk tugas-tugas jangka pendek dan jangka panjang, dan kemampuan untuk terlibat dalam interaksi berbasis teks dengan GPT-4 dan GPT-3.5. Di sisi lain, LangChain berfungsi sebagai *framework* yang dirancang untuk mengembangkan aplikasi yang menggunakan *large language models* (LLM). Tujuan utamanya adalah untuk menyederhanakan proses bagi para pengembang untuk mengintegrasikan sumber data lain dan berinteraksi dengan berbagai aplikasi (Russakovsky et al., 2015). LangChain unggul dalam berinteraksi secara efektif dengan beragam sumber data dan aplikasi, menawarkan komponen yang dapat disesuaikan dan rantai yang disesuaikan untuk kasus penggunaan tertentu, sehingga mempercepat pengembangan LLM. Terakhir, Chat GPT mewakili model bahasa yang memfasilitasi interaksi yang lebih alami dan percakapan antara pengguna dan komputer. GPT yang merupakan singkatan dari "*Generative Pre-trained Transformer*", adalah model bahasa alami berbasis *artificial intelligence* yang terkenal untuk menghasilkan konten original.

Natural language processing (NLP) tidak berfokus pada pengubahan suara menjadi data digital atau sebaliknya, tujuan utamanya adalah untuk memahami makna yang disampaikan dalam bahasa alami yang diucapkan dan menawarkan respons yang sesuai, yang mungkin melibatkan pelaksanaan tindakan tertentu atau menyajikan informasi tertentu. Tujuan ini melibatkan prosedur tiga tahap. Tahap kedua, yang dikenal sebagai interpretasi semantik, berusaha untuk menangkap makna dengan cara yang netral terhadap *context-independent* untuk keperluan lebih lanjut. Selanjutnya, tahap ketiga, interpretasi kontekstual, bertujuan untuk menangkap makna dalam *context dependent* dan melihat tujuan di balik penggunaan kalimat tersebut.



Gambar 2.3 Elemen *chatbot*

Pada Gambar 2.3, komponen-komponen yang terlibat dalam *natural language processing* (Pengolahan Bahasa Alami) dijelaskan sebagai berikut: *parser*, *lexicon*, *understander*, *knowledge base*, dan *generator*. *Parser* bertanggung jawab untuk mengenali kata-kata individual, sedangkan *lexicon* terdiri dari repositori program untuk kata-kata yang dikenali. *Understander* memainkan peran penting dalam memastikan arti dari kalimat yang diberikan. *Knowledge base* berfungsi sebagai tempat penyimpanan kata dan frasa. Terakhir, *generator* menghasilkan output berdasarkan input yang diproses (Lisangan, 2013).

2.5 Large Language Models

Language models adalah model komputasi yang memiliki kemampuan untuk memahami dan menghasilkan bahasa manusia. LM memiliki kemampuan transformatif untuk memprediksi kemungkinan urutan kata atau menghasilkan teks baru berdasarkan input yang diberikan (Devlin et al., 2018). Model N-gram adalah jenis LM yang paling umum, memperkirakan probabilitas kata berdasarkan konteks sebelumnya. Namun, LM juga menghadapi tantangan, seperti masalah kata-kata yang jarang atau tidak terlihat, masalah *overfitting*, dan kesulitan dalam menangkap fenomena linguistik yang kompleks. Para peneliti terus berupaya meningkatkan arsitektur LM dan metode pelatihan untuk mengatasi tantangan-tanganan ini (Brown et al., 1992).

Large Language Models (LMS) (Chen et al., 2021) adalah model bahasa tingkat lanjut dengan ukuran parameter yang sangat besar dan kemampuan pembelajaran yang luar biasa. Modul inti di balik banyak LLM, seperti GPT-3 (Floridi & Chiriatti, 2020), InstructGPT (Ouyang et al., 2022), dan GPT-4 (OpenAI, 2023) adalah modul *self-attention* di *transformer* yang berfungsi sebagai blok bangunan fundamental untuk tugas pemodelan bahasa. *Transformer* telah merevolusi bidang NLP dengan kemampuannya menangani data berurutan dengan efisien memungkinkan paralelisasi dan menangkap ketergantungan jarak jauh dalam

teks. Salah satu fitur LLM adalah pembelajaran dalam konteks (*context-learning*). Model dilatih untuk menghasilkan teks berdasarkan konteks atau perintah yang diberikan. Hal ini memungkinkan LLM untuk menghasilkan respons yang lebih koheren dan relevan secara kontekstual, sehingga cocok untuk aplikasi interaktif dan percakapan. *Reinforcement Learning from Human Feedback* (RLHF) (Christiano et al., 2017) adalah aspek penting lainnya dari LLM. Teknik ini melibatkan penyempurnaan model dengan menggunakan respons yang dihasilkan manusia sebagai *rewards*, memungkinkan model untuk belajar dari kesalahannya dan meningkatkan kinerjanya dari waktu ke waktu.

Dalam *language models autoregresif*, seperti GPT-3 dan PaLM (Chowdhery et al., 2022), diberikan sebuah urutan konteks X, tugas LM bertujuan untuk memprediksi token y berikut. $P(y|X) = P(X_1, X_2, \dots, X_{t-1})$, $x_1, x_2, \dots, x_{t-1}$ dengan menggunakan aturan rantai, probabilitas bersyarat dapat dikomposisikan menjadi produk dari probabilitas di setiap posisi:

$$P(y|X) = \prod_{t=1}^{T} P(y_t|x_1, x_2, \dots, x_{t-1}),$$

Dengan cara ini, model ini memprediksi setiap posisi yang diambil dalam model *autoregresif*, menghasilkan urutan teks yang lengkap.

Tabel 2.1 Perbandingan antara *Traditional* ML, *Deep Learning,* dan LLM

No.	*Comparison*	Traditional ML	DL	LLMS
1.	Training Data Size	Large	Large	Very Large
2.	Feature Engineering	Manual	Automatic	Automatic
3.	Model Complexity	Limited	Complex	Very Complex
4.	Interpretability	Good	Poor	Poorer
5.	Performance	Moderate	High	Highest
6.	Interpretability	Low	High	Very High

Salah satu pendekatan umum untuk berinteraksi dengan LLM adalah *prompt engineering* (Clavié et al., 2023) dimana pengguna mendesain dan menyediakan teks *prompt* khusus untuk memandu LLM dalam menghasilkan respons yang diinginkan atau menyelesaikan tugas tertentu. Hal ini diadopsi secara luas dalam upaya evaluasi yang ada. Orang-orang juga dapat terlibat dalam interaksi tanya-jawab (Jansson et al., 2021), Mereka mengajukan pertanyaan

kepada model dan menerima jawaban, atau terlibat dalam interaksi dialog, melakukan percakapan bahasa alami dengan LLM. Kesimpulannya, LLM dengan arsitektur *transformer*, *supervised learning*, dan kemampuan RLHF, telah merevolusi NLP dan menjanjikan dalam berbagai aplikasi. Tabel 2.1 memberikan perbandingan singkat tentang ML tradisional, *deep learning*, dan LLM.

2.6 Prompt Engineering

Prompt engineering adalah proses merancang, mengoptimalkan, atau memilih *prompt,* teks *prompt* yang akan digunakan dalam interaksi dengan model bahasa yang telah dilatih (P. Liu et al., 2021). Design *prompt* melibatkan pemilihan kata-kata, struktur kalimat, pengaturan kalimat, dan pengaturan konteks yang tepat untuk merangsang respons yang diinginkan dari model. Terbukti bahwa penggunaan prompt yang disusun dengan baik memiliki potensi besar untuk meningkatkan kinerja *large language models* (LLM) di berbagai sektor.

Metode pembelajaran berdasarkan *prompt engineering* telah menjadi pendekatan yang umum dalam pengembangan *language models* (LM). Sebagai alternatif dari cara sebelumnya yang menggunakan bahasa pra-pelatihan untuk menyelesaikan tugas tertentu melalui pendekatan teknik yang rumit, metode pembelajaran berbasis *prompt engineering* mengubah cara dalam pendekatan tugas tersebut dengan menggunakan *prompt engineering*. Dengan melakukan ini, tugas yang harus diselesaikan menjadi lebih mirip dengan tugas yang model bahasa dipelajari selama pelatihan awal (Haque et al., 2022).

Tujuan dari *prompt engineering* adalah untuk mencapai hasil maksimal dalam respons model dengan menghasilkan jawaban yang relevan, konsisten, dan bermanfaat. Proses ini melibatkan serangkaian eksperimen untuk menemukan kombinasi kata-kata dan instruksi yang paling efektif dalam mengarahkan model menuju keluaran yang sesuai dengan harapan pengguna. Untuk merancang *prompt* yang berhasil, pemahaman mendalam tentang karakteristik model dan konteks penggunaan sangat penting. Dengan demikian, *prompt engineering* menjadi langkah kunci dalam memaksimalkan potensi model dan memastikan respons yang memuaskan dalam berbagai situasi. Model *prompt engineering* dapat menjadi paradigma baru untuk interaksi, pengguna hanya perlu mencari tahu bagaimana cara meminta model untuk mendapatkan pengetahuan dan abstraksi spesifik yang diperlukan untuk menyelesaikan tugas (V. Liu & Chilton, 2022).

Terdapat tiga langkah utama dalam pendekatan *prompt engineering* yang digunakan dalam model bahasa. Pertama, *task description*, model membutuhkan penjelasan tugas dan

instruksi yang diberikan agar dapat memahami dengan baik apa yang diminta. Kedua, *feature description,* model perlu informasi lebih lanjut tentang fitur-fitur atau aspek-aspek yangrelevan dengan tugas tersebut. Ketiga, *domain knowledge*, model dapat memanfaatkan pengetahuan dalam domain tertentu yang diperlukan untuk mengeksekusi tugas dengan lebih baik (Deldjoo, 2023). Menurut (Dai et al., 2023), ada empat proses dalam *prompt engineering*yang digunakan. Pertama, *question,* adalah masalah yang harus dipecahkan. Kedua, *thought*, adalah ide dan proses berpikir dari masalah. Ketiga, *action*, adalah operasi yang dipilih oleh AI setelah mempertimbangkan mana yang menurut AI paling cocok untuk menyelesaikan tugas. Keempat, *action input*, digunakan sebagai input dari fungsi. Dari empat proses di atas ada tiga tahapan output yang akan didapatkan, yaitu Pertama, *observation*, adalah hasil dari fungsi untuk menginspirasi pemikiran AI selanjutnya. Kedua, *thought*, menunjukkan hasil dari pemikiran AI tentang *observation*. Ketiga, *final answer*, Jika AI berpikir bahwa hasil saat ini dapat menjawab pertanyaan awal, AI akan mengembalikan jawaban akhir, jika tidak, AI akan terus berpikir dan memanggil fungsi lain.

Prompt dapat direkayasa untuk memprogram LLM agar dapat melakukan lebih dari sekedar mendikte jenis keluaran atau menyaring informasi yang diberikan kepada model. Dengan *prompt* yang tepat, dimungkinkan untuk membuat paradigma interaksi yang sama sekali baru, seperti membuat LLM menghasilkan dan memberikan kuis yang terkait dengan konsep atau alat rekayasa perangkat lunak. Selain itu, *prompt* memiliki potensi untuk beradaptasi sendiri, menyarankan *prompt* lain untuk mengumpulkan informasi tambahan atau menghasilkan artefak terkait. Kemampuan *prompt* yang canggih ini menyoroti pentingnya merekayasa *prompt* untuk memberikan nilai lebih dari sekedar pembuatan teks atau kode (White et al., 2023).

2.7 Scrum

Scrum adalah salah satu model yang termasuk dalam metode *agile* untuk mengelola pengembangan proyek (Firdaus, 2017). Penggunaan *scrum* dalam rekayasa perangkat lunak telah terbukti sebagai metode yang efektif dan relatif mudah diterapkan dalam pengembangan aplikasi (Rizky & Sugiarti, 2022). *Scrum* memiliki fokus utama pada aktivitas "inspect dan adapt" yang bertujuan untuk mengidentifikasi dan mengatasi masalah yang muncul dengan cepat. Dengan aliran kerja dan prinsip-prinsip yang dimiliki oleh *scrum*, metodologi ini sangat cocok untuk digunakan dalam situasi pekerjaan harus diselesaikan dengan cepat dan dalam lingkungan bisnis yang berubah-ubah.

Sebagai metodologi yang muncul untuk mempercepat proses pengembangan, *scrum* memiliki alur atau tahapan khusus untuk mencapai tujuan tersebut. *Scrum* terbagi menjadi beberapa tahapan, yakni *product backlog*, *sprint planning*, *daily sprint*, *sprint review*, dan *sprint retrospective*. Setiap tahapan pada *Scrum* melibatkan beberapa peran yang dikenal dengan istilah *scrum team*.

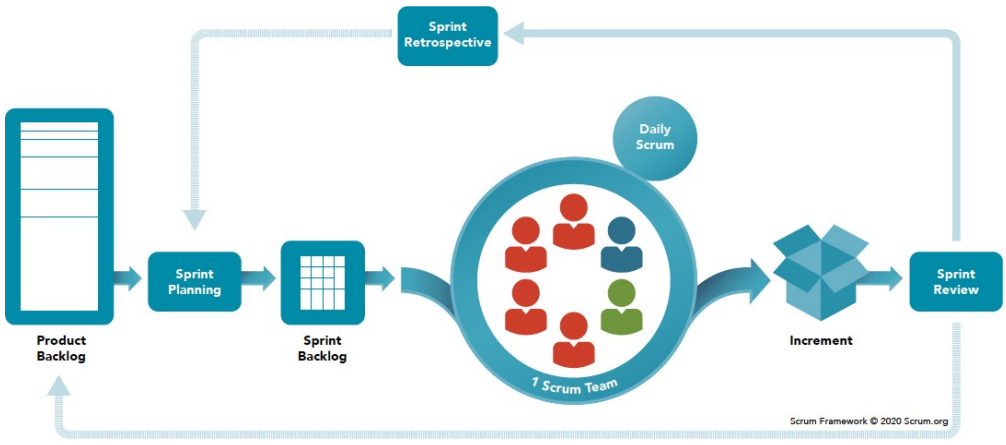
2.7.1 Scrum Team

Scrum team merupakan kumpulan individu yang umumnya terdiri dari 5 hingga 9 dengan memiliki tanggung jawab atau perang masing-masing dalam proses pengembangan. Secara umum terdapat 3 perang di dalam *scrum team*, diantaranya:

a. *Product Owner*, bertanggung jawab dalam menentukan prioritas fungsional sekaligus memastikan output dari hasil *sprint* telah memenuhi DoD.
b. *Scrum Master*, memiliki tanggung jawab untuk memastikan bahwa semua anggota tim *Scrum* memahami dan mengikuti nilai dan aturan yang terkait dengan *Scrum*.
c. *Development Team*, bertanggung jawab dalam mengembangkan sistem sesuai dengan *product backlog*.

2.7.2 Alur *Scrum*

Scrum memiliki alur kerja yang adaptif dan interaktif, seperti yang diperlihatkan dalam Gambar 2.3. Dalam alur kerja *scrum*, salah satu elemen kunci adalah *sprint*, yang melibatkan tim dalam menyelesaikan tugas yang tercantum dalam *sprint backlog* yang telah ditetapkan.



Gambar 2.3 Alur pengembangan dengan metodologi *scrum*

Sumber: (*Scrum* org, 2023)

Adapun penjelasan alur *scrum* adalah sebagai berikut:

a. *Product Backlog*

Product backlog secara sederhana dapat diartikan sebagai daftar poin yang harus dikerjakan oleh tim yang terurut secara prioritas, point tersebut diperoleh dari *product owner*. *Product backlog* merupakan sumber tunggal dari pekerjaan yang dapat dilakukan oleh *scrum team* (Schwaber & Sutherland, 2020).

b. *Sprint Planning*

Sprint Planning secara sederhana dapat diartikan sebagai aktivitas penyusunan kembali daftar poin dari *product backlog* menjadi PBI (*product backlog items*) yang akan dieksekusi pada saat *sprint* berlangsung.

c. *Sprint*

Sprint secara sederhana dapat diartikan sebagai aktivitas *scrum team* menyelesaikan *increment* dalam periode waktu tertentu. Dalam 1 kali *sprint* biasanya *scrum team* membutuhkan waktu 1 hingga 4 minggu sesuai dengan kesulitan *items* yang dikerjakan. Untuk menjaga proses pengembangan tetap berjalan dan sesuai dengan *goals*, terdapat aktivitas daily *scrum* di dalam aktivitas *sprint*. Daily *scrum* merupakan kegiatan rutin harian yang dilakukan *development team* tidak kurang dari 15 menit. Biasanya kegiatan ini berisi penyampaian *progress, plan, problem* dari aktivitas yang telah dikerjakan.

d. *Sprint Review*

Sprint review secara sederhana dapat diartikan sebagai aktivitas meninjau hasil *sprint* melalui presentasi yang dilakukan oleh *scrum team* kepada *stakeholder* sekaligus menentukan hal apa yang akan dilakukan selanjutnya (Schwaber & Sutherland, 2020). Pada umumnya, aktivitas ini maksimal berjalan selama 4 jam dalam *sprint* yang berdurasi 1 bulan. Jika durasi *sprint* kurang dari satu bulan, *sprint review* akan berjalan lebih singkat.

e. *Sprint Retrospective*

Sprint retrospective secara sederhana dapat diartikan sebagai aktivitas evaluasi terhadap *scrum team* untuk menentukan rencana agar meningkatkan efektivitas dan kualitas *team* (Schwaber & Sutherland, 2020). Poin yang dibahas pada aktivitas ini menitikberatkan pada proses pengembangan itu sendiri seperti aktivitas yang berdampak positif selama *sprint* berlangsung, problem yang dijumpai, hingga *problem solving* dari masalah tersebut, Sehingga dari pembahasan detail tersebut, pengembangan yang dilakukan oleh *scrum team* lebih efektif dan efisien.

BAB III
PELAKSANAAN MAGANG

Di Jala Tech, mereka memilih metodologi pengembangan aplikasi yang dikenal sebagai *scrum*. Metode *scrum* adalah salah satu pendekatan dalam pengembangan perangkat lunak yang menjalankan prinsip-prinsip *agile*. *Scrum* terbukti sangat efisien ketika digunakan dalam proyek-proyek yang kompleks dan memiliki jadwal yang ketat (Pressman, 2005). Proses pengembangan perangkat lunak di Jala Tech terdiri dari beberapa tahap kunci, termasuk *sprint planning*, *development*, *sprint review*, dan *sprint retrospective*.

Dalam konteks kegiatan magang penulis, setiap hari dimulai dengan meninjau pekerjaan yang telah dilakukan pada hari sebelumnya. Ini dilakukan sebagai persiapan untuk rapat harian bersama seluruh tim *software* di Jala Tech. Penulis aktif berpartisipasi dalam *daily meeting* ini bersama dengan anggota tim lainnya di ruang pertemuan. Tujuan dari pertemuan harian adalah untuk membahas kemajuan proyek dan berbagi rencana kegiatan hari itu untuk setiap anggota tim. Setelah *daily meeting* selesai, penulis melanjutkan dengan menjalankan tugas-tugas sesuai dengan rencana yang telah ditetapkan sebelumnya. Jika penulis menghadapi kendala atau hambatan selama menjalankan tugas yang diberikan, biasanya penulis akan menghubungi senior atau *supervisor* untuk meminta bimbingan dan berdiskusi mengenai masalah yang dihadapi.

Setiap Jumat sore, penulis mengadakan *weekly meeting* yang melibatkan semua anggota tim *software*. *Weekly meeting* ini diadakan untuk meninjau kemajuan proyek yang telah berjalan selama seminggu penuh dan untuk menguraikan tujuan untuk minggu yang akan datang. Biasanya, pertemuan ini diawasi oleh *supervisor* penulis, Farid Inawan yang menjabat sebagai *principal engineer* di Jala Tech.

3.1 Penerapan Metodologi *Scrum* di Jala Tech

Jala Tech menggunakan metodologi *scrum* untuk setiap proyek pengembangan produk perangkat lunak, metode *scrum* yang bersifat cepat dan adaptif sehingga memungkinkan sinkronisasi dan eksekusi yang cepat dari fase perencanaan, pengembangan, termasuk implementasi, pengujian, dan evaluasi (Grebić & Stojanović, 2021). Dalam kerangka kerja *scrum* di Jala Tech, ada dua fase utama: fase perencanaan dan fase pengembangan. Fase-fase ini mencakup empat tahap yang berbeda, yaitu *sprint planning*, *sprint* (*daily scrum*), *sprint*

review, dan *sprint retrospective*. Berikut ini adalah rincian dari setiap tahap dalam dua fase tersebut:

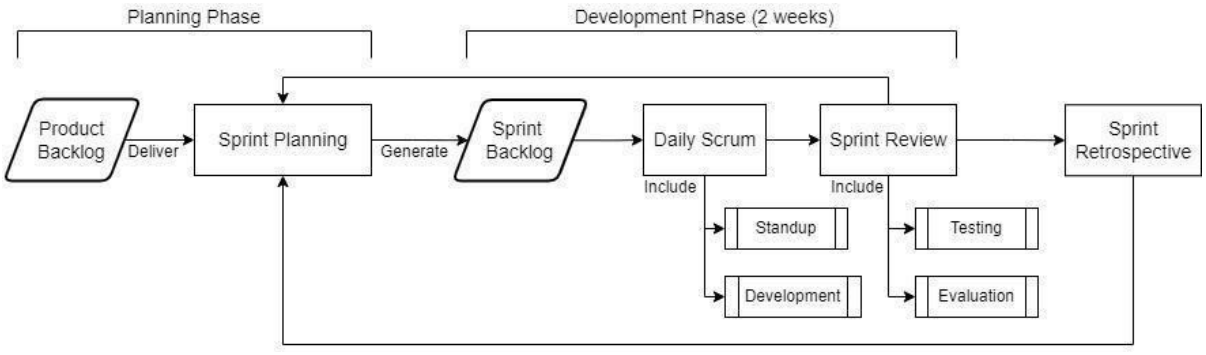
a. *Sprint planning* adalah fase yang didedikasikan untuk perencanaan implementasi proyek. Dimulai dengan diskusi *product backlog* yang dipimpin oleh *product owner*, diikuti dengan pembahasan *sprint backlog*, yang merupakan katalog tugas yang berasal dari *product backlog*. Selanjutnya, *product manager* memberikan tugas-tugas ini kepada masing-masing anggota tim.

b. *Sprint development*, merupakan tahap pengembangan terkait dengan pelaksanaan daftar pekerjaan yang diformulasikan selama tahap perencanaan. Selama tahap pengembangan ini, ada *daily meeting* dan *weekly meeting* rutin untuk memastikan bahwa kemajuan proyek tetap tersinkronisasi di seluruh anggota tim.

c. *Sprint review* adalah fase yang didedikasikan untuk mengevaluasi hasil dari upaya implementasi mingguan dan *sprint*. Tahap ini juga mencakup presentasi hasil pengujian fitur sesuai dengan spesifikasi produk yang dikembangkan.

d. *Sprint retrospective* merupakan fase yang melibatkan evaluasi kinerja setiap anggota tim dari tahap pengembangan. Selain itu, semua anggota tim terlibat dalam diskusi mengenai tantangan yang dihadapi selama pelaksanaan tugas dan merancang solusi untuk mengatasi rintangan tersebut. Proses ini berfungsi sebagai pengalaman belajar yang berharga dan meningkatkan efektivitas tugas dalam *sprint* berikutnya.

Adapun ilustrasi tahapan dari metode pengembangan *scrum* di Jala Tech yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 Alur metode pengembangan *scrum* yang diterapkan di Jala Tech

3.2 Sprint Planning

Selama fase ini, sebuah pertemuan diadakan dengan semua anggota tim proyek untuk membahas *product backlog* yang disediakan oleh *product owner*. *Product backlog* terdiri dari katalog persyaratan fungsional sistem yang perlu diimplementasikan oleh *developer* (Bhavsar et al., 2020). Dalam konteks *sprint planning*, tugas dibagi berdasarkan *product backlog* dan kemudian ditugaskan kepada *developer*. Pembagian tugas ini dilakukan dengan mengkategorikan tugas sesuai dengan bobot dan cakupan fitur kebutuhan sistem. Daftar detail *backlog* yang ditangani dalam proyekipengembangan fitur *chatbot* dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar *backlog* proyek pengembangan fitur *chatbot*

No.	***Backlog***
1.	Melakukan setup Chat GPT, Auto-GPT, dan LangChain
2.	Integrasi dengan JALA API – List
3.	Integrasi dengan JALA API – Action
4.	Integrasi dengan JALA DATABASE – Query

Setelah *product manager* memahami daftar fitur yang akan dikembangkan dalam *chatbot*, mereka terlibat dalam perbincangan dengan pengembang untuk merencanakan cara mengalokasikan tugas-tugas tersebut. Tugas-tugas ini direncanakan berdasarkan daftar fitur yang ada dalam *product backlog*, yang kemudian membentuk *sprint backlog*. *Sprint backlog* adalah daftar tugas yang disusun secara teratur berdasarkan prioritas dari product backlog, dan ini akan dijalankan dalam konteks pengembangan (Bhavsar et al., 2020). Rincian mengenai *sprint backlog*, yang memerlukan pelaksanaan, dapat ditemukan dalam Tabel 3.2.

Tabel 3.2 *sprint backlog* yang telah disusun

No.	**Nama *Backlog***	**Definisi Tugas**	**Bobot Tugas (Skala 1 – 10)**
1.	Melakukan setup Chat GPT, Auto-GPT, dan LangChain	Melakukan setup Chat GPT dengan membuat *environment* serta memasukkan api-key dari openAI	5

No.	Nama *Backlog*	Definisi Tugas	Bobot Tugas (Skala 1 – 10)
		Melakukan setup Auto-GPT dengan membuat *environment* serta memasukkan api-key dari openAI	5
		Melakukan setup LangChain dengan membuat *environment* yang dibutuhkan	5
2.	Integrasi dengan JALA API – List	Melakukan integrasi model dengan API *jala_list_regions*	5
		Melakukan integrasi model dengan API *jala_list_shrimp_prices*	5
		Melakukan integrasi model dengan API *jala_list_farms*	5
		Melakukan integrasi model dengan API *jala_create_farm*	5
		Melakukan integrasi model dengan API *jala_update_farm*	5
		Melakukan integrasi model dengan API *jala_create_farm*	5
		Melakukan integrasi model dengan API *jala_list_ponds_in_farm*	5
		Melakukan integrasi model dengan API *jala_create_multi_measurement*	5
3.	Integrasi dengan JALA API – Action	Melakukan integrasi model dengan API *browse_list_sites*	5

No.	Nama *Backlog*	Definisi Tugas	Bobot Tugas (Skala 1 – 10)
		Melakukan integrasi model dengan API *browse_summarize_site*	5
4.	Integrasi dengan JALA Database – Query	Melakukan integrasi model dengan query Jala	5

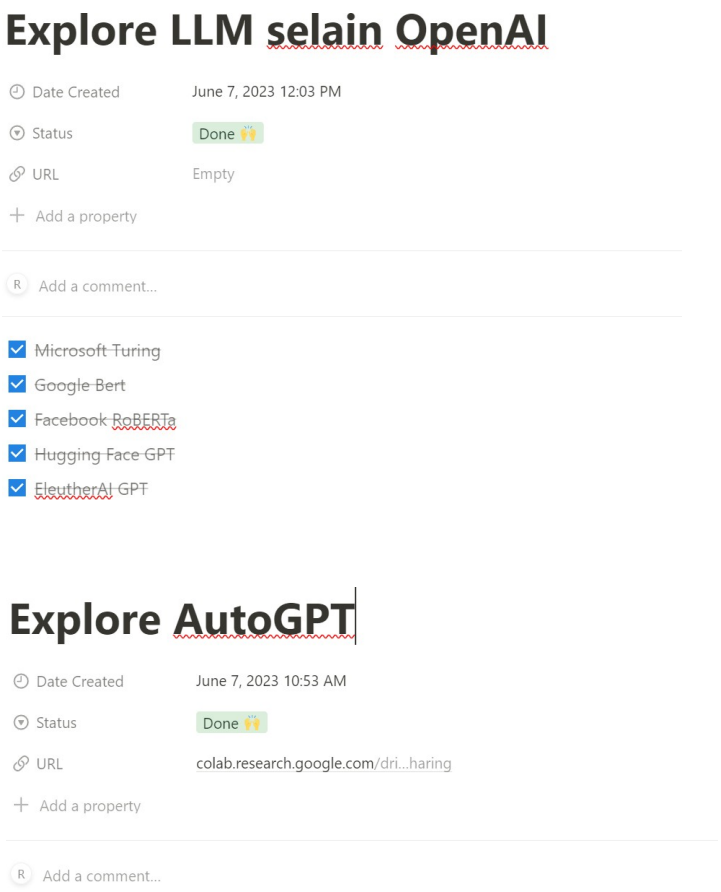
3.3 Sprint Development

Setelah perencanaan *sprint backlog*, fase *development* dimulai. Para *developer* melakukan tugas-tugas yang diberikan yang diuraikan dalam *sprint backlog*. Fase *development* ini berlangsung selama dua minggu dan mencakup tahap-tahap utama seperti *daily scrum* dan *sprint review*, yang berlangsung pada hari terakhir setiap *sprint*. *Daily scrum* berfungsi sebagai platform untuk kolaborasi tim, yang melibatkan diskusi tentang kemajuan pengembangan. Di Jala Tech, *daily scrum* mencakup aktivitas seperti *daily meeting*. Anggota tim membahas status tugas individu, rencana aktivitas harian, dan tantangan apa pun yang dihadapi saat mengerjakan tugas-tugas yang ada di *backlog*.

Selama minggu pertama *sprint*, fokusnya adalah pada implementasi tugas, sedangkan minggu kedua didedikasikan untuk menguji fitur yang dikembangkan di minggu awal. Hari terakhir *sprint* menyelenggarakan sesi peninjauan, tugas-tugas yang sudah selesai dan yang belum selesai dibahas. Selain itu, *product manager* menggunakan *sprint review* untuk membuat keputusan mengenai perlunya *sprint* tambahan dalam proyek *chatbot* Jala. Selain itu, pada hari terakhir *sprint*, *sprint retrospective* dilakukan untuk mengatasi masalah dan merancang solusi terkait hambatan implementasi. Proses ini membantu meningkatkan efisiensi bagi semua anggota tim dalam pelaksanaan tugas mereka.

3.3.1 Sprint Ke-1

Penulis masuk ke proyek *chatbot* pada *sprint* ke-1 diawal bulan Juni, tepatnya tanggal 5 Juni 2023. Sprint ke-1 ini berlangsung pada tanggal 5 Juni 2023 hingga 16 Juni 2023. Pada saat itu penulis berkontribusi sebagai *Software Engineer* yang memiliki tanggung jawab melakukan pengembangan *chatbot.* Pada sprint ke-1 ini penulis melakukan eksplorasi model dan *architecure* berbagai model *chatbot,* membandingkan kinerja antara model dengan alur sederhana. Pada Gambar 3.2 terdapat bukti penugasan pada Notion untuk ekplorasi Auto-GPT, ChatGPT, dan *explore* LLM selain Auto-GPT dan OpenAI.



Gambar 3.2 Bukti penugasan pada Notion untuk *eksplorasi* dan setup model *chatbot*

3.3.1.1 Melakukan *Setup* Chat GPT, Auto-GPT, dan LangChain

Untuk menyelesaikan tugas ini, penulis telah membuat dua file, yaitu *setup.py* dan *OpenAIStream.js*. Kedua file ini berisi fungsi-fungsi yang diperlukan untuk mengintegrasikan *chatbot* Jala dengan akun Jala, serta untuk mengintegrasikan dengan Chat GPT, Auto-GPT, dan LangChain. Pada file *setup.py*, terdapat dua fungsi yang perlu dibuat. Hal ini dikarenakan dalam fitur integrasi akun Jala, pengguna harus memasukkan data akun Jala yang aktifsehingga *chatbot* dapat melakukan permintaan ke API Jala yang terkait dengan integrasi akunJala. Di dalam proses ini, ada langkah-langkah autentikasi dan otorisasi yang diperlukan untukmenjaga keamanan akun Jala.

```
authorization_base_url = 'https://app.jala.tech/oauth/authorize'
token_url = 'https://app.jala.tech/oauth/token'
redirect_uri = 'https://app.jala.tech/callback'
jala_oauth = OAuth2Session(client_id,
redirect_uri=redirect_uri)
```

```
authorization_url,                          state                          =
jala oauth.authorization url(authorization base url)

authorization_response = input('Enter the full callback URL: ')
token = jala_oauth.fetch_token(token_url,
client_secret=client_secret,
authorization response=authorization response)
```

Gambar 3.3 Kode implementasi untuk *setup* autentikasi akun Jala.

Selain itu dalam mengintegrasikan dengan Chat GPT, Auto-GPT, dan LangChain dibutuhkan sebuah *prompt* yang baik sehingga hasil yang diinginkan tercapai. Penulis membuat file baru bernama *prompt*.js untuk menyimpan *prompt* yang digunakan dalam *chatbot* terdapat dua jenis *prompt* yakni *prompt* untuk mengakses API serta *prompt* untuk mengakses *database*. Kode program pada *prompt* dapat dilihat pada Gambar 3.4.

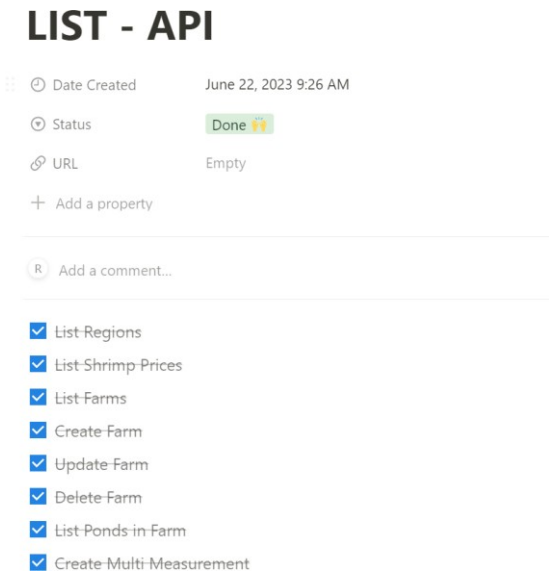
```
Export const system_message = ‘Anda adalah JALA-GPT,
asisten AI yang dikembangkan oleh JALA Tech,
sebuah startup teknologi yang fokus pada budidaya udang.
Anda akan menjawab pertanyaan seputar JALA Tech dan topik terkait budidaya
udang.
Ketika berbicara tentang JALA, konteksnya adalah JALA Tech.
Anda akan menjawab pertanyaan dengan sejujur mungkin.`

export const SQL_PREFIX = `You are an agent designed to interact with a SQL
database.
Given an input question, create a syntactically correct {dialect} query to
run, then look at the results of the query and return the answer.
Always limit your query to at most {top_k} results using the LIMIT clause.
You can order the results by a relevant column to return the most interesting
examples in the database.
Never query for all the columns from a specific table, only ask for a the
few relevant columns given the question.
If you get a "no such table" error, rewrite your query by using the table in
quotes.
DO NOT use a column name that does not exist in the table.
You have access to tools for interacting with the database.
Only use the below tools. Only use the information returned by the below
tools to construct your final answer.
```

```
You MUST double check your query before executing it. If you get an error
while executing a query, rewrite a different query and try again.
DO NOT try to execute the query more than three times.
If the question does not seem related to the database, just return "I don't
know" as the answer.
If you cannot find a way to answer the question, just return the best answer
you can find after trying at least three times.`
```

Gambar 3.4 Prompt Auto-GPT dan LangChain

3.3.2 Sprint Ke-2 dan Ke-3

Sprint ke-2 yang berlangsung dari tanggal 19 Juni 2023 hingga 30 Juni 2023, serta Sprint ke-3 yang berlangsung dari tanggal 3 Juli 2023 hingga 14 Juli 2023, penulis memegang tanggung jawab utama untuk meneruskan pengembangan model dan melakukan integrasi dengan layanan yang telah disediakan di Jala Apps menggunakan API – List. Pada Gambar 3.5, terdapat bukti penugasan yang terdokumentasi dengan jelas di platform Notion, menunjukkan perincian tugas yang mencakup pembuatan beberapa API – List dengan menggunakan bahasa pemrograman Python. Tugas ini menunjukkan fokus dan kontribusi penulis dalam merancang solusi yang sesuai dengan kebutuhan proyek pada periode Sprint tersebut.

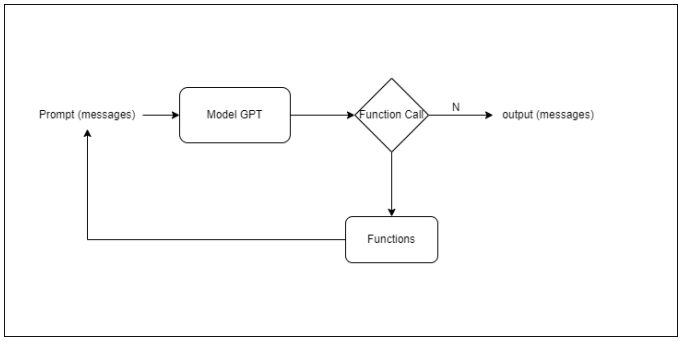


Gambar 3.5 Bukti penugasan pada Notion untuk integrasi dengan beberapa API Jala

3.3.2.1 Integrasi Dengan Jala API – List

Integrasi API Jala – List adalah tugas yang bertujuan untuk menghubungkan *chatbot* dengan berbagai layanan yang tersedia di platform Jala. Dalam model interaksi percakapan tanya-jawab antara *chatbot* dan pengguna, beberapa jenis API perlu dimuat. Ini mencakup layanan *regions* (wilayah), *list* harga udang, *list* tambak (kolam ikan), pembuatan tambak, pembaruan tambak, penghapusan tambak, *list* daftar kolam dalam tambak, dan pembuatan pengukuran multi. Untuk memudahkan pengembangan, API ini perlu dibagi menjadi beberapa bagian sesuai dengan jenisnya. Oleh karena itu, penulis telah membuat API terpisah untuk setiap jenis data ini, sehingga definisi fungsi dapat disusun dengan lebih mudah.

Sebelum melanjutkan dengan implementasi, penulis melakukan diskusi bersama Farid Inawan untuk merancang alur interaksi yang tepat serta menetapkan fungsi-fungsi yang harus menjadi prioritas utama. Hasil diskusi disepakati alur interaksi yang akan diikuti bersama dengan daftar fungsi yang harus menjadi prioritas input. Berikut adalah gambaran alur integrasi fungsi dengan Jala API, seperti yang diilustrasikan dalam Gambar 3.6.



Gambar 3.6 Diagram alur interaksi function dengan Jala API

Tabel 3.3 Daftar parameter pada Jala API - List

Nama *Function*	Tipe Data
jala_list_regions	String | Priority
jala_list_shrimp_prices	String | Priority
jala_list_farms	String | Priority
jala_create_farm	String
jala_update_farm	String
jala_delete_farm	String
jala_list_ponds_in_farm	String
jala_create_multi_measurement	String

Pada Tabel 3.3, disajikan data-data parameter yang tersedia pada *function list*. Parameter yang bersifat prioritas utama adalah parameter yang akan diproses oleh *chatbot* berdasarkan *prompt* yang dimasukkan oleh pengguna. Pada langkah ini, aktivitas yang dilakukan adalah membuat *function* untuk menghubungkan Chat GPT dan Auto-GPT dengan service *list regions* menjadi JSON sehingga *frontend* dapat bekerja dengan data yang diterima dengan mudah. Aktivitas ini dimulai dari membuat file baru yang bernama *model.py*. Kemudian dilakukan penulisan beberapa blok kode seperti yang dapat dilihat pada Gambar 3.5.

```
class RequestParamsModel(BaseModel):
  search: Optional[str] = Field(min_length=1)
  page: Optional[int] = 1
  per_page: Optional[int] = 15

class ListRegionParamsModel(BaseModel):
  request_params: RequestParamsModel = RequestParamsModel()

def jala_list_regions(params: ListRegionParamsModel):
  """Get list of regions"""

  response: Response = jala_oauth.get(
      "https://app.jala.tech/api/regions",
      params=params.request_params.dict()
  )

  return json.dumps({
      "status_code": response.status_code,
      "reason": response.reason,
      "content": response.json()
  })
```

Gambar 3.7 Kode program implementasi *function jala_list_regions*

Jika mengacu pada Gambar 3.7 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan permintaan HTTP GET ke API Jala untuk mengambil daftar wilayah (*regions*). Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **RequestParamsModel:** Ini adalah sebuah model data yang mendefinisikan menggunakan Pydantic. Model ini digunakan untuk mewakili parameter yang diperlukan untuk melakukan permintaan API Jala. Model ini memiliki tiga bidang (*field*):

- **‘search’:** Sebuah *string* opsional yang memiliki panjang minimal 1 karakter. Ini akan digunakan sebagai kata kunci pencarian.
- **‘page’:** Sebuah integer opsional dengan nilai *default* 1. Ini akan digunakan untuk menentukan halaman yang akan diambil.
- **‘per_page’**: Sebuah integer opsional dengan nilai *default* 15. Ini akan digunakan untuk digunakan untuk menentukan jumlah item per halaman.

- **‘ListRegionParamModel’:** Ini adalah model data lainnya yang juga menggunakan Pydantic. Model ini memiliki satu bidang yaitu **‘request_params’**, yang merupakan objek dari tipe **‘RequestParamModel’**. Model ini digunakan untuk menggabungkan parameter permintaan dalam satu objek.
- **‘jala_list_regions(params):** Ini adalah fungsi yang digunakan untuk melakukan permintaan ke API Jala untuk mengambil daftar wilayah (region).

```
class RequestParamsModel(BaseModel):
  search: Optional[str] = Field(min_length=1)
  page: Optional[int] = 1
  per_page: Optional[int] = 15

class ListShrimpPriceParamsModel(BaseModel):
  request_params: RequestParamsModel = RequestParamsModel()

def jala_list_shrimp_prices(params: ListShrimpPriceParamsModel):
  """Get list of shrimp prices from JALA"""

  response: Response = jala_oauth.get(
      "https://app.jala.tech/api/shrimp_prices",
      params=params.request_params.dict()
  )

  return json.dumps({
      "status_code": response.status_code,
      "reason": response.reason,
      "content": response.json()
  })
```

Gambar 3.8 Kode program implementasi *function jala_list_shrimp_prices*

Jika mengacu pada Gambar 3.8 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan permintaan HTTP GET ke API Jala. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **'ListShrimpPriceParamsModel':** Ini adalah model data yang digunakan untuk menggabungkan parameter permintaan dalam satu objek yang berhubungan dengan daftar harga udang (*shrimp prices*). Model ini memiliki satu bidang yaitu **'request_params'**, yang merupakan objek dari tipe **'RequestParamsModel'**. Dengan cara ini, kita dapat menggabungkan parameter yang sama seperti yang digunakan dalam permintaan sebelumnya ke dalam objek ini.
- **'jala_list_shrimp_prices(params)':** Ini adalah fungsi yang digunakan untuk melakukan permintaan ke API Jala untuk mengambil daftar harga udang (*shrimp prices*).

```
class RequestParamsModel(BaseModel):
  search: Optional[str] = Field(min_length=1)
  page: Optional[int] = 1
  per_page: Optional[int] = 15

class ListFarmParamsModel(BaseModel):
  request_params: RequestParamsModel = RequestParamsModel()

def jala_list_farms(params: ListFarmParamsModel):
  """Get list of farms from JALA"""

  response: Response = jala_oauth.get(
      "https://app.jala.tech/api/farms",
      params=params.request_params.dict()
  )

  return json.dumps({
      "status_code": response.status_code,
      "reason": response.reason,
      "content": response.json()
  })
```

Gambar 3.9 Kode program implementasi *function jala_list_farms*

Jika mengacu pada Gambar 3.9 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan permintaan HTTP GET ke API Jala. Berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **'ListFarmParamsModel'**: Ini adalah model data yang digunakan untuk menggabungkan parameter permintaan dalam satu objek yang berhubungan dengan daftar kolam (*farm*). Model ini memiliki satu bidang yaitu **'request_params'**, yang merupakan objek dari tipe **'RequestParamsModel'**. Dengan cara ini, kita dapat menggabungkan parameter yang sama seperti yang digunakan dalam permintaan sebelumnya ke dalam objek ini.
- **'jala_list_farms(params)**: Ini adalah fungsi yang digunakan untuk melakukan permintaan ke API Jala untuk mengambil daftar kolam (*farm*) dari JALA.

```
class CreateFarmParamsModel(BaseModel):
  name: str
  region_id: str = Field(description="Refer to list regions: jala_list_regions")
  timezone: str = Field(description="Timezone format is ±HH:mm, ex: +07:00", regex=r'^[+-]\d{2}:\d{2}$')

def jala_create_farm(params: CreateFarmParamsModel):
  """Create farm from JALA"""

  response: Response = jala_oauth.post(
      "https://app.jala.tech/api/farms",
      data=params.dict()
  )

  return json.dumps({
      "status_code": response.status_code,
      "reason": response.reason,
      "content": response.json()
  })
```

Gambar 3.10 Kode program implementasi *function jala_create_farm*

Jika mengacu pada Gambar 3.10 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan

permintaan HTTP POST ke API Jala untuk membuat entitas tambak baru (*farm*). Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **CreateFramParamsModel'**: Ini adalah model data yang digunakan untuk menggambarkan data yang diperlukan untuk membuat tambak baru di API Jala. Model ini memiliki tiga bidang (fields):
  - **'name'**: Ini adalah *string* yang harus diisi yang mewakili nama tambak yang akan dibuat.
  - **'region_id'**: Ini adalah *string* yang mewakili ID dari wilayah (*regions*) tempat peternakan akan dibuat. Bidang ini memiliki deskripsi yang menjelaskan bahwa harus merujuk ke daftar wilayah yang mungkin diperoleh melalui fungsi 'jala_list_regions'.
  - **'timezone'**: Ini adalah *string* yang mewakili zona waktu tambak dalam format ±HH:mm. Bidang ini juga memiliki deskripsi yang menjelaskan format yang diharapkan dan menggunakan ekspresi reguler (*regex*) untuk memvalidasi formatnya.
- **'jala_create_farm(params)**: Ini adalah fungsi yang digunakan untuk membuat permintaan HTTP POST ke API Jala untuk membuat tambak baru berdasarkan data yang diberikan dalam objek ` CreateFarmParamsModel`.

```
class UpdateFarmParamsModel(BaseModel):
  id: int = Field(title="Farm ID", description="Refer to list farms: jala_list_farms")
  name: Optional[str]
  region_id: Optional[str] = Field(description="Refer to list regions: jala_list_regions")
def jala_update_farm(params: UpdateFarmParamsModel):
  """Update farm from JALA"""
  response: Response = jala_oauth.put(
      f"https://app.jala.tech/api/farms/{params.id}",
      data=params.dict())
  return json.dumps({
      "status_code": response.status_code,
      "reason": response.reason,
      "content": response.json()
```

Gambar 3.11 Kode program implementasi *function jala_update_farm*

Jika mengacu pada Gambar 3.11 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan permintaan HTTP PUT ke API Jala untuk memperbarui informasi tambak (*farm*). Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **'UpdateFarmParamsModel'**: Ini adalah model data yang digunakan untuk menggambarkan data yang diperlukan untuk memperbarui tambak di API Jala. Model ini memiliki tiga bidang (*fields*):
  - **'id'**: Ini adalah integer yang mewakili ID peternakan yang akan diperbarui. Bidang ini memiliki judul ("Farm ID") dan deskripsi yang menjelaskan bahwa harus merujuk ke daftar tambak yang mungkin diperoleh melalui fungsi 'jala_list_farms'.
  - **'name'**: Ini adalah *string* opsional yang mewakili nama baru untuk tambak.
  - **'region_id'**: Ini adalah *string* opsional yang mewakili ID dari wilayah (*regions*) baru tempat tambak berada. Bidang ini memiliki deskripsi yang menjelaskan bahwa Anda harus merujuk ke daftar wilayah yang mungkin diperoleh melalui fungsi 'jala_list_regions'.
- **'jala_update_farm(params)**: Ini adalah fungsi yang digunakan untuk membuat permintaan HTTP PUT ke API Jala untuk memperbarui informasi tentang peternakan berdasarkan data yang diberikan dalam objek **'UpdateFarmParamsModel'**.

```
class DeleteFarmParamsModel(BaseModel):
  id: int = Field(title="Farm ID", description="Refer to list farms: jala_list_farms")

def jala_delete_farm(params: DeleteFarmParamsModel):
  """Delete farm from JALA"""
  response: Response = jala_oauth.delete(
      f"https://app.jala.tech/api/farms/{params.id}"
  )
  return json.dumps({
      "status_code": response.status_code,
      "reason": response.reason,
      "content": response.json()
  })
```

Gambar 3.12 Kode program implementasi *function jala_delete_farm*

Jika mengacu pada Gambar 3.12 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan permintaan HTTP DELETE ke API Jala untuk menghapus tambak. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **'DeleteFarmParamsModel'**: Ini adalah model data yang digunakan untuk menggambarkan data yang diperlukan untuk menghapus tambak dari API Jala. Model ini memiliki satu bidang (field):
    - **'id'**: Ini adalah integer yang mewakili ID tambak yang akan dihapus. Bidang ini memiliki judul ("Farm ID") dan deskripsi yang menjelaskan bahwa harus merujuk ke daftar tambak yang diperoleh melalui fungsi `jala_list_farms`.
- **'jala_delete_farm(params)**: Ini adalah fungsi yang digunakan untuk melakukan permintaan HTTP DELETE ke API Jala untuk menghapus tambak berdasarkan ID yang diberikan dalam objek `**DeleteFarmParamsModel**`.

```
class RequestParamsModel(BaseModel):
  search: Optional[str] = Field(min_length=1)
  page: Optional[int] = 1
  per_page: Optional[int] = 15

class  ListPondInFarmParamsModel(BaseModel):
  request_params: RequestParamsModel = RequestParamsModel()
  farm_id: int = Field(description="Refer to list farms: jala_list_farms")

def jala_list_ponds_in_farm(params: ListPondInFarmParamsModel):
  """Get list of ponds in farm from JALA"""

  response = jala_oauth.get(
      f"https://app.jala.tech/api/farms/{params.farm_id}/ponds",
      params=params.request_params.dict()
  )

  return json.dumps({
      "status_code": response.status_code,
      "reason": response.reason,
      "content": response.json()
  })
```

Gambar 3.13 Kode program implementasi *function jala_list_ponds_in_farm*

Jika mengacu pada Gambar 3.13 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan permintaan HTTP GET ke API Jala untuk mengambil daftar kolam (*ponds*) dalam suatu tambak (*farm*). Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **`ListPondInFarmParamsModel`**: Ini adalah model data yang digunakan untuk menggambarkan data yang diperlukan untuk mengambil daftar kolam dalam suatu peternakan dari API Jala. Model ini memiliki dua bidang (fields):
  - **`request_params`**: Ini adalah objek dari tipe ‘RequestParamsModel` yang digunakan untuk mengatur parameter permintaan seperti pencarian, halaman, dan jumlah item per halaman. Parameter ini opsional dan memiliki nilai *default* dari objek ` RequestParamsModel`.
  - **`farm_id`**: Ini adalah integer yang mewakili ID tambak (*farm*) yang kolam-kolamnya akan diambil. Bidang ini memiliki deskripsi yang menjelaskan bahwa harus merujuk ke daftar peternakan yang mungkin diperoleh melalui fungsi `jala_list_farms`.
- **`jala_list_ponds_in_farm(params)`**: Ini adalah fungsi yang digunakan untuk melakukan permintaan HTTP GET ke API Jala untuk mengambil daftar kolam dalam suatu peternakan berdasarkan ID peternakan yang diberikan dalam objek **`ListPondInFarmParamsModel`**.

```
class CreateMultiMeasurementParamsModel(BaseModel):
  pond_id: int = Field(description="Refer to list ponds: jala_list_ponds")
  measured_at: str = Field(description="Datetime of measurements, example
format: 2023-06-26\T07:00:00+07:00")
  temperature: Optional[float] = Field(title="Temperature (C)")
  ph: Optional[float] = Field(title="pH")
  do: Optional[float] = Field(title="Dissolved Oxygen")
  salinity: Optional[float] = Field(title="Salinity")

def                                    jala_create_multi_measurement(params:
CreateMultiMeasurementParamsModel):
  """Create multi measurement to JALA"""
  response: Response = jala_oauth.post(
      f"https://app.jala.tech/api/multi_measurements",
      data=params.dict()
  )
```

```
    return json.dumps({
        "status_code": response.status_code,
        "reason": response.reason,
        "content": response.json()
    })
```

Gambar 3.14 Kode program implementasi *function* jala_create_multi_measurement

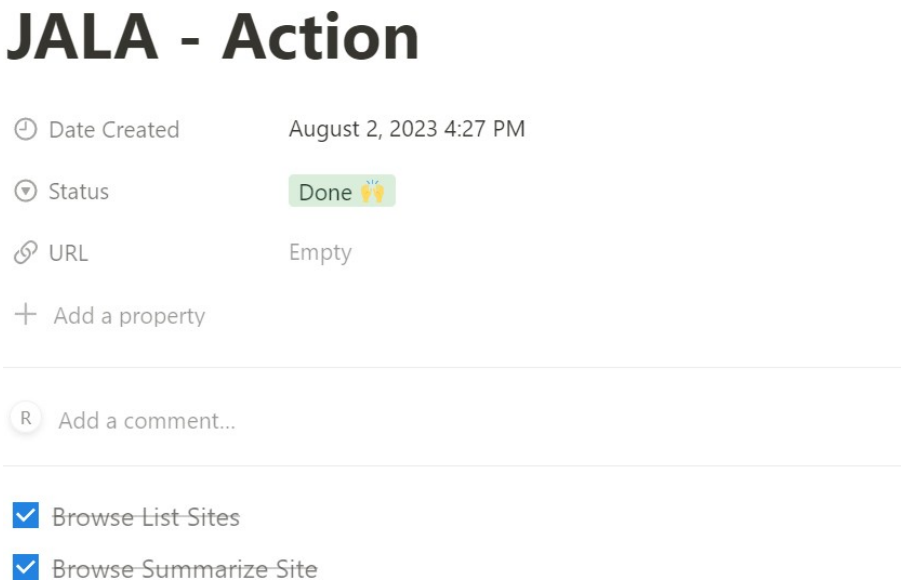
Jika mengacu pada Gambar 3.14 tersebut, dapat dilihat menggunakan modul Pydantic untuk mendefinisikan model data dan menggunakan modul HTTPx untuk melakukan permintaan HTTP POST ke API Jala untuk membuat entitas pengukuran multi-parameter. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **`CreateMultiMeasurementParamsModel`**: Ini adalah model data yang digunakan untuk menggambarkan data yang diperlukan untuk membuat pengukuran multi-parameter di API Jala. Model ini memiliki beberapa bidang (fields):
    - **'pond_id`**: Ini adalah integer yang mewakili ID kolam (pond), deskripsi pada bidang ini menegaskan bahwa ID tersebut harus mengacu pada daftar kolam yang dapat ditemukan melalui fungsi **'jala_list_ponds'**. Sedangkan, **'measured_at'** adalah sebuah *string* yang mencerminkan tanggal dan waktu pengukuran dalam format yang telah ditentukan, contohnya, "2023-06-26T07:00:00+07:00". Deskripsi pada bidang ini menjelaskan format yang diharapkan."
    - **`temperature`, `ph`, `do`, `salinity`**: Ini adalah bidang opsional yang mewakili pengukuran suhu (temperature), pH, kadar oksigen terlarut (dissolved oxygen), dan salinitas. Dapat memasukkan nilai-nilai ini jika memiliki data pengukuran. Masing-masing dari bidang ini juga memiliki judul yang menjelaskan jenis pengukuran yang dilakukan.
- **'jala_create_multi_measurement(params)'**: Ini adalah fungsi yang digunakan untuk membuat permintaan HTTP POST ke API Jala untuk membuat pengukuran multi parameter berdasarkan data yang diberikan dalam objek **`CreateMultiMeasurementParamsModel`**.

3.3.3 Sprint Ke-4 dan Sprint Ke-5

Sprint ke-4 berlangsung dari tanggal 17 Juli 2023 hingga 28 Juli 2023, sementara sprint ke-5 berlangsung dari tanggal 31 Juli 2023 hingga 11 Agustus 2023. Pada kedua sprint tersebut, penulis memiliki tanggung jawab untuk melanjutkan pengembangan model dan melakukan

integrasi dengan layanan yang tersedia di Jala Apps menggunakan API – Action. API – Action dibangun dengan tujuan memungkinkan *chatbot* melakukan tugas secara mandiri, melatih diri sendiri, dan dapat mengakses informasi dari sumber eksternal. Gambar 3.15 menunjukkan bukti penugasan pada platform Notion yang merinci tugas pembuatan beberapa API – Action menggunakan bahasa pemrograman Python. Tugas ini mencerminkan fokus penulis dalam merancang solusi yang mendukung kemampuan chatbot untuk berinteraksi secara otonom dan menggali informasi eksternal.



Gambar 3.15 Bukti penugasan pada Notion untuk integrasi dengan beberapa API Jala - Action

3.3.3.1 Integrasi Dengan Jala API - Action

Setelah menyelesaikan integrasi dengan Jala API - List, langkah selanjutnya adalah mengimplementasikan integrasi dengan Jala API - Action. Dengan integrasi ini, *chatbot* akan memiliki kemampuan untuk melakukan pelatihan otomatis dan mengambil data dari Jala secara eksternal. Tujuan utamanya adalah untuk memperkuat dan meningkatkan kualitas informasi yang disampaikan oleh *chatbot*.

Dalam tahap ini, *chatbot* akan mampu menjalankan tindakan atau aksi tertentu berdasarkan permintaan pengguna dan mengintegrasikannya dengan Jala API. Contoh tindakan ini mungkin melibatkan operasi seperti menambahkan data ke dalam sistem Jala, memperbarui informasi, atau melakukan pelatihan model berdasarkan data yang diperoleh dari Jala.

```
class ListSiteParamsModel(BaseModel):
  q: str = Field(title="Google search query")
  hl: Optional[str] = Field(title="User interface language. Ex: id")
  num: Optional[int] = Field(title="Number of search results to return.", min=1, max=10)
  start: Optional[int] = Field(title="The index of the first result to return")


class SummarizeSiteParamsModel(BaseModel):
  link: str = Field(title="URL to the website, preferably coming from list_sites")
  question: str = Field(title="Context for summarization of the content")


def browse_list_sites(params: ListSiteParamsModel):
  """Get list of sites that related to query"""

  from googleapiclient.discovery import build
  from googleapiclient.errors import HttpError

  service = build("customsearch", "v1", developerKey='AIzaSyDhfuGCVYf13lvccwIRJcriUuVpjL1rcCo')
  try:
    response = service.cse().list(cx='a1271872f41374a11', **params.dict()).execute()
    content = {
        "items": list(map(lambda x: ({
            "title": x["title"],
            "link": x["link"],
            "snippet": x["snippet"]
        }), response["items"]))
    }
    return json.dumps({
        "status_code": 200,
        "reason": "OK",
        "content": content
    })
  except HttpError as e:
    return json.dumps({
```

```
        "status_code": e.status_code,
        "reason": e.reason,
        "content": str(e)
    })
```

Gambar 3.16 Kode program implementasi *function browse_list_sites*

Jika mengacu pada Gambar 3.16 tersebut, dapat dilihat bahwa kode program untuk mengintegrasikan *chatbot* dengan *custom search engine* (CSE) dari Google untuk melakukan pencarian situs web berdasarkan *query* pengguna. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **'ListSiteParamsModel'**: Ini adalah model data yang digunakan untuk menggambarkan parameter yang diperlukan untuk melakukan pencarian situs web dengan layanan *Google Custom Search Engine*. Model ini memiliki beberapa bidang (fields):
  - **'q'**: Ini adalah *string* yang mewakili kueri pencarian Google yang akan digunakan untuk mencari situs web. Bidang ini memiliki judul ("Google search query").
  - **'h1'**: Ini adalah *string* opsional yang mewakili bahasa antarmuka pengguna. Misalnya, "id" untuk Bahasa Indonesia. Bidang ini memiliki judul ("*User interface language*").
  - **'num'**: Ini adalah integer opsional yang menentukan jumlah hasil pencarian yang akan dikembalikan. Nilai ini harus antara 1 dan 10. Bidang ini memiliki judul ("*Number of search results to return*").
  - **'start'**: Ini adalah integer opsional yang menentukan indeks hasil pertama yang akan dikembalikan. Bidang ini memiliki judul ("*The index ofithe first result toireturn*").
- **'SummarizeSiteParamsModel'**: Ini adalah model data yang digunakan untuk menggambarkan parameter yang diperlukan untuk merangkum konten dari situs web yang telah ditemukan dalam hasil pencarian. Model ini memiliki dua bidang (fields):
  - **'link'**: Ini adalah *string* yang mewakili URL situs web yang akan dirangkum. Idealnya, URL ini harus berasal dari hasil pencarian sebelumnya. Bidang ini memiliki judul ("*URL to the website, preferably coming from list_sites*").
  - **'question'**: Ini adalah *string* yang digunakan sebagai konteks untuk merangkum konten dari situs web yang dimaksud. Bidang ini memiliki judul ("*Context for summarization of the content*").

- **‘browse_list_sites(params)’**: Ini adalah fungsi yang digunakan untuk melakukan pencarian situs web menggunakan layanan *Google Custom Search Engine* (CSE) dengan parameter yang diberikan dalam objek **‘ListSiteParamsModel’**.

```
def browse_summarize_site(params: SummarizeSiteParamsModel):
  """Get summary of site that related to the question"""

  try:
    response = requests.get(params.link)
  except ConnectionError as e:
    return json.dumps({
        "status_code": e.errno,
        "reason": e.strerror,
        "content": str(e)
    })

  from bs4 import BeautifulSoup

  soup = BeautifulSoup(response.text, 'html.parser')
  script_tags = soup.find_all('script')
  for script_tag in script_tags:
      script_tag.extract()
  content = "\n".join(soup.find_all(string=True))
  messages = [{
      "role": "user",
      "content": f"\"{params.question}\" based on:\n\n{content}"
  }]
  response = openai.ChatCompletion.create(
      model="gpt-3.5-turbo-16k-0613",
      messages=messages,
      temperature=0.5,
  )
  message = response["choices"][0]["message"]
  return json.dumps({
    "status_code": 200,
    "reason": "OK",
    "content": message["content"]
  })
```

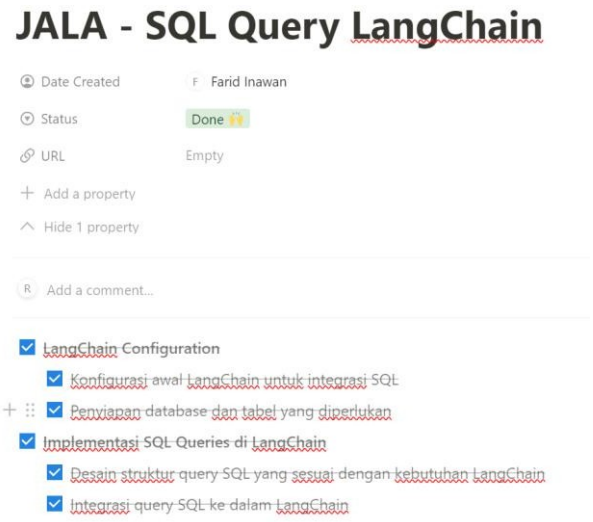
Gambar 3.17 Kode program implementasi *function browse_summarize_site*

Jika mengacu pada Gambar 3.17 tersebut, dapat dilihat bahwa kode program untuk merangkum konten dari situs web berdasarkan URL yang diberikan dan pertanyaan yang diberikan oleh pengguna. Kode ini juga menggunakan layanan OpenAI GPT-3.5 Turbo untuk menghasilkan ringkasan. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **'browse_summarize_site'**: Ini adalah fungsi yang digunakan untuk melakukan rangkuman (*summarization*) dari konten situs web berdasarkan parameter yang diberikan dalam objek **'SummarizeSiteParamsModel'**.

3.3.4 Sprint Ke-6

Sprint ke-6 berlangsung dari tanggal 14 Agustus 2023 hingga 25 Agustus 2023. Pada sprint ini, penulis memiliki tanggung jawab untuk melanjutkan pengembangan model dan melakukan integrasi dengan *database* Jala menggunakan bantuan LangChain. Dengan implementasi SQL melalui LangChain, diharapkan dapat menghubungkan dan mengelola berbagai data yang terdapat di Jala Apps secara efektif. Tugas ini menjadi kunci dalam memastikan bahwa *chatbot* yang dikembangkan dapat mengakses dan memanfaatkan informasi dari *database* Jala dengan baik.



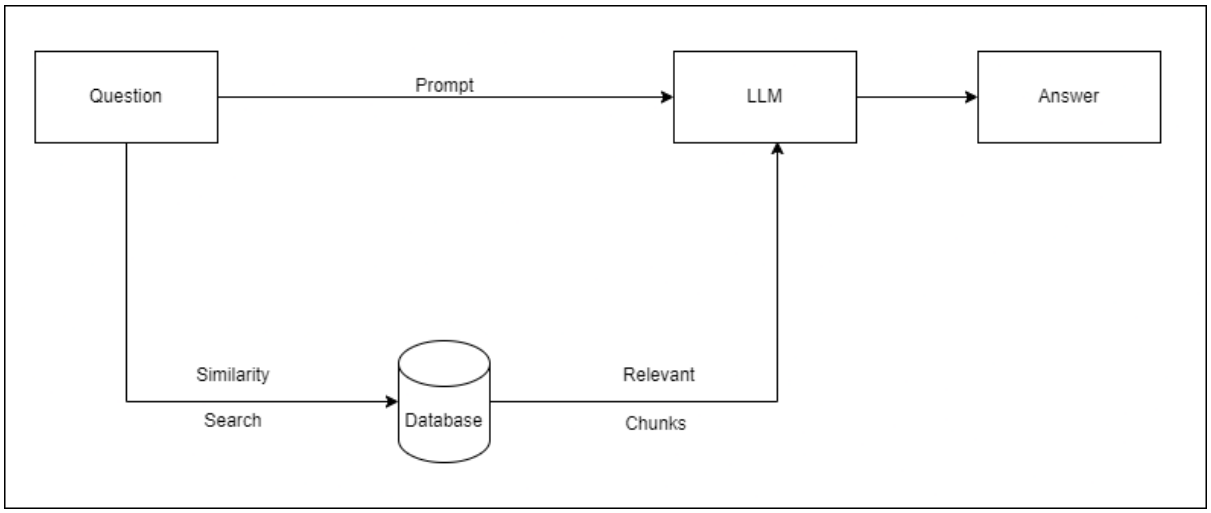
Gambar 3.18 Bukti penugasan pada Notion untuk integrasi dengan beberapa API Jala – Query

3.3.4.1 Integrasi Dengan Jala Database - Query

Setelah menyelesaikan integrasi dengan Jala API - Action, langkah selanjutnya adalah mengimplementasikan Jala API - Query dengan bantuan LangChain. Integrasi ini akan memberikan *chatbot* kemampuan multibahasa yang memungkinkannya untuk berkomunikasi dalam berbagai bahasa. Tujuannya adalah untuk memungkinkan *chatbot* mengakses *database* dengan cepat melalui agen LangChain, sehingga dapat memberikan informasi yang diinginkan oleh pengguna yang tersimpan dalam *database*.

Dengan integrasi ini, *chatbot* akan menjadi lebih adaptif terhadap preferensi bahasa pengguna, dan mampu memberikan respons yang lebih akurat dan relevan dalam bahasa yang dapat dipahami oleh pengguna. Dengan demikian, integrasi ini akan menghasilkan peningkatan yang signifikan dalam pengalaman pengguna dan membuat *chatbot* menjadi lebih efektif dalam menyediakan informasi yang dibutuhkan oleh pengguna.

Sebelum melanjutkan dengan implementasi, penulis telah melakukan diskusi bersama Farid Inawan untuk merancang alur interaksi yang optimal. Setelah berdiskusi, kami berhasil menyepakati alur interaksi yang akan kami ikuti bersama. Kami menggambarkan secara detail alur integrasi fungsi dengan Jala API - Query, sebagaimana diilustrasikan dalam Gambar 3.19.



Gambar 3.19 Diagram alur interaksi LangChain dengan *chatbot*

```
export const handler = async (req: NextApiRequest, res: NextApiResponse) =>
{
  // Inisialisasi sumber data dan database SQL
  const datasource = new DataSource({
    type: 'sql',
    database: 'jala.database.id',
  })
```

```
const db = await SqlDatabase.fromDataSourceParams({
  appDataSource: datasource,
})
// Inisialisasi toolkit SQL
const toolkit = new SqlToolkit(db)
// Inisialisasi model OpenAI
const model = new OpenAI({
  openAIApiKey: process.env.OPENAI_API_KEY,
  temperature: 0,
})

// Inisialisasi executor SQL
const executor = createSqlAgent(model, toolkit, {
  topK: 10,
  prefix: SQL_PREFIX,
  suffix: SQL_SUFFIX,
})
// Mendapatkan permintaan dari pengguna
const { query: Prompt } = req.body
// Inisialisasi respons awal
let response = {
  Prompt: Prompt,
  sqlQuery: '',
  result: [],
  error: '',
}

// Eksekusi perintah SQL
const result = await executor.call({ input: Prompt })
// Persiapan pesan-pesan yang akan digunakan dalam interaksi dengan model
let messages: ChatCompletionRequestMessage[] = [
  {
    role: 'system',
    content: SQL_PREFIX,
  },
]

let message: ChatCompletionRequestMessage = {
  role: 'user',
  content: Prompt,
```

```
  }

  messages.push(message)
```

Gambar 3.20 Kode program implementasi integrasi *database*

Jika mengacu pada Gambar 3.20 tersebut, dapat dilihat bahwa kode program untuk penanganan permintaan HTTP (*handler*) yang digunakan dalam *chatbot*. Fungsi ini menginisialisasi sumber data SQL, *database* SQL, *toolkit* SQL, model OpenAI dan *executor* SQL. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **Inisialisasi Sumber Data dan Database SQL**: Pada bagian ini, kode menginisialisasi sumber data dan koneksi ke *database* SQL Jala. Hal ini dilakukan dengan menyediakan informasi seperti jenis sumber data (SQL) dan alamat *database*, dalam contoh ini 'jala.database.id'.
- **Inisialisasi Toolkit SQL**: Setelah mendapatkan koneksi ke *database*, *toolkit* SQL diinisialisasi menggunakan objek *database* yang telah dibuat sebelumnya. *Toolkit* SQL digunakan untuk menjalankan perintah SQL di *database*.
- **Inisialisasi Model OpenAI**: Bagian ini bertanggung jawab atas inisialisasi model OpenAI yang akan digunakan untuk pemrosesan bahasa alami. Dalam hal ini, model OpenAI diatur dengan kunci API yang sesuai dan parameter temperatur.
- **Inisialisasi Executor SQL**: *Executor* SQL digunakan untuk menjalankan perintah-perintah SQL yang diberikan oleh pengguna. *Executor* ini menggunakan model OpenAI yang telah diinisialisasi sebelumnya, *toolkit* SQL, serta beberapa parameter seperti topK, prefix, dan suffix.
- **Eksekusi Perintah SQL**: Perintah SQL yang diberikan oleh pengguna dieksekusi menggunakan *executor* SQL yang telah diinisialisasi sebelumnya. Hasil eksekusi perintah SQL akan digunakan dalam pembentukan respons akhir.

```
// Iterasi melalui langkah-langkah perantara dalam hasil eksekusi
  result.intermediateSteps.forEach((step: any) => {
    if (step.action.tool === 'query-sql') {
      response.sqlQuery = step.action.toolInput
      response.result = JSON.parse(step.observation)

      // Menambahkan perintah SQL ke dalam pesan
```

```
      messages.push({
        role: 'assistant',
        content: response.sqlQuery,
        function_call: {
          name: 'executorSQL',
          arguments: JSON.stringify({
            sqlQuery: response.sqlQuery,
            sqlResult: JSON.stringify(response.result),
          }),
        },
      })
    }
  })

  try {
    // Interaksi dengan model bahasa GPT-3.5 Turbo
    let sqlresult: any = await openai.createChatCompletion({
      model: 'gpt-3.5-turbo-16k-0613',
      messages: messages,
      temperature: 0.25,
      function: generateFunctionArray(),
      function_call: 'auto',
    })
    // Melanjutkan interaksi dengan model sampai selesai
    while (sqlresult.data.choices[0]?.finish_reason === null) {
      const message: ChatCompletionRequestMessage = {
        role: 'user',
        content: sqlresult.data.choices[0]?.message?.content,
      }
      messages.push(message)
      if (sqlresult.data.choices[0]?.finish_reason === 'function_call') {
        try {
          const funcName =
            sqlresult.data.choices[0]?.message?.function_call?.name
          if (funcName) {
            const func = functionMap[funcName]
            const params = func({
              ...JSON.parse(

sqlresult.data.choices[0]?.message?.function_call?.arguments ??
```

```
                '{}'
            ),
          })
          const function_response = await func(params)
```

Gambar 3.21 Kode program implementasi iterasi pada LangChain

Jika mengacu pada Gambar 3.21 tersebut, dapat dilihat digunakan untuk berinteraksi dengan API OpenAI, khususnya dengan model bahasa GPT-3.5 Turbo, untuk menghasilkan respons teks berdasarkan interaksi dengan model tersebut. Berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **Menambahkan Pesan Hasil Fungsi ke dalam Interaksi**
  - Kode dimulai dengan menambahkan pesan hasil fungsi ke dalam variabel ‘messages’. Pesan ini digunakan sebagai interaksi awal dengan model.
  - Setiap pesan memiliki beberapa atribut, seperti *role* yang menyatakan peran pesan (dalam konteks ini, '*function*'), name yang merupakan nama fungsi atau sumber pesan, dan *content* yang berisi teks respons dari fungsi atau pesan tersebut.
  - Pesan ini ditambahkan ke dalam array ‘messages’.
- **Melanjutkan Interaksi dengan Model**
  - Setelah pesan hasil fungsi atau pesan kesalahan ditambahkan ke dalam ‘messages’, program melanjutkan interaksi dengan model GPT-3.5 Turbo menggunakan pesan-pesan yang telah ditambahkan.
  - Model dijalankan dengan menggunakan metode ‘openai.createChatCompletion()’ dengan beberapa parameter seperti model yang digunakan, pesan-pesan yang telah disiapkan, suhu (*temperature*) yang mengatur seberapa kreatif respons model, dan sebagainya.

```
//Menambahkan pesan hasil fungsi ke dalam interaksi
            const functionMessage: ChatCompletionRequestMessage = {
              role: 'function',
              name: funcName,
              content: function_response,
            }
            messages.push(functionMessage)
          }
        } catch (e) {
```

```
          const error = e as Error
          const functionMessage: ChatCompletionRequestMessage = {
            role: 'function',
            name: sqlresult.data.choices[0]?.message?.function_call?.name,
            content: JSON.stringify({
              status_code: 422,
              reason: error.name,
              content: error.message,
            }),
          }
          messages.push(functionMessage)
        }
        // Melanjutkan interaksi dengan model berdasarkan pesan-pesan yang
ditambahkan
        sqlresult = await openai.createChatCompletion({
          model: 'gpt-3.5-turbo-16k-0613',
          messages: messages,
          temperature: 0.25,
          function: generateFunctionArray(),
          function_call: 'auto',
        })
      } else {
        break
      }
    }
```

Gambar 3.22 Kode program implementasi *Function Call* pada LangChain

Jika mengacu pada Gambar 3.22 tersebut, dapat dilihat digunakan untuk menambahkan pesan hasil fungsi atau pesan kesalahan ke dalam interaksi dengan model, kemudian melanjutkan interaksi berdasarkan pesan-pesan yang telah ditambahkan. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **Menambahkan Pesan Hasil Fungsi ke dalam Interaksi**
  - Kode ini memulai dengan menciptakan sebuah objek *'functionMessage'* yang merupakan pesan hasil fungsi. Pesan ini akan ditambahkan ke dalam interaksi dengan model.
  - **'*function*Message'** memiliki tiga atribut:
    - **'role'**: Menyatakan peran pesan, dalam hal ini, '*function*'.

- **‘name’**: Nama fungsi atau sumber pesan. Nama ini diberikan oleh variabel funcName.
- **‘content’**: Isi pesan yang berisi respons dari fungsi yang dijalankan sebelumnya, diberikan oleh variabel *function_response*.

o Pesan ini kemudian ditambahkan ke dalam array *‘messages’*.

- **Melanjutkan Interaksi dengan Model Berdasarkan Pesan-pesan yang Ditambahkan**
  - Setelah pesan hasil fungsi atau pesan kesalahan ditambahkan ke dalam array *‘messages’*, program melanjutkan interaksi dengan model GPT-3.5 Turbo.
  - Model dijalankan menggunakan metode openai.createChatCompletion(). Dalam pemanggilan ini, pesan-pesan yang telah ditambahkan ke dalam array *‘messages’* digunakan sebagai konteks interaksi dengan model.
  - Parameter lainnya yang digunakan termasuk model yang digunakan, suhu (*temperature*) yang mengatur seberapa kreatif respons model, dan fungsi-fungsi tambahan yang mungkin diperlukan yang dihasilkan oleh generate*Function*Array().

```
    const finalresponse = await openai.createChatCompletion({
      model: 'gpt-3.5-turbo-16k-0613',
      messages: messages,
      temperature: 0.8,
      function: generateFunctionArray(),
      function_call: 'auto',
    })

    const responseMessage = finalresponse.data.choices[0]?.message
    if (responseMessage) {
      const formattedContent = responseMessage?.content?.replace(/\n/g, '')
      responseMessage.content = formattedContent
    }

    const customOutput = responseMessage || {}
    // Mengirimkan respons akhir sebagai JSON ke klien
    res.status(200).json({ output: customOutput })
  } catch (error) {
    // Penanganan kesalahan

    console.error('Error calling OpenAI API:', error)
    res
```

```
      .status(500)
      .json({ error: 'An error occurred while processing the request.' })
  }
```

Gambar 3.23 Kode program implementasi final response LangChain

Jika mengacu pada Gambar 3.23 tersebut, dapat dilihat digunakan untuk mendapatkan respons teks dari model bahasa GPT-3.5 Turbo dan mengirimkannya sebagai respons JSON ke klien melalui HTTP. Kemudian berikut merupakan uraian penjelasan dari beberapa blok kode tersebut.

- **Mendapatkan Respons dari Model**
  - Kode ini menggunakan metode openai.createChatCompletion() untuk mengambil respons akhir dari model GPT-3.5 Turbo.
  - Beberapa parameter yang digunakan dalam pemanggilan ini termasuk model yang digunakan (dalam hal ini, 'gpt-3.5-turbo-16k-0613'), pesan-pesan yang telah dihasilkan sepanjang interaksi yang disimpan dalam array messages, suhu (temperature) yang mengatur seberapa kreatif respons model, fungsi-fungsi tambahan yang mungkin diperlukan yang dihasilkan oleh generate*Function*Array(), dan *function*_call yang diatur ke 'auto'.
- **Membuat Objek JSON untuk Respons Akhir**
  - Setelah mengolah respons akhir, kode membuat sebuah objek JSON yang disebut *customOutput*. Jika *responseMessage* tidak terdefinisi (null atau *undefined*), maka *customOutput* akan menjadi objek kosong {}. Jika respons Message terdefinisi, *customOutput* akan mengambil nilai dari *responseMessage*.
  - Terakhir, kode mengirimkan respons akhir sebagai respons JSON ke klien melalui protokol HTTP. Respons ini dikirim dalam bentuk objek JSON dengan kunci output, yang berisi customOutput.

3.4 Sprint Retrospective

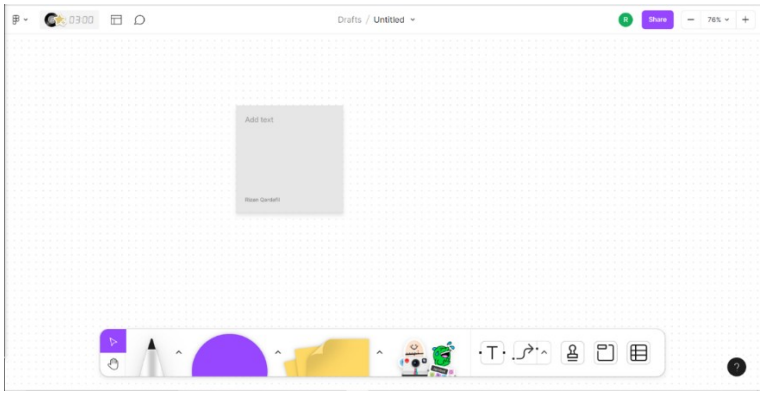
Pada hari terakhir setiap *sprint*, tim melaksanakan sesi *sprint retrospective*, yang merupakan bagian penting dari metodologi pengembangan. Sesi ini mencakup beberapaelemen utama, termasuk penilaian kinerja individu dalam tim, diskusi tentang tantangan yangdihadapi selama proses implementasi, dan perencanaan strategi untuk meningkatkan efektivitas pelaksanaan di *sprint-sprint* berikutnya (Grebić & Stojanović, 2021). *Sprint retrospective*

adalah momen kolaboratif di mana setiap anggota tim berpartisipasi aktif, bersama-sama mengatasi kendala yang mereka alami, baik dalam kerja tim maupun secara pribadi.

Selama sesi *retrospective*, tim melakukan evaluasi mendalam terhadap pencapaian mereka selama *sprint* yang berlangsung. Mereka mengidentifikasi tindakan yang telah membawa hasil positif dan harus diulang di masa depan untuk mempertahankan kinerja yang baik. Selain itu, tim juga mengenali tindakan yang tidak berjalan dengan baik selama *sprint* ini, yang akan dijadikan bahan evaluasi dan dihindari dalam *sprint-sprint* berikutnya. Terakhir, tim mencari peluang untuk mengidentifikasi tindakan baru yang sebelumnya belum dieksplorasi namun memiliki potensi untuk meningkatkan kinerja tim di masa depan.

Sprint *retrospective* dilakukan 6 kali setelah sprint *review.* Sprint *retrospective* pertama membawa solusi untuk masalah yang muncul pada saat sprint pertama. Memahami proses model dan *architecture* yang cukup rumit dan kompleks mengakibatkan penulis membuahkan hasil yang kurang optimal, sehingga menjadi *overtime* pada sprint pertama berlangsung. Untuk meningkatkan kualitas dan efisiensi, pada sprint *retrospective* menyimpulkan bahwa masuk pada sprint 2, aktivitas brainstorming dilakukan maksimal dua kali dalam satu hari. Hal ini terbukti efektif sehinggga tidak ada pencatatan dari kegiatan sprint *retrospective* kedua sampai keenam (terakhir).

Penting untuk dicatat bahwa di Jala, proses *sprint retrospective* menjadi lebih efisien dan berkolaborasi berkat penggunaan Figjam, sebuah fitur yang terintegrasi dengan aplikasi web Figma. Figjam memungkinkan anggota tim untuk berpartisipasi dalam sesi *retrospective* secara virtual, menciptakan suasana seperti mereka sedang berdiskusi di sekitar papan tulis secara langsung. Gambaran antarmuka Figjam yang diilustrasikan dapat dilihat pada Gambar 3.24 dalam laporan. Ini adalah contoh bagaimana teknologi dapat digunakan untuk meningkatkan kolaborasi dan efisiensi dalam proses pengembangan perangkat lunak.



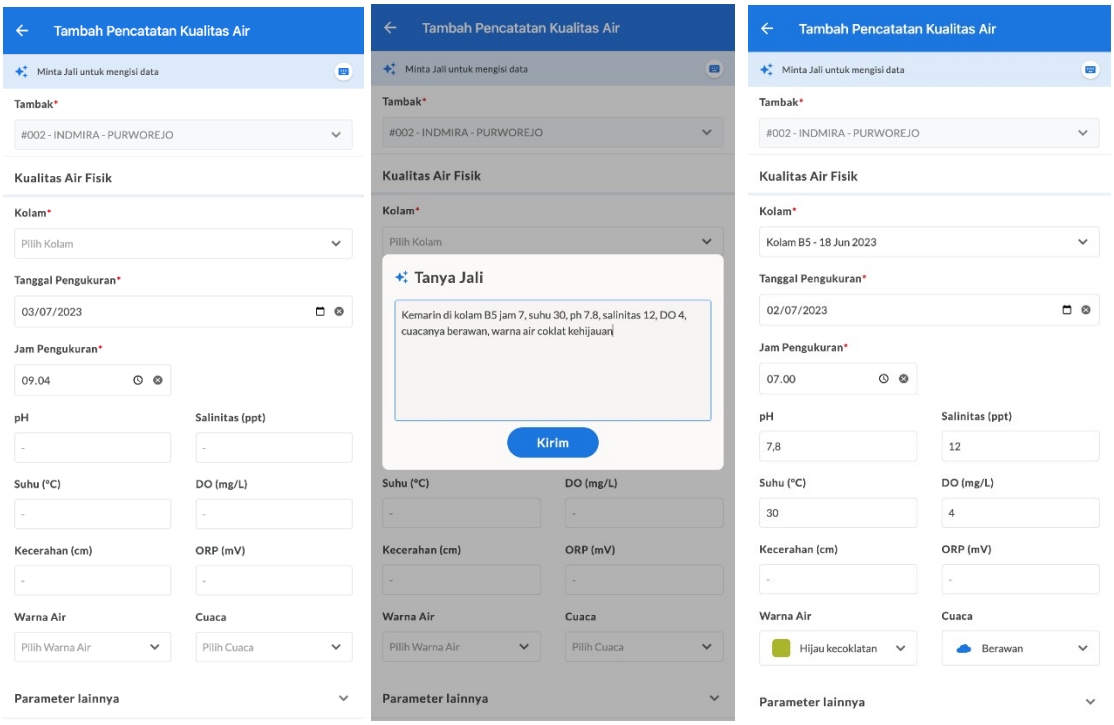
Gambar 3.24 Tampilan antarmuka *Figjam*

3.5 Dampak Implementasi

Implementasi yang telah dilakukan oleh pemagang telah dicatat oleh *product manager*. Sebelum masuk ke tahap produksi, produk ini akan diuji coba secara beta oleh pengguna terlebih dahulu. Hasil wawancara dengan *principal engineer* Jala Tech, Farid Inawan, dan *product manager*, Wildan Avian Pratama, menunjukkan bahwa *chatbot* yang telah dikembangkan menggabungkan tiga teknologi yang berbeda dengan sangat baik. Hal ini memungkinkan chatbot untuk secara otomatis *filling form*, *create data*, dan *query data*. Oleh karena itu, pengembangan ini sangat bermanfaat bagi pengguna dalam meningkatkan produktivitas mereka dalam pengelolaan tambak.

3.5.1 Filling Form

Hasil dari implementasi Chat GPT, Auto-GPT, dan LangChain pada *chatbot* Jala memungkinkan *chatbot* dapat mengisi *form* yang ada berdasarkan teks yang diberikan oleh pengguna secara otomatis.



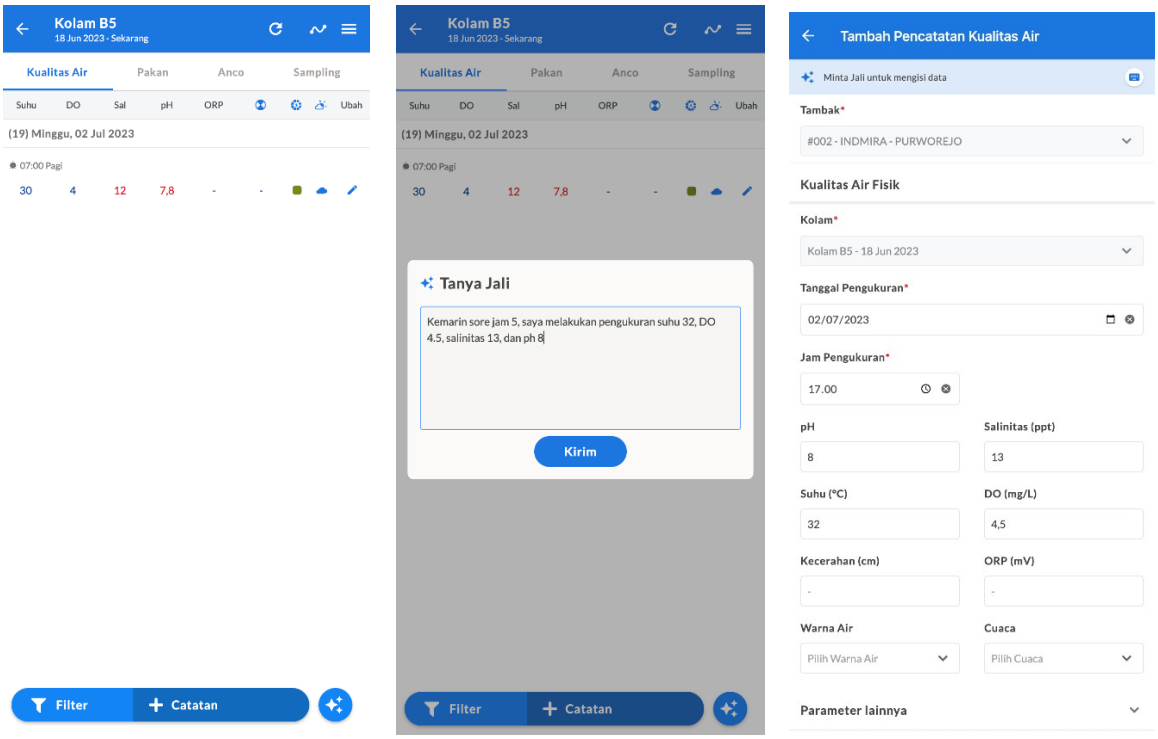
Gambar 3.25 Tampilan *filling form* pada aplikasi Jala

Berdasarkan Gambar 3.25, pencatatan otomatis pada *form* yang terdapat dalam aplikasi Jala dapat dilakukan secara otomatis dengan menggunakan informasi yang diperoleh dari teks yang disediakan oleh pengguna. *Chatbot* dapat menganalisis teks yang diberikan oleh pengguna serta mengidentifikasi elemen-elemen informasi yang relevan, dan secara otomatis memasukkan data tersebut ke dalam *form* yang tepat dalam aplikasi Jala. Dengan kata lain,

chatbot bertindak sebagai asisten yang efisien dalam mengisi *form* berdasarkan informasi yang berikan oleh pengguna. Menurut Farid Inawan, kemampuan *chatbot* untuk mengisi data secara akurat pada *form* berdasarkan informasi dari teks pengguna sangat bermanfaat bagi pengguna yang sering merasa malas mengisi banyak *form* dalam manajemen tambak udang. Ini tidak hanya meningkatkan efisiensi dalam manajemen tambak udang, tetapi juga mempercepat prosesnya secara signifikan.

3.5.2 Create Data

Hasil dari implementasi Chat GPT, Auto-GPT, dan LangChain pada *chatbot* Jala, hasilnya adalah kemampuan *chatbot* dapat secara otomatis membuat dan memasukkan data baru berdasarkan teks yang diberikan oleh pengguna.



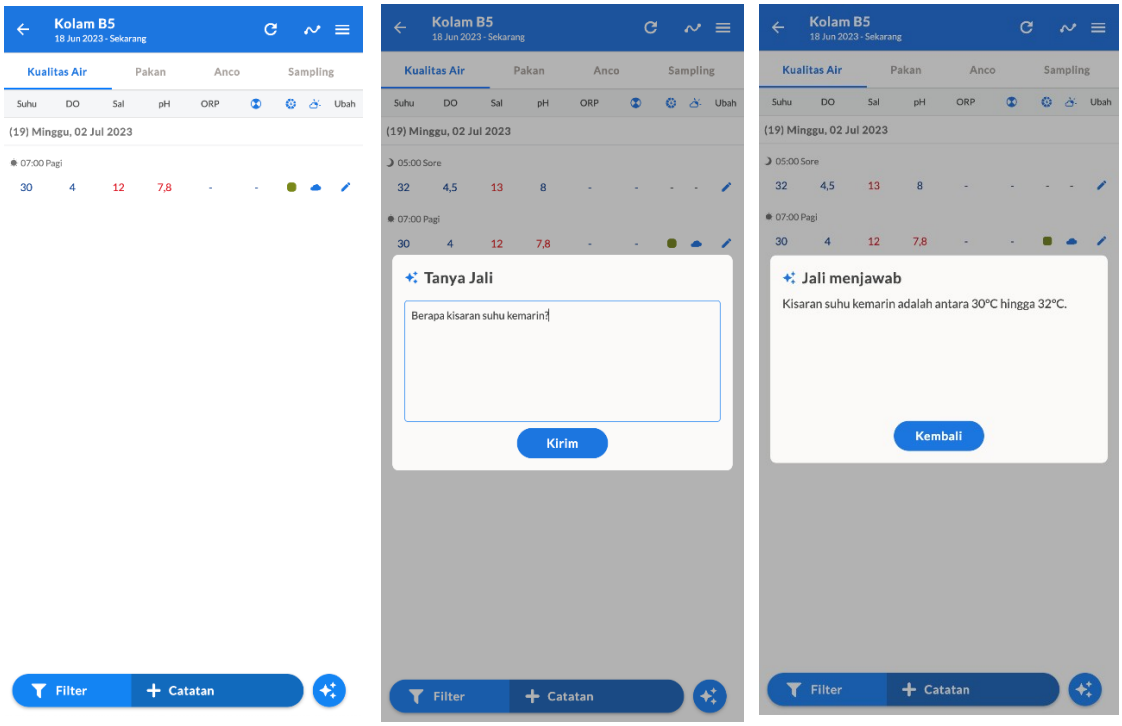
Gambar 3.26 Tampilan *create data* pada aplikasi Jala

Berdasarkan Gambar 3.26, terlihat bahwa pembuatan data secara otomatis dan pencatatan otomatis dalam aplikasi Jala dapat terjadi dengan menggunakan informasi yang diperoleh dari teks yang diberikan oleh pengguna. *Chatbot* akan mengolah teks yang diberikan oleh pengguna dan akan memasukkannya ke dalam layanan yang sesuai secara otomatis. Menurut Farid Inawan, kemampuan *chatbot* untuk secara otomatis memasukkan data ke dalam layanan yang sesuai sangat membantu pengguna yang seringkali harus mencari layanan yang cukup banyak

dalam aplikasi Jala. Dengan *chatbot*, pengguna hanya perlu memberikan informasi melalui teks, dan data tersebut akan masuk ke dalam layanan yang sesuai serta diinputkan secara otomatis. Ini tidak hanya meningkatkan efisiensi dalam manajemen tambak udang, tetapi juga signifikan dalam mempercepat prosesnya.

3.5.3 Query Data

Hasil dari implementasi Chat GPT, Auto-GPT, dan LangChain pada *chatbot* Jala, hasilnya adalah kemampuan *chatbot* dapat secara otomatis membuat *query* dan menampilkan informasi tersebut kepada pengguna berdasarkan teks informasi yang berikan oleh pengguna.



Gambar 3.27 Tampilan *query data* pada aplikasi Jala

Berdasarkan Gambar 3.27, terlihat bahwa proses *query* data dilakukan secara otomatis dan menghasilkan informasi yang kemudian ditampilkan kepada pengguna. Menurut Wildan Avian Pratama, bahwa kemampuan untuk membuat *query* secara otomatis berdasarkan teks yang diberikan oleh pengguna sangat memudahkan pengguna dalam mencari informasi yang relevan, termasuk data dari tambak pada hari sebelumnya.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Selama magang di Jala Tech, penulis menghadapi situasi penerapan teori yang telah dipelajari membutuhkan adaptasi karena keadaan yang kurang ideal. Pengalaman ini menggaris bawah fakta bahwa aplikasi di dunia nyata tidak selalu selaras dengan hasil teori ideal yang disampaikan dalam perkuliahan. Banyak kejadian tak terduga yang dapat terjadi, seperti tantangan tak terduga yang muncul saat mengerjakan proyek dengan menggunakan metodologi pengembangan *scrum*.

4.1.1 Metodologi *Scrum*

Pengerjaan proyek *chatbot* dilakukan dengan menerapkan metodologi pengembangan agile, khususnya menggunakan *framework scrum*. *Scrum* adalah salah satu kerangka kerja yang membantu dalam pengembangan proyek dengan sejumlah manfaat yang signifikan. Salah satu keuntungan utama dari menerapkan *scrum* adalah struktur yang lebih baik dalam pengerjaan tugas. Dengan memulai proyek dengan pembentukan *product backlog*, tim dapat mengalokasikan tugas dengan lebih terarah, memprioritaskan pekerjaan yang perlu diselesaikan terlebih dahulu.

Selain itu, penerapan *scrum* juga memberikan manfaat dari segi kolaborasi tim. Melalui transparansi dan fleksibilitas yang diberikan oleh *scrum*, tim dapat berkomunikasi dengan lebih baik. *Daily scrum*, misalnya, memungkinkan anggota tim untuk mengetahui pekerjaan yang telah dilakukan sebelumnya dan mengatasi kendala yang mungkin muncul selama pelaksanaan tugas. Ini memungkinkan tim untuk saling membantu mengatasi kendala dan membuat penyesuaian jadwal jika diperlukan, serta merencanakan pekerjaan yang akan dilakukan pada *sprint* berikutnya.

Penerapan kerangka kerja *scrum* dalam pemilihan peran pada proyek *chatbot* yang dilakukan oleh penulis telah sesuai dengan konsep yang telah dijelaskan sebelumnya. Tim proyek melibatkan peran seperti *product owner*, *scrum master*, dan *developer*. Selama tahap pengerjaan proyek, penulis menerapkan langkah-langkah *scrum* yang mencakup inisiasi, perencanaan dan estimasi, ulasan, dan rilis. Pendekatan ini membantu mengatur pekerjaan dengan lebih baik karena proses *scrum* dimulai dengan fase inisialisasi, kebutuhan proyek ditentukan dengan jelas. Dengan memiliki pemahaman yang baik tentang kebutuhan proyek

dari awal, tim dapat lebih mudah menjalankan proses pengerjaan karena mereka memiliki panduan yang kuat untuk melanjutkan pekerjaan.

Untuk memastikan pembelajaran yang efektif, perlu ditekankan pentingnya edukasi dan praktik dalam penggunaan *framework scrum* dalam setiap mata kuliah yang melibatkan proyek pengembangan. Hal ini disebabkan oleh prevalensi yang tinggi penggunaan *scrum* dalam industri IT dan sejumlah manfaat yang signifikan yang dapat diperoleh dari penerapannya. Menurut penulis, manfaat dari penggunaan *scrum* termasuk pelatihan dalam kolaborasi, pengembangan kreativitas dalam merancang strategi untuk menyelesaikan tugas, dan sejumlah manfaat lainnya yang dapat dijelaskan lebih lanjut.

4.1.2 Pengujian Sistem Secara Manual

Dalam proyek pengembangan *chatbot* Jala, tim *quality assurance* melakukan pengujian dengan metode manual dan independen, karena tidak ada teknologi pengujian otomatis yang cocok untuk digunakan dalam penilaian *chatbot* Jala. Selama penerapan sistem *Work From Office* (WFO), penulis harus berkolaborasi secara langsung dengan tim *quality assurance* untuk mengatasi masalah yang muncul dalam proses pengujian.

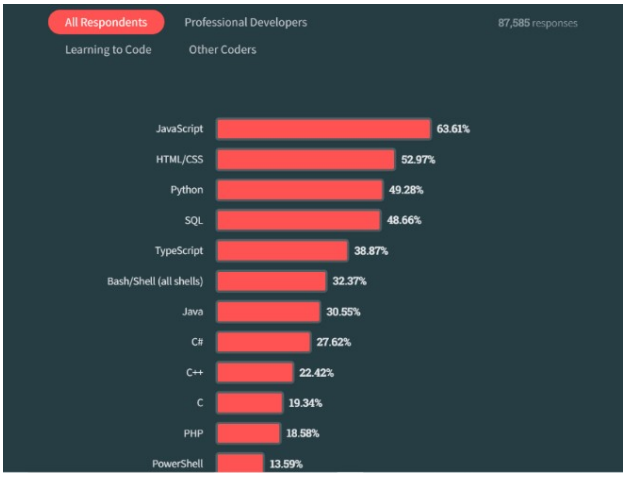
Selama fase akademis, fokus penulis terutama pada aspek-aspek lain di bidang teknologi, seperti pemrograman dan pengembangan perangkat lunak. Akibatnya, penulis tidak memiliki kesempatan untuk mendalami pengujian manual sebagai bagian dari kurikulum. Namun, ketika penulis memulai magang di Jala Tech, hal ini menandai pengalaman pertama dalam memahami dan menjalankan prosedur pengujian manual.

4.1.3 Penerapan Chat GPT, Auto-GPT, dan LangChain Pada *Chatbot* Jala

Sejalan dengan teori yang sudah ada sebelumnya, integrasi teknologi Chat GPT, Auto-GPT, dan LangChain ditujukan untuk meningkatkan kualitas *chatbot* Jala secara keseluruhan. Teknologi-teknologi ini ketika diterapkan pada *chatbot* Jala, memiliki potensi untuk meningkatkan kemampuan interaksinya dan kualitas layanan yang diberikannya. Secara khusus, Chat-GPT dapat berfungsi sebagai kecerdasan inti dari *chatbot*, Auto-GPT dapat meningkatkan kemampuan beradaptasi pengguna, dan pemanfaatan LangChain memfasilitasi penggabungan berbagai teknologi AI untuk menawarkan pengalaman pengguna yang lebih baik.

Selama masa perkuliahan, penulis diperkenalkan dengan bahasa pemrograman *JavaScript* dalam konteks mata kuliah Pengembangan Aplikasi Berbasis Web. *JavaScript* ditekankan sebagai komponen dasar, yang sering disebut sebagai "otak", dalam arsitektur aplikasi web. Meskipun *JavaScript* dapat digunakan secara independen dengan *Node.js* untuk

berbagai tujuan, termasuk pengembangan game dengan *Unity*, penulis juga diperkenalkan dengan bahasa pemrograman *Python* dalam mata kuliah Pemrosesan Bahasa Alami. *Python* terkenal dengan kemampuannya yang luar biasa dalam pemrosesan bahasa alami dan aplikasi kecerdasan buatan. Oleh karena itu, eksplorasi lebih lanjut terhadap *JavaScript* dan *Python* dianggap penting, mengingat posisi mereka yang menonjol dalam lanskap pemrograman. Patut dicatat bahwa kedua bahasa pemrograman ini memiliki peringkat yang signifikan dibandingkan dengan bahasa lain seperti *Java*, *C++*, dan lainnya, seperti yang diilustrasikan dalam grafik tren perkembangan bahasa pemrograman dari survei *Stack Overflow* 2023 (lihat Gambar 4.1).



Gambar 4.1 Grafik bahasa pemrograman terpopuler dalam survey *stackoverflow* 2023

Sumber: (Stack Overlow, 2023)

4.2 Pembelajaran Magang

Magang merupakan langkah yang berharga dalam meningkatkan kapabilitas seseorang dalam mempersiapkan dan membentuk jalur karir yang diinginkan. Ini disebabkan oleh kenyataan bahwa magang memberikan peluang kepada pesertanya untuk menerapkan pengetahuan teoritis yang telah mereka peroleh ke dalam situasi dunia nyata. Selama menjalani pengalaman magang, penulis memperoleh berbagai pengalaman dan manfaat, baik dalam aspek teknis maupun non-teknis. Beberapa dari manfaat ini akan diuraikan lebih lanjut di bawah ini.

4.2.1 Teknis

Selama periode magang di Jala Tech, penulis merasakan banyak keuntungan, terutama dalam mendapatkan pemahaman yang lebih dalam tentang tahapan pengembangan produk dalam industri teknologi informasi dan mendapatkan pengalaman langsung dalam

pemrograman menggunakan bahasa *Python* dan *JavaScript*. Manfaat ini sebagian besar merupakan hasil dari pembelajaran mandiri. Namun, penting untuk ditekankan bahwa pengetahuan teoritis yang diperoleh selama perkuliahan tetap sangat relevan dan berharga dalam konteks praktis kerja magang.

Selama fase akademis, mata kuliah Pengembangan Aplikasi Berbasis Web menarik minat penulis dalam bidang pemrograman. Pengembangan web, seperti yang dirasakan oleh penulis, memiliki daya pikat yang unik, terutama dalam bidang pengembangan *frontend*. Domain ini melibatkan penerjemahan konsep desain ke dalam antarmuka aplikasi web yang nyata, bersama dengan orkestrasi interaksi pengguna. Selain itu, *frontend* memainkan peran penting dalam penataan distribusi data, memungkinkan *backend* untuk memproses dan menyajikan informasi dalam aplikasi web.

Perjalanan pengembangan aplikasi dieksplorasi lebih lanjut dalam mata kuliah Pengembangan Sistem Informasi, yang tidak hanya mempelajari proses pengembangan tetapi juga memberikan wawasan dalam merancang proses bisnis dari perspektif perusahaan. Pemahaman penulis tentang proses bisnis spesifik perusahaan semakin diperdalam melalui magang di Jala Tech, sebuah organisasi yang berspesialisasi dalam pengembangan dan pemeliharaan produk. Untuk manfaat lebih teknis dan wawasan tambahan.

A. Pemahaman Pemrograman Dengan JavaScript dan Python

Melalui partisipasi aktif dalam proyek *chatbot* Jala, penulis mendapatkan wawasan yang luas tentang pemrograman dengan *JavaScript* dan *Python*. Khususnya, pengalaman belajar ini mencakup pemanfaatan teknologi seperti *React, Vue.js, Flask*, dan banyak lagi. Pengembangan *chatbot* Jala terutama mengandalkan *JavaScript* yang diimplementasikan dengan *Node.js*, yang memfasilitasi pemrograman mandiri. Selain itu, *Python* digunakan untuk mengintegrasikan *chatbot* dengan Chat-GPT, Auto-GPT, dan LangChain. Akibatnya, upaya pemrograman ini berbeda dari praktek pemrograman web konvensional yang sebagian besar memanfaatkan *JavaScript*.

Sepanjang perjalanan ini, penulis berhasil memperoleh kemahiran dalam konsep dasar dan lanjutan dalam *JavaScript* dan *Python*. Hal ini mencakup aspek-aspek seperti manipulasi data dalam larik dan objek, serta pemanfaatan operator terner yang mahir untuk mewujudkan implementasi kondisi minimal.

B. Menggunakan Multiple Promise

JavaScript menggabungkan fitur yang dikenal sebagai konkurensi, yang memungkinkan eksekusi kode terjadi secara berurutan. Salah satu kelas *JavaScript* yang mampu mencapai

concurrency adalah kelas Promise. Promise memungkinkan pengembang untuk mengeksekusi kode secara *asynchronous*, terutama untuk tugas-tugas yang melibatkan operasi yang memakan waktu. Promises memberikan hasil setelah eksekusi selesai.

Secara *default*, ketika kode yang menggunakan promise dieksekusi, hal ini terjadi secara bersamaan, sehingga memungkinkan kode berikutnya untuk berjalan tanpa menunggu kode promise selesai. Namun, jika ada persyaratan untuk mengeksekusi kode hanya setelah promise menyelesaikan tugasnya, kata kunci "await" ikut berperan. Penggunaan "await" memastikan bahwa kode berikutnya menunggu sampai eksekusi promise selesai sebelum melanjutkan.

Selain itu, *JavaScript* menyediakan kemampuan untuk mengatur eksekusi kode secara bersamaan dan berurutan secara bersamaan melalui pemanfaatan fungsi Promise.all(). Untuk mengimplementasikan hal ini, fungsi Promise.all() menerima larik sebagai argumennya, dengan setiap elemen larik merepresentasikan sebuah await. Ketika dipanggil, Promise.all() akan memberikan hasil setelah semua await dalam argumen berhasil menyelesaikan eksekusi. Hasilnya, semua kode janji yang ditentukan dalam argumen akan dieksekusi secara paralel. Namun, kode berikutnya setelah pemanggilan Promise.all() akan dieksekusi hanya setelah Promise.all() menyelesaikan tugasnya. Hal ini dicapai dengan menggunakan kata kunci "await" sebelum kode berikutnya. Contoh ilustrasi penggunaan Promise.all() disediakan pada Gambar 4.2.

```
await Promise.all([
    fetchFarms(),
    fetchPonds(),
    fetchFinances()
)]
```

Gambar 4.2 Penggunaan Promise.all()

C. Menggunakan High Order *Function*

High Order Function dalam *JavaScript* adalah jenis fungsi yang mampu menerima fungsi lain sebagai argumen atau mengembalikan fungsi sebagai hasil. Di antara fungsi-fungsitingkat tinggi yang umum digunakan dalam *JavaScript* adalah map, filter, dan reduce.

Fungsi map secara khusus digunakan untuk mengubah struktur larik. Fungsi ini membutuhkan sebuah fungsi sebagai argumen, yang harus mengembalikan nilai dari semua tipe data. Fungsi ini diterapkan berulang kali pada setiap elemen dalam larik, dan nilai yang dihasilkan dari eksekusi fungsi tersebut dikumpulkan menjadi larik baru, membentuk hasil akhir. Contoh yang mengilustrasikan penggunaan fungsi map dapat ditemukan pada Gambar 4.3.

```
const names = ['rizan', 'rifa', 'qardafil', 'husniyyah'];

const namesWithIndex = names.map(function (element, index) {
    return `${index}. ${element}`;
});
console.log(namesWithIndex);
// Hasil: ['0. rizan', '1. rifa', '2. qardafil', '3. husniyyah']
```

Gambar 4.3 Penggunaan *function* map

Fungsi filter digunakan untuk menyaring larik dan mempertahankan elemen yang memenuhi kriteria tertentu. Fungsi ini menerima fungsi sebagai argumennya, yang seharusnya mengembalikan nilai boolean. Elemen yang bernilai salah saat diproses oleh fungsi ini akan dihilangkan dari larik yang dihasilkan, yang dibuat oleh fungsi filter. Anda dapat menemukan contoh yang menampilkan penggunaan fungsi filter pada Gambar 4.4.

```
// Hasil: [‘0. rizan', ‘1. rifa', ‘2. qardafil', ‘3. husniyyah']
const names = ['rizan', 'rifa', 'qardafil', 'husniyyah'];

const namesWithoutRizan = names.filter(function (element) {
    return element !== 'rizan';
});
console.log(namesWithoutRizan);
// Hasil: [‘rifa', ‘qardafil', ‘husniyyah']
```

Gambar 4.4 Penggunaan *function* filter.

Fungsi kurangi digunakan untuk melakukan operasi pada setiap elemen dalam larik. Tidak seperti fungsi map dan filter, fungsi kurangi menghasilkan nilai keluaran tunggal. Kode yang akan dieksekusi untuk setiap elemen ditentukan dalam parameter fungsi reduce. Sebuah contoh yang menunjukkan penggunaan fungsi reduce digambarkan pada Gambar 4.5.

```
const numbers = [1, 2, 3, 4, 5];
const accumulation = numbers.reduce(function (acc, cur) {
    return acc + cur;
});
console.log(accumulation);
// Hasil: 15
```

Gambar 4.5 Penggunaan *function* reduce.

4.2.2 Non Teknis

Selain memberikan manfaat dalam aspek teknis, magang juga memiliki nilai yang signifikan dalam pengembangan sisi pribadi, yang memiliki peran penting dalam konteks pekerjaan sehari-hari. Proses pengembangan diri ini berkontribusi dalam membentuk kepribadian yang positif dan memungkinkan individu untuk beradaptasi dengan baik dalam situasi atau lingkungan yang baru. Selain itu, hal ini juga berdampak positif dalam memperkuat dan memfasilitasi kerjasama yang efisien dengan rekan kerja. Selama masa magang di Jala Tech, penulis telah merasakan berbagai manfaat dan pengalaman yang diharapkan akan membantu penulis menjadi seorang profesional yang lebih kompeten. Manfaat ini akan dijelaskan lebih lanjut dalam paragraf berikutnya.

A. Mendapatkan Pengalaman Bekerja Secara Profesional

Salah satu hal yang tidak dapat diperoleh diperoleh melalui pendidikan di bangku kuliah adalah pengalaman dalam bekerja secara professional. Ketika menjalani magang di Jala Tech, harapannya adalah individu dapat menjalankan tugas sesuai dengan standar professional yang berlaku dalam pekerjaan yang diberikan. Pengalaman ini tidak hanya memberikan pemahaman tentang aspek praktis dalam dunia kerja, tetapi juga mengajarkan nilai-nilai seperti integritas, tanggung jawab, dan disiplin dalam lingkungan kerja yang sesungguhnya.

Penulis memiliki kesempatan untuk mengaplikasikan pengetahuan teoritis yang telah mereka pelajari selama kuliah dalam situasi dunia nyata. Dalam konteks kerja tim, kemampuan untuk berkolaborasi dan berkomunikasi dengan rekan kerja sangat penting untuk mencapai tujuan bersama. Penulis bekerja sama dengan pengembang lain, desainer produk, dan manajemen, yang membantu meningkatkan keterampilan kolaborasi dan komunikasi penulis.

Pengalaman magang juga memberikan wawasan tentang dinamika organisasi dan budaya perusahaan yang mungkin berbeda dari lingkungan akademis. Hal ini membantu pemagang untuk memahami berbagai faktor yang mempengaruhi produktivitas dan kolaborasi di tempat kerja. Dengan demikian, meskipun pendidikan formal memberikan dasar yang kuat, pengalaman magang di Jala Tech memberikan lapisan tambahan dalam pembentukan seorang profesional yang kompeten dan siap menghadapi tantangan dalam dunia kerja.

Dalam konteks Scrum di Jala Tech, perlu dicatat bahwa setiap *sprint* tidak hanya melibatkan pemilihan task dan bobot tugas yang tetap dan seragam bagi setiap pengembang. Sebaliknya, fleksibilitas diberikan kepada setiap *programmer* untuk memilih tugas dan menentukan bobotnya sendiri, tergantung pada kemampuan dan keahlian individu. Hal ini menciptakan dinamika berbeda di setiap *sprint,* di mana para *programmer* memiliki kebebasan untuk menilai kompleksitas suatu *task* dan menentukan bobot tugas yang sesuai dengan kemampuan. Keberagaman ini tidak hanya mencerminkan kompleksitas pekerjaan yang dihadapi, tetapi juga menciptakan kesempatan bagi setiap *programmer* untuk tumbuh dan

mengasah keterampilan mereka dalam area spesifik.

Sejalan dengan semangat kolaboratif Scrum, tim proyek dan *Product Owner* bersama-sama berpartisipasi dalam diskusi mengenai alokasi tugas dan bobotnya, dengan mempertimbangkan kompleksitas dan urgensi setiap elemen *backlog.* Adanya perbedaan dalam *task* dan bobot antar *programmer* menjadi faktor yang diperhitungkan, dan kolaborasi ini membantu memastikan bahwa tim dapat mencapai tujuan *sprint* dengan cara yang paling efektif dan efisien. Dengan demikian, pengalaman magang di Jala Tech tidak hanya mengekspos pemagang kepada konsep-konsep Scrum, tetapi juga memperlihatkan bagaimana aplikasi dapat bervariasi di setiap *sprint* dan di antara setiap anggota tim, memperkaya pengalaman penulis dalam kerangka kerja manajemen scrum ini.

B. Belajar Manajemen Diri

Ketika mengikuti program magang, seringkali pemagang dihadapkan pada tantangan manajemen waktu karena harus menjalani magang sekaligus mengikuti kuliah. Penulis perlu memiliki keterampilan untuk mengenali prioritas agar bisa menentukan pekerjaan yang harus diselesaikan terlebih dahulu. Kemampuan ini sangat penting untuk menjaga efisiensi dalam alokasi waktu antara magang dan kuliah. Dalam lingkup magang, tugas-tugas yang harus diatasi tidak terbatas pada satu proyek saja, melainkan sering melibatkan beberapa pekerjaan dalam periode waktu tertentu, seperti dalam satu *sprint* atau satu minggu. Oleh karena itu, memiliki kemampuan manajemen diri yang kuat memegang peran kunci dalam mencapai kesuksesan ketika menjalani magang dan kuliah secara bersamaan.

Pemagang perlu memiliki kemampuan untuk perencanaan yang baik, memahami prioritas, menerapkan konsep manajemen proyek, berkomunikasi dengan efektif, multitasking, dan melakukan evaluasi serta penyesuaian yang teratur dalam jadwal mereka. Dengan mengembangkan kemampuan-kemampuan ini, pemagang dapat mengatasi tantangan manajemen waktu yang dihadapi ketika menjalani magang dan kuliah secara bersamaan.

C. Pengalaman Beradaptasi

Saat menjalani magang, pemagang dihadapkan pada lingkungan perusahaan yang seringkali baru bagi pemagang. Setiap perusahaan memiliki budaya kerja yang unik. Oleh karena itu, salah satu tantangan utama yang dihadapi oleh pemagang adalah kemampuan mereka untuk beradaptasi dengan lingkungan perusahaan yang berbeda ini. Adaptasi merupakan faktor kunci dalam kesuksesan magang, karena tidak hanya mempermudah komunikasi, tetapi juga memungkinkan pemagang untuk mengambil inisiatif yang sesuai dengan budaya kerja perusahaan tersebut.

Kemampuan beradaptasi dengan teknologi yang digunakan oleh perusahaan juga merupakan faktor penting dalam kesuksesan pelaksanaan magang. Hal ini disebabkan oleh

perbedaan antara teknologi yang diajarkan di perguruan tinggi dengan yang digunakan di dunia kerja. Pemagang perlu memiliki tingkat keterbukaan yang tinggi terhadap teknologi-teknologi yang digunakan oleh perusahaan. Dengan sikap yang terbuka ini, pemagang akan lebih mudah untuk terus memperoleh pengetahuan tentang teknologi baru yang mungkin diterapkan oleh perusahaan.

D. Mendapatkan Pandangan terhadap Dunia Industri

Perbedaan lingkungan antara lingkungan perkuliahan dan dunia industri sangat mencolok. Di perguruan tinggi, proses kerja dalam proyek sering kali terstruktur dengan jangka waktu yang relatif panjang. Di dunia industri, sebaliknya tempo kerja biasanya sangat cepat untuk mengatasi persaingan dan memenuhi kebutuhan pelanggan sesegera mungkin. Selain itu, di dunia industri, seringkali situasi yang tidak sesuai dengan harapan dapat mengganggu jadwal pengerjaan proyek. Oleh karena itu, para pekerja, termasuk pemagang, harus memiliki kemampuan untuk beradaptasi dengan situasi yang tidak selalu sesuai dengan ekspektasi.

Selain perbedaan dalam fase kerja, orientasi antara mahasiswa dan pekerja di dunia industri juga berbeda. Mahasiswa ketika mengerjakan proyek cenderung berorientasi pada pencapaian tugas akademis dan meraih nilai tinggi. Di sisi lain, di dunia industri, pekerja menjalankan proyek dengan fokus pada penghasilan dan pemenuhan kebutuhan hidup. Mereka melaksanakan pekerjaan sesuai dengan arahan yang telah diberikan sebelumnya. Motivasi untuk menjalankan pekerjaan dengan optimal oleh pekerja di dunia industri mungkin lebih rendah dibandingkan dengan mahasiswa.

Dengan demikian, menjadi sebuah tantangan bagi pemagang untuk menjaga motivasi dalam menjalankan tugas-tugas yang diberikan oleh perusahaan. Selain mematuhi instruksi yang diberikan, pemagang juga perlu berupaya untuk selalu termotivasi agar dapat mengambil inisiatif yang dapat memberikan kontribusi positif bagi perusahaan.

E. Mendapatkan Wadah Aktualisasi Keilmuan

Salah satu alasan utama untuk terlibat dalam magang adalah keinginan untuk menerapkan pengetahuan yang telah diperoleh selama masa kuliah. Ada kepuasan yang didapatkan dari penggunaan pengetahuan tersebut untuk memberikan kontribusi pada lingkungan sekitar, terutama dalam konteks perusahaan. Pemagang merasa bersyukur karena diberikan kesempatan untuk merasakan pengalaman bekerja sesuai dengan bidang ilmu yang telah mereka pelajari selama kuliah.

Melalui magang, mereka memiliki kesempatan untuk mengaplikasikan pengetahuan mereka sebaik mungkin di dalam perusahaan. Mereka dapat melihat bagaimana teori yang mereka pelajari di kelas diimplementasikan dalam praktik sehari-hari. Hal ini membantu memperdalam pemahaman mereka tentang aspek praktis dari ilmu yang dipelajari.Harapannya, implementasi pengetahuan tersebut dapat berkembang dan memberikan dampakyang lebih luas

pada masyarakat. Dengan demikian, magang tidak hanya menjadi peluang untuk pengembangan pribadi, tetapi juga untuk memberikan nilai tambah bagi perusahaan dan masyarakat secara keseluruhan.

F. Bekerja di Bawah Tekanan

Salah satu motivasi utama untuk terlibat dalam program magang adalah keinginan untuk mengaplikasikan pengetahuan yang telah diperoleh selama masa kuliah. Ada kepuasan yang mendalam dalam menerapkan pengetahuan tersebut untuk memberikan kontribusi dalam lingkungan kerja, khususnya di dalam lingkup perusahaan. Para peserta magang merasa bersyukur karena mereka mendapat kesempatan untuk mengalami pekerjaan sesuai dengan bidang ilmu yang telah mereka pelajari selama masa kuliah.

Melalui magang, mereka memiliki kesempatan untuk menerapkan sebanyak mungkin ilmu yang mereka miliki kepada perusahaan. Mereka dapat melihat bagaimana teori dan konsep yang mereka pelajari di kelas diimplementasikan dalam praktik sehari-hari. Hal ini membantu memperdalam pemahaman mereka tentang aspek praktis dari ilmu yang dipelajari. Harapannya adalah bahwa penerapan ilmu tersebut tidak hanya bermanfaat bagi perusahaan, tetapi juga dapat terus berkembang dan memiliki dampak yang lebih luas pada masyarakat secara umum. Dengan demikian, magang menjadi peluang yang sangat berharga dalam pengembangan pribadi dan dalam memberikan nilai tambah bagi dunia kerja dan masyarakat.

G. Mendapatkan Berbicara di Depan Umum

Selama periode magang, pemagang secara rutin diminta untuk melakukan presentasi setiap minggu yang berisi laporan mengenai pekerjaan yang telah mereka lakukan selama seminggu terakhir. Melalui pengalaman ini, pemagang mengembangkan rasa percaya diri mereka dalam melakukan presentasi karena telah terbiasa dengan tugas tersebut.

Ketidakpercayaan diri yang sering muncul saat melakukan presentasi biasanya disebabkan oleh kekhawatiran akan kemungkinan membuat kesalahan selama proses presentasi. Namun, seiring berjalannya waktu dan setelah beberapa kali melakukan presentasi, pemagang mulai menyadari bahwa baik pembicara maupun pendengar adalah manusia yang kadang-kadang melakukan kesalahan. Bahkan, komentar negatif yang mungkin diterima selama presentasi dapat dijadikan sebagai pelajaran berharga untuk terus meningkatkan keterampilan presentasi di masa depan. Untuk meningkatkan kualitas presentasi, langkah-langkah yang bisa diambil mencakup berlatih secara konsisten dan memiliki pemahaman yang mendalam tentang materi yang akan disampaikan.

H. Mendapatkan Pengalaman Belajar dari Mentor

Ketika menjalani masa magang, pemagang umumnya memiliki seorang mentor atau supervisor yang bertanggung jawab memberikan bimbingan serta arahan terkait tugas-tugas pekerjaan mereka. Peran mentor ini tidak terbatas hanya pada membahas aspek pekerjaan,

tetapi juga mencakup memberikan wawasan mengenai aspek-aspek non-teknis, seperti budaya perusahaan dan bagaimana menjadi seorang karyawan yang memiliki kepribadian yang positif.

Selain menjadi sumber pengetahuan, hubungan dengan mentor seringkali menjadi jaringan yang berharga dalam konteks karier dan industri. Koneksi ini dapat dimanfaatakan oleh pemagang untuk mendapatkan informasi yang mungkin berguna di masa depan. Proses transfer pengetahuan yang dilakukan oleh mentor juga memiliki nilai yangsignifikan bagi pemagang. Pengetahuan yang diterima dari mentor dapat menjadi berharga jika pemagang suatu saat diminta untuk membagikan pengetahuan tersebut kepada orang lain, terutama ketika mereka berinteraksi dengan rekan kerja baru di tempat kerja mereka di masa depan.

I. Pandangan terhadap Pengelolaan Perusahaan

Pemagang memiliki ambisi untuk suatu saat memiliki perusahaannya sendiri. Untuk mencapai tujuan tersebut, pemagang merasa perlu untuk memahami bagaimana suatu perusahaan dikelola dalam berbagai aspek. Dengan mengikuti magang, pemagang mendapatkan gambaran tentang berbagai aspek yang terlibat dalam pengelolaan perusahaan.

Di Jala Tech, sebuah perusahaan startup yang berfokus pada teknologi akuakultur, budaya kerjanya memiliki karakteristik yang berbeda jika dibandingkan dengan perusahaan korporat konvensional. Di perusahaan ini, hierarki jabatan tidak menjadi penghambat dalam berkomunikasi, dan komunikasi dilakukan dengan sikap yang setara, bahkan antara individu-individu yang memiliki posisi berbeda. Hal ini menciptakan atmosfer kerja yang terbuka dan responsif. Namun, penting untuk dicatat bahwa meskipun komunikasi berlangsung secara setara antara semua karyawan, seorang pemimpin tetap harus mempertahankan integritasnya dalam menjalankan tugasnya.

Melalui pengalaman tersebut, pemagang menyadari bahwa lingkungan perusahaan dan gaya komunikasi di dalamnya dapat mengalami perubahan seiring berjalannya waktu. Pengalaman ini memberikan wawasan kepada pemagang bahwa jika suatu saat mereka memiliki perusahaan sendiri, mereka perlu memiliki kemampuan untuk menyesuaikan perusahaan dengan perubahan yang terjadi di sekitarnya.

Saat menjalani masa magang, pemagang juga dimintai pertanyaan yang bersifat personal, yang tidak hanya terkait dengan pekerjaan mereka. Pertanyaan-pertanyaan ini bertujuan untuk memastikan kesejahteraan mental pemagang. Pengalaman ini memberikan dampak positif kepada pemagang dan menunjukkan bahwa perusahaan memiliki perhatian terhadap kesejahteraan mental karyawan. Hal ini memberikan pelajaran kepada pemagang bahwa jika suatu saat mereka memiliki perusahaan sendiri, sangat penting untuk memastikan bahwa perusahaan tersebut memberikan perhatian yang memadai terhadap kesehatan mental karyawan, bukan hanya fokus pada aspek finansial. Perusahaan harus mampu memenuhi kebutuhan karyawan dalam hal kesehatan mental mereka.

J. Penerapan Etika Profesi Islami Dalam Bekerja

Saat menjalani studi di perguruan tinggi, penulis telah mengikuti mata kuliah Etika Profesi yang membahas tentang norma dan aturan dalam dunia kerja. Penulis telah berhasil mengaplikasikan konsep-konsep yang dipelajari dengan baik, yang sangat membantu penulis dalam menyelesaikan magang dengan sukses. Etika kerja memainkan peran yang sangat penting dalam membangun hubungan dan kerjasama yang baik dengan rekan kerja di perusahaan, serta dalam menjaga kedisiplinan dengan mematuhi peraturan yang berlaku.

Menurut Hidayat (2006), ada beberapa prinsip yang harus dipegang oleh setiap Muslim dalam mengembangkan karier mereka di dunia kerja. Penulis telah mengimplementasikan beberapa prinsip ini dalam pekerjaan mereka, seperti menyelesaikan setiap tugas dengan dedikasi dan profesionalisme. Penulis juga sangat menghargai nilai-nilai kejujuran dan tanggung jawab dalam menyelesaikan tugas-tugas, menganggapnya sebagai cara untuk mencari keridhoan Allah SWT. Oleh karena itu, penulis berharap agar Allah SWT senantiasa memberikan keridhoan-Nya dan mempermudah penulis selama menjalani magang di Jala Tech. Ini adalah penerapan Etika Profesi Islami dalam Konteks Dunia Kerja.

K. Mendapatkan Pengalaman Visit Tambak

Selama menjalani magang, pemagang diberikan kesempatan yang sangat berharga untuk mengunjungi tambak udang yang merupakan salah satu mitra perusahaan Jala. Kunjungan ini memiliki tujuan utama, yaitu memberikan pemagang pemahaman yang mendalam tentang praktik budidaya udang yang baik dan benar. Dalam kunjungan ini, pemagang dapat secara langsung mengamati berbagai aspek penting dalam budidaya udang, mulai dari pemilihan lokasi tambak yang tepat hingga manajemen lingkungan yang berkelanjutan. Selain itu, pemagang juga memiliki kesempatan untuk berinteraksi dengan para petani udang yang berpengalaman, memperoleh wawasan praktis, dan bertanya mengenai berbagai tantangan yang sering dihadapi dalam budidaya udang.

Kunjungan ke tambak udang ini bukan hanya memberikan pemahaman teoritis, tetapi juga pengalaman praktis yang sangat berharga. Pemagang dapat melihat secara langsung bagaimana perawatan harian dan pengelolaan tambak dilakukan dengan teliti, termasuk pemantauan kualitas air, pengendalian penyakit, dan manajemen pakan udang. Dengan demikian, pemagang dapat memahami betapa pentingnya keberlanjutan dalam budidaya udang, yang tidak hanya bermanfaat bagi lingkungan, tetapi juga untuk menjaga kelangsungan usaha dan kualitas produk.

Selain itu, kunjungan ini juga membuka peluang bagi pemagang untuk melihat peran teknologi dalam budidaya udang. Mereka dapat melihat penggunaan peralatan canggih, seperti sistem pemantauan otomatis dan pengendalian suhu, yang membantu meningkatkan produktivitas dan keamanan tambak. Dengan demikian, kunjungan ke tambak udang partner Jala ini menjadi langkah penting dalam mempersiapkan pemagang untuk menjadi profesional yang kompeten dan memiliki pemahaman yang mendalam tentang praktik budidaya udang yang baik dan berkelanjutan.

Gambar 4.6 Kegiatan visit tambak Jala Tech

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pelaksanaan magang, dapat disimpulkan bahwa implementasi Chat GPT, Auto-GPT, dan LangChain pada fitur *chatbot* aplikasi Jala secara signifikan meningkatkan efektivitas penyediaan informasi kepada pengguna. Pengguna kini dapat dengan mudah mengakses informasi yang mereka perlukan melalui fitur Function Call, yang memungkinkan mereka memanggil fungsi yang relevan untuk mendapatkan jawaban atas pertanyaan mereka dengan cepat dan tanpa kerumitan.

Fitur Searching External dan Training Mandiri memberikan keunggulan dalam mengakses informasi eksternal dan melakukan pembaharuan mandiri. Dengan adanya fitur ini, *chatbot* dapat terus memperbarui pengetahuannya secara otomatis, menjaga keakuratan dan kelengkapan informasi yang disampaikan kepada pengguna. Proses pelatihan mandiri ini memberikan daya adaptasi kepada *chatbot*, sehingga dapat tetap relevan dan responsif terhadap perkembangan informasi terbaru.

Efisiensi dalam mengakses dan menampilkan data dengan cepat menjadi poin kunci dalam meningkatkan kualitas informasi. Chat GPT, Auto-GPT, dan LangChain memungkinkan *chatbot* untuk dengan cepat dan efisien merespon pertanyaan pengguna, memberikan pengalaman berinteraksi yang lebih memuaskan. Dengan demikian, integrasi teknologi ini tidak hanya meningkatkan kecepatan respons *chatbot*, tetapi juga memastikan informasi yang diberikan selalu terkini, akurat, dan relevan bagi pengguna aplikasi Jala.

5.2 Saran

Untuk pengembangan masa depan aplikasi Jala, kami memiliki beberapa saran yang kami ingin sampaikan. Pertama, dalam jangka panjang, kami merekomendasikan melakukan penelitian yang lebih mendalam terkait dengan Chat GPT, Auto-GPT, dan LangChain. Penelitian ini diarahkan untuk mengoptimalkan penggunaan fitur-fitur tersebut agar aplikasi dapat menghadirkan informasi dengan lebih efektif dan efisien. Perkembangan pesat dalam teknologi *chatbot* menunjukkan bahwa pemahaman yang mendalam sangat penting untuk penerapan jangka panjang.

Selanjutnya, kami menekankan pentingnya melakukan peninjauan kode terhadap seluruh kode yang telah dikembangkan oleh tim pengembang. Ini akan membantu dalam menjaga kebersihan dan kualitas kode yang terkait dengan aplikasi Jala. Selain meningkatkan kualitas

kode, peninjauan kode juga memberikan peluang bagi pengembang untuk terus meningkatkan keterampilan mereka dan menjadi pemrogram yang lebih terampil.

Terakhir, penulis menyarankan untuk lebih fokus pada dokumentasi komponen yang telah dibuat oleh pengembang. Dokumentasi ini akan menjadi acuan berharga jika pengembang lain harus bekerja pada fitur yang sama di masa depan. Dengan dokumentasi yang baik, mereka dapat dengan cepat memahami fungsi dari komponen yang telah ditulis sebelumnya. Dokumentasi juga akan sangat berguna jika pengembang ingin menggunakan ulang komponen yang telah ada, karena mereka akan memiliki pemahaman yang jelas tentang fungsi komponen tersebut.

DAFTAR PUSTAKA

Alamanda, R., Suhery, C., & Brianorman, Y. (2016). Aplikasi Pendeteksi Plagiat Terhadap Karya Tulis Berbasis Web Menggunakan Natural Language Processing dan Algoritma Knuth-Morris-Pratt. *Coding Jurnal Komputer Dan Aplikasi*, *4*(1).

Bhavsar, K., Shah, V., & Gopalan, S. (2020). Scrum: An Agile Process Reengineering In Software Engineering. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, *9*(3), 840–848.

Brown, P. F., deSouza, P. V, L. Mercer, R., Pietra, V. J. Della, & Lai, J. C. (1992). *Class-Based n-gram Models of Natural Language* . *18*(4), 467–480.

Chen, M., Tworek, J., Jun, H., Yuan, Q., & Pinto, H. (2021). *Evaluating Large Language Models Trained on Code*. https://arxiv.org/abs/2107.03374v2

Chowdhery, A., Narang, S., & Devlin, J. (2022). *PaLM: Scaling Language Modeling with Pathways*. https://arxiv.org/abs/2204.02311v5

Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*.

Clavié, B., Ciceu, A., Naylor, F., Soulié, G., & Brightwell, T. (2023). Large Language Models in the Workplace: A Case Study on Prompt Engineering for Job Type Classification. *International Conference on Applications of Natural Language to Information Systems*, 3–17.

Dai, H., Li, Y., Liu, Z., Zhao, L., Wu, Z., Song, S., Shen, Y., Zhu, D., Li, X., Li, S., Yao, X., & Shi, L. (2023). *AD-AutoGPT:AnAutonomousGPTfor Alzheimer'sDiseaseInfodemiology*.

Deldjoo, Y. (2023). *Fairness of ChatGPT and the Role Of Explainable-Guided Prompts*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://arxiv.org/abs/1810.04805v2

Firdaus, M. A. (2017). Implementasi Kerangka Kerja Scrum Pada Manajemen Pengembangan Sistem Informasi. *Semnasteknomedia Online*, *5*(1), 1–2.

Floridi, L., & Chiriatti, M. (2020). *GPT-3: Its Nature, Scope, Limits, and Consequences*. *30*, 681–694.

Grebić, B., & Stojanović, A. (2021). Application of the Scrum Framework on Projects in IT Sector. *European Project Management Journal*, *11*(2), 37–46.

Guzman, & Ines. (2016). *Accenture Chatbots Customer Service*. https://www.scribd.com/document/371520067/Accenture-Chatbots-Customer-Service-pdf#

Hadiwijaya, S. (2018). *Tech Blog*. Https://Medium.Com/Kata-Engineering/Berkenalan-Dengan-Chatbot-Part2-439524a52790. https://medium.com/kata-engineering/berkenalan-dengan-chatbot-part2-439524a52790

Haque, S., Eberhart, Z., Bansal, A., & McMillan, C. (2022). Semantic Similarity Metrics for Evaluating Source Code Summarization. *IEEE International Conference on Program Comprehension*, *2022-March*, 36–47. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Iswandi, A. (2018). *Implementasi Chatbot pada Order Management System Usaha Mikro Kecil Menengah (Studi Kasus Hdkreasi)*.

Iswandi, I., Suwardi, I. S., & Maulidevi, N. U. (2013). Penelitian Awal: Otomatisasi Interpretasi Data Akuntansi Berbasis Natural Language Processing. *JSI: Jurnal Sistem Informasi (E-Journal)*, *5*(2).

Jala Tech. (2023). *Jala Tech Web App*. Https://App.Jala.Tech/. https://app.jala.tech/

Jansson, M., Hrastinski, S., Stenbom, S., & Enoksson, F. (2021). Online question and answer sessions: How students support their own and other students' processes of inquiry in a text-based learning environment. *The Internet and Higher Education*.

Kamble, R., & Shah, D. (2018). Applications of Artificial Intelligence in Human Life. *International Journal of Research - GRANTHAALAYAH*, *6*, 178–188.

Lavena, C. U. (2019). *Apa Itu Chatbot*.

Lisangan, E. A. (2013). Natural Language Processing Dalam Memperoleh Informasi Akademik Mahasiswa UAJM. *Jurnal Tematika*, 1–9.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2021). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computing Surveys*, *55*, 1–35. https://api.semanticscholar.org/CorpusID:236493269

Liu, V., & Chilton, L. B. (2022). *Design Guidelines for Prompt Engineering Text-to-Image Generative Models* .

OpenAI. (2023). *GPT-4 Technical Report*.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, W., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). *Training language models to follow instructions with human feedback*.

Pressman, R. S. (2005). *Software Engineering A PRACTITIONER'S APPROACH*.

Rizky, M., & Sugiarti, Y. (2022). Pengunaan Metode Scrum Dalam Pengembangan Perangkat Lunak: Literature Review. *Journal of Computer Science and Engineering (JCSE)*, 41–48. https://doi.org/10.36596/jcse.v3i1.353

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Russel, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. *New Jersey: Pearson Education, Inc.*

Schwaber, K., & Sutherland, J. (2020). *Panduan Scrum*.

Shawar, B. A., & Atwell, E. (2004). *A Comparison Between Alice and Elizabeth Chatbot Systems*.

Sihombing, D., & Wirapraja, A. (2018). Tren Penerapan Artificial Intelligence Pada Bidang Akuntansi, Energi Terbarukan dan Proses Industri Manufaktur (Studi Literatur). *Jurnal EKSEKUTIF*, *15*, 47–54.

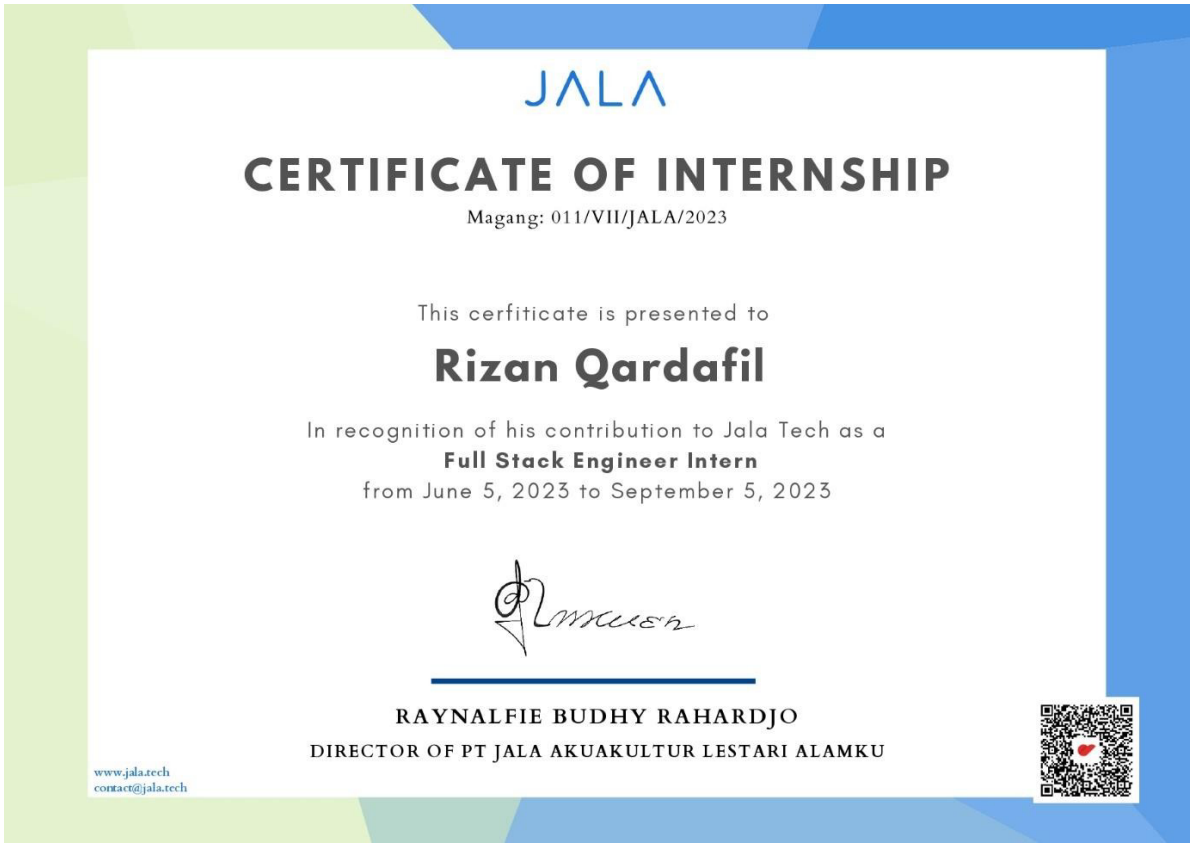
Suparman. (1991). Mengenal Artificial Intelligent. *Yogyakarta: Andi Offset Yogyakarta*.

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. (2023). *A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT*.

Yunanto, A. A., & Herumurti, D. (2016). Face recognition based on Extended Symmetric Local Graph Structure. *International Conference on Information & Communication Technology and Systems (ICTS)*.

LAMPIRAN

Lampiran Sertifikat Magang



JALA

CERTIFICATE OF INTERNSHIP

Magang: 011/VII/JALA/2023

This cerfiticate is presented to

Rizan Qardafil

In recognition of his contribution to Jala Tech as a
Full Stack Engineer Intern
from June 5, 2023 to September 5, 2023

RAYNALFIE BUDHY RAHARDJO
DIRECTOR OF PT JALA AKUAKULTUR LESTARI ALAMKU

www.jala.tech
contact@jala.tech