BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan program magang di Divisi *Digital Business & Technology* (DBT) PT Telkom Indonesia, penempatan dilakukan pada posisi *Backend Developer Intern* dalam tim pengembangan internal. Tugas utama yang dijalankan meliputi dukungan terhadap pengembangan *backend*, khususnya dalam proyek strategis bertajuk *AI Code Review System*. Posisi dan mekanisme koordinasi mengacu pada struktur organisasi Telkom STO Kalibata sebagaimana disajikan dalam Gambar 2.4.2 dan telah dijelaskan pada Bab II.

Dalam praktik kerja, pendampingan teknis diberikan secara intensif oleh *developer senior* sebagai mentor, serta keterlibatan aktif dalam tim Digital *Product Engineering* menjadi bagian integral dari proses magang. Selama periode tersebut, kolaborasi lintas tim secara rutin dilakukan, terutama dalam sesi *knowledge sharing* dan pelaksanaan kerja kolaboratif pada proyek-proyek berskala lebih besar.

Alur Koordinasi Kerja

a). Direktur Utama hingga EVP *Digital Business & Technology* memiliki tanggung jawab strategis dan menetapkan kebijakan makro bagi seluruh divisi dan tim pengembangan.

b). *Chapter Leader* dan *Manager Product Engineering* bertugas mendistribusikan tugas, mengelola *project*, serta memantau jalannya implementasi pengembangan produk digital.

c). *Developer* (Shofyan) bertindak sebagai pembimbing harian yang memberikan arahan teknis, evaluasi berkala, serta pendampingan dalam pelaksanaan tugas-tugas magang.

d). *Backend Intern* (Rafi Aldino) menjalankan tugas-tugas yang diberikan secara mandiri maupun kolaboratif, serta aktif dalam sesi diskusi, presentasi hasil kerja, dan menerima umpan balik (*feedback*) secara berkala.

lingkungan industri digital diperoleh, sekaligus pengalaman langsung dalam menyusun, mengembangkan, dan mengevaluasi solusi *backend* berskala besar berbasis *cloud* dan teknologi *AI*.

Pada awal masa magang, pelatihan internal diberikan yang mencakup dokumentasi teknis, pemahaman *tools*, serta sesi *Golang Workout* untuk memperkuat kemampuan dasar sebelum keterlibatan penuh dalam pengembangan proyek utama. Selain itu, beberapa *mini project* seperti pembuatan fitur *CRUD* juga dikerjakan sebagai bentuk penerapan pembelajaran mandiri dan latihan pengembangan awal.

Selama program magang berlangsung, koordinasi dilakukan secara *hybrid*, menggabungkan komunikasi daring dan luring. Komunikasi harian umumnya berlangsung secara langsung di ruang diskusi internal, sedangkan sesi diskusi teknis yang melibatkan kolaborasi lintas divisi biasanya diadakan dalam forum mingguan atau bulanan melalui *platform Microsoft Teams*. Fleksibilitas untuk belajar, bertanya, dan berdiskusi secara aktif diberikan, khususnya saat mengerjakan *mini project, main project*, serta dalam sesi *sharing* internal *department*.

3.2 Tugas dan Uraian Kerja Magang

Berdasarkan *timeline* kegiatan magang yang ditampilkan pada Tabel 1.1 *Timeline* Tugas dan *Project* Magang, keseluruhan proses pelaksanaan magang dibagi ke dalam tiga fase utama yang disusun secara bertahap guna mendukung pengembangan kompetensi teknis, pemahaman terhadap sistem arsitektural, serta kesiapan profesional di lingkungan industri digital. Fase-fase tersebut terdiri atas:

1). Fase Pembelajaran Dasar dan Penguatan Konsep.

2). Fase *Mini Project* dengan Implementasi *Clean Architecture*.

3). Fase Proyek Utama Pengembangan Sistem *AI Code Review*.

Pelaksanaan kegiatan magang di PT Telkom Indonesia Divisi Digital *Business & Technology* dilakukan secara bertahap dan terstruktur dalam beberapa fase utama. Setiap fase dirancang dengan ruang lingkup kerja dan tujuan pembelajaran yang terfokus, serta dijalankan secara berurutan agar proses peningkatan kapabilitas teknis dapat dilakukan secara sistematis. Pendekatan *progressive enhancement* digunakan untuk memastikan bahwa kompleksitas tantangan meningkat secara bertahap, selaras dengan tingkat penguasaan materi dan lingkungan pengembangan.

Table 3. 1 Realisasi Rincian Kegiatan Magang

No	Tema Kegiatan	Kegiatan yang di Lakukan	Waktu Mulai	Waktu Selesai
Fase Pembelajaran				
1	Perkenalan	Onboarding Magang	10-02-2025	10-02-2025
2	Golang Workout	Golang Workout Modul A	10-02-2025	18-02-2025
3		Golang Workout Modul B	19-02-2025	27-02-2025
4		Golang Workout Modul C	20-02-2025	11-03-2025
5		Golang Workout Modul D	11-03-2025	20-03-2025
Fase Mini Project				
6	Book Season	Reading Book	24-02-2025	3-03-2025
7		Pembuatan Materi Presentasi	3-03-2025	5-03-2025
8		Sesi Sharing + Revisi	5-03-2025	7-03-2025
9	CRUD Project	Intro CRUD Architecture	21-03-2025	24-03-2025
10		Update CRUD usecase user dan repo	25-03-2025	26-03-2025
11		Import Docker build dan Compose	25-03-2025	8-04-2025
12		Membuat pool patern di postgresql connection	26-03-2025	7-04-2025
13		Membuat ENV Template dan git ignore	26-03-2025	8-04-2025
14		Membuat Unit Testing	27-03-2025	5-04-2025

No	Tema Kegiatan	Kegiatan yang di Lakukan	Waktu Mulai	Waktu Selesai
15		Menambahkan liveiless, readiness, gracefull shutdown.	27-03-2025	7-04-2025
16		Membuat teknik sql manual tidak pakai gorm	27-03-2025	9-04-2025
17		Membuat Interface di setiap layer	9-04-2025	9-04-2025
18		Menambahkan jaeger dan Circuite Breaker	9-04-2025	10-04-2025
19		Membuat validasi untuk usecase dan handler	9-04-2025	10-04-2025
20		Menambahkan kafka-ui atau kafka drop	10-04-2025	14-04-2025
21		Membuat state circuit breaker Ketika di kirim alert ke telegram	14-04-2025	22-04-2025
22		Presentasi + Sharing Season	2-05-2025	2-05-2025
23	Power Automate	Pembuatan form jadwal tugas	26-05-2025	26-06-2025
24		Integrasi Email dan Ms.teams	26-05-2025	27-05-2025
25		Update planner	27-05-2025	28-05-2025
26	Automation n8n	Membuat Flow n8n	2-06-2025	3-06-2025
27		Import form dan membuat API Key	2-06-2025	2-06-2025
28		Membuat system database sementara menggunakan Groq	2-06-2025	2-06-2025
28		Hasil output ke botchat n8n	3-06-2025	3-06-2025
30		Membuat flow n8n untuk AI (gpt 4o-mini, Deepseek, qwen-32)	3-06-2025	4-06-2025
31		Membuat deskripsi flow setiap AI yang di gunakan	4-06-2025	9-06-2025
32		Melakukan merge setiap hasil Ai	5-06-2025	7-06-2025
33		Convert hasil menjadi pdf, html, txt dan markdown	7-06-2025	11-06-2025
34		Trigger via telegram bot dan botchat	12-06-2025	12-06-2025
Fase Main Project				
35	AI Code Review	Pembuatan materi mengenai LSTM, dan AI	14-02-2025	27-03-2025

No	Tema Kegiatan	Kegiatan yang di Lakukan	Waktu Mulai	Waktu Selesai
36		Pembuatan perbandingan dan alternative aplikasi untuk Jira Service Desk	18-02-2025	21-02-2025
37		Pengenalan dan pembekalan AI Code Review	20-02-2025	11-03-2025
38		Testing Postman	26-02-2025	19-03-2025
39		Clone repo/ data AI Code untuk di lanjutkan	11-03-2025	11-03-2025
40		Transferknowlage AI Code	11-03-2025	17-03-2025
41		Export data Jira Service	17-03-2025	17-03-2025
42		Test dan perbaikan Resource dan role AI COde	18-03-2025	20-03-2025
43		Pembuatan dokumentasi AI Code	25-03-2025	15-04-2025
44		Sesi Presentasi	27-03-2025	27-03-2025
45		Pembuatan tutorial untuk run AI Code Review (Postman dan readme.md) + Merge Request test	24-04-2025	25-04-2025
46		Request Akses user, ui, code review	28-04-2025	28-04-2025
47		Pembuatan branch baru dan penghapusan modul dan repo AI Code	6-05-2025	6-05-2025
48		Pembuatan unit test untuk modul baru	7-05-2025	8-05-2025
49		Kunjungan dan sesi Sprint Meeting DFU 2025	8-05-2025	8-05-2025
50		Implementasi Circuite Breaker kedalam AI Code	9-05-2025	23-05-2025
51		Pembuatan video tutorial menjalankan AI Code dan postman + Update dokumentasi	16-05-2025	20-05-2025
52		Deploy Local Ai Code review	19-05-2025	20-05-2025
53		Checking API Start dan Queue, docker, dan env	19-05-2025	21-05-2025
54		Validasi Buat circuit breaker untuk API ke APT LLM + Alert ke MS.Teams	22-05-2025	28-05-2025
55		Testing ai code review menggunakan trace, metric,	3-06-2025	20-06-2025

No	Tema Kegiatan	Kegiatan yang di Lakukan	Waktu Mulai	Waktu Selesai
		dan log ke dalam open observe		

Tabel 3.1 Realisasi Rincian Kegiatan Magang, menyajikan rangkuman kegiatan yang telah dilaksanakan sepanjang periode magang, yaitu dari tanggal 10 Februari hingga 17 Juli 2025. Setiap agenda diklasifikasikan berdasarkan tahapan yang telah disusun secara sistematis dan progresif, mulai dari pengenalan lingkungan kerja, pelatihan teknis berbasis modul, hingga keterlibatan langsung dalam pengembangan proyek aktual. Lebih dari sekadar pencatatan kronologis, tabel ini juga dirancang untuk menggambarkan pembentukan kompetensi teknis dan profesional yang sejalan dengan kebutuhan industri teknologi informasi, khususnya di lingkungan Divisi Digital *Business & Technology* (DBT) PT Telkom Indonesia.

Dengan demikian, Tabel 3.1 Realisasi Rincian Kegiatan Magang, tidak hanya berfungsi sebagai dokumentasi waktu dan jenis kegiatan, tetapi juga merepresentasikan peta penguasaan keahlian secara bertahap yang menghasilkan kontribusi nyata dalam pengembangan sistem berbasis kecerdasan buatan. Rangkaian program tersebut disusun untuk mendorong pencapaian kemampuan teknis secara menyeluruh, sekaligus membekali kemampuan dengan pengalaman langsung dalam merancang sistem *backend* modern. Fokus utama diarahkan pada implementasi teknologi *AI* dalam proses pengembangan perangkat lunak, integrasi arsitektur sistem terdistribusi, serta penerapan prinsip *DevOps* yang didukung oleh *containerization* dan otomasi.

Seluruh tahapan dilaksanakan melalui pendekatan bertingkat dan terstruktur, dimulai dari fase orientasi dan pembelajaran dasar, dilanjutkan dengan *mini project* sebagai bentuk penerapan praktis, hingga berkontribusi dalam proyek utama berskala industri. Berikut merupakan uraian detail kegiatan yang dilakukan selama masa magang, disusun berdasarkan empat fase utama pelaksanaan:

3.2.1 Fase Awal dan Pembelajaran

Pada fase awal ini, fokus diarahkan pada pembentukan pemahaman yang kuat terhadap konsep dasar pemrograman *backend* serta pengenalan terhadap ekosistem teknologi yang digunakan di lingkungan Divisi Digital *Business & Technology*. Fase ini difokuskan untuk memberikan pemahaman awal mengenai konsep-konsep fundamental yang diperlukan selama pelaksanaan magang. Selain itu, fase ini juga digunakan sebagai media orientasi terhadap sistem kerja, alur komunikasi, serta pengenalan terhadap lingkungan pengembangan yang akan digunakan. Tujuan utama dari fase ini adalah membangun landasan teoritis dan teknis sebelum berlanjut ke tahap penerapan. Pembelajaran dilakukan secara mandiri dan terstruktur melalui *platform* internal yang telah dirancang khusus, dengan materi yang mencakup sintaks dasar, struktur data, pengelolaan alur program, hingga pengenalan konsep *concurrency*. Selain itu, pemahaman terhadap alur kerja tim dan *tools* kolaboratif seperti GitLab, Telegram *Bot*, dan Microsoft Teams juga diperkenalkan untuk mendukung kesiapan teknis dan adaptasi terhadap lingkungan kerja digital.

3.2.1.1 Onboarding dan Orientasi Awal

Tahap awal program magang diawali dengan pelaksanaan sesi *onboarding* yang bertujuan untuk memberikan pemahaman menyeluruh mengenai struktur organisasi, nilai-nilai perusahaan, serta lingkungan kerja di PT Telkom Indonesia, khususnya di bawah Divisi Digital *Business & Technology* (DBT). Kegiatan ini menjadi fondasi penting dalam membangun keselarasan antara tim dengan kultur kerja profesional yang diterapkan oleh tim internal.

Materi orientasi mencakup pengenalan terhadap sistem kerja kolaboratif, etika komunikasi internal, serta tata kelola keamanan informasi sesuai dengan regulasi perusahaan. Selain itu, diperkenalkan pula berbagai perangkat dan teknologi yang akan digunakan selama program magang, di antaranya:

a). Bahasa pemrograman Golang, sebagai alat utama dalam pengembangan layanan *backend*,

b). GitLab, sebagai *platform version control* dan manajemen kolaborasi kode,

c). PostgreSQL, sebagai sistem basis data relasional utama,

d). serta lingkungan pengembangan seperti Microsoft Teams, Telegram, dan Docker untuk kebutuhan komunikasi dan *containerization*.

Melalui proses orientasi ini, kesiapan kerja secara teknis dan *non*-teknis dibentuk sejak awal, guna mempercepat proses adaptasi terhadap alur kerja, standar pengembangan, serta budaya organisasi yang berlaku.

Gambar 3. 1 Sesi Onboarding Magang

Gambar 3.1 memperlihatkan dokumentasi pelaksanaan onboarding sebagai titik awal integrasi setiap individu ke dalam lingkungan kerja. Sesi ini dilaksanakan secara *hybrid* dan dipandu langsung oleh perwakilan tim DBT. Selain sebagai sarana penyampaian informasi administratif dan teknis, *onboarding* juga berperan sebagai medium pembentukan budaya kerja kolaboratif yang adaptif dan produktif. Suasana interaktif yang tercipta selama sesi berlangsung turut mendorong percepatan adaptasi terhadap sistem kerja dan dinamika proyek yang akan dijalankan. Oleh karena itu, tahapan ini tidak hanya berfungsi sebagai pengenalan awal, tetapi juga sebagai landasan strategis

dalam mempersiapkan segala hal untuk menghadapi kompleksitas proyek berskala industri selama masa magang.

3.2.1.2 Pembelajaran Golang Workout Modul

Fase pembelajaran teknis dimulai dengan program intensif yang berfokus pada penguasaan dasar-dasar pemrograman menggunakan bahasa Go (Golang). Hal ini disesuaikan dengan kebutuhan pengembangan sistem *backend* di lingkungan Divisi Digital *Business & Technology* (DBT), PT Telkom Indonesia, yang secara aktif mengimplementasikan Golang dalam proyek-proyek produksinya.

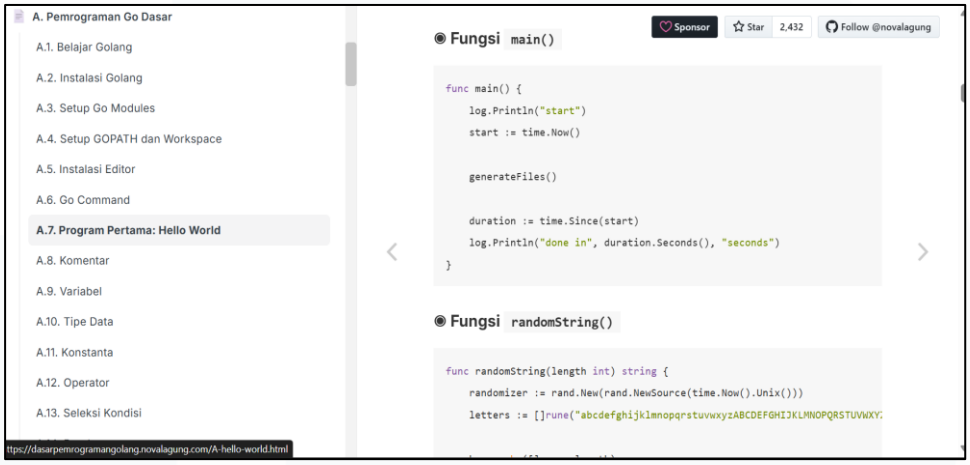
Kegiatan pembelajaran difasilitasi melalui *platform* internal bernama Golang *Workout*, yang terdiri dari lebih dari 120 modul interaktif dan tersusun dalam empat tingkat progresif, yaitu *Level* A (Dasar), *Level* B (Menengah), *Level* C (Lanjutan), dan *Level* D (*Challenges*). Setiap level dirancang untuk memperkuat pemahaman secara bertahap, dengan pendekatan praktik langsung dan evaluasi berkelanjutan.

Materi yang dipelajari mencakup topik-topik fundamental dalam pemrograman *backend*, antara lain:

a). Pemahaman struktur data dasar seperti *array, slice*, dan map,
b). Pengelolaan kontrol alur program menggunakan *conditional* dan *looping*,
c). Implementasi *goroutine* dan *channel* untuk menangani proses secara konkuren (*concurrency*),
d). Penanganan *error* secara idiomatik,
e). Penerapan prinsip modularitas dan penyusunan *package*,
f). Hingga pembuatan REST API sederhana dengan pendekatan *handler* dan *router*.

Setiap modul dilengkapi dengan latihan praktis, studi kasus, serta evaluasi berbasis tes kode untuk memastikan pemahaman yang

menyeluruh, tidak hanya secara teoritis tetapi juga dalam konteks implementasi nyata.



Gambar 3. 2 Modul Pembelajaran Golang Workout

Gambar 3.2 menunjukkan tampilan antarmuka dari *platform* git hub Golang *Workout* yang digunakan sebagai media utama pembelajaran mandiri. Melalui *platform* ini, tersedia berbagai materi yang dapat diakses secara langsung, menyelesaikan soal, dan memperoleh umpan balik otomatis terhadap setiap latihan yang dikumpulkan. Penyusunan modul yang sistematis serta integrasi fitur evaluasi membuat proses pembelajaran berlangsung secara adaptif dan efisien. *Platform* ini juga memungkinkan penyesuaian tingkat kesulitan sesuai progres yang telah di selesaikan, menjadikannya sarana yang efektif dalam membangun fondasi penguasaan Golang sebelum memasuki fase pengembangan proyek.

3.2.2 Fase Mini Project

Setelah penguasaan dasar dianggap cukup, tahap selanjutnya diarahkan pada penerapan keterampilan teknis melalui proyek berskala kecil berupa pengembangan layanan *CRUD backend*, serta beberapa tugas lainnya. Fase ini ditujukan sebagai tahap awal penerapan keterampilan yang telah dipelajari. Melalui aktivitas ini, pemahaman yang sebelumnya bersifat konseptual mulai diarahkan ke dalam praktik terstruktur. Selain melatih

kemampuan teknis, fase ini juga berperan dalam membangun keterampilan *problem solving*, kerja mandiri, serta pemahaman terhadap pola kerja sistematis. Seperti *project CRUD* sistem dirancang menggunakan pendekatan *Clean Architecture* untuk mendorong modularitas, *maintainability*, dan pengujian yang sistematis. Dalam fase ini, konsep observabilitas mulai diperkenalkan melalui integrasi Jaeger, serta prinsip *event-driven architecture* diaplikasikan menggunakan Apache Kafka. Sistem juga dihubungkan dengan Telegram *Bot* sebagai media notifikasi otomatis. Fase ini menjadi tahap penting untuk menguji pemahaman, membangun fondasi teknis, dan mengevaluasi kemampuan *debugging* serta desain sistem.

3.2.2.1.Season Membaca Buku dan Sharing

Sebagai bagian dari fase mini *project* yang mengintegrasikan aspek penguatan *soft skill* dan pemahaman konseptual, telah dilaksanakan kegiatan literasi teknis melalui pembacaan buku "*Designing Data-Intensive Applications"* karya Martin Kleppmann. Buku ini dipilih karena dianggap relevan dengan konteks pengembangan sistem *backend* modern, khususnya dalam hal pengelolaan data, skalabilitas, dan arsitektur sistem terdistribusi.

Fokus pembelajaran diarahkan pada Bab 7 hingga Bab 9, yang mencakup topik lanjutan seperti:

a). Replikasi dan konsistensi data,
b). Sistem terdistribusi dan penanganan konflik,
c). Pemrosesan aliran data (*stream processing*),
d). serta prinsip desain arsitektur yang dapat diskalakan secara horizontal.

Setelah proses pembacaan, telah dilakukan sesi diskusi dan *sharing* bersama tim Divisi DBT. Kegiatan ini dipimpin dan di jalankan bersama-sama antar seluruh tim yang ada di divisi DBT, sebagai upaya untuk menyampaikan *insight* yang diperoleh dan mengaitkannya

dengan kebutuhan aktual proyek yang sedang dikembangkan. Diskusi dilakukan secara terbuka dan interaktif, sehingga memungkinkan terjadinya pertukaran ide teknis lintas *level* serta penguatan literasi teknologi dalam lingkup tim.

Gambar 3. 3 Sharing Season Book

Gambar 3.3 menampilkan dokumentasi pelaksanaan sesi diskusi yang berlangsung secara *hybrid*. Kegiatan ini tidak hanya dimanfaatkan sebagai forum presentasi materi, tetapi juga sebagai sarana validasi pemahaman dan simulasi komunikasi teknis di lingkungan profesional. Berdasarkan evaluasi yang dilakukan oleh mentor dan tim, kegiatan *sharing* ini berdampak positif dalam memperluas perspektif teknis anggota tim serta memperdalam pemahaman terhadap prinsip-prinsip sistem berskala besar yang relevan dengan implementasi *backend service* di industri.

3.2.2.2.Mini project CRUD

Mini project ini merupakan fondasi awal dari pengembangan sistem *backend* berskala industri. Setiap individu diminta mengembangkan layanan *CRUD* (*Create, Read, Update, Delete*) sederhana namun *robust* dengan menerapkan prinsip *Clean Architecture*

menggunakan bahasa pemrograman Golang. Proyek ini bertujuan mengasah kemampuan desain arsitektur modular, pemisahan logika bisnis dari infrastruktur, serta pengelolaan dependensi yang baik.

Selain sebagai ajang penerapan teori yang telah dipelajari selama masa pelatihan, *mini project* ini juga dirancang untuk memberikan pengalaman praktis dalam pengembangan perangkat lunak *backend* yang kompleks dan terintegrasi. Sistem *CRUD* yang dibangun tidak hanya mencakup fitur dasar, tetapi juga mencerminkan praktik industri melalui penerapan arsitektur yang *maintainable* dan *scalable*.

Stack teknologi yang digunakan meliputi:

a). PostgreSQL sebagai sistem basis data relasional,
b). Jaeger untuk *observabilitas* dan *distributed tracing*,
c). Apache Kafka sebagai *messaging queue* untuk mendukung integrasi *event-driven*,
d). Telegram *Bot* API untuk notifikasi otomatis yang dikirimkan berdasarkan aksi pengguna.

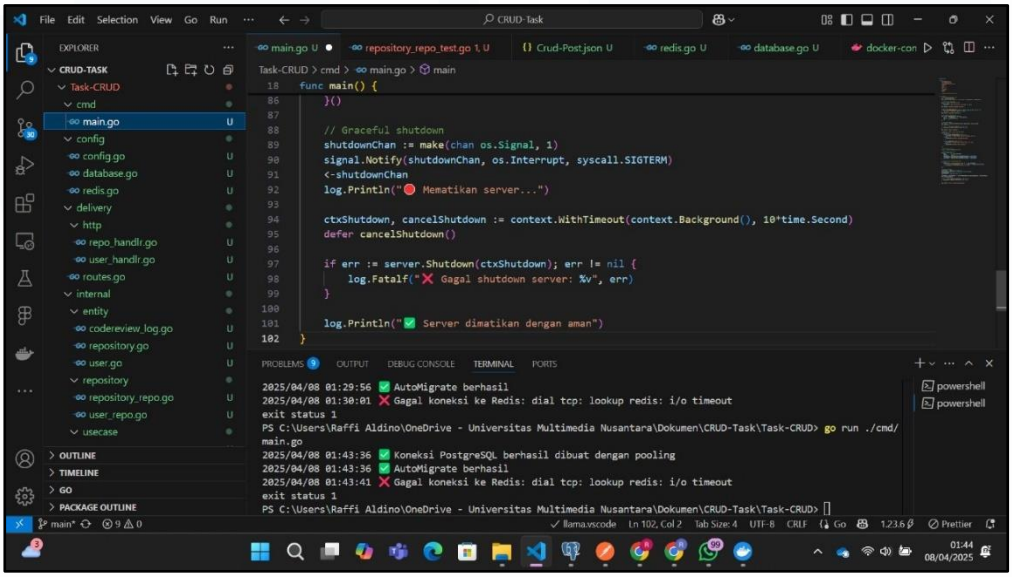
Pengembangan dilakukan dengan memisahkan kode ke dalam beberapa *layer: entity, usecase, delivery (handler),* dan *repository*. Hal ini bertujuan untuk memastikan bahwa setiap komponen dapat diuji secara independen dan mudah diperluas di masa depan. Unit *testing* juga dilakukan untuk menjamin reliabilitas fungsi utama.

Beberapa tantangan teknis yang dihadapi antara lain adalah:

1). Konfigurasi koneksi *database* yang aman dan efisien
2). Penanganan *error* antar-*layer* dan standardisasi *response* API
3). Penerapan *distributed tracing* dengan *span* yang konsisten antar-layanan
4). Penyusunan struktur proyek yang sesuai dengan prinsip *separation of concerns*

5). *Debugging* dan optimasi performa sistem saat melakukan transaksi data besar

Mini project ini memberikan fondasi kuat dalam pemahaman pengembangan *backend service* skala kecil-menengah dengan pendekatan yang profesional dan siap untuk diintegrasikan dalam sistem berskala besar.



Gambar 3. 4 CRUD Project

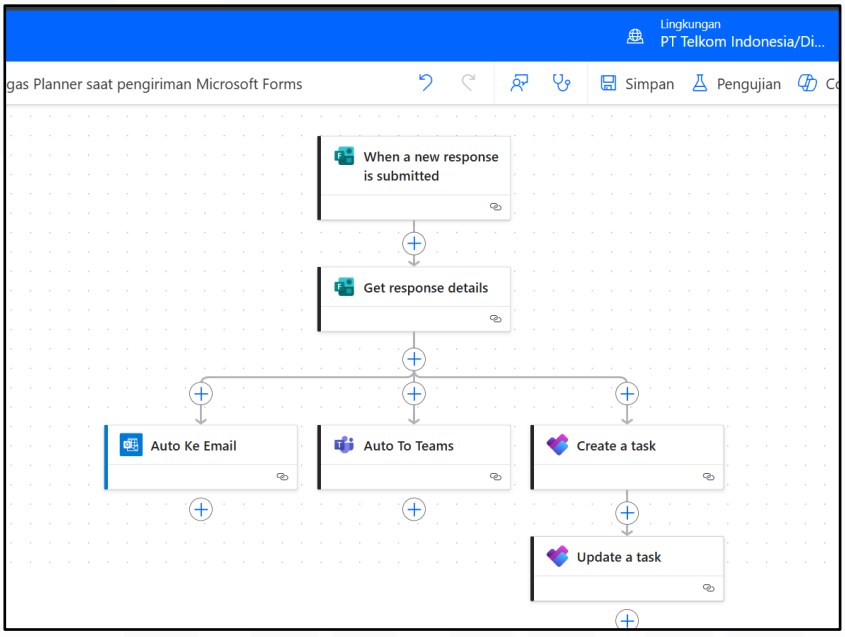
Gambar 3.4 memperlihatkan *Architecture code* dan representasi visual dari hasil pengembangan *mini project CRUD* yang dibangun dengan pendekatan *Clean Architecture* menggunakan bahasa pemrograman Golang. Dalam tampilan tersebut, dapat diamati struktur direktori proyek yang telah dipisahkan ke dalam beberapa *layer* utama, seperti *entity, usecase, delivery*, dan *repository*, yang masing-masing memiliki tanggung jawab tersendiri dalam menjalankan alur logika aplikasi.

Visualisasi ini sekaligus menggambarkan bagaimana praktik desain modular diterapkan secara konsisten untuk mendukung prinsip *separation of concerns* dalam pengembangan *backend service*. Selain itu, elemen-elemen penting dari proyek seperti konfigurasi koneksi ke PostgreSQL, integrasi Apache Kafka untuk pemrosesan *event* secara

asynchronous, serta penggunaan Jaeger untuk *distributed tracing* dapat dilihat pada file konfigurasi dan *package* yang telah terstruktur.

Penggunaan Telegram *Bot* untuk notifikasi otomatis juga tercermin melalui *file* integrasi API eksternal yang ditautkan ke alur aksi pengguna, seperti pembuatan atau pembaruan data. Gambar ini mendemonstrasikan kesiapan *mini project* tidak hanya sebagai latihan teknis, tetapi juga sebagai prototipe yang merefleksikan praktik pengembangan perangkat lunak *backend* berskala industri secara nyata dan fungsional.

3.2.2.3.Power Automate Autentikasi



Gambar 3. 5 Power Automate Autentikasi

Gambar 3.5 menampilkan autentikasi untuk pengembangan automasi menggunakan *power automate*. Sebagai inisiatif lanjutan dalam peningkatan efisiensi operasional, dilakukan pengembangan sistem otomatisasi menggunakan *Microsoft Power Automate* untuk membantu proses pelaporan dan perencanaan tugas mingguan karyawan di lingkungan Divisi DBT PT Telkom Indonesia. Sistem ini dirancang untuk merespons setiap input yang dikirim melalui *Microsoft Forms*, kemudian memprosesnya secara otomatis dan mengirimkan hasilnya ke

berbagai layanan internal seperti *Microsoft Teams*, *Outlook* (*Email*), dan *Microsoft Planner*.

Alur kerja dimulai dari pemicu utama, yaitu ketika ada tanggapan baru pada *Microsoft Forms* (*When a new response is submitted*). Setelah itu, sistem akan mengambil detail data isian (*Get response details*), dan mendistribusikannya secara otomatis melalui tiga jalur utama:

a). *Auto* ke *Email* – Mengirimkan rekap tugas ke *email* pengguna sebagai pengingat pribadi.
b). *Auto to Teams* – Mengirimkan notifikasi otomatis ke kanal *Microsoft Teams* yang sesuai.
c). *Create/Update Task* di *Microsoft Planner* – Membuat atau memperbarui entri tugas dalam sistem manajemen kerja terpusat.

Gambar 3. 6 Hasil Power Automate Triger ke Teams dan Email

Gambar 3.6 menampilkan hasil visual dari sistem otomatisasi tugas yang dikembangkan menggunakan *Microsoft Power Automate*. Sistem ini menunjukkan notifikasi mingguan yang dikirim secara otomatis melalui *platform Microsoft Teams* dan *Email (Outlook),* berdasarkan input yang sebelumnya dikumpulkan dari *Microsoft Forms*.

Dalam gambar tersebut, dapat dilihat bahwa pengguna menerima pesan langsung melalui kanal komunikasi internal yang berisi informasi terstruktur mengenai:

1). Judul tugas dan aktivitas mingguan (misalnya: *Develop Application*)
2). Tanggal mulai dan selesai pengerjaan
3). Fokus pengerjaan (seperti penekanan pada laporan hasil akhir)
4). Catatan penting sebagai pengingat untuk memperbarui status pekerjaan, menghindari hambatan komunikasi (*stuck*), serta ajakan untuk menjaga konsistensi kolaboratif di tim.

Pesan ini juga dipersonalisasi dengan gaya komunikasi yang ringan namun produktif, serta dilengkapi emoji untuk mendukung *engagement* visual. Format tersebut tidak hanya meningkatkan keterbacaan, tetapi juga menciptakan lingkungan kerja yang lebih interaktif dan suportif.

Seluruh proses dijalankan dalam konteks *Microsoft 365* yang telah diamankan menggunakan sistem autentikasi *Azure Active Directory* (AAD). Hanya pengguna yang terautentikasi dan tergabung dalam domain organisasi PT Telkom Indonesia yang dapat mengakses dan menjalankan alur kerja ini. Hal ini menjamin privasi data, akurasi distribusi pesan, serta kepatuhan terhadap kebijakan keamanan informasi perusahaan.

Hasil akhir dari implementasi alur ini adalah:

a). Proses perencanaan tugas menjadi lebih sistematis dan terdokumentasi.

b). Komunikasi lintas kanal (*Forms* → *Email* → *Teams* → *Planner*) berlangsung otomatis tanpa intervensi manual.

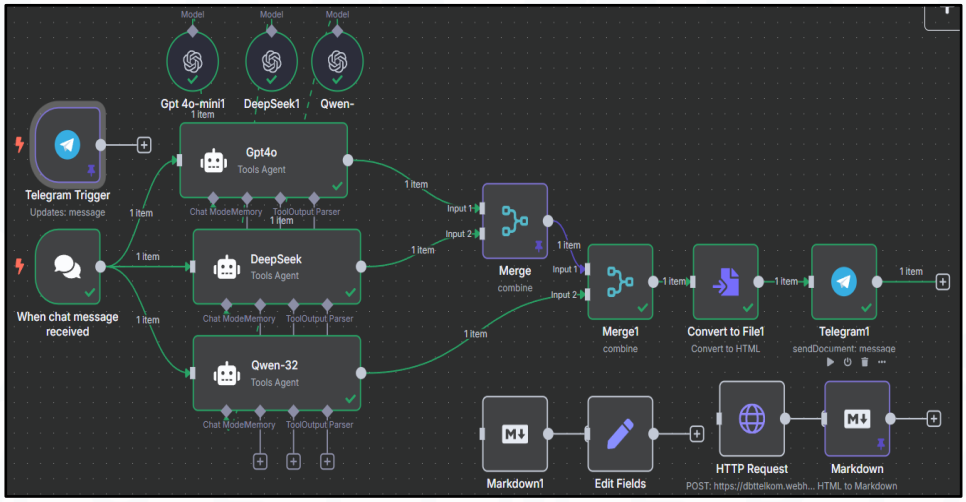
c). Efisiensi pelaporan dan konsistensi penyebaran informasi meningkat secara signifikan.

d). Kolaborasi tim diperkuat dengan kejelasan alur kerja dan keterlibatan aktif setiap individu dalam manajemen tugas mingguannya.

Dengan demikian, integrasi *Power Automate* ini telah berhasil mendemonstrasikan bagaimana pendekatan *low-code* automation dapat diimplementasikan untuk mempercepat proses internal, sekaligus meningkatkan pengalaman kerja digital di lingkungan DBT PT Telkom Indonesia.

Seluruh proses ini berjalan dalam lingkungan kerja *Microsoft 365* yang diamankan melalui autentikasi berbasis *Azure Active Directory* (AAD). Dengan demikian, hanya pengguna yang telah terautentikasi dan memiliki hak akses tertentu yang dapat memicu dan memanfaatkan alur ini. Autentikasi ini tidak hanya menjamin keamanan dan validitas pengguna, tetapi juga memastikan bahwa alur kerja hanya berjalan di dalam konteks organisasi Telkom Indonesia.

3.2.2.4.Autentikasi and Compare Result n8n



Gambar 3. 7 n8n Autentikasi Compare AI

Sebagai bagian dari pengembangan dan eksplorasi lanjutan dalam proyek *AI Code Review*, dilakukan integrasi alur otomatis menggunakan n8n untuk memfasilitasi perbandingan hasil tinjauan kode dari beberapa model kecerdasan buatan secara *real-time,* Seperti yang terdapat pada Gambar 3.7 n8n Autentikasi *Compare AI*. Implementasi ini bertujuan untuk mengevaluasi konsistensi serta kedalaman analisis dari berbagai model *AI*, sekaligus mengotomatisasi proses komunikasi antar sistem melalui *platform* Telegram.

Proses ini diawali dengan pemicu pesan dari Telegram yang digunakan sebagai *entry point* sistem. Ketika pengguna mengirimkan pesan (berupa *prompt* atau kode) ke *bot* Telegram, n8n secara otomatis meneruskan permintaan tersebut ke tiga model *AI* yang berbeda, yaitu *GPT-4o*, *DeepSeek*, dan *Qwen-32*, secara paralel. Masing-masing model diakses melalui *Tools Agent* node di dalam n8n, dengan hasil *output* yang diparsing dalam format *JSON*.

Setelah ketiga model memberikan respons, hasilnya dikombinasikan menggunakan *node Merge* dan *Merge1* untuk membentuk struktur perbandingan terpadu. Hasil penggabungan ini kemudian diubah ke format *HTML* dan *Markdown* melalui *node* konversi, serta dikirimkan kembali ke pengguna Telegram melalui API *sendDocument*.

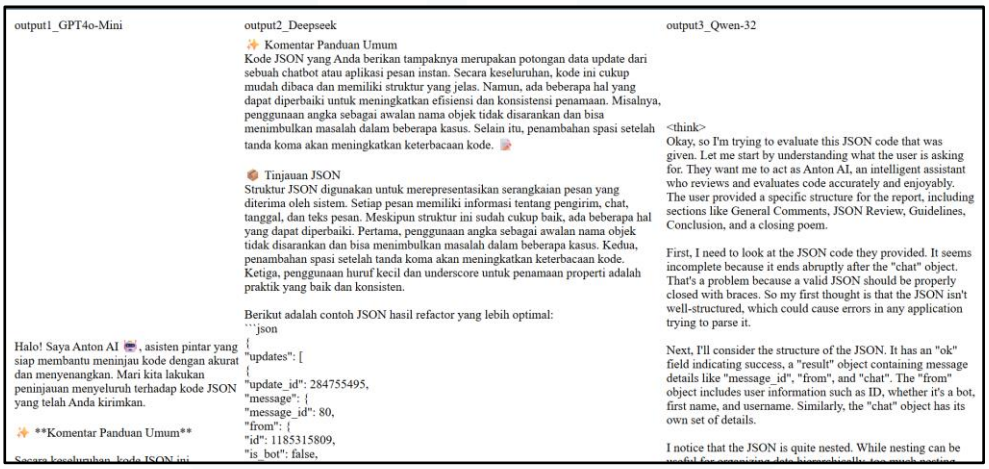
Integrasi ini memungkinkan proses:

a). Pengujian otomatis terhadap berbagai model *AI* untuk *use-case* tinjauan kode.

b). Evaluasi komparatif terhadap kualitas respons *AI* berdasarkan *prompt* yang sama.

c). Dokumentasi cepat dan estetis terhadap hasil *review* dalam format yang siap digunakan atau dianalisis lebih lanjut.

Melalui penerapan alur ini, proses validasi model *AI* dapat dijalankan secara efisien dan fleksibel. Hal ini berkontribusi terhadap strategi pengembangan *AI Code Review* yang adaptif, di mana evaluasi antar model sangat dibutuhkan untuk menentukan *output* terbaik dan paling sesuai dengan standar kualitas perangkat lunak yang diterapkan di Telkom Indonesia.



Gambar 3. 8 Hasil Compare AI n8n

Gambar 3.8 menampilkan hasil evaluasi dan perbandingan respons dari tiga model kecerdasan buatan berbeda *GPT-4o-Mini, DeepSeek,* dan *Qwen-32* terhadap sebuah input berupa potongan kode *JSON*. Masing-masing model diminta untuk memberikan masukan atau review terhadap struktur dan kualitas kode *JSON* yang sama, dengan gaya respons yang khas sesuai arsitektur model masing-masing.

1). *Output1_GPT4o-Mini*:

Memberikan respons yang bersifat ramah dan komunikatif, diawali dengan salam pembuka dari karakter asisten *AI* bernama Anton *AI*. Model ini menggunakan gaya bahasa natural dan ringan, menyisipkan bagian *review* dalam bentuk poin-poin jelas, seperti Komentar Panduan Umum. Penekanannya adalah pada struktur pesan *JSON* secara keseluruhan dan keterbacaan.

2). *Output2_DeepSeek*:

Respons DeepSeek lebih sistematis dan teknis, dengan pembagian yang terstruktur ke dalam dua bagian utama: Komentar Panduan Umum dan Tinjauan *JSON*. Model ini secara eksplisit menyarankan perbaikan sintaksis dan konsistensi penggunaan karakter (seperti spasi dan tanda koma), serta memberikan contoh kode *JSON* yang telah direfaktor agar lebih optimal dan sesuai praktik terbaik pengembangan perangkat lunak.

3). *Output3_Qwen-32*:

Qwen-32 menampilkan respons dalam bentuk esai teknis berbahasa Inggris. Gaya penjelasannya cenderung analitis dan mendalam, diawali dengan *think aloud reasoning*, diikuti oleh evaluasi struktur, validitas sintaks, dan potensi *error parsing*. Model ini juga membahas detail atribut seperti *ok*, *message_id*, dan *chat*, serta menyoroti kekurangan struktur *nesting* pada *JSON*.

Perbandingan ini menggambarkan beragam pendekatan dan kedalaman analisis dari masing-masing model *AI* terhadap input yang identik. Hal ini sangat bermanfaat untuk:

a). Menilai gaya komunikasi masing-masing model (ramah, teknikal, atau akademis).

b). Mengidentifikasi model mana yang lebih cocok digunakan untuk *review* kode di konteks profesional tertentu (misalnya, dokumentasi teknis vs interaksi dengan *end-user*).

c). Membantu proses seleksi model *AI* yang akan diintegrasikan secara permanen dalam sistem *AI Code Review*.

Dengan visualisasi ini, pengujian komparatif tidak hanya dilakukan secara teknis, tetapi juga mempertimbangkan aspek kejelasan,

struktur *respons*, dan kualitas saran yang diberikan oleh masing-masing model *AI*.

Pendekatan ini juga mencerminkan praktik nyata dalam pengembangan sistem *backend* modern berbasis *event-driven architecture* dan *workflow automation*, serta membuka peluang integrasi lanjutan untuk pengambilan keputusan otomatis berbasis model evaluasi *AI*.

3.2.3 Fase Main project

Fase inti dari kegiatan magang ditandai dengan keterlibatan langsung dalam pengembangan sistem *AI Code Review* sebuah *platform* berbasis kecerdasan buatan yang dirancang untuk mengevaluasi kualitas kode secara otomatis. Fase ini difungsikan sebagai tahap kontribusi terhadap kegiatan berskala lebih besar yang melibatkan proses kerja nyata di dalam tim. Pada tahap ini, pemahaman teknis dan pengalaman yang telah diperoleh sebelumnya mulai diimplementasikan dalam lingkup yang lebih kompleks dan terstruktur. Fase ini juga memberikan kesempatan untuk memahami dinamika kerja profesional, alur kolaborasi, dan tanggung jawab yang lebih luas. Pada tahap ini, skema API *backend* dirancang ulang untuk dapat berinteraksi dengan model *AI*, sistem *caching* dioptimalkan menggunakan Redis, serta pencatatan hasil analisis dilakukan untuk tujuan historis. *Middleware* validasi, penerapan *circuit breaker*, dan observasi performa melalui Jaeger turut dilakukan untuk meningkatkan keandalan sistem. Proyek ini memberikan gambaran nyata mengenai kompleksitas sistem berskala industri dan pentingnya kolaborasi dalam pengembangan berbasis *microservices*.

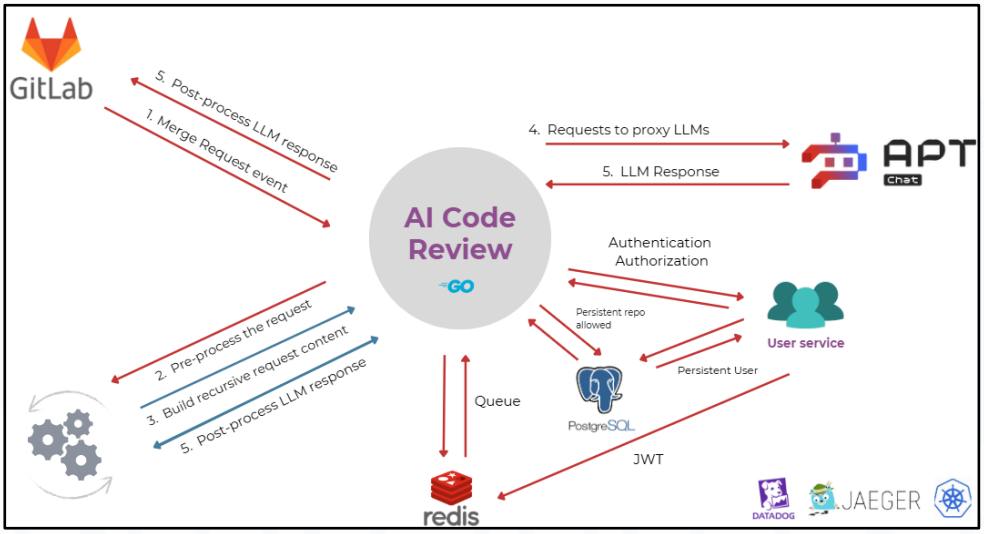
3.2.3.1. AI Code Review

Fungsi utama dari sistem *AI Code Review* adalah untuk menganalisis kode secara otomatis dan memberikan saran atau komentar yang relevan terhadap *merge request* yang diajukan oleh *developer*

melalui GitLab. Proses ini dirancang agar berjalan secara *real-time* dan terintegrasi langsung dengan *workflow* pengembangan perangkat lunak yang telah ada di perusahaan.

Kontribusi teknis yang diberikan pada fase ini meliputi perancangan ulang struktur API *backend*, implementasi *middleware* baru untuk kebutuhan autentikasi dan validasi input, serta integrasi data AI *Review Log* ke dalam basis data sebagai dokumentasi historis dari hasil tinjauan kode. Selain itu, sistem juga telah dioptimalkan melalui penerapan Redis sebagai *caching layer*, serta observabilitas sistem ditingkatkan dengan menggunakan Jaeger untuk *distributed tracing*. Ketahanan sistem turut ditingkatkan dengan penerapan pola *circuit breaker*, dan integritas data dijamin melalui penerapan pustaka validator yang ketat.

Seluruh komponen dan alur proses dalam sistem ini digambarkan pada Gambar 3.9 *Flow AI Code Review*, yang menjelaskan bagaimana integrasi antar layanan dan modul dilakukan secara menyeluruh dan efisien.



Gambar 3. 9 Flow AI Code Review

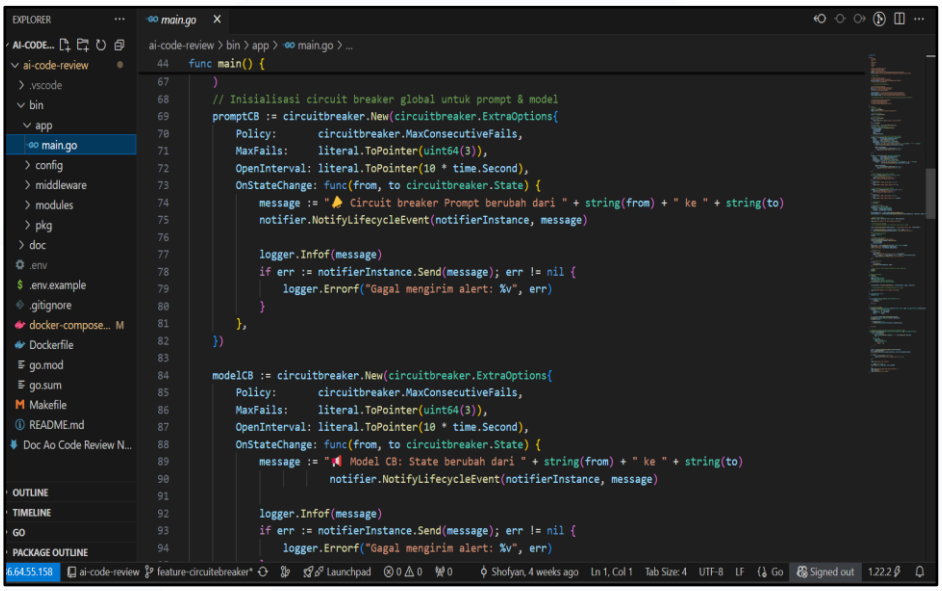
Pada Gambar 3.9 menujukkan *Flow* dari *AI Code Review*, dimulai saat terjadi *Merge Request Event* yang dikirimkan dari GitLab ke layanan utama *AI Code Review*. Permintaan ini kemudian diproses secara internal

untuk diuraikan dan disusun kontennya, yang selanjutnya dikirimkan ke layanan APT *Chat* yang bertugas sebagai *proxy* bagi model *AI* eksternal (LLM). Respons dari model AI kemudian diterima dan diproses kembali, untuk kemudian dikirim secara otomatis ke GitLab dalam bentuk komentar yang menyatu dengan kode.

Di sisi lain, proses autentikasi dan otorisasi dilakukan melalui layanan *User Service*, di mana informasi pengguna dan hak akses diverifikasi menggunakan JWT *Token*, serta didukung oleh PostgreSQL untuk penyimpanan data secara persisten. Proses *caching* juga diterapkan menggunakan Redis untuk mengurangi beban permintaan berulang dan mempercepat waktu respons.

Setiap permintaan dan proses dalam sistem dimonitor secara *end-to-end* menggunakan Jaeger, yang memungkinkan pelacakan performa dan analisis *bottleneck*. Sistem ini juga telah disiapkan agar dapat berkembang ke arah arsitektur berbasis *event-driven* dan dapat diskalakan sesuai kebutuhan tim *developer* di masa depan.

Melalui keterlibatan langsung dalam proyek utama ini, pemahaman teknis dan pengalaman profesional yang mendalam berhasil diperoleh, khususnya dalam membangun sistem *backend* modern yang terintegrasi dengan kecerdasan buatan, serta memperkuat praktik *DevOps* melalui pemanfaatan observabilitas, *containerization*, dan *workflow* otomatis.



Gambar 3. 10 AI Code Review Project

Gambar 3.10 menunjukkan *code* dari *architecture AI Code Review.* Setelah seluruh tahapan pelatihan dan *mini project* terselesaikan, kontribusi dalam proyek utama difokuskan pada pengembangan sistem *AI Code Review,* yaitu sebuah *platform* berbasis kecerdasan buatan yang dirancang untuk menganalisis kualitas kode secara otomatis dan memberikan masukan berbasis model *AI* melalui integrasi dengan *platform* kolaboratif seperti GitLab.

Tugas teknis yang dilibatkan dalam proyek ini meliputi perancangan ulang struktur API *backend* agar selaras dengan skema *input/output* model *AI*. Penyesuaian ini mencakup penyusunan *payload JSON*, penyesuaian format *respons*, serta pengelolaan komunikasi dengan GitLab API. *Endpoint* baru dan *middleware* juga disiapkan untuk mendukung validasi *input*, autentikasi pengguna, dan pemetaan terhadap model *AI* tertentu berdasarkan konteks permintaan pengguna.

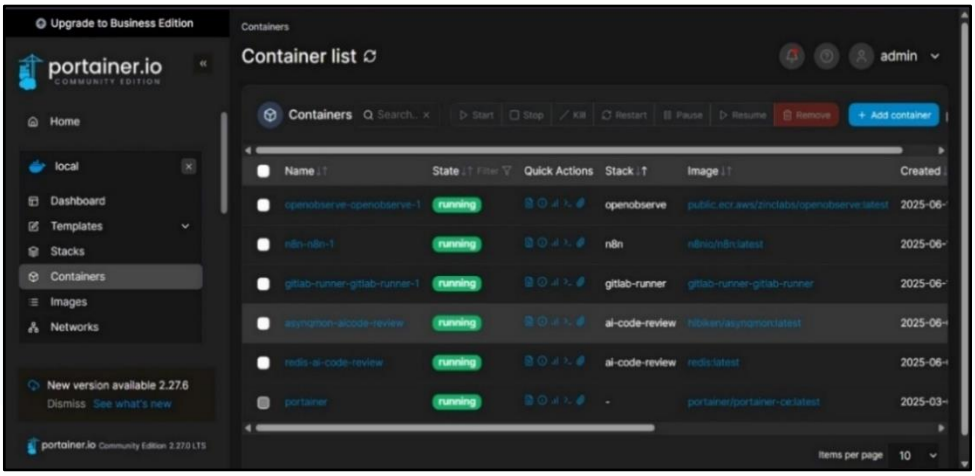
Optimalisasi performa sistem diperkuat melalui penerapan Redis sebagai *caching layer*. Dengan menyimpan hasil sementara untuk permintaan yang serupa atau berulang, beban pemrosesan berhasil

dikurangi secara signifikan, sekaligus meningkatkan kecepatan respons sistem.

Guna menjaga ketersediaan layanan, pola *circuit breaker* diimplementasikan. Ketika sistem mendeteksi kegagalan beruntun terhadap layanan *AI, circuit breaker* memutus sementara jalur komunikasi tersebut dan mencegah terjadinya penurunan performa menyeluruh. Mekanisme ini juga dihubungkan dengan sistem notifikasi untuk mendukung observabilitas *real-time.*

Pemantauan performa sistem dilakukan secara menyeluruh menggunakan Jaeger. Teknologi *distributed tracing* ini memungkinkan pelacakan permintaan secara *end-to-end* dan sangat membantu dalam proses *debugging* serta identifikasi *bottleneck*.

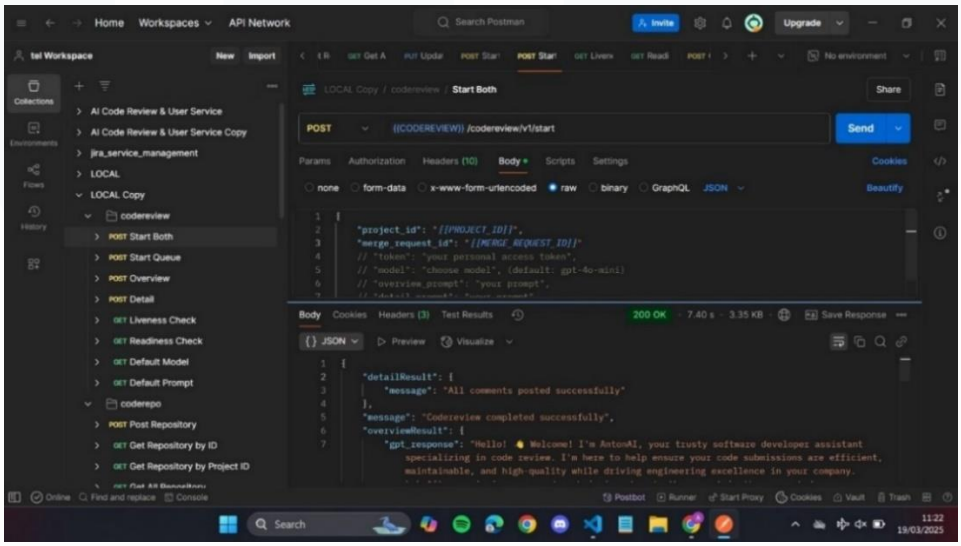
Aspek keamanan data diperkuat melalui validasi input otomatis menggunakan pustaka validator, yang memastikan setiap permintaan telah memenuhi persyaratan struktur, tipe data, dan keamanan *input* sebelum diproses oleh model *AI*.



Gambar 3. 11 Portainer Ai Code Review

Gambar 3.11 Menunjukkan *portainer* yang sedang dijalankan oleh *AI Code Review*. *Portainer* digunakan sebagai *platform* manajemen kontainer Docker yang menyederhanakan pemantauan layanan. Gambar

ini memperlihatkan bahwa berbagai *service* yang membentuk sistem *AI Code Review* (seperti Redis, GitLab *Runner*, *engine AI*, dan *OpenObserve*) telah berhasil dijalankan dalam lingkungan kontainer secara simultan. Status "*running*" menunjukkan kestabilan dan keberhasilan integrasi *deployment* antar *service*. *Portainer* juga memungkinkan *restart, update*, atau *logging container* tanpa perlu akses *CLI* langsung, sangat berguna dalam operasi *DevOps*.

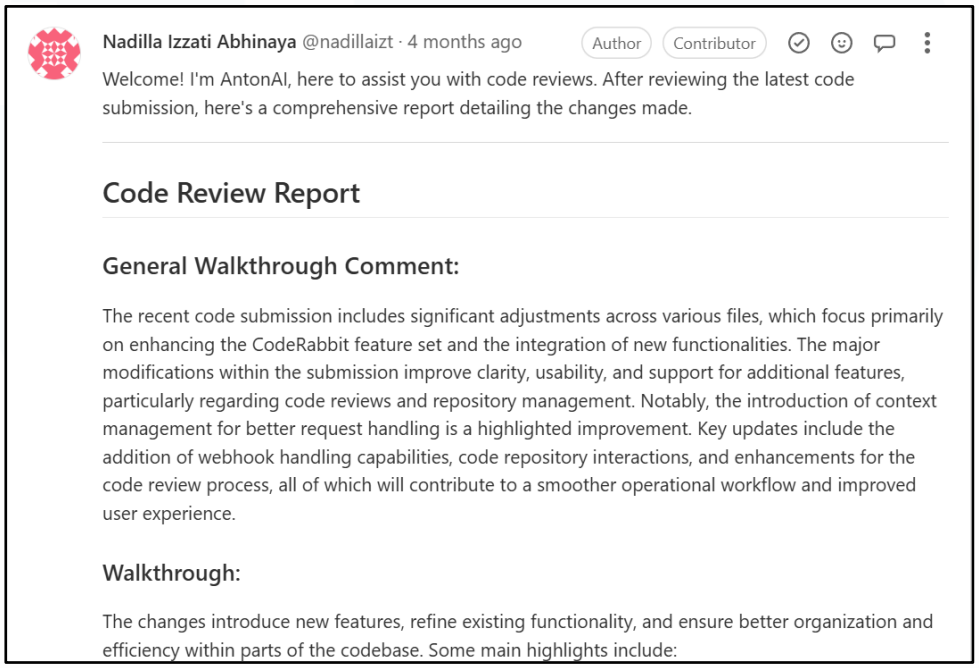


Gambar 3. 12 Postman Testing Ai Code

Gambar 3.12 menampilkan postman dan hasil *testing* dari *AI Code Review*. Postman digunakan untuk menguji fungsionalitas *endpoint /codereview/v1/start*, yang merupakan titik utama dalam *workflow AI Code Review*. Dalam gambar tersebut, terlihat data seperti *project_id, merge_request_id*, serta *prompt* dan model *AI* yang digunakan dikirim melalui metode *POST*. *Respons* API menunjukkan hasil evaluasi kode dari model *AI* (di sini *GPT-4o*), serta komentar yang secara otomatis akan diberikan ke repositori GitLab. Hal ini memperlihatkan keterhubungan penuh antara sistem *backend*, layanan *AI*, dan *platform* kolaboratif pengembang.

Secara keseluruhan, sistem *AI Code Review* ini dirancang dan dikembangkan sebagai solusi *backend* yang cerdas, modular, dan mampu

mendukung proses peninjauan kode secara otomatis, cepat, dan akurat. Pendekatan ini tidak hanya memperkuat kualitas teknis sistem, namun juga meningkatkan skalabilitas dan efisiensi dalam proses pengembangan perangkat lunak secara umum.



Gambar 3. 13 Hasil Overview Merge request AI Code Review

Gambar 13 menampilkan hasil evaluasi kode dalam Mode *Overview* yang dihasilkan oleh sistem *AI Code Review*. Mode ini merupakan salah satu fitur utama yang dirancang untuk memberikan gambaran umum terhadap perubahan kode dalam suatu *merge request*, tanpa perlu meninjau *file* demi *file* secara *manual*. Mode ini sangat bermanfaat dalam skenario di mana *reviewer* atau pengembang membutuhkan ringkasan cepat untuk mengambil keputusan awal terkait validitas dan arah perubahan yang dilakukan.

Dalam gambar ditampilkan bahwa *AI Code Review* memberikan komentar pengantar berbasis *Large Language Model (LLM)*, yang menyatakan bahwa telah terjadi perubahan signifikan terhadap berbagai *file* dalam repositori. Fokus utama perubahan berkaitan dengan penambahan fitur, peningkatan kejelasan arsitektur, serta integrasi fungsi

baru dalam komponen sistem, khususnya pada area pengelolaan *repository* dan proses *code review* itu sendiri. Selain itu, juga disebutkan adanya penambahan fitur *webhook*, perbaikan validasi *status*, serta peningkatan sistem komando *query-handler* untuk interaksi dengan *database*.

Sebagaimana tercermin dari kutipan komentar *AI* dalam bagian "*General Walkthrough Comment*", sistem juga mampu mengidentifikasi bahwa fitur-fitur lama yang tidak lagi relevan telah dihapus, dan dilakukan pemurnian terhadap struktur kode agar lebih ringkas, terjaga kejelasannya, dan mudah dirawat.

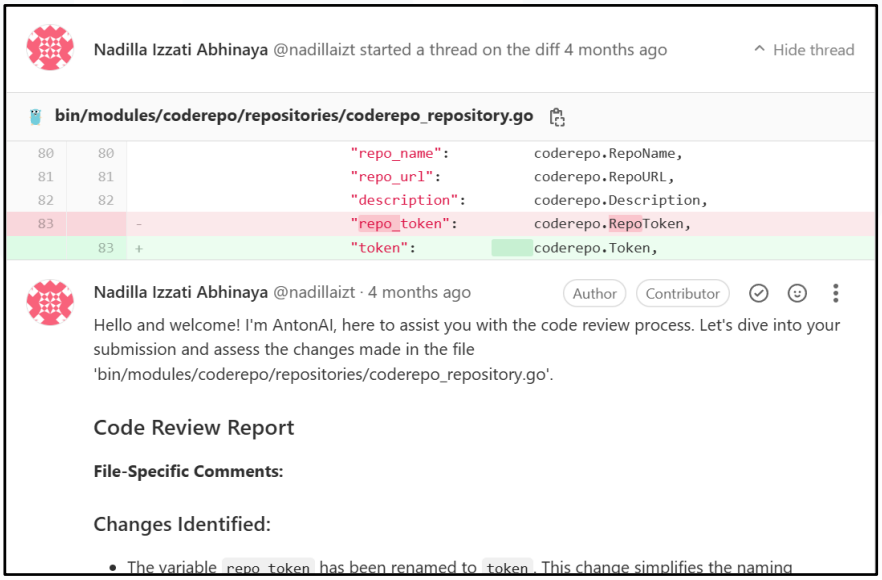
Lebih lanjut, sistem menyajikan tabel ringkasan perubahan (*Changes Summary Table*) yang menguraikan *file* yang terpengaruh beserta deskripsi singkatnya. Misalnya, *file .vscode/settings.json* dihapus untuk membersihkan konfigurasi pengembangan lokal yang tidak lagi diperlukan, dan dokumentasi teknis pada *codeRabbit-analysis.md* diperbarui untuk mencerminkan fitur baru dan hasil perbandingan sistem terkini.

Jika dibandingkan dengan mode *file-level*, *Mode Overview* memiliki keunggulan dalam hal efisiensi waktu dan pemahaman konteks secara makro. Alih-alih memberikan komentar *granular* di setiap bagian kode, sistem menyampaikan narasi strategis dan dampak arsitektural dari perubahan yang terjadi, termasuk bagaimana perubahan tersebut memengaruhi *workflow*, kualitas kode, serta *maintainability* sistem secara keseluruhan.

Fungsi *Mode Overview* juga dapat dikaitkan dengan praktik *lightweight review* dalam industri perangkat lunak, yaitu pendekatan tinjauan yang fokus pada nilai strategis dari perubahan, bukan sekadar *detail* sintaks. Hal ini menjadi sangat penting ketika proses *merge* perlu

dilakukan cepat namun tetap bertanggung jawab terhadap integritas sistem.

Dengan kemampuan ini, sistem *AI Code Review* telah menunjukkan tidak hanya kemampuannya dalam menganalisis kode secara *mikro*, tetapi juga memahami konteks pengembangan secara holistik sebuah pencapaian yang menjadi indikator kuat atas kesiapan teknologi ini untuk diadopsi dalam pipeline *DevOps* berskala besar.



Gambar 3. 14 Hasil File Specific Merge Request AI Code Review Repo File

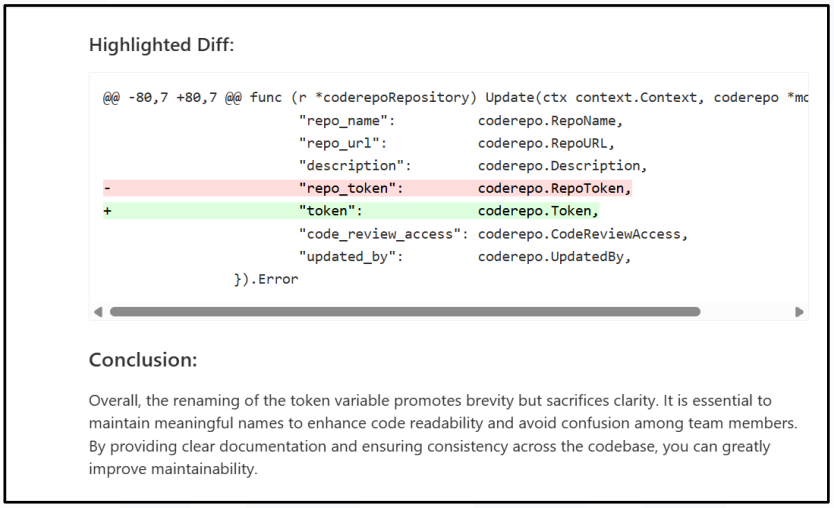
Gambar 3.14 Menjelaskan hasil *AI Code Review* dalam bentuk file, hasil tersebut menampilkan code yang teridentifikasi bermasalah. Dalam kasus ini, *file coderepo_repository.go* dianalisis, dengan fokus utama pada perubahan nama *variabel* dari *repo_token* menjadi *token*. Perubahan ini dinilai sebagai upaya penyederhanaan sintaks, namun sistem memberikan komentar bahwa tindakan ini dapat mengurangi kejelasan konteks semantik, terutama dalam lingkungan kerja yang melibatkan banyak kontributor. Oleh karena itu, *AI* menyarankan agar nama *variabel* diubah menjadi *api_token* untuk menjelaskan maksud *token* secara eksplisit dan mencegah kebingungan.

Analisis teknis yang dihasilkan oleh sistem mencakup:

a). Kejelasan Penamaan: Disarankan penggunaan nama yang lebih deskriptif untuk meningkatkan keterbacaan.

b). Pembaruan Dokumentasi: Penting dilakukan untuk mencerminkan perubahan dan mendukung *maintainability*.

c). Deteksi Potensi *Bug*: Sistem memperingatkan bahwa penggunaan RepoToken di tempat lain dalam kode harus diperbarui secara menyeluruh untuk menghindari *error undefined*.



Gambar 3. 15 File Specific Merge Request AI Code Review highlighted diff

Gambar 3.14 menunjukkan hasil evaluasi sistem *AI Code Review* dalam *Mode Detail*, yaitu *mode* yang secara khusus dirancang untuk melakukan analisis mendalam baris demi baris terhadap *file* yang mengalami perubahan dalam suatu *Merge Request*. Mode ini sangat berguna dalam proses pengembangan perangkat lunak berskala besar, di mana standar penamaan, efisiensi logika, dan struktur kode menjadi aspek yang sangat krusial.

Hasil evaluasi juga divisualisasikan dalam bentuk *highlighted diff*, yang memperlihatkan secara jelas baris kode yang dihapus

(*repo_token*) dan ditambahkan (*token*). Hal ini memperkuat prinsip transparansi dalam proses audit kode dan memungkinkan tim teknis melakukan verifikasi langsung terhadap cakupan perubahan.

Jika dibandingkan dengan *Mode Overview* yang hanya memberikan ringkasan umum terhadap perubahan kode dalam *Merge Request*, *Mode Detail* memberikan informasi lebih teknis dan *granular*, serta memungkinkan sistem:

1). Memeriksa konsistensi penamaan dan konvensi pengkodean internal,

2). Mendeteksi ketidakefisienan logika kode,

3). Memberikan saran perbaikan berbasis konteks fungsional kode.

Secara keseluruhan, *Mode Detail* memperlihatkan kemampuan *AI Code Review* bukan hanya sebagai pemeriksa sintaksis, tetapi sebagai asisten teknis berbasis kecerdasan buatan yang mampu mengevaluasi kualitas semantik kode dan memberikan rekomendasi yang mendalam dalam menjaga standar teknis proyek.

3.3 Kendala yang Ditemukan

Selama pelaksanaan kegiatan magang di PT Telkom Indonesia, sejumlah kendala ditemukan, baik dari sisi teknis maupun *non*-teknis. Berbagai tantangan tersebut merupakan bagian dari proses pembelajaran dan adaptasi terhadap lingkungan kerja *professional* serta penggunaan teknologi yang tergolong baru. Adapun rincian kendala yang dihadapi selama masa magang adalah sebagai berikut:

1). Adaptasi Bahasa Pemrograman (Golang)

Penggunaan bahasa pemrograman Golang menjadi tantangan tersendiri karena belum adanya pengalaman sebelumnya. Diperlukan waktu

untuk memahami sintaksis, konsep *concurrency*, serta pola pengembangan backend berbasis Golang secara efektif.

2). Kompleksitas *Tools* dan Teknologi

Integrasi berbagai *tools* seperti Jaeger, Kafka, Telegram *Bot*, serta penggunaan Docker, Redis, dan Postman menjadi tantangan tersendiri karena keterbatasan pengalaman praktis dalam menggunakannya secara bersamaan dalam satu *project*.

3). Minimnya Pengalaman diposisi *Backend* dan Perpindahan Peran ke *Backend Engineer*

Latar belakang sebelumnya yang lebih berfokus pada analisis data menyebabkan peran sebagai *backend engineer* menjadi hal yang baru. Proses seperti *debugging*, *error tracing*, serta penulisan unit *testing* memerlukan penyesuaian konsep dan peningkatan keterampilan teknis.

4). Skala *Project AI Code Review*

Project utama yang berskala besar menuntut pemahaman mendalam mengenai *pipeline AI*, arsitektur *microservices*, *CI/CD*, serta *deployment* berbasis *cloud*, yang secara kompleksitas jauh melampaui *project* akademik biasa.

5). Time Management

Tantangan muncul dari pelaksanaan berbagai aktivitas secara paralel, seperti menyelesaikan modul pelatihan, membaca literatur teknis, mengerjakan *mini project*, serta terlibat dalam proyek utama. Hal ini menuntut keterampilan manajemen waktu yang baik agar seluruh tanggung jawab dapat diselesaikan secara optimal dan tepat waktu.

Secara keseluruhan, kendala-kendala tersebut memberikan tantangan yang signifikan namun juga berkontribusi besar terhadap peningkatan kapasitas teknis dan *professional*. Kemampuan untuk mengidentifikasi, menganalisis, serta menghadapi hambatan tersebut menjadi bagian penting dalam proses pengembangan kompetensi selama program magang berlangsung.

3.4 Solusi atas Kendala yang Ditemukan

Mengatasi berbagai kendala yang muncul selama pelaksanaan magang, sejumlah strategi adaptif dan solusi praktis diterapkan, baik dari sisi teknis maupun *non*-teknis. Upaya-upaya ini tidak hanya membantu dalam penyelesaian tugas-tugas harian, tetapi juga mendorong peningkatan kompetensi secara menyeluruh. Berikut adalah solusi yang dilakukan berdasarkan jenis kendala yang telah diuraikan sebelumnya:

1). Adaptasi terhadap Bahasa Pemrograman Golang

Berbagai referensi pembelajaran digunakan secara aktif, seperti dokumentasi resmi Golang, video *tutorial* pada *platform* daring, serta latihan langsung melalui *mini project* internal dengan pendekatan *Clean Architecture*. Selain itu, sesi *pair programming* dan *code review* bersama *engineer senior* dimanfaatkan untuk mempercepat proses adaptasi terhadap idiom dan pola pengembangan *backend* menggunakan Golang.

2). Mengatasi Kompleksitas Penggunaan *Tools* dan Teknologi

Pemahaman terhadap *tools* seperti Jaeger, Kafka, Telegram *Bot*, Docker, dan Postman ditingkatkan melalui eksplorasi mandiri yang dilakukan secara terstruktur. Beberapa langkah yang diambil antara lain:

a). Membaca dokumentasi resmi dan artikel *medium*.

b). Mengikuti tutorial praktis dalam bentuk video atau *blog*.

c). Melakukan eksperimen langsung pada lingkungan *development* pribadi.

Hasil eksplorasi juga didokumentasikan agar dapat dijadikan referensi internal dan berkontribusi terhadap pengetahuan kolektif tim.

3). Penyesuaian terhadap Peran Baru sebagai *Backend Engineer*

Untuk menghadapi perpindahan peran dari *data analyst* ke *backend engineer,* pembelajaran materi fundamental seperti *RESTful* API, struktur *folder* pada *Clean Architecture*, serta praktik *debugging* dilakukan secara aktif. Beberapa *tools* seperti Postman dan GORM *Debug* digunakan dalam kegiatan *testing* dan *error tracing*, sementara *unit testing* diterapkan sebagai bagian dari evaluasi kerja mandiri.

4). Mengelola Kompleksitas Skala *Project AI Code Review*

Proyek berskala besar dipecah menjadi *milestone* kecil agar lebih terstruktur dan mudah dikendalikan. Perencanaan harian dan mingguan dilakukan menggunakan *Trello* dan *Notion* untuk memonitor progres serta memecah pekerjaan ke dalam *task* yang *manageable*. Diskusi teknis secara rutin juga dilakukan bersama rekan *engineer* dan data *scientist* guna memperluas pemahaman terhadap arsitektur sistem dan *pipeline AI* yang digunakan.

5). Strategi Manajemen Waktu

Manajemen waktu diterapkan dengan pendekatan berbasis prioritas menggunakan *Eisenhower Matrix* dan teknik *time blocking*. Penggunaan *Google Calendar* dan *task tracker* membantu dalam mengalokasikan waktu secara seimbang antara proyek utama, *side task*, dan aktivitas pembelajaran. Evaluasi berkala terhadap beban kerja dilakukan untuk mengidentifikasi tugas yang dapat ditunda atau didelegasikan, sehingga fokus tetap terjaga pada pekerjaan dengan nilai strategis tinggi.

Melalui penerapan solusi-solusi di atas, seluruh tanggung jawab selama program magang dapat diselesaikan secara optimal. Selain memberikan peningkatan signifikan dalam aspek teknis dan kolaboratif, pengalaman tersebut juga memperkuat keterampilan manajemen diri dan menjadi bekal berharga dalam pengembangan karier *professional* di bidang *software engineering* dan sistem berbasis kecerdasan buatan.