

**APLIKASI MANAJEMEN TUGAS *MULTI-TENANT* BERBASIS
WEB MENGGUNAKAN SISTEM *DYNAMIC CONTEXT-AWARE*
ROLE BASED ACCESS CONTROL (RBAC)
(Studi Kasus : PT Bharata Internasional Pharmaceutical)**

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata Satu
Program Studi Informatika



**ALDIKA GAGAH PRASETYA
22EO10042**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026**

**APLIKASI MANAJEMEN TUGAS *MULTI-TENANT* BERBASIS
WEB MENGGUNAKAN SISTEM *DYNAMIC CONTEXT-AWARE*
ROLE BASED ACCESS CONTROL (RBAC)
(Studi Kasus : PT Bharata Internasional Pharmaceutical)**

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata Satu
Program Studi Informatika



**ALDIKA GAGAH PRASETYA
22EO10042**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026**

HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya menyatakan bahwa penulisan skripsi dengan judul “Aplikasi Manajemen Tugas *Multi-Tenant* Berbasis Web Menggunakan Sistem *Dynamic Context-Aware Role Based Access Control* (RBAC) (Studi Kasus : PT Bharata Internasional Pharmaceutical)” adalah hasil karya saya dengan arahan dari pembimbing dan belum diajukan kepada pihak manapun. Sumber informasi yang dikutip dalam skripsi ini telah dicantumkan dalam Daftar Pustaka.

Demikian pernyataan ini dibuat dengan sebenarnya. Apabila di kemudian hari terdapat ketidaksesuaian dalam pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.



Cilacap, 27 Juli 2026

Aldika Gagah Prasetya
22EO10042

LEMBAR PENGESAHAN

Skripsi Saudara

Nama : Aldika Gogah Prasetya
NIM : 22EO10042
Fakultas/Prodi : Fakultas MIKOM / Informatika
Judul : Aplikasi Manajemen Tugas *Multi-Tenant* Berbasis Web Menggunakan Sistem *Dynamic-Context Aware Role Based Access Control* (RBAC) (Studi Kasus : PT Bharata Internasional Pharmaceutical)

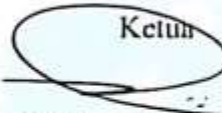
Telah disidangkan oleh Dewan Penguji Fakultas Matematika dan Ilmu Komputer Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap pada hari / tanggal :
Senin, 27 Juli 2026

Dan dapat diterima sebagai pemenuhan tugas akhir mahasiswa Program Strata Satu (S.1) Program Studi Informatika (S.Kom.) Fakultas Matematika dan Ilmu Komputer (FMIKOM) pada Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap.

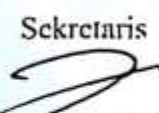
Cilacap, 12 Agustus 2026

Dewan Sidang

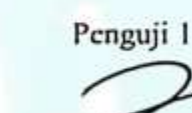
Ketua


Tri Anggoro, M.Kom.
NIDN. 0601128103

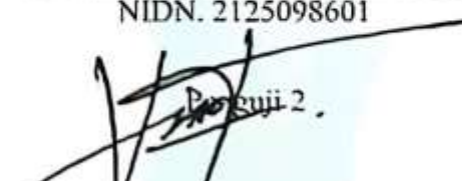
Sekretaris


M.T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

Penguji I


M.T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

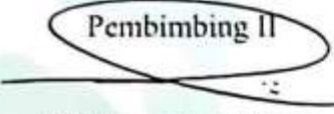
Penguji 2


M. Novriansyah Dasaprawira, M.Pd.
NIDN. 0610119301

Pembimbing I


Lasim, M.Kom.
NIDN. 0605048602

Pembimbing II


Tri Anggoro, M.Kom.
NIDN. 0601128103

Mengetahui,

Dekan Fakultas Matematika dan Ilmu Komputer


M.T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601



HALAMAN NOTA KONSULTAN

Mochamad T.A. Aziz Zein, S.Si., M.Kom.

Dosen Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali Cilacap

NOTA KONSULTAN

Hal : Skripsi Saudara Aldika Gagah Prasetya
Lampiran : -

Yth.
Dekan Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali Cilacap
di Cilacap

Assalamu'alaikum Wr. Wb.

Setelah membaca, memeriksa dan melakukan perbaikan seperlunya maka skripsi saudara/i:

Nama : Aldika Gagah Prasetya
NIM : 22EO10042
Prodi : Informatika
Judul : Aplikasi Manajemen Tugas *Multi-Tenant* Berbasis Web
Menggunakan Sistem *Dynamic-Context Aware Role Based Access Control* (RBAC) (Studi Kasus : PT Bharata Internasional Pharmaceutical)

Dapat diajukan ke Fakultas Matematika dan Ilmu Komputer, Universitas Nahdlatul Ulama Al Ghazali Cilacap untuk memenuhi syarat memperoleh gelar Strata Satu (S1).

Wassalamu'alaikum Wr. Wb.

Cilacap, 12 Agustus 2026
Konsultan



M.T.A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

HALAMAN MOTO

“Allah adalah tempat meminta segala sesuatu”
(Qs. Al-Ikhlas :2)

"It always seems impossible until it's done."
(Nelson Mandela)

HALAMAN PERSEMBAHAN

Segala puji bagi Allah SWT, Rabb semesta alam yang senantiasa memberikan karunia sehingga penulis mampu menyelesaikan penulisan skripsi ini. Karya ini saya persembahkan kepada :

1. Orang tua tercinta Bapak Mujiman dan Ibu Resmi Isnaeni yang selalu mendoakan, mensupport anaknya agar selalu yakin dengan proses yang sedang dijalani, dan segala upaya yang dilakukan agar kebutuhan anaknya selalu bisa dipenuhi.
2. Mbah putri dan mbah kakung tercinta, orang yang sangat baik hati terhadap cucunya, selalu memberi doa, nasehat dan memberikan petuah baik dalam kehidupan.
3. Kakak tercinta Andika Prasetya dan saudara perempuan tersayang Alvita Lily Prasasty yang selalu memberikan do'a dan semangat tiada henti di setiap langkahku, sehingga keyakinan dalam diri ini selalu ada.
4. Umi Wahidah sahabat yang selalu ada, selalu mendukung dan orang yang bisa dipercaya dalam hal apapun selama proses ini, semoga hal baik selalu menyertaimu.
5. Teman teman seperjuangan di LKP ESA, tidak hanya belajar *emglish language* tapi tempat dimana pembelajaran kehidupan, kedisiplinan dan realistik kehidupan mulai terbentuk.
6. Teman teman informatika 8B yang luar biasa dalam mensupport satu sama lain, semoga kalian menemukan jalan sukses masing-masing.
7. Keluarga FMIKOM Angkatan 2022 yang selalu memberikan keceriaan, kebersamaan dan motivasi.
8. Keluarga program studi Teknik Informatika, Sistem Informasi, Matematika yang saya banggakan
9. Seluruh teman UNUGHA yang telah memberikan do'a, dukungan, dan semangat.

KATA PENGANTAR

Puji syukur kepada Allah SWT atas segala limpahan nikmat dan karuniaNya, sehingga penulis dapat menyelesaikan penulisan skripsi. Sholawat dan salam senantiasa penulis haturkan kepada Nabi Muhammad SAW sebagai pembimbing seluruh umat manusia.

Skripsi ini tidak akan selesai tanpa adanya bantuan dari banyak pihak, karena itu penulis menyampaikan terima kasih kepada :

1. Bapak M.T. Abdul Aziz Zain. M.Kom, Dekan FMIKOM UNUGHA
2. Bapak Lasimin, M.Kom, pembimbing I yang telah dengan sabar memberikan bimbingan dan petunjuk dalam menyelesaikan skripsi ini.
3. Bapak Tri Anggoro, M.Kom, pembimbing II yang telah dengan sabar memberikan bimbingan dan petunjuk dalam menyelesaikan skripsi ini.
4. Bapak Varry, S.T., M.Kom., sebagai Pembimbing Akademis
5. PT Bharata Internasional Pharmaceutical Cipari sebagai pelaksana skripsi
6. Bapak-Ibu dosen Program Studi Informatika FMIKOM UNUGHA
7. Semua pihak yang tidak dapat kami sebutkan satu persatu

Semoga Allah SWT membalas jerih payah dan pengorbanan yang telah diberikan dengan balasan yang lebih baik. Amiin. Penulis berharap semoga karya kecil ini bermanfaat bagi pembaca.

Cilacap, 17 Juli 2026



Penulis

Aldika Gagah Prasetya

ABSTRAK

Aldika Gagah Prasetya. *Aplikasi Manajemen Tugas Multi-Tenant Berbasis Web Menggunakan Sistem Dynamic Context-Aware Role-Based Access Control (RBAC) (Studi Kasus: PT Bharata Internasional Pharmaceutical)*. Dibimbing oleh Lasimin, M.Kom. dan Tri Anggoro, M.Kom.

Perkembangan sistem manajemen tugas berbasis web pada lingkungan *multi-tenant* menuntut mekanisme pengendalian akses yang mampu menjaga keamanan data antar divisi tanpa mengurangi fleksibilitas proses bisnis. Pada PT Bharata Internasional Pharmaceutical, proses pengelolaan tugas masih dilakukan secara manual sehingga menyulitkan pelacakan pekerjaan, meningkatkan risiko akses tidak sah, serta belum memiliki mekanisme validasi akses berbasis konteks dan pencatatan aktivitas pengguna. Selain itu, penerapan *Role-Based Access Control* (RBAC) konvensional yang hanya mempertimbangkan peran pengguna belum mampu mengakomodasi kebutuhan otorisasi dinamis pada lingkungan *multi-tenant*.

Penelitian ini bertujuan mengembangkan aplikasi manajemen tugas *multi-tenant* berbasis web dengan menerapkan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC). Metode penelitian yang digunakan adalah *Design Science Research* (DSR) yang meliputi identifikasi masalah, penetapan tujuan solusi, perancangan dan pengembangan artefak, demonstrasi, pengujian, serta evaluasi. Sistem dikembangkan menggunakan bahasa pemrograman Go dengan framework Fiber, MongoDB sebagai basis data, *JSON Web Token* (JWT) untuk autentikasi, serta *WebSocket* untuk sinkronisasi data secara *real-time*. Mekanisme otorisasi menggunakan *role* sebagai dasar RBAC dengan mempertimbangkan tiga konteks, yaitu *ownership*, *tenant boundary*, dan *resource state* melalui validasi *workflow transition*. Hasil pengujian menggunakan metode *Black Box Testing* terhadap 16 skenario yang mencakup *authentication*, *tenant boundary*, *ownership*, *resource state*, *audit trail*, dan *real-time notification* menunjukkan 16 skenario berstatus valid dan 0 skenario tidak valid, sehingga diperoleh tingkat validitas sebesar 100%. Hasil tersebut menunjukkan bahwa mekanisme yang dikembangkan dapat menjalankan validasi akses berdasarkan konteks operasional, menjaga isolasi data antar *tenant*, mengendalikan perubahan status tugas sesuai alur kerja, serta mendukung pencatatan aktivitas dan penyampaian informasi secara *real-time*. Dengan demikian, artefak yang dikembangkan dapat mendukung pengelolaan tugas secara terstruktur pada lingkungan organisasi multi-divisi.

Kata kunci: *audit trail, dynamic context-aware RBAC, multi-tenant, sistem manajemen tugas, workflow validation.*

ABSTRACT

Aldika Gagah Prasetya. *Web-Based Multi-Tenant Task Management Application Using a Dynamic Context-Aware Role-Based Access Control (RBAC) System (Case Study: PT Bharata Internasional Pharmaceutical). Supervised by Lasimin, M.Kom. and Tri Anggoro, M.Kom.*

The development of web-based task management systems in multi-tenant environments requires an access control mechanism capable of maintaining data security across divisions without reducing business process flexibility. At PT Bharata Internasional Pharmaceutical, task management processes were previously conducted manually, making task tracking difficult, increasing the risk of unauthorized access, and lacking context-based access validation and user activity recording mechanisms. In addition, conventional Role-Based Access Control (RBAC), which primarily considers user roles, is limited in addressing dynamic authorization requirements in a multi-tenant environment.

*This study aims to develop a web-based multi-tenant task management application by implementing a Dynamic Context-Aware Role-Based Access Control (RBAC) mechanism. The research employed the Design Science Research (DSR) method, consisting of problem identification, definition of solution objectives, artifact design and development, demonstration, testing, and evaluation. The system was developed using Go with the Fiber framework, MongoDB as the database, JSON Web Token (JWT) for authentication, and WebSocket for real-time data synchronization. The authorization mechanism uses role as the basis of RBAC while considering three dynamic contexts: ownership, tenant boundary, and resource state through workflow transition validation. The results of Black Box Testing on **16 test scenarios** covering authentication, tenant boundary, ownership, resource state, audit trail, and real-time notification showed that 16 scenarios were valid and 0 scenarios were invalid, resulting in a 100% validity rate. These results indicate that the developed mechanism can perform access validation based on operational context, maintain data isolation between tenants, control task status changes according to the defined workflow, and support activity recording and real-time information delivery. Therefore, the developed artifact can support structured task management in a multi-division organizational environment.*

Keywords: *audit trail, dynamic context-aware RBAC, multi-tenant, task management system, workflow validation.*

DAFTAR ISI

HALAMAN PERNYATAAN KEASLIAN SKRIPSI	Error! Bookmark not defined.
LEMBAR PENGESAHAN	Error! Bookmark not defined.
HALAMAN NOTA KONSULTAN	i
HALAMAN MOTO	i
HALAMAN PERSEMBAHAN	ii
KATA PENGANTAR	Error! Bookmark not defined.
ABSTRAK	i
<i>ABSTRACT</i>	ii
DAFTAR ISI	iii
DAFTAR GAMBAR	i
DAFTAR TABEL	i
BAB I PENDAHULUAN	2
A. Latar Belakang	2
B. Rumusan Masalah	5
C. Batasan Masalah	5
D. Tujuan Penelitian	6
E. Manfaat Penelitian	6
BAB II TINJAUAN PUSTAKA	7
A. Penelitian Terkait	7
B. Landasan Teori	18
a) Mekanisme Kontrol Akses (<i>Access Control</i>)	18
b) Arsitektur Multi-Tenancy dan Isolasi Data	19
c) Paradigma Keamanan Zero Trust	20
d) Teknologi Pengembangan Sistem	20
e) <i>Clean Architecture</i>	21
f) <i>JSON Web Token (JWT)</i> pada Skema <i>Zero Trust</i>	21
h. Teknologi Kontainerisasi (Docker)	22
BAB III METODOLOGI	24
A. Waktu dan Tempat Penelitian	24
B. Alat dan Bahan	24

C. Prosedur Penelitian	25
a. Tahap Identifikasi Masalah	26
b. Tahap Penetapan Tujuan Solusi	28
c. Tahap Perancangan dan Pengembangan Artefak	29
d. Tahap Demonstrasi Artefak	67
e. Tahap Pengujian Artefak	68
f. Tahap Evaluasi	73
D. Jadwal Penelitian	74
BAB IV HASIL DAN PEMBAHASAN	75
A. Hasil	75
1. Hasil Pengembangan Artefak	75
a) Pengembangan Infrastruktur Backend	75
b) Pengembangan Modul <i>Workspace</i>	76
c) Implementasi Mekanisme Pengelolaan Tugas	79
d) Pengembangan Mekanisme Keamanan	83
e) Pengembangan Modul Pendukung	88
2. Hasil Pengujian Mekanisme <i>Dynamic Context-Aware Role-Based Access Control (RBAC)</i>	91
a) Pengujian <i>Authentication</i>	91
b) Pengujian <i>Tenant Boundary</i>	92
c) Pengujian <i>Ownership</i>	93
d) Pengujian <i>Resource State</i>	94
e) Pengujian <i>Audit Trail</i>	95
f) Pengujian <i>Real-Time Notification</i>	96
g) Rekapitulasi Hasil Pengujian <i>Black Box</i>	96
h) Sintesis Hasil Evaluasi	97
B. Pembahasan	98
1) Pembahasan Hasil Pengembangan <i>Backend</i> Sistem Manajemen Tugas <i>Multi-Tenant</i>	98
2) Pembahasan Hasil Penerapan <i>Dynamic Context-Aware Role-Based Access Control RBAC</i>	98
3) Kontribusi Penelitian terhadap Sistem Manajemen Tugas <i>Multi-Tenant</i>	99
BAB V KESIMPULAN	101
A. Kesimpulan	101

B. Saran	101
DAFTAR PUSTAKA	102
LAMPIRAN	105

DAFTAR GAMBAR

Gambar 2. 1 Mind Map	17
Gambar 3. 1 Flowchart Penelitian.....	25
Gambar 3. 2 Use Case Diagram Superadmin.....	30
Gambar 3. 3 Use Case Diagram Supervisor.....	32
Gambar 3. 6 Use Case Diagram Staff	34
Gambar 3. 7 Activity Diagram Login	35
Gambar 3. 8 Activity Diagram membuat akun supervisor	37
Gambar 3. 9 Activity Diagram membuat akun staff	39
Gambar 3. 10 Activity Diagram membuat workspace divisi.....	41
Gambar 3. 11 Activity Diagram pengajuan task.....	43
Gambar 3. 12 Activity Diagram approval task	45
Gambar 3. 13 Activity Diagram perubahan status task.....	47
Gambar 3. 14 Sequence diagram login dan validasi token JWT.....	49
Gambar 3. 15 Sequence diagram manajemen space dengan validasi RBAC context aware	51
Gambar 3. 16 Sequence Diagram Membuat Task Baru dengan Real-time Notification	53
Gambar 3. 17 Sequence Diagram Perubahan Status Task	55
Gambar 3. 18 Perancangn Arsitektur Sistem	57
Gambar 4. 1 Struktur Folder Backend.....	75
Gambar 4. 2 Implementasi Pembentukan Workspace	77
Gambar 4. 3 Implementasi Model Data Workspace	78
Gambar 4. 4 Tampilan Halaman Workspace	78
Gambar 4. 5 Implementasi Model Data Task	79
Gambar 4. 6 Implementasi Mekanisme Workflow Tugas	80
Gambar 4. 7 Model Data Tugas	81
Gambar 4. 8 Implementasi Logika Bisnis Pengelolaan Tugas	82
Gambar 4. 9 Tampilan Pengelolaan Tugas	82
Gambar 4. 10 Implementasi Authentication	83
Gambar 4. 11 Tampilan Halaman Authentication	84
Gambar 4. 12 Implementasi Route Level Authorization	85
Gambar 4. 13 Space Level Authorization	86
Gambar 4. 14 Operation Level Authorization.....	87
Gambar 4. 15 Implementasi Isolasi Akses Pengguna Antar Divisi	88
Gambar 4. 16 Implementasi Audit Trail	89
Gambar 4. 17 Audit Trail	89
Gambar 4. 18 Implementasi Websocket Service	90
Gambar 4. 19 Implementasi Notifikasi	91

DAFTAR TABEL

Tabel 2. 1 Penelitian Terkait	7
Tabel 2. 2 Gap & Novelty	12
Tabel 3. 1 Alat dan Bahan.....	24
Tabel 3. 2 Collection Users.....	58
Tabel 3. 3 Collection spaces	59
Tabel 3. 4 Collection Tasks.....	60
Tabel 3. 5 Collection Tasks.....	63
Tabel 3. 6 Collection Audits	64
Tabel 3. 7 Rencana Pengujian Umum.....	69
Tabel 3. 8 Skenario Pengujian Spesifik	70
Tabel 3. 9 Tabel Penelitian.....	74
Tabel 4. 1 Komponen Infrastruktur Backend.....	76
Tabel 4. 2 Hierarki Role Sistem.....	83
Tabel 4. 3 Hasil Pengujian Authentication	92
Tabel 4. 4 Hasil Pengujian Tenant Boundary	92
Tabel 4. 5 Hasil Pengujian Ownership.....	93
Tabel 4. 6 Pengujian Resource State.....	94
Tabel 4. 7 Pengujian Audit Trail.....	95
Tabel 4. 8 Pengujian Real-Time Notification	96
Tabel 4. 9 Rekapitulasi Hasil Pengujian	97
Tabel 4. 10 Keunggulan Penerapan Dynamic Context Aware RBAC.....	98
Tabel 4. 11 Kontribusi Penelitian terhadap Sistem Manajemen Tugas Multi-Tenant.....	99

BAB I PENDAHULUAN

A. Latar Belakang

Transformasi digital dalam komputasi awan (*cloud computing*) menuntut efisiensi operasional yang tinggi, salah satunya melalui digitalisasi sistem manajemen tugas berbasis *Software as a Service* (SaaS). Model SaaS memungkinkan organisasi untuk menskalakan alur kerja secara dinamis tanpa terbebani biaya investasi perangkat keras awal yang besar [1]. Selain itu, model ini juga mengurangi kompleksitas instalasi dan pemeliharaan sistem karena seluruh infrastruktur dikelola oleh penyedia layanan [2]. Dengan demikian, SaaS menjadi salah satu pendekatan utama dalam meningkatkan efisiensi operasional dan fleksibilitas sistem informasi organisasi [3].

Untuk mendukung efisiensi tersebut, pengembangan sistem SaaS modern umumnya menerapkan arsitektur *multi-tenant*, yaitu suatu pendekatan yang memungkinkan banyak pengguna atau organisasi berbagi satu sistem yang sama dengan tetap mempertahankan isolasi data masing-masing [4]. Melalui implementasi komponen teknis pada lapisan data dan komputasi yang tepat, arsitektur ini mampu meningkatkan skalabilitas sistem hingga 70% serta menekan biaya operasional sebesar 60% [4]. Namun, penggunaan infrastruktur bersama (*shared infrastructure*) menjadikan mekanisme pengendalian akses sebagai salah satu aspek yang paling kritis dalam menjaga keamanan sistem. Setiap permintaan akses harus dipastikan hanya dapat menjangkau sumber daya yang menjadi hak pengguna sesuai dengan batas organisasi (*tenant boundary*) agar tidak terjadi kebocoran data antar-tenant (*cross-tenant data leakage*).

Permasalahan tersebut menjadi semakin kompleks pada sistem manajemen tugas yang melibatkan banyak pengguna dengan tanggung jawab berbeda dalam satu lingkungan *multi-tenant*. Pada sistem seperti ini, keputusan otorisasi tidak cukup hanya ditentukan berdasarkan peran (*role*) pengguna, tetapi juga dipengaruhi oleh hubungan pengguna terhadap tugas yang diakses (*ownership*), kondisi atau tahapan pekerjaan (*resource state*), serta batas organisasi tempat data berada (*tenant boundary*). Oleh karena itu, mekanisme kontrol akses yang mampu mempertimbangkan konteks operasional secara dinamis menjadi kebutuhan penting dalam sistem manajemen tugas berbasis *multi-tenant*.

Salah satu tantangan dalam arsitektur *multi-tenant* adalah menjaga isolasi data serta memastikan setiap akses dilakukan sesuai hak akses pengguna. Dalam praktiknya, model ini rentan terhadap kebocoran data lintas tenant dan pelanggaran otorisasi akibat miskonfigurasi hak akses. Permasalahan tersebut menunjukkan bahwa mekanisme kontrol akses tidak hanya memverifikasi identitas pengguna, tetapi juga mampu memvalidasi konteks akses terhadap sumber daya yang digunakan [5].

Sebagian besar sistem informasi konvensional masih menggunakan pendekatan *Role-Based Access Control* (RBAC) statis sebagai mekanisme pengelolaan hak akses pengguna. Meskipun RBAC efektif pada sistem yang sederhana, model ini memiliki keterbatasan dalam lingkungan *multi-tenant* karena keputusan otorisasi hanya didasarkan pada peran pengguna tanpa mempertimbangkan konteks sumber daya yang diakses [6]. Selain itu, peningkatan jumlah pengguna dan variasi kebutuhan akses dapat menyebabkan kompleksitas jumlah peran (*role explosion*) yang mempersulit tata kelola keamanan [7]. akibatnya, RBAC statis belum mampu mengakomodasi kebutuhan validasi berdasarkan kepemilikan tugas (*ownership*), status sumber daya (*resource state*), dan batas penyewa (*tenant boundary*) yang diperlukan dalam proses manajemen tugas *multi tenant*.

Seiring dengan berkembangnya kebutuhan keamanan sistem berbasis cloud, pendekatan keamanan modern mulai mengarah pada konsep *Zero Trust Architecture*. Pendekatan ini mengadopsi prinsip *never trust, always verify*, yaitu setiap permintaan akses harus divalidasi secara berkelanjutan berdasarkan identitas dan konteks akses [8]. Dengan demikian, paradigma *zero trust* menjadi landasan dalam pengembangan mekanisme otorisasi dinamis sehingga keputusan akses tidak hanya ditentukan oleh peran pengguna, tetapi juga mempertimbangkan *ownership*, *resource state*, dan *tenant boundary* pada setiap permintaan akses.

Kebutuhan akan isolasi data dan manajemen tugas yang terstruktur ditemukan pada PT Bharata Internasional Pharmaceutical. Sebagai perusahaan industri kosmetik dan obat herbal berskala nasional, perusahaan ini memiliki struktur organisasi kompleks yang terdiri dari 10 divisi operasional. Kompleksitas ini menuntut pemisahan hak akses yang sangat ketat. Dalam skema ini, setiap divisi mempresentasikan sebuah entitas penyewa (*tenant*) yang memerlukan isolasi data mandiri dalam satu ekosistem sistem manajemen tugas.

Berdasarkan hasil observasi, pengelolaan alur pengajuan tugas masih dilakukan secara manual melalui komunikasi lisan, aplikasi pesan instan seperti WhatsApp, dan pencatatan sederhana. Kondisi ini menyebabkan kesulitan dalam pelacakan tugas, koordinasi antar divisi, serta pengendalian aktifitas pengguna. Penelitian menunjukkan bahwa penggunaan media komunikasi informal dalam manajemen tugas mengakibatkan 70% proyek gagal mencapai akibat inefisiensi koordinasi dan kendala dalam pelacakan progres [9]. Selain berdampak pada efektivitas koordinasi, kondisi tersebut juga menyulitkan organisasi dalam menerapkan mekanisme otorisasi yang konsisten sehingga meningkatkan risiko akses tidak sah dan pelanggaran *tenant boundary* antar-divisi.

Meskipun berbagai platform manajemen tugas berbasis SaaS seperti Trello, Notion dan Asana telah menyediakan fitur kolaborasi dan pengelolaan proyek [10] [11], platform tersebut dikembangkan untuk memenuhi kebutuhan organisasi secara umum. Sementara itu, PT Bharata Internasional Pharmaceutical memerlukan

mekanisme otorisasi yang dapat disesuaikan dengan kebijakan internal perusahaan, khususnya dalam memvalidasi hak akses berdasarkan *role*, *ownership*, *resource state*, dan *tenant boundary*. Selain itu, penggunaan platform pihak ketiga juga membatasi fleksibilitas organisasi dalam mengembangkan mekanisme keamanan sesuai kebutuhan operasional perusahaan, serta menimbulkan ketergantungan terhadap penyedia layanan (*vendor lock-in*) dan keterbatasan kendali terhadap pengolahan data organisasi [12].

Permasalahan ini menjadi semakin relevan mengingat PT Bharata Internasional Pharmaceutical telah memiliki infrastruktur server internal yang mandiri. Ketersediaan tersebut seharusnya dapat dimanfaatkan untuk mengimplementasikan sistem kustom dengan model *deployment on-premise*. Langkah ini diambil untuk menjaga integritas dan keamanan data secara optimal dalam kendali penuh organisasi sekaligus memitigasi risiko kedaulatan data yang sering menjadi kendala pada platform SaaS public. Dengan demikian, diperlukan pengembangan sistem internal yang mengadopsi fleksibilitas fitur manajemen proyek sebagaimana dikembangkan dalam konsep SaaS [13], serta menerapkan mekanisme otorisasi yang mampu menyesuaikan keputusan akses berdasarkan konteks operasional dinamis.

Dalam pengembangan sistem manajemen tugas berbasis *multi-tenant*, mekanisme otorisasi merupakan aspek krusial karena miskonfigurasi pada bagian ini berpotensi memicu kebocoran data antar-divisi (*cross-tenant data leakage*). Berdasarkan studi literatur terhadap penelitian dalam lima tahun terakhir, ditemukan beberapa celah penelitian (*research gap*) yang menjadi landasan posisi penelitian ini.

Pertama, penelitian oleh [8] menekankan pentingnya paradigma *zero trust* dalam lingkungan *multi-tenant* untuk validasi berkelanjutan, namun focus penelitian tersebut masih berada pada kerangka kerja keamanan global di level infrastruktur jaringan, bukan pada implementasi logika otorisasi di tingkat operasional fungsional aplikasi. Kedua, penelitian oleh [9] telah merancang sistem manajemen proyek berbasis web dengan *multi-level* RBAC, tetapi mekanisme otorisasi yang digunakan masih bersifat statis karena hanya mempertimbangkan peran pengguna. Dengan demikian, masih diperlukan mekanisme otorisasi yang mampu mengintegrasikan validasi *role*, *ownership*, *resource state*, dan *tenant boundary* dalam proses pengambilan keputusan akses pada sistem manajemen tugas berbasis *multi-tenant*.

Berdasarkan permasalahan dan *research gap* tersebut, penelitian ini mengusulkan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) sebagai model otorisasi dinamis yang mengintegrasikan validasi *role*, *ownership*, *resource state*, dan *tenant boundary* dalam proses pengambilan keputusan akses. Mekanisme tersebut diimplementasikan pada aplikasi manajemen

tugas berbasis web sebagai media implementasi untuk mengevaluasi model otorisasi yang diusulkan. Dengan demikian, kontribusi utama penelitian ini terletak pada pengembangan model otorisasi dinamis yang mampu menyesuaikan keputusan akses berdasarkan konteks operasional pada lingkungan *multi-tenant*.

Berdasarkan urgensi tersebut, penelitian ini diangkat dengan judul “Aplikasi Manajemen Tugas *Multi-Tenant* Berbasis Web Menggunakan Sistem *Dynamic Context Aware Role-Based Access Control*”.

B. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah:

- a. Bagaimana membangun sistem manajemen tugas berbasis *multi tenant* sebagai solusi terhadap proses pengelolaan tugas konvensional di PT. Bharata Internasional Pharmaceutical?
- b. Bagaimana hasil pengujian mekanisme otorisasi *Dynamic Context-Aware Role-Based Access Control (RBAC)* berbasis tiga variabel konteks (*ownership, resource state, dan tenant boundary*) dalam mengelola otorisasi dan isolasi data antar divisi pada sistem manajemen tugas *multi tenant* PT Bharata Internasional Pharmaceutical?

C. Batasan Masalah

Batasan masalah membahas mengenai :

- a. Penelitian difokuskan pada pengembangan *backend* API menggunakan pendekatan *clean architecture*. Antarmuka pengguna (*frontend*) hanya digunakan sebagai media interaksi dan pengujian sistem jika dibutuhkan dalam pemaparan serta tidak menjadi fokus pengembangan maupun evaluasi penelitian.
- b. Implementasi arsitektur *multi-tenant* dibatasi pada model *Single Database – Shared Schema* pada basis data MongoDB untuk menjaga isolasi data antar divisi.
- c. Mekanisme otorisasi sistem dibatasi pada penerapan *Dynamic Context-Aware RBAC*, dengan evaluasi berbasis tiga variabel konteks : *Ownership, Resource State, dan Tenant Boundary*.
- d. Sistem autentikasi pengguna dikembangkan secara mandiri (*stateless*) menggunakan JSON Web Token (JWT), tanpa melibatkan integrasi sistem autentikasi pihak ketiga seperti *Single Sign-On (SSO)*.
- e. Pengujian keamanan sistem difokuskan pada evaluasi mekanisme otorisasi dan isolasi data antar tenant (*cross tenant data leakage*) pada tingkat perangkat lunak, tanpa melibatkan pengujian keamanan pada lapisan jaringan (*network penetration testing*).

D. Tujuan Penelitian

Tujuan Penelitian ini adalah :

- a. Mengetahui hasil perancangan dan implementasi sistem manajemen tugas berbasis *multi-tenant* sebagai solusi terhadap proses pengelolaan tugas konvensional di PT Bharata Internasional Pharmaceutical.
- b. Mengetahui hasil pengujian mekanisme otorisasi *Dynamic Context-Aware Role-Based Access Control* (RBAC) berbasis tiga variabel konteks, yaitu *ownership*, *resource state*, dan *tenant boundary*, dalam mengelola otorisasi dan isolasi data antar divisi pada sistem manajemen tugas berbasis *multi-tenant* di PT Bharata Internasional Pharmaceutical.

E. Manfaat Penelitian

Adapun manfaat yang diharapkan dari hasil penelitian ini dapat diklasifikasikan ke dalam dua aspek penerima manfaat utama, yaitu :

- a. Manfaat bagi perusahaan (Praktis)

Penelitian ini memberikan kontribusi praktis bagi PT. Bharata Internasional Pharmaceutical dalam bentuk rancangan dan implementasi sistem backend manajemen tugas *multi-tenant* berbasis *web*. Sistem ini diharapkan mampu membantu perusahaan dalam menjaga isolasi data antar divisi, mencegah akses tidak sah serta meningkatkan efektivitas pengelolaan tugas melalui sistem yang terintegrasi dan terstruktur.

- b. Manfaat bagi akademisi dan pengembang perangkat lunak (praktis dan teoritis).

Hasil penelitian ini dapat menjadi referensi bagi akademisi dan pengembang perangkat lunak dalam memahami penerapan arsitektur backend berbasis *Clean Architecture*. Selain itu, penelitian ini juga memberikan gambaran implementasi sistem *multi-tenant* dengan pendekatan *shared schema* pada basis data NoSQL (MongoDB), serta penerapan mekanisme keamanan berbasis *Dynamic Context-Aware RBAC* dalam pengelolaan hak akses sistem.

BAB II TINJAUAN PUSTAKA

A. Penelitian Terkait

Sebagai referensi dan juga acuan mengenai berbagai hal dalam penelitian maka digunakan beberapa sumber yang memiliki relevansi dengan penelitian ini, berikut adalah beberapa penelitian yang serupa dengan penelitian ini adalah:

Tabel 2. 1 Penelitian Terkait

No	Penulis & Tahun	Keterangan
(1)	(2)	(3)
1	M Sahyudi & Erliyan Redy Susanto (2025)	Judul Penelitian : Analisis Implementasi Sistem Keamanan Basis Data Berbasis Role-Based Access Control (RBAC) pada Aplikasi Enterprise Resource Planning. Metode : Systematic Literature Review (SLR) Tools/Software : <i>Machine Learning (Decision Tree & Random Forest)</i> NLP, Blockchain. Hasil : Integrasi Machine Learning, NLP, dan Blockchain meningkatkan efektivitas implementasi RBAC, dengan peningkatan keamanan hingga 37%. Otomatisasi RBAC juga menurunkan biaya operasional sebesar 42% dan waktu pengelolaan hingga 65% dibandingkan pendekatan manual..
2	Loay Abdelrazek, Leyli Karacay, & Marin Orlic (2025)	Judul Penelitian : <i>Authorization of Knowledge-base Agents in an Intent-based Management Function</i> . Metode : Perancangan Hybrid Authorization Framework Tools/Software : RBAC, ABAC, RelBAC, Knowledge Graph, Zero Trust Architecture. Hasil : Penelitian mengembangkan <i>Hybrid Authorization Framework</i> yang menggabungkan RBAC, ABAC, dan RelBAC untuk menghasilkan otorisasi yang lebih dinamis dan granular. Framework menerapkan <i>least privilege</i> , <i>continuous authorization</i> , dan <i>dynamic policy update</i> pada lingkungan multi-tenant.
3	Saurabh Deochake & Vrushali Channpattan (2022)	Judul Penelitian : <i>Identity and Access Management Framework for Multi-Tenant Resources in Hybrid Cloud Computing</i> .

No	Penulis & Tahun	Keterangan
(1)	(2)	(3)
		Metode : Perancangan dan implementasi <i>Identity and Access Management (IAM) Framework</i>. Tools/Software : <i>Google Cloud Platform (GCP)</i>, <i>LDAP</i>, <i>Kerberos</i>, <i>OAuth 2.0</i>, <i>Vault</i>. Hasil : Penelitian menghasilkan <i>IAM Framework</i> berbasis <i>mirror identity</i> untuk mengelola akses pada lingkungan <i>hybrid cloud multi-tenant</i>. Framework menerapkan prinsip <i>Authentication</i>, <i>Authorization</i>, <i>Auditing (AAA)</i> dan <i>Least Privilege</i> guna meningkatkan keamanan serta skalabilitas pengelolaan identitas.
4	Aziz Catur Laksono & Bety Wulan Sari (2025)	Judul Penelitian : Pengembangan Framework Tata Kelola Akses <i>Multi-Tenant</i> untuk Mitigasi Ancaman Insider di Cloud Publik. Metode : <i>Research and Development (R&D)</i> berbasis <i>Design Science Research (DSR)</i> dengan evaluasi komparatif terhadap <i>RBAC</i> dan <i>ABAC</i> pada lingkungan simulasi <i>OpenStack multi-tenant</i>. Tools/Software : <i>OpenStack</i>, <i>RBAC</i>, <i>ABAC</i>, <i>Markov Chain</i>, <i>STRIDE</i>, <i>Apache Benchmark</i>, <i>Splunk</i>, <i>Zero Trust Architecture</i>. Hasil : Framework yang dikembangkan mengintegrasikan kontrol akses, evaluasi perilaku kontekstual, dan mitigasi adaptif berbasis <i>trust score</i> untuk mendeteksi ancaman insider pada lingkungan cloud publik multi-tenant. Hasil evaluasi menunjukkan framework memiliki akurasi deteksi dan efisiensi respons yang lebih baik dibandingkan <i>RBAC</i> dan <i>ABAC</i>, meskipun menghasilkan peningkatan overhead yang masih berada dalam batas operasional yang dapat diterima
5	Fajar Krismahendra Bintang Pangestu, et al. (2023)	Judul Penelitian : Pengembangan Model Kontrol Akses Berbasis Atribut (<i>ABAC</i>) yang Selaras dengan NIST untuk Keamanan Rekam Medis Elektronik pada Lingkungan <i>Hybrid Cloud</i>. Metode : Studi literatur, perancangan model arsitektur berbasis <i>XACML</i>, dan analisis skenario uji.

No	Penulis & Tahun	Keterangan
(1)	(2)	(3)
		Tools/Software : Standar NIST, Hybrid Cloud, XACML. Hasil : Model ABAC yang dirancang mampu menerapkan kebijakan akses secara granular dan dinamis berdasarkan atribut subjek, objek, dan lingkungan. Pendekatan ini dinilai lebih fleksibel dibandingkan RBAC dalam mengelola kontrol akses pada lingkungan hybrid cloud serta mendukung kepatuhan terhadap standar NIST.
6	Samuel Nugraha & Dian W. Chandra (2023)	Judul Penelitian : Peningkatan Keamanan Database pada Layanan Azure Melalui Metode <i>Multi-Tenant</i> Dengan Pendekatan <i>Separate Database</i>. Metode : mplementasi arsitektur <i>multi-tenant</i> dengan pendekatan <i>separate database</i> pada layanan <i>Microsoft Azure</i> menggunakan <i>Azure App Service</i> dan <i>Azure SQL Database</i>. Tools/Software : Microsoft Azure, Azure App Service, Azure SQL Database. Hasil : Pendekatan <i>separate database</i> meningkatkan keamanan data antar tenant karena setiap tenant memiliki basis data yang terpisah sehingga mengurangi risiko kebocoran data. Selain itu, penggunaan <i>Azure App Service</i> memberikan efisiensi biaya dibandingkan <i>Azure Virtual Machine</i>..
7	Yoga Agung Prasetya & Danny Manongga (2024)	Judul Penelitian : <i>Role-Based Access Control</i> (RBAC) untuk Sistem Otorisasi Terpusat Berbasis Flask: Studi Kasus PT.XYZ Metode : Studi literatur, identifikasi masalah, perancangan sistem, implementasi, dan pengujian sistem menggunakan pendekatan RBAC. Tools/Software : Python Flask, PostgreSQL, RBAC, PL/pgSQ. Hasil : Penelitian menghasilkan sistem otorisasi terpusat berbasis RBAC yang menyederhanakan pengelolaan hak akses dengan mengelompokkan pengguna berdasarkan jabatan. Pendekatan ini meningkatkan efisiensi administrasi hak akses, mempermudah pemeliharaan sistem, serta lebih

No	Penulis & Tahun	Keterangan
(1)	(2)	(3)
		sesuai diterapkan pada organisasi dengan struktur jabatan yang jelas dibandingkan pemberian hak akses secara langsung kepada setiap pengguna.
8	Arun Kumar Akuthota (2025)	Penelitian menunjukkan bahwa implementasi RBAC pada lingkungan cloud modern mampu meningkatkan keamanan, efisiensi administrasi, dan kepatuhan regulasi. Integrasi <i>AI</i> , <i>Machine Learning</i> , serta <i>Zero Trust Architecture</i> membuat RBAC lebih adaptif dalam mendeteksi ancaman, mengoptimalkan manajemen peran, dan mendukung tata kelola keamanan cloud berskala besar.
9	Alexander (2026)	Judul Penelitian : Studi Perbandingan Model Keamanan Data pada Cloud Computing. Metode : <i>Systematic Literature Review</i> (SLR) terhadap berbagai studi keamanan data terindeks. Tools/Software : Algoritma enkripsi (AES, RSA, ECC) serta mekanisme kontrol akses RBAC dan ABAC. Hasil : Memberikan analisis komprehensif mengenai efektivitas model RBAC dan ABAC dalam melindungi privasi data pada lingkungan <i>multi-tenant</i> serta merumuskan strategi mitigasi risiko terhadap ancaman keamanan data cloud.
10	Liliana et al. (2024)	Judul Penelitian : <i>Exploration of Software as a Service (SaaS) as a Project Management Tools</i> Metode : <i>Design Science Research</i> (DSR), studi literatur, perancangan sistem, implementasi menggunakan Laravel, serta pengujian <i>Black Box</i> dan kuesioner pengguna. Tools/Software : <i>Laravel 8.0</i> , <i>SaaS</i> , <i>Multi-Tenant</i> , <i>Multi-Schema Database</i> , <i>Cloudflare DNS</i> , <i>Google Calendar</i> , <i>Speech-to-Text</i> . Hasil : Penelitian berhasil mengembangkan aplikasi project management berbasis SaaS dengan arsitektur multi-tenant menggunakan pendekatan multi-schema sehingga setiap tenant memiliki database yang terpisah. Hasil pengujian menunjukkan 77,8% pengguna menyatakan konsep SaaS memberikan fleksibilitas

No	Penulis & Tahun	Keterangan
(1)	(2)	(3)
		penggunaan sistem, sedangkan 76,16% pengguna menyatakan fitur yang tersedia membantu proses manajemen proyek secara efektif.
11	Rishi Kumar Sharma (2025)	Judul Penelitian : Multi-tenant Architectures in Modern Cloud Computing: A Technical Deep Dive Metode : Analisis teknis mendalam (<i>technical deep dive</i>) dan tinjauan studi kasus ekstensif terhadap komponen inti arsitektur <i>multi-tenant</i>. Tools/Software : Ekosistem infrastruktur AWS (Lambda, ECS, Aurora Serverless, S3, IAM, KMS), protokol OAuth 2.0, serta model analitik AI dan NLP. Hasil : Penelitian menunjukkan bahwa arsitektur multi-tenant mampu meningkatkan efisiensi penggunaan sumber daya, menjaga isolasi data antar tenant, serta meningkatkan keamanan melalui penerapan IAM, RBAC, OAuth 2.0, KMS, dan mekanisme enkripsi tenant. Implementasi tersebut juga meningkatkan skalabilitas sistem dan menurunkan biaya operasional dibandingkan arsitektur single-tenant.
12	Mukul Gupta et al. (2024)	Judul Penelitian : Exploring the Impact of Software as a Service (SaaS) on <i>Human Life</i>. Metode : Analisis kualitatif dampak SaaS terhadap operasional bisnis Tools/Software : Layanan komputasi awan AWS dan Azure (fokus pada solusi keamanan). Hasil : Penelitian menunjukkan bahwa implementasi SaaS meningkatkan skalabilitas bisnis melalui fleksibilitas infrastruktur, efisiensi biaya, kemudahan integrasi, otomatisasi proses, serta peningkatan kolaborasi. Meskipun demikian, keberhasilan implementasi SaaS tetap dipengaruhi oleh aspek keamanan data, integrasi sistem, dan kesiapan organisasi.
13	Antony R.R. Panguraj (2025)	Judul Penelitian : Systematic Approach to Security Testing in Multi-Tenant Cloud Systems

No	Penulis & Tahun	Keterangan
(1)	(2)	(3)
		Metode :. Kerangka kerja pengujian keamanan sistematis (<i>Security Testing Framework</i>) Tools/Software : Lingkungan <i>cloud-native</i> (kontainer, <i>microservices</i>, <i>serverless</i>), <i>pipeline</i> CI/CD, serta mekanisme akses RBAC dan ABAC. Hasil : Penelitian menghasilkan kerangka kerja security testing untuk sistem cloud multi-tenant yang menekankan penerapan <i>continuous monitoring</i>, <i>fine-grained access control</i>, RBAC, ABAC, data <i>isolation</i>, enkripsi, serta integrasi keamanan ke dalam proses <i>DevSecOps</i>. Pendekatan tersebut mampu meningkatkan deteksi dini terhadap kerentanan dan mengurangi risiko kebocoran data maupun akses lintas tenant.
14	Ramanan Hariharan (2025)	Judul Penelitian : <i>Zero Trust Security in Multi-Tenant Cloud Environments</i> Metode :. Kerangka kerja keamanan <i>Zero Trust</i> untuk otorisasi dan isolasi data Tools/Software : IAM, AWS KMS, Azure Key Vault, <i>Open Policy Agent</i> (OPA). Hasil : Penelitian menunjukkan bahwa penerapan <i>Zero Trust</i> pada lingkungan cloud multi-tenant meningkatkan keamanan melalui prinsip <i>never trust</i>, <i>always verify</i>, autentikasi berkelanjutan (<i>continuous authentication</i>), <i>tenant-scoped</i> RBAC, <i>fine-grained</i> ABAC, <i>identity federation</i>, serta evaluasi akses berbasis konteks secara <i>real-time</i>.

Berikut adalah tabel gap penelitian dan Novelty yang digunakan dalam penelitian.

Tabel 2. 2 Gap & Novelty

No	Penulisan	Gap Penelitian	Keterbaruan Penelitian
(1)	(2)	(3)	(4)
1	M Sahyudi & Erliyan Redy	Penelitian mengkaji implementasi RBAC melalui <i>Systematic</i>	Mengembangkan mekanisme <i>Dynamic Context-Aware</i> RBAC yang

No	Penulisan	Gap Penelitian	Keterbaruan Penelitian
(1)	(2)	(3)	(4)
	Susanto (2025)	<i>Literature Riview</i> sehingga belum mengembangkan mekanisme otorisasi yang mempertimbangkan konteks operasional pengguna pada lingkungan aplikasi <i>multi-tenant</i> .	mengintegrasikan <i>role, ownership, tenant boundary, dan resource state</i> sebagai dasar pengambilan keputusan otorisasi pada sistem manajemen tugas SaaS <i>multi-tenant</i> .
2	Loay Abdelrazek et al (2025)	Penelitian mengembangkan <i>hybrid authorization</i> pada lingkungan <i>Intent-Based Management</i> , namun belum mengimplementasikan mekanisme tersebut pada aplikasi manajemen tugas berbasis <i>workspace multi-tenant</i> .	Menerapkan mekanisme <i>Dynamic Context-Aware RBAC</i> pada aplikasi manajemen tugas dengan validasi <i>ownership, tenant boundary, dan resource state</i> untuk mendukung otorisasi adaptif.
3	Saurabh Deochake & Vrushali Channpattan (2022)	Penelitian berfokus pada <i>Identity and Access Management</i> untuk <i>hybrid cloud</i> , namun belum membahas validasi hak akses berdasarkan hubungan penugasan pengguna terhadap sumber daya pada level aplikasi.	Menambahkan mekanisme <i>ownership validation</i> sehingga hak akses terhadap task ditentukan berdasarkan hubungan penugasan yang diberikan supervisor.
4	Azis Catur Laksono & Bety Wulan Sari (2025)	Penelitian mengembangkan <i>framework</i> tata kelola akses berbasis <i>trust score</i> untuk mitigasi <i>insider threat</i> , namun belum mengintegrasikan validasi <i>workflow</i> operasional pada sistem manajemen tugas.	Mengintegrasikan <i>workflow transition validation</i> bersama <i>role, ownership, dan tenant boundary</i> dalam proses otorisasi untuk memastikan perubahan status task sesuai alur kerja yang ditetapkan.
5	Fajar Krismahendra et al. (2023)	Penelitian menerapkan <i>ABAC</i> pada rekam medis elektronik sesuai standar NIST, namun belum mengimplementasikan mekanisme <i>Dynamic</i>	Mengembangkan mekanisme otorisasi berbasis <i>Dynamic Context-Aware RBAC</i> yang memanfaatkan konteks operasional pengguna tanpa

No	Penulisan	Gap Penelitian	Keterbaruan Penelitian
(1)	(2)	(3)	(4)
		<i>Context-Aware</i> RBAC pada aplikasi SaaS <i>multi-tenant</i> .	mengadopsi kompleksitas penuh model ABAC.
6	Samuel Nugraha & Dian W. Chandra (2023)	Penelitian berfokus pada isolasi basis data menggunakan pendekatan <i>separate database</i> , namun belum membahas pengendalian akses pada level aplikasi.	Mengimplementasikan <i>tenant boundary validation</i> pada mekanisme otorisasi sehingga akses terhadap <i>workspace</i> dan <i>task</i> dibatasi sesuai divisi (<i>tenant</i>) pengguna.
7	Yoga Agung Prasetya & Danny Manongga (2024)	Penelitian menerapkan RBAC <i>statis</i> sehingga keputusan akses hanya ditentukan berdasarkan role pengguna.	Mengembangkan mekanisme <i>Dynamic Context-Aware</i> RBAC dengan keputusan akses berdasarkan <i>role</i> , <i>ownership</i> , <i>tenant boundary</i> , dan <i>resource state</i> .
8	Arun Kumar Akuthota (2025)	Penelitian membahas tata kelola RBAC pada cloud modern secara konseptual, namun belum mengimplementasikan mekanisme otorisasi kontekstual pada aplikasi manajemen tugas.	Mengimplementasikan mekanisme <i>Dynamic Context-Aware</i> RBAC pada backend aplikasi SaaS <i>multi-tenant</i> melalui validasi konteks operasional pengguna secara real-time.
9	Alexander (2026)	Penelitian membandingkan berbagai model keamanan <i>cloud</i> tanpa mengembangkan mekanisme kontrol akses pada level aplikasi.	Mengembangkan mekanisme kontrol akses berbasis <i>Dynamic Context-Aware</i> RBAC yang disesuaikan dengan kebutuhan aplikasi manajemen tugas <i>multi-tenant</i> .
10	Liliana et al. (2024)	Penelitian mengembangkan aplikasi <i>project management</i> berbasis SaaS <i>multi-tenant</i> , namun belum membahas mekanisme otorisasi yang mempertimbangkan	Menambahkan mekanisme <i>assignment-based ownership validation</i> dan <i>resource state</i> sebagai bagian dari proses otorisasi pada sistem manajemen tugas SaaS <i>multi-tenant</i> .

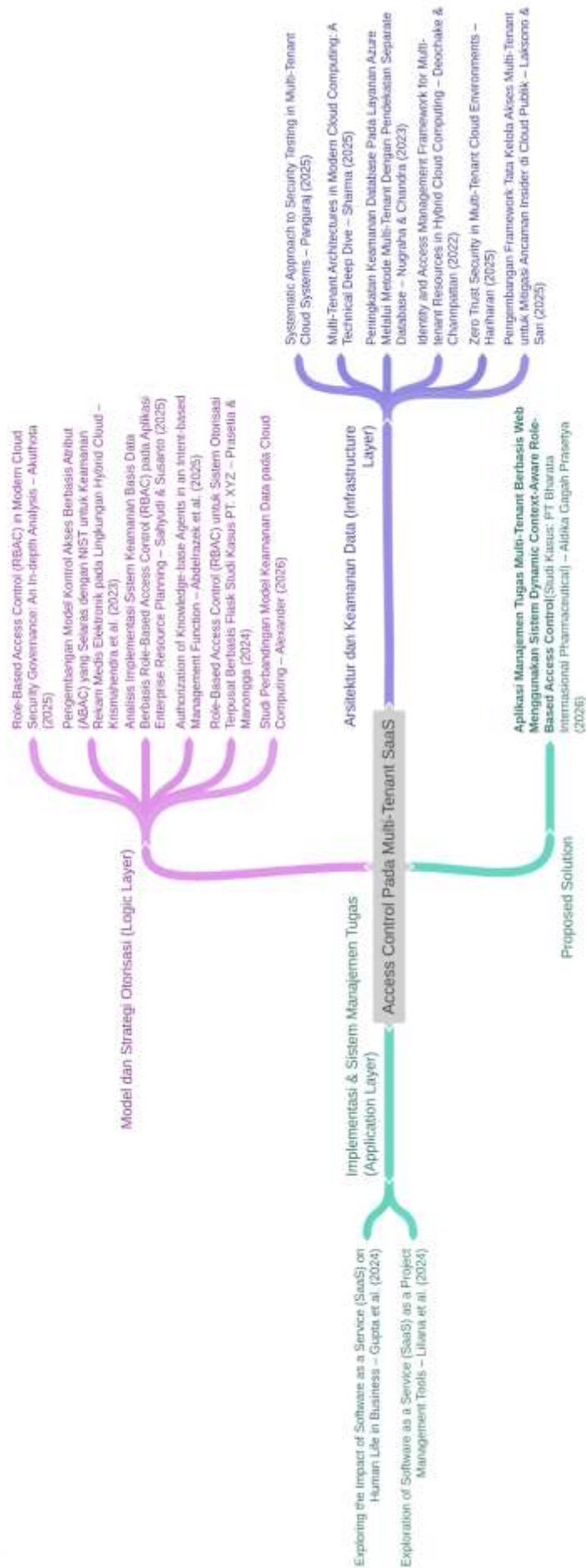
No	Penulisan	Gap Penelitian	Keterbaruan Penelitian
(1)	(2)	(3)	(4)
		hubungan penugasan pengguna maupun validasi <i>workflow</i> .	
11	Rishi Kumar Sharma (2025)	Penelitian membahas arsitektur <i>multi-tenant</i> pada tingkat infrastruktur <i>cloud</i> , namun belum mengimplementasikan mekanisme tenant isolation pada proses otorisasi aplikasi.	Mengimplementasikan <i>tenant boundary validation</i> sebagai bagian dari mekanisme <i>Dynamic Context-Aware</i> RBAC untuk menjaga isolasi data antar-divisi.
12	Mukul Gupta et al. (2024)	Penelitian berfokus pada dampak SaaS terhadap skalabilitas bisnis tanpa membahas mekanisme kontrol akses berbasis konteks.	Mengombinasikan arsitektur SaaS dengan mekanisme <i>Dynamic Context-Aware</i> RBAC untuk mendukung keamanan pengelolaan <i>task</i> pada lingkungan <i>multi-tenant</i> .
13	Antony R.R. Panguraj (2025)	Penelitian mengusulkan <i>framework security testing</i> pada <i>cloud multi-tenant</i> , namun belum mengimplementasikan mekanisme otorisasi kontekstual pada proses bisnis aplikasi.	Mengintegrasikan <i>Dynamic Context-Aware</i> RBAC, <i>audit trail</i> , dan <i>real-time notification</i> untuk mendukung keamanan serta ketertelusuran aktivitas pada sistem manajemen tugas <i>multi-tenant</i> .
14	Ramanan Hariharan (2025)	Penelitian membahas <i>Zero Trust</i> , <i>tenant-scoped</i> RBAC, dan <i>fine-grained</i> ABAC pada tingkat konseptual, namun belum mengimplementasikan mekanisme tersebut pada aplikasi manajemen tugas berbasis SaaS.	Mengimplementasikan <i>Dynamic Context-Aware</i> RBAC yang mengadopsi prinsip <i>Zero Trust</i> melalui validasi <i>role</i> , <i>ownership</i> , <i>tenant boundary</i> , dan <i>resource state</i> pada backend aplikasi.

Berdasarkan hasil pemetaan terhadap 14 penelitian terdahulu, dapat disimpulkan bahwa sebagian besar penelitian masih berfokus pada aspek keamanan

tertentu secara parsial, seperti kontrol akses berbasis RBAC atau ABAC, manajemen identitas (*Identity and Access Management*), isolasi data pada arsitektur *multi-tenant*, keamanan *cloud*, maupun penerapan *Zero Trust*. Meskipun beberapa penelitian telah mengembangkan mekanisme otorisasi yang lebih adaptif melalui pendekatan berbasis atribut, *trust-based access control*, maupun integrasi kecerdasan buatan, sebagian besar penelitian tersebut masih berorientasi pada keamanan infrastruktur, tata kelola identitas, atau kontrol akses umum sehingga belum secara spesifik mengakomodasi kebutuhan otorisasi dinamis pada level aplikasi manajemen tugas berbasis *multi-tenant*.

Selain itu, pendekatan RBAC yang digunakan pada penelitian terdahulu umumnya masih menjadikan peran (*role*) sebagai dasar utama dalam pengambilan keputusan akses. Pendekatan tersebut belum mempertimbangkan konteks operasional secara terpadu, seperti hubungan penugasan pengguna terhadap sumber daya (*assignment-based ownership*), batas isolasi antar-tenant (*tenant boundary*), serta status alur kerja (*workflow transition*) yang memengaruhi hak akses selama proses bisnis berlangsung. Kondisi tersebut berpotensi meningkatkan risiko akses lintas tenant maupun pemberian hak akses yang tidak lagi sesuai dengan kondisi operasional aplikasi.

Berdasarkan kesenjangan tersebut, penelitian ini mengembangkan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) pada sistem manajemen tugas berbasis SaaS *multi-tenant*. Mekanisme yang diusulkan tetap menggunakan RBAC sebagai dasar pengendalian akses, namun melengkapinya dengan validasi *assignment-based ownership*, *workflow transition*, dan *tenant boundary* sehingga keputusan otorisasi dapat menyesuaikan konteks operasional secara dinamis. Dengan demikian, kebaruan penelitian ini terletak pada integrasi keempat komponen tersebut ke dalam mekanisme *Dynamic Context-Aware* RBAC untuk mendukung otorisasi yang lebih kontekstual sekaligus memperkuat isolasi data antar-tenant pada aplikasi manajemen tugas berbasis SaaS.



Gambar 2. 1 Mind Map

B. Landasan Teori

Landasan teori pada penelitian ini mencakup tinjauan komprehensif mengenai mekanisme kontrol akses, strategi isolasi data pada arsitektur *multi-tenant*, prinsip keamanan *cloud*, serta teknologi pengembangan sistem yang mendukung operasional sistem manajemen tugas.

a) Mekanisme Kontrol Akses (*Access Control*)

Kontrol akses merupakan mekanisme keamanan informasi yang berfungsi untuk mengatur izin penggunaan sistem melalui proses verifikasi identitas dan otorisasi hak akses. Dalam ekosistem *multi-tenant*, mekanisme ini berperan dalam mencegah akses ilegal antardivisi yang berbagi infrastruktur yang sama.

i. *Role-Based Access Control (RBAC)*

RBAC adalah model otorisasi yang menetapkan hak akses berdasarkan peran (*role*) pengguna dalam organisasi. Secara konseptual, RBAC menyederhanakan administrasi akses dengan memetakan hak akses ke dalam peran tertentu, sehingga setiap pengguna memperoleh akses yang sesuai dengan fungsinya [6]. Namun, RBAC memiliki keterbatasan intrinsik dalam menangani kondisi dinamis, di mana hak akses tidak dapat menyesuaikan secara otomatis berdasarkan status objek data [14].

ii. *Dynamic Context-Aware RBAC*

Dynamic Context-Aware RBAC merupakan pengembangan dari mekanisme RBAC yang mengintegrasikan informasi kontekstual ke dalam proses pengambilan keputusan otorisasi akses. Berbeda dengan RBAC statis yang hanya mendasarkan keputusan akses pada peran (*role*) pengguna, pendekatan ini juga memanfaatkan atribut dan hubungan (*relationships*) yang relevan sehingga keputusan akses dapat disesuaikan secara dinamis berdasarkan kondisi operasional sistem [15]. Dengan demikian, mekanisme ini mampu menghasilkan kontrol akses yang lebih granular, adaptif, serta mendukung penerapan prinsip *least privilege* pada lingkungan yang dinamis.

Selain memanfaatkan informasi kontekstual, pengambilan keputusan otorisasi juga dapat mempertimbangkan atribut yang berkaitan dengan pengguna (*subject*), sumber daya (*object*), maupun lingkungan (*environment*) sehingga keputusan akses dapat dilakukan secara *real-time* sesuai karakteristik permintaan akses yang diterima [16]. Pendekatan tersebut menjadikan *Dynamic Context-Aware RBAC* lebih sesuai diterapkan pada sistem berbasis *multi-tenant* yang memerlukan mekanisme kontrol akses adaptif terhadap perubahan kondisi operasional.

Dalam penelitian ini, konsep *Dynamic Context-Aware RBAC* diadopsi sebagai landasan dalam pengembangan mekanisme otorisasi pada sistem

manajemen tugas berbasis *multi-tenant*. Implementasi mekanisme tersebut disesuaikan dengan kebutuhan sistem melalui empat konteks utama, yaitu *role*, *ownership*, *resource state*, dan *tenant boundary*. Keempat konteks tersebut digunakan sebagai dasar dalam proses pengambilan keputusan otorisasi agar hak akses tidak hanya ditentukan berdasarkan peran pengguna, tetapi juga mempertimbangkan hubungan pengguna terhadap sumber daya, kondisi sumber daya, serta batas akses antar-*tenant*. Penjelasan mengenai implementasi keempat konteks tersebut dijabarkan lebih lanjut pada Bab III sebagai bagian dari perancangan mekanisme otorisasi penelitian

b) Arsitektur Multi-Tenancy dan Isolasi Data

Multi-Tenancy merupakan arsitektur perangkat lunak di mana satu *instance* aplikasi melayani banyak penyewa (*tenant*) secara simultan. Dalam penelitian ini, 10 divisi pada PT Bharata direpresentasikan sebagai penyewa yang memerlukan privasi data.

i. Isolasi pada Infrastruktur Cloud

Tantangan utama pada lingkungan *multi-tenant* adalah penggunaan infrastruktur bersama (*shared infrastructure*) yang berpotensi memicu kebocoran data antar-penyewa (*cross-tenant data leakage*) [5]. Oleh karena itu, diperlukan teknik isolasi pada layer komputasi untuk menjamin keamanan informasi serta mendukung skalabilitas sistem tanpa mengorbankan integritas data [4].

ii. Strategi *Single Database – Shared Schema*

Pendekatan *single-database – shared schema* merupakan strategi arsitektur *multi-tenancy* di mana seluruh penyewa (*tenant*) ditampung dalam satu basis data dan skema yang sama. Dalam penelitian ini, penyewa direpresentasikan oleh 10 divisi organisasi. Pendekatan ini lazim digunakan dalam pengembangan layanan *Software as a Service* (SaaS) karena menawarkan efisiensi dalam pemanfaatan sumber daya server, kemudahan pemeliharaan (*maintenance*), serta skalabilitas operasional yang optimal [4].

Dalam skema ini, pemisahan data antar-divisi dilakukan secara logis melalui penggunaan atribut khusus, seperti *tenant_id* atau *division_id*, yang disematkan pada setiap dokumen dalam koleksi MongoDB untuk memastikan isolasi data pada tingkat aplikasi.

Namun, penggunaan infrastruktur yang berbagi ruang penyimpanan (*shared storage*) secara inheren menimbulkan risiko keamanan signifikan, khususnya terkait potensi terjadinya kebocoran data antarpenyewa (*cross-*

tenant data leakage). Karakteristik arsitektur yang terpusat ini menyebabkan mekanisme keamanan sangat bergantung pada lapisan aplikasi. Jika mekanisme otorisasi hanya mengandalkan *Role-Based Access Control* (RBAC) yang bersifat statis, maka risiko akses ilegal lintas divisi akan meningkat, terutama ketika terjadi kegagalan validasi pada parameter identitas.

Oleh karena itu, implementasi *Dynamic Context-Aware* RBAC menjadi komponen krusial dalam penelitian ini. Pendekatan ini berperan sebagai mekanisme keamanan tambahan yang tidak hanya memverifikasi identitas dan peran pengguna, tetapi juga melakukan validasi berkelanjutan terhadap konteks permintaan akses, sehingga mampu memitigasi risiko kebocoran data pada arsitektur *single database shared* secara lebih komprehensif [5].

c) Paradigma Keamanan Zero Trust

i. Prinsip Never Trust, Always Verify

Paradigma Zero Trust menekankan penerapan autentikasi berkelanjutan (*continuous authentication*) terhadap setiap permintaan akses. Proses verifikasi didasarkan pada identitas pengguna dan konteks permintaan akses tersebut [8]. Pendekatan ini penting untuk memitigasi risiko ancaman internal (*insider threats*) di mana pengguna dalam satu divisi berupaya melampaui hak aksesnya terhadap data divisi lain [17].

d) Teknologi Pengembangan Sistem

Efisiensi operasional dari sistem otorisasi dinamis memerlukan dukungan teknologi yang memiliki kemampuan menangani proses secara bersamaan serta fleksibilitas dalam pengelolaan struktur data.

i. Bahasa Pemrograman Golang

Golang dipilih karena efisiensinya dalam menangani banyak proses secara paralel melalui fitur *goroutines*. Karakteristik ini sangat krusial saat sistem harus melakukan pengecekan konteks dinamis secara cepat dan berulang pada setiap permintaan akses, tanpa menurunkan performa aplikasi secara keseluruhan.

ii. Basis Data NoSQL MongoDB

MongoDB menyediakan fleksibilitas melalui skema data yang bersifat *schema-less*. Karakteristik ini memungkinkan penyimpanan data tugas dengan atribut konteks yang bervariasi antardivisi tanpa memerlukan perubahan skema yang kompleks. Hal ini selaras dengan kebutuhan sistem manajemen proyek berbasis *Software as a Service* (SaaS) yang menuntut fleksibilitas dan adaptivitas tinggi [13].

e) *Clean Architecture*

Pengembangan sistem berbasis Golang dalam penelitian ini menerapkan prinsip *clean architecture* untuk memastikan kode program memiliki struktur yang modular serta meminimalkan ketergantungan terhadap *framework* eksternal.

i. Pemisahan Lapisan Sistem

Clean Architecture mengorganisasi kode perangkat lunak ke dalam lapisan konsentris yang memiliki tanggung jawab masing-masing. Disesuaikan dengan alur kerja sistem otorisasi, implementasi ini membagi arsitektur sistem ke dalam empat lapisan utama :

- 1) Lapisan *Delivery/Handler*, yang bertanggung jawab dalam menangani dan memformat komunikasi protocol HTTP.
- 2) Lapisan *Use Case/Service*, yang berperan sebagai pusat eksekusi logika bisnis, termasuk algoritma *Dynamic Context-Aware* RBAC.
- 3) Lapisan *Repository* yang bertanggung jawab dalam mengelola interaksi dan kueri data ke basis data MongoDB.
- 4) Lapisan *Entity/Domain* yang mendefinisikan struktur dan aturan data dalam sistem [18].

Pemisahan lapisan ini memastikan bahwa logika otorisasi pada sistem *multi-tenant* tetap terpisah dari mekanisme akses data, sehingga tidak bergantung langsung pada implementasi database.

ii. Kemudahan Pemeliharaan dan Kolaborasi.

Penerapan arsitektur ini penting dalam pengembangan sistem berskala besar yang melibatkan lebih dari satu pengembang. Struktur lapisan yang jelas tidak hanya mempermudah pemahaman alur kerja sistem, tetapi juga mendukung proses pemeliharaan (*maintenance*) serta pengembangan fitur di masa mendatang tanpa mengganggu stabilitas sistem secara keseluruhan [18].

f) *JSON Web Token (JWT)* pada Skema *Zero Trust*

Untuk mendukung paradigma *Zero Trust* yang meminimalkan kepercayaan otomatis, sistem menggunakan *JSON Web Token (JWT)* sebagai mekanisme autentikasi dan otorisasi yang bekerja secara *stateless*.

i. Validasi Identitas *Stateless*

JWT digunakan karena kemampuannya dalam memverifikasi identitas pengguna pada setiap permintaan akses tanpa memerlukan penyimpanan data sesi di sisi server. Sifat *stateless* ini selaras dengan karakteristik performa Golang yang efisien dalam menangani beban kerja tinggi, di

mana validasi dilakukan secara berkelanjutan melalui *Header HTTP Authorization:Bearer* [8][18].

ii. Optimasi Payload untuk Isolasi Tenant

JWT memungkinkan penyisipan klaim identitas penting seperti *user_id*, *role*, dan terutama *tenant_id*. Informasi ini memungkinkan algoritma *Dynamic Context-Aware* RBAC untuk memverifikasi hak akses terhadap sumber daya divisi secara cepat pada setiap permintaan akses, tanpa memerlukan kueri basis data secara berulang [19]. Pendekatan ini meningkatkan keamanan isolasi data lintas penyewa serta efisiensi proses otorisasi.

h. Teknologi Kontainerisasi (Docker)

Docker merupakan platform kontainerisasi yang mencerminkan pergeseran dari virtualisasi berbasis perangkat keras menuju virtualisasi berbasis sistem operasi [20]. Teknologi ini memungkinkan aplikasi berjalan dalam lingkungan terisolasi dengan memanfaatkan kernel sistem operasi dari mesin induk, sehingga meningkatkan efisiensi penggunaan sumber daya komputasi [20] [21]. Untuk mendukung kinerja, Docker memanfaatkan fitur kernel Linux berupa *namespacing* untuk isolasi proses dan *control groups* untuk pengelolaan alokasi sumber daya secara independen [21]. Karakteristik ini menghasilkan lingkungan yang ringan dan konsisten, sehingga aplikasi dapat berjalan secara seragam di berbagai infrastruktur [20]

i. Arsitektur dan Komponen Inti

Sistem docker bekerja melalui integrasi beberapa komponen utama dalam mengelola siklus hidup aplikasi [20]. Proses pembangunan aplikasi dimulai dari *Docker Image*, yaitu *template read-only (immutable)* yang tersusun atas lapisan (*layers*) dan didefinisikan melalui *Dockerfile* [20] [22]. Image ini kemudian dijalankan sebagai *Docker Container*, yaitu instance aplikasi yang dieksekusi dalam lingkungan terisolasi [22]. Pengelolaan aplikasi dapat dioptimalkan dengan Menyusun instruksi *Dockerfile* secara efisien untuk mengurangi jumlah lapisan serta menghapus data yang tidak diperlukan [22]. Untuk mendukung skalabilitas, alat orkestrasi seperti *Docker Compose* digunakan untuk mengelola aplikasi *multi-kontainer* secara terintegrasi [22].

ii. Mekanisme Keamanan dan Isolasi-Tenant

Dalam sistem *multi-tenant*, Docker menyediakan lapisan isolasi infrastruktur yang penting untuk menjaga stabilitas operasional [22]. Setiap kontainernya berjalan dalam lingkungan terpisah dengan sistem berkas dan *variable* lingkungan yang terisolasi, sehingga mencegah gangguan antar kontainer [22]. Keamanan dapat diperkuat melalui prinsip *Zero Trust*, di mana isolasi jaringan memungkinkan penerapan *mikro-segmentasi* dengan

membatasi komunikasi antar layanan hanya pada jalur yang telah ditentukan [22]. Dengan demikian, kombinasi isolasi pada tingkat infrastruktur dan mekanisme otorisasi pada tingkat aplikasi mampu meningkatkan keamanan sistem secara menyeluruh [20], [22].

BAB III METODOLOGI

A. Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan selama enam bulan, terhitung dari Februari hingga Juli 2026. Tahapan penelitian meliputi studi literatur dan observasi sistem, dilanjutkan dengan perancangan arsitektur *Clean Architecture*, serta diakhiri dengan pengujian sistem dan penyusunan laporan akhir skripsi. Penelitian ini dilaksanakan di PT Bharata Internasional Pharmaceutical, yang beralamat di Jl. Pertamina No. 19, Cipari, Kecamatan Cipari, Kabupaten Cilacap.

B. Alat dan Bahan

Pada tahap production dibutuhkan perlengkapan/komponen yang mendukung dalam pembuatan sistem backend di aplikasi manajemen tugas. Berikut spesifikasi perangkat keras maupun perangkat lunak yang peneliti gunakan dalam pembuatan aplikasi ini :

Tabel 3. 1 Alat dan Bahan

NO	NAMA	SPESIFIKASI
PERANGKAT KERAS/HARDWARE		
1.	Laptop Dell Latitude E7470	Prosesor Intel Core i7, 2801Mhz
		16 GB RAM
		SSD 240 GB
PERANGKAT LUNAK/SOFTWARE		
2.	Visual Studio Code	1.126.0 version
3.	MongoDB Compose	db version v8.2.1
4.	Postman	-
5.	Docker Desktop	Docker version 29.1.2
6.	Windows	10 Version Pro

C. Prosedur Penelitian



Gambar 3. 1 Flowchart Penelitian

Pendekatan pengembangan sistem pada penelitian ini menggunakan metodologi *Design Science Research* (DSR). Metodologi ini dipilih karena sesuai dengan tujuan penelitian yang berorientasi pada pengembangan artefak berupa sistem *backend* manajemen tugas multi-tenant serta evaluasi kinerjanya di lingkungan operasional nyata. DSR terdiri atas enam tahapan utama yang saling berurutan, yaitu identifikasi masalah, penetapan tujuan solusi, perancangan dan pengembangan artefak, demonstrasi, pengujian, serta evaluasi. Adapun rincian setiap tahapan yang disesuaikan dengan konteks penelitian ini diuraikan sebagai berikut.

a. Tahap Identifikasi Masalah

Tahap identifikasi masalah merupakan tahapan awal penelitian yang bertujuan untuk memahami kondisi eksisting serta mengidentifikasi kesenjangan antara sistem manajemen tugas yang sedang berjalan dengan kebutuhan keamanan pada lingkungan *multi-tenant*. Tahap ini menjadi dasar dalam merumuskan kebutuhan sistem yang akan dikembangkan pada penelitian.

i. Pengumpulan Data

Pada tahap ini dilakukan proses pengumpulan data untuk memperoleh informasi mengenai proses bisnis, mekanisme pengelolaan tugas, serta pengendalian hak akses yang diterapkan di PT Bharata Internasional Pharmaceutical. Pengumpulan data dilakukan melalui beberapa teknik, yaitu;

1) Observasi Lapangan

Observasi dilakukan terhadap proses pengelolaan tugas antar-divisi melibatkan supervisor dan staff IT Tech untuk mengidentifikasi alur kerja, mekanisme koordinasi, proses persetujuan (*approval*), serta mekanisme pengendalian akses yang diterapkan pada kondisi aktual.

2) Wawancara

Wawancara dilakukan dengan pihak yang terlibat dalam proses pengelolaan tugas untuk memperoleh informasi mengenai kendala operasional, kebutuhan sistem, serta mekanisme otorisasi yang diharapkan pada lingkungan *multi-tenant*.

3) Studi Dokumentasi

Studi dokumentasi dilakukan dengan mempelajari dokumen proses bisnis, struktur organisasi, serta referensi yang berkaitan dengan sistem manajemen tugas sebagai pendukung proses identifikasi kebutuhan sistem.

Hasil pengumpulan data berupa informasi mengenai proses bisnis, mekanisme pengelolaan tugas, pengendalian hak akses, serta kebutuhan pengguna yang selanjutnya digunakan sebagai masukan pada tahap analisis permasalahan.

ii. Analisis Permasalahan

1) Fragmentasi Tata Kelola dan Risiko Integritas Data

a) Pengelolaan alur kerja dan pengajuan tugas antar-divisi masih dilakukan secara informal melalui komunikasi lisan dan aplikasi pesan instan, sehingga menyulitkan proses pelacakan progres, koordinasi pekerjaan, dan pertanggungjawaban tugas secara terstruktur.

- b) Status tugas sering kali tidak terdokumentasi dengan jelas, sehingga menyebabkan kesulitan dalam memantau perkembangan pekerjaan dan menentukan kondisi aktual suatu tugas pada setiap tahapan proses kerja.
 - c) Proses persetujuan (*approval*) tugas belum memiliki mekanisme pencatatan yang terintegrasi, sehingga riwayat persetujuan, perubahan status, dan penugasan tugas tidak dapat ditelusuri kembali secara sistematis.
 - d) Tidak adanya rekam jejak aktivitas (*audit trail*) yang terintegrasi menyebabkan perubahan status dan aktivitas pengguna dalam sistem tidak terdokumentasi dengan baik, sehingga berisiko terhadap validitas dan akuntabilitas informasi operasional.
 - e) Belum terdapat mekanisme validasi alur kerja yang memastikan perubahan status tugas dilakukan secara bertahap sesuai *workflow*, hanya oleh pengguna yang memiliki kewenangan, serta setelah seluruh prasyarat operasional seperti penugasan anggota dan persetujuan terpenuhi.
- 2) Keterbatasan Otorisasi Statis pada Lingkungan *Multi-Tenant*.
- a) Sistem otorisasi konvensional berbasis Role-Based Access Control (RBAC) statis masih memiliki keterbatasan dalam memvalidasi konteks akses secara dinamis pada lingkungan multi-tenant yang memiliki banyak divisi dengan kebutuhan akses yang berbeda.
 - b) Tidak adanya batas pemisah yang jelas (*tenant boundary*) antar-divisi berpotensi menimbulkan akses data yang tidak sah dan meningkatkan risiko kebocoran data antar-divisi (*cross-tenant data leakage*).
 - c) Keputusan otorisasi yang hanya didasarkan pada identitas peran pengguna tanpa mempertimbangkan konteks operasional, seperti kepemilikan tugas (*ownership*), status sumber daya (*resource state*), dan batas divisi (*tenant boundary*) menyebabkan mekanisme otorisasi kurang adaptif terhadap dinamika kolaborasi lintas divisi dan perubahan status tugas pada lingkungan *multi-tenant*.
 - d) Sistem belum memiliki mekanisme validasi akses yang mampu memastikan bahwa perubahan status tugas hanya dapat dilakukan oleh pengguna yang memiliki kewenangan sesuai peran, kepemilikan tugas (*ownership*), batas tenant (*tenant boundary*), serta kondisi sumber daya (*resource state*) yang berlaku pada saat proses berlangsung.

Berdasarkan hasil identifikasi masalah tersebut, diperoleh kebutuhan sistem berupa mekanisme otorisasi dinamis yang mampu memvalidasi setiap permintaan akses berdasarkan konteks operasional secara real-time, menjaga

isolasi data antar-divisi (*tenant boundary*), mengendalikan perubahan status tugas sesuai kondisi sumber daya (*resource state*), serta mendukung pencatatan aktivitas melalui mekanisme *audit trail*. Berdasarkan kebutuhan tersebut dirancang mekanisme *Dynamic Context-Aware Role-Based Access Control (RBAC)* yang mengintegrasikan dimensi *ownership*, *resource state*, dan *tenant boundary* sebagai dasar pengambilan keputusan otorisasi pada sistem manajemen tugas multi-tenant.

b. Tahap Penetapan Tujuan Solusi

Tahapan ini bertujuan untuk menetapkan solusi yang akan dikembangkan berdasarkan permasalahan yang telah diidentifikasi pada lingkungan sistem manajemen tugas *multi-tenant* di PT Bharata Internasional Pharmaceutical. Solusi yang dirancang difokuskan pada pengembangan sistem manajemen tugas yang terintegrasi serta penerapan mekanisme otorisasi dinamis untuk meningkatkan keamanan hak akses dan menjaga isolasi data antar divisi. Berdasarkan hasil identifikasi masalah, penelitian ini menetapkan beberapa tujuan solusi sebagai berikut :

i. Pengembangan Sistem Manajemen Tugas *Multi-Tenant*

- 1) Mengembangkan sistem manajemen tugas berbasis *multi-tenant* dengan setiap divisi bertindak sebagai tenant yang memiliki isolasi data dan pengelolaan tugas masing-masing.
- 2) Mengembangkan sistem pengelolaan tugas yang mampu mendukung proses penugasan, pemantauan progres pekerjaan, dan pengelolaan *workflow* secara terstruktur.
- 3) Menyediakan mekanisme *audit trail* untuk mencatat aktivitas pengguna, perubahan status tugas, dan proses penugasan secara terintegrasi.
- 4) Menyediakan mekanisme penyampaian informasi tugas secara *real-time* guna mendukung koordinasi dan komunikasi antar pengguna dalam sistem.

ii. Pengembangan Mekanisme *Dynamic Context-Aware RBAC*

- 1) Mengembangkan mekanisme otorisasi berbasis *Dynamic Context-Aware Role-Based Access Control (RBAC)* untuk memvalidasi hak akses pengguna secara dinamis berdasarkan konteks operasional.
- 2) Menerapkan validasi akses berdasarkan dimensi *ownership* guna memastikan bahwa setiap operasi terhadap tugas hanya dapat dilakukan oleh pengguna yang memiliki hubungan operasional atau kepemilikan yang sah sesuai mekanisme penugasan dalam sistem..
- 3) Menerapkan validasi *resource state* melalui mekanisme *workflow transition* untuk memastikan perubahan status tugas berlangsung secara bertahap sesuai alur *Request, Todo, On Going, Testing, Done*, dan *Archive*.

- 4) Menerapkan *tenant boundary validation* untuk menjaga isolasi data antar-divisi serta mencegah akses data tidak sah pada lingkungan *multi-tenant*.
- 5) Mengintegrasikan struktur peran pengguna dengan mekanisme validasi *ownership*, *resource state*, dan *tenant boundary* dalam proses pengambilan keputusan otorisasi secara dinamis.

Hasil dari tahap penetapan tujuan solusi ini menjadi dasar dalam proses perancangan artefak sistem manajemen tugas *multi-tenant* yang dilengkapi dengan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC). Fokus utama solusi yang dikembangkan adalah membangun mekanisme pengambilan keputusan otorisasi (*authorization decision*) yang mengintegrasikan validasi *ownership*, *resource state*, dan *tenant boundary* sehingga *backend* mampu menghasilkan keputusan akses yang adaptif sesuai konteks operasional pada lingkungan *multi-tenant*.

c. Tahap Perancangan dan Pengembangan Artefak

1. Tahap perancangan dan pengembangan artefak merupakan proses transformasi hasil analisis kebutuhan ke dalam rancangan teknis yang menjadi acuan implementasi sistem. Dalam pendekatan *Design Science Research*, artefak digunakan sebagai hasil perancangan yang merepresentasikan solusi terhadap permasalahan yang diteliti, termasuk artefak yang digunakan untuk merepresentasikan dan mengomunikasikan rancangan sistem[23]. Dalam penelitian ini, artefak yang dikembangkan berupa sistem manajemen tugas berbasis *multi-tenant* dengan penerapan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) pada *backend*.

Perancangan dilakukan secara bertahap mencakup pemodelan fungsional melalui *Use Case Diagram*, pemodelan alur proses melalui *Flowchart* dan *Sequence Diagram*, perancangan struktur data pada basis data MongoDB, serta perancangan arsitektur sistem menggunakan prinsip *Clean Architecture*.

i. Perancangan *Use Case Diagram*

Tahap ini bertujuan untuk menggambarkan interaksi antara pengguna dengan sistem berdasarkan hak akses dan fungsi yang dimiliki masing-masing peran. *Use Case Diagram* digunakan untuk memvisualisasikan kebutuhan fungsional sistem serta batasan akses pada lingkungan *multi-tenant*.

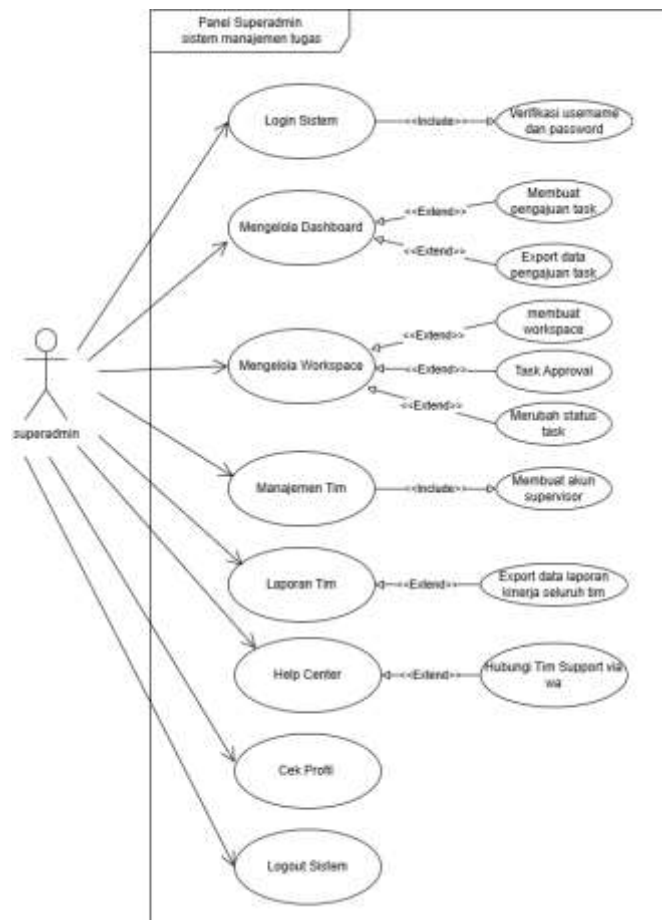
Pada sistem yang dikembangkan terdapat tiga aktor utama, yaitu:

1. Superadmin
2. Supervisor
3. Staff

Masing-masing aktor memiliki hak akses yang berbeda sesuai peran dan tanggung jawab dalam sistem. Superadmin memiliki hak akses penuh terhadap pengelolaan sistem, supervisor memiliki hak akses penuh terhadap tugas dalam divisinya, sedangkan staff hanya dapat melakukan pengerjaan dan perpindahan status tugas berdasarkan penugasan yang diberikan supervisor.

Berikut merupakan rancangan *Use Case Diagram* pada sistem yang dikembangkan.

1) *Use Case Diagram Superadmin*



Gambar 3. 2 *Use Case Diagram Superadmin*

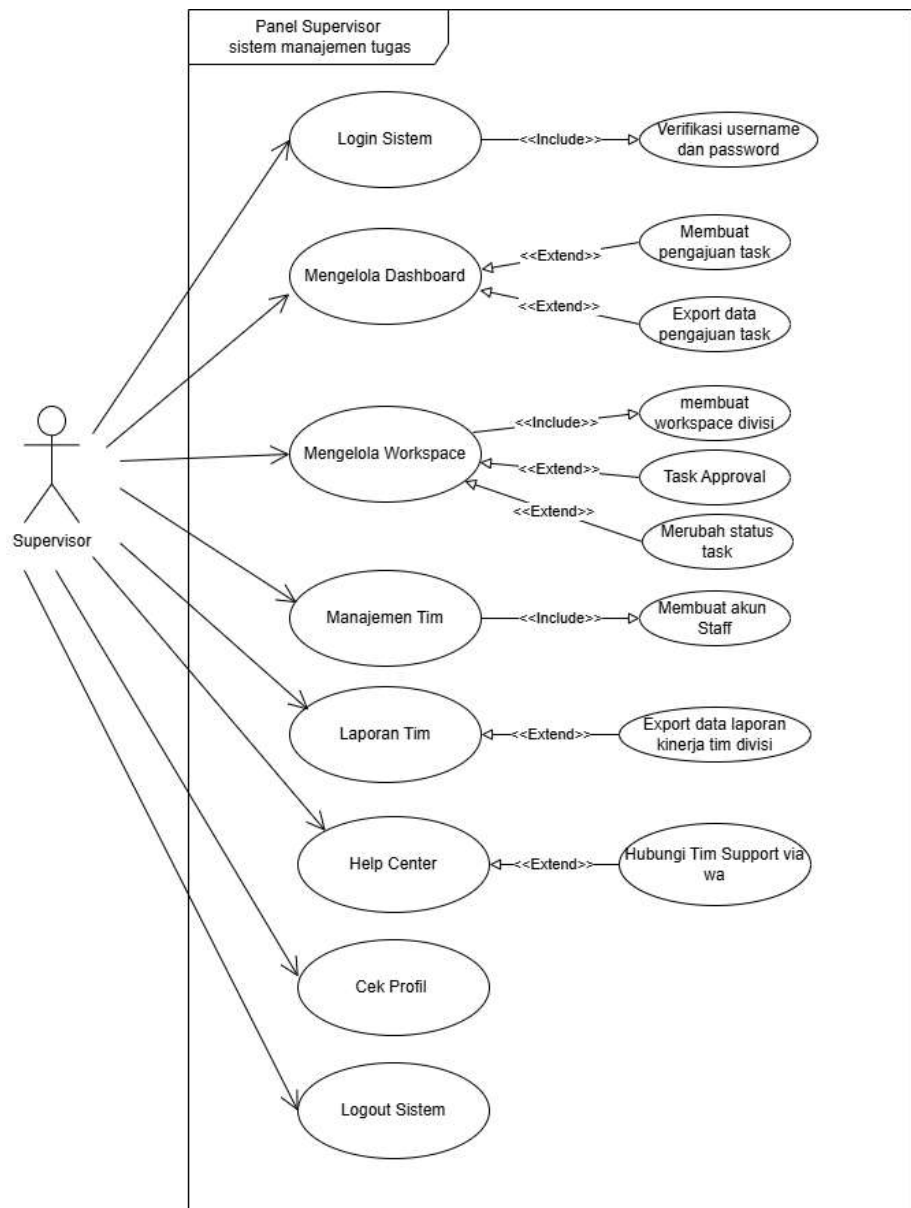
Gambar di atas menunjukkan *Use Case Diagram* panel *Superadmin* pada Sistem Manajemen Tugas. Diagram tersebut menggambarkan interaksi aktor *Superadmin* sebagai pengguna dengan hak akses tertinggi dalam sistem.

Fungsionalitas utama yang dimiliki meliputi proses *autentikasi* dan manajemen sesi seperti *login*, verifikasi akun, pengecekan profil, dan *logout*. Selain itu, *Superadmin* memiliki akses penuh dalam pengelolaan *workspace*, pengelolaan tugas, persetujuan tugas (*task approval*), perubahan status tugas, serta pengelolaan *dashboard* untuk pengajuan dan ekspor data tugas.

Pada sisi administrasi, *Superadmin* juga dapat mengelola tim melalui pembuatan akun *supervisor* dan melakukan monitoring kinerja melalui fitur laporan tim. Sistem turut menyediakan fitur *Help Center* yang terhubung dengan layanan dukungan teknis melalui WhatsApp.

Secara umum, diagram ini menunjukkan bahwa *Superadmin* berperan sebagai administrator utama yang mengontrol pengelolaan pengguna, divisi, *workspace*, dan operasional tugas dalam sistem.

2) Use Case Diagram Supervisor



Gambar 3. 3 Use Case Diagram Supervisor

Gambar di atas menunjukkan *Use Case Diagram* panel *Supervisor* pada Sistem Manajemen Tugas. Diagram tersebut menggambarkan interaksi aktor *Supervisor* sebagai pengguna dengan wewenang pengelolaan pada tingkat divisi.

Fungsionalitas utama meliputi proses autentikasi dan manajemen sesi seperti *login*, verifikasi kredensial, pengelolaan profil, dan *logout*. Selain itu, *Supervisor* memiliki akses untuk mengelola *workspace*, melakukan *task approval*, memperbarui status tugas, serta mengakses *dashboard* untuk pengajuan dan ekspor data tugas.

Pada aspek manajemen tim, *Supervisor* dapat membuat akun *staff* dan memantau kinerja anggota melalui fitur laporan tim. Sistem juga menyediakan fitur *Help Center* untuk layanan bantuan teknis melalui WhatsApp.

Secara umum, diagram ini menunjukkan bahwa hak akses *Supervisor* dibatasi berdasarkan divisi (*tenant division boundary*), sehingga pengelolaan tim, *workspace*, dan operasional tugas hanya dapat dilakukan pada divisi yang menjadi tanggung jawabnya.

3) Use Case Diagram Staff



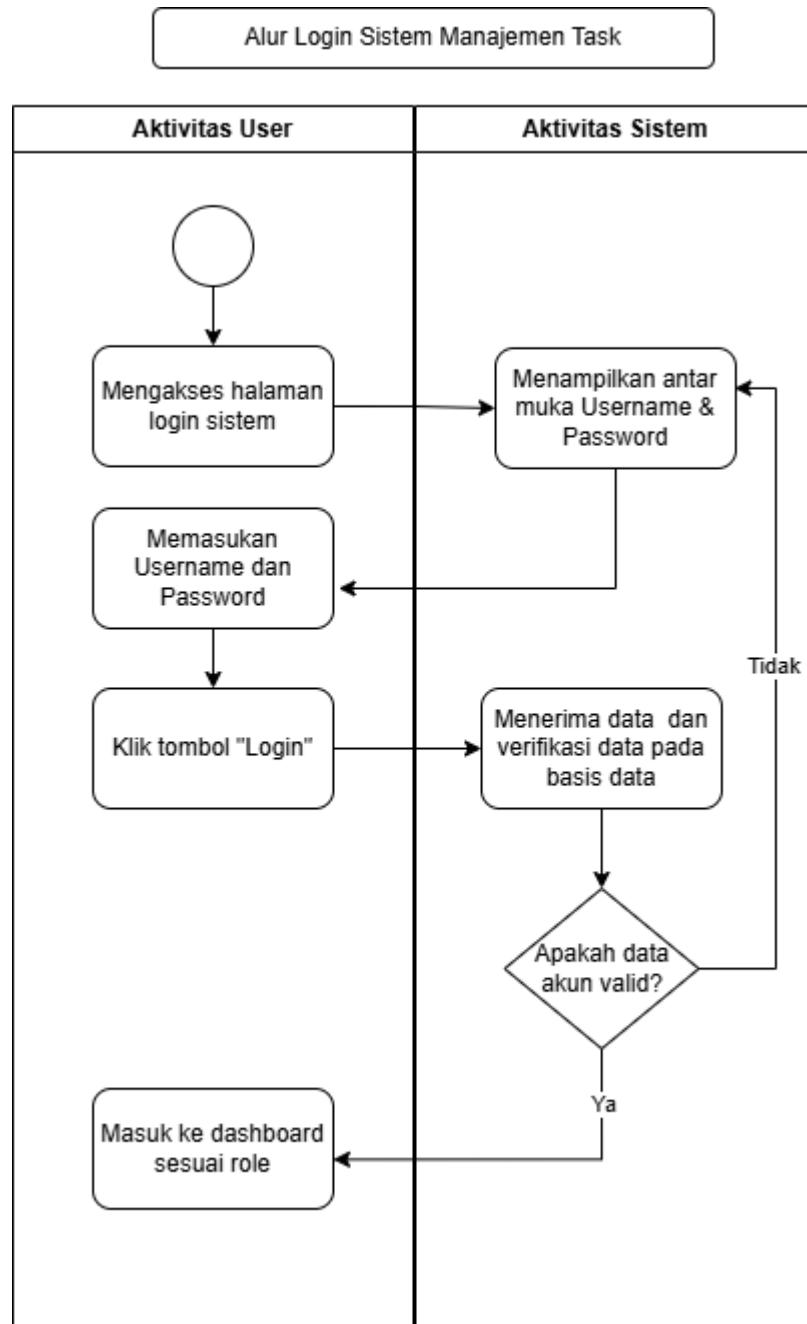
Gambar 3. 4 Use Case Diagram Staff

Gambar di atas menunjukkan *Use Case Diagram* panel *Staff* pada Sistem Manajemen Tugas. Diagram tersebut menggambarkan interaksi aktor *Staff* sebagai pengguna operasional (*end-user*) dengan hak akses terbatas di dalam sistem.

Fungsionalitas utama meliputi proses autentikasi dan manajemen sesi seperti *login*, verifikasi kredensial, pengecekan profil, dan *logout*. Pada aspek operasional tugas, *Staff* dapat mengakses *dashboard* untuk membuat pengajuan tugas, mengerjakan tugas yang diberikan, menerima notifikasi, serta mengelola *Kanban Board* melalui perubahan status tugas yang menjadi tanggung jawabnya.

ii. Activity Diagram

1) Activity Diagram Login



Gambar 3. 5 Activity Diagram Login

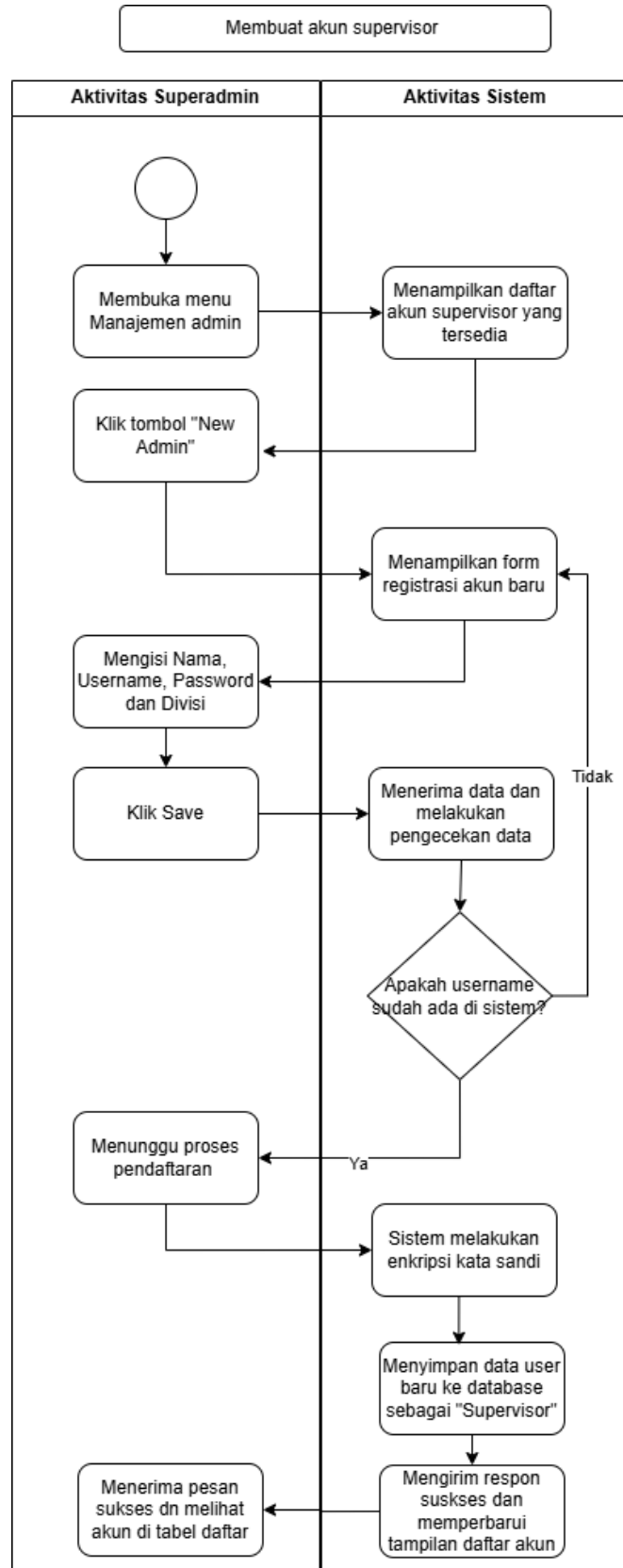
Gambar di atas menunjukkan *Activity Diagram* alur *login* pada Sistem Manajemen Tugas. Diagram tersebut menggambarkan alur interaksi antara pengguna dan sistem dalam proses autentikasi akses pengguna.

Proses dimulai ketika pengguna mengakses halaman *login*, kemudian sistem menampilkan antarmuka *username* dan *password*. Selanjutnya pengguna memasukkan kredensial dan menekan tombol *login* untuk mengirimkan data ke sistem. Sistem kemudian melakukan proses verifikasi data akun ke basis data.

Pada tahap keputusan (*decision node*), sistem akan memvalidasi apakah data akun sesuai atau tidak. Jika kredensial tidak valid, sistem akan mengembalikan pengguna ke halaman *login* untuk melakukan percobaan ulang. Sebaliknya, jika data valid, sistem memberikan otorisasi akses dan mengarahkan pengguna ke *dashboard* sesuai peran (*role*) masing-masing.

Secara umum, diagram ini menunjukkan mekanisme autentikasi yang memisahkan aktivitas pengguna sebagai pengirim data dan sistem sebagai pihak yang melakukan validasi serta pemberian hak akses.

2) Activity Diagram Membuat Akun Supervisor



Gambar 3. 6 Activity Diagram membuat akun supervisor

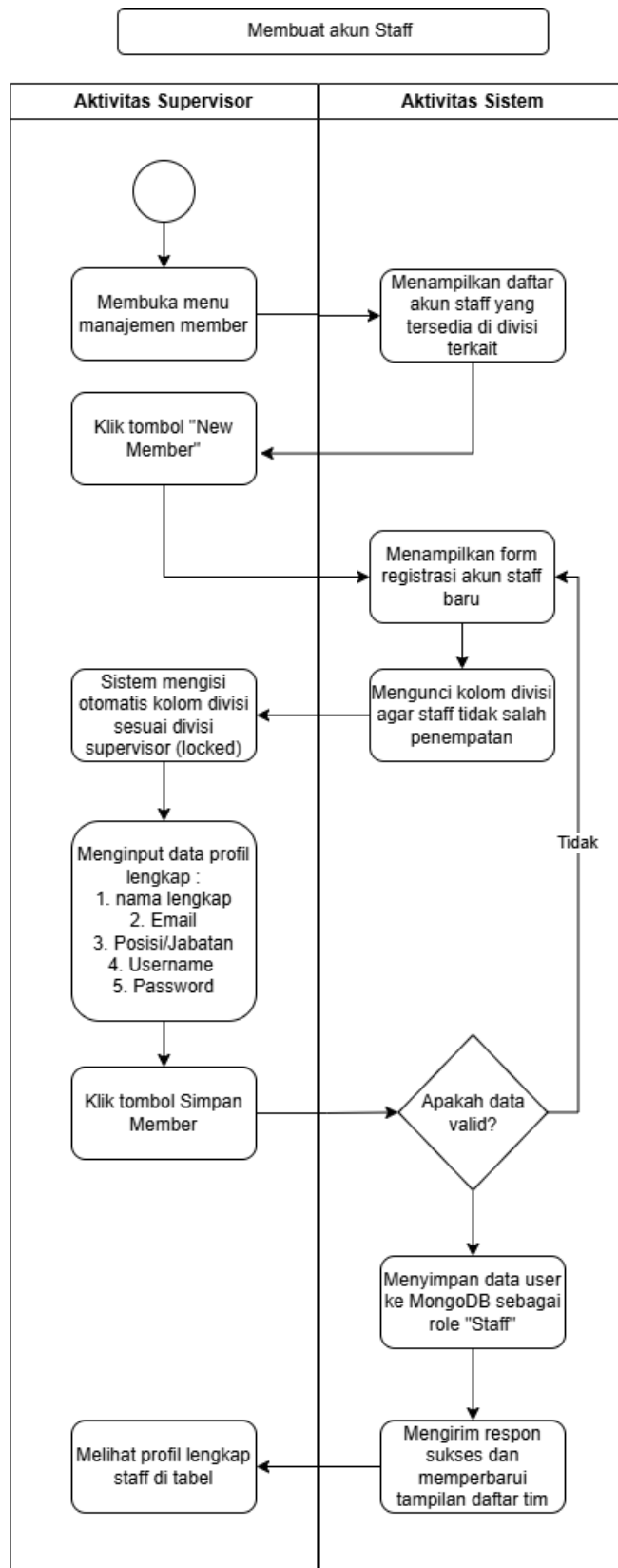
Gambar di atas menunjukkan *Activity Diagram* proses pembuatan akun *Supervisor* oleh *Superadmin*. Diagram tersebut menggambarkan interaksi antara aktivitas *Superadmin* sebagai penginput data dan sistem sebagai pengelola proses validasi serta penyimpanan data.

Proses dimulai ketika *Superadmin* membuka menu manajemen admin, kemudian sistem menampilkan daftar akun *Supervisor* yang tersedia. Selanjutnya *Superadmin* memilih tombol *New Admin* sehingga sistem menampilkan formulir pendaftaran akun baru. Pada tahap ini, *Superadmin* mengisi data seperti nama, *username*, *password*, dan divisi, lalu menekan tombol *Save*.

Setelah data dikirim, sistem melakukan validasi keunikan data pada basis data melalui proses pengecekan *username*. Jika ditemukan data yang sudah terdaftar, sistem akan mengembalikan proses ke formulir registrasi untuk melakukan input ulang. Sebaliknya, jika data valid dan unik, sistem melakukan proses *hashing password* untuk keamanan data, kemudian menyimpan akun baru ke dalam basis data dengan *role* sebagai *Supervisor*.

Proses diakhiri dengan sistem menampilkan notifikasi berhasil serta memperbarui daftar akun *Supervisor* pada antarmuka *Superadmin*.

3) Activity Diagram Membuat Akun Staff



Gambar 3. 7 Activity Diagram membuat akun staff

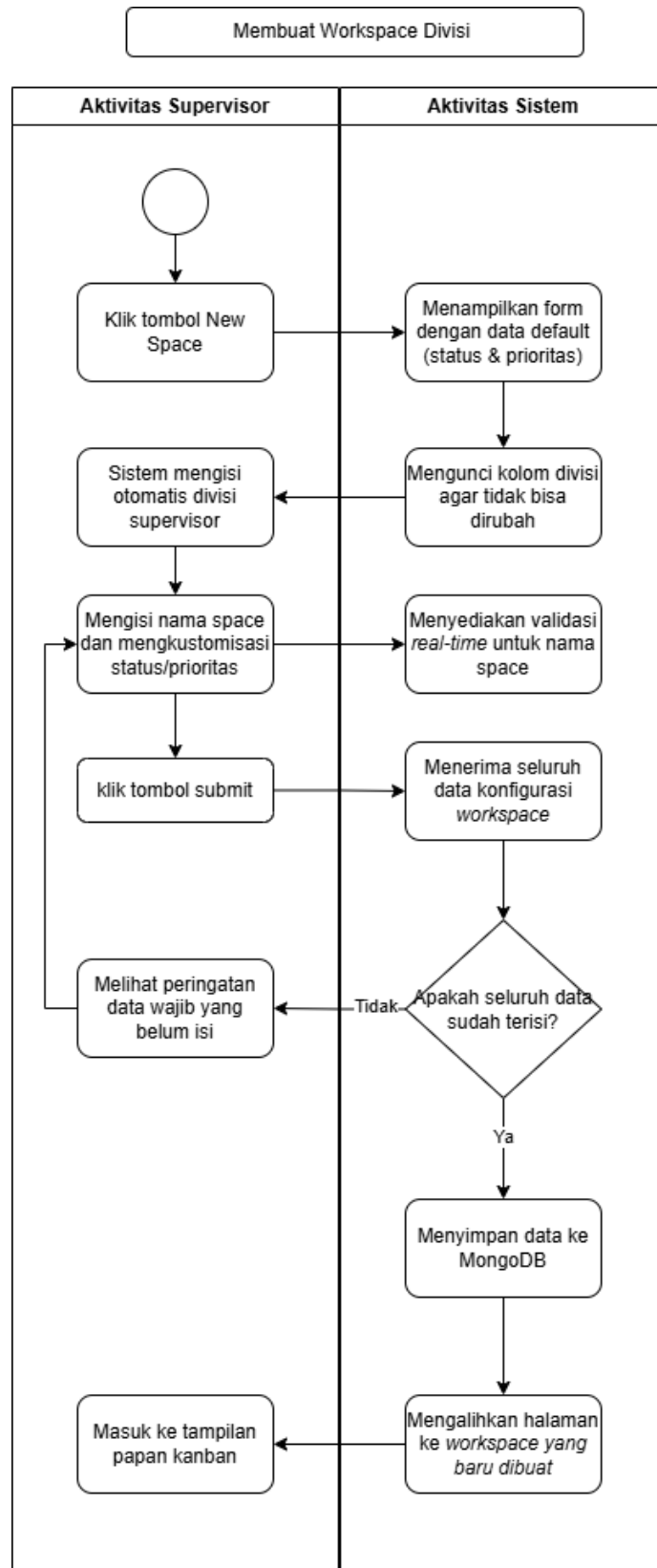
Gambar di atas menunjukkan *Activity Diagram* proses pembuatan akun *Staff* oleh *Supervisor*. Diagram tersebut menggambarkan alur interaksi antara *Supervisor* sebagai pengelola divisi dan sistem dalam menambahkan anggota tim baru secara aman.

Proses dimulai ketika *Supervisor* membuka menu manajemen anggota (*member management*), kemudian sistem menampilkan daftar akun *Staff* sesuai divisi yang dikelola. Selanjutnya *Supervisor* memilih tombol *New Member* sehingga sistem menampilkan formulir registrasi akun baru. Pada tahap ini, sistem secara otomatis mengunci kolom divisi sesuai divisi milik *Supervisor* untuk menjaga batas isolasi antar-divisi (*tenant boundary*).

Setelah itu, *Supervisor* mengisi data pengguna seperti nama lengkap, email, jabatan, *username*, dan *password*, kemudian menekan tombol simpan. Sistem akan melakukan validasi data melalui proses pengecekan kelengkapan dan keabsahan data pengguna. Jika data tidak valid, sistem mengembalikan proses ke formulir registrasi. Sebaliknya, jika data valid, sistem menyimpan akun baru ke dalam basis data MongoDB dengan *role* sebagai *Staff*.

Proses diakhiri dengan sistem menampilkan notifikasi berhasil serta memperbarui daftar anggota tim pada antarmuka *Supervisor*.

4) Activity Diagram Membuat Workspace Divisi oleh Supervisor



Gambar 3. 8 Activity Diagram membuat workspace divisi

Gambar di atas menunjukkan *Activity Diagram* proses pembuatan *workspace* divisi oleh *Supervisor*. Diagram tersebut menggambarkan interaksi antara *Supervisor* sebagai pengelola ruang kerja dan sistem dalam membangun ruang kolaborasi berbasis divisi.

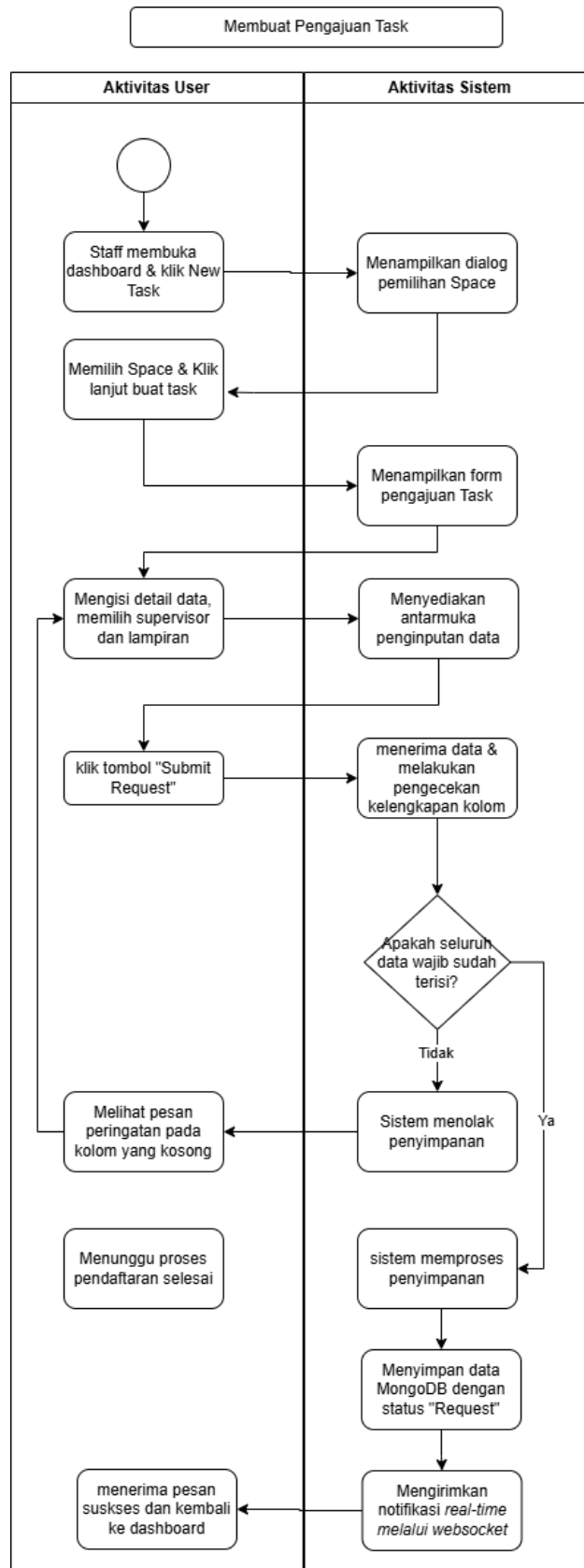
Proses dimulai ketika *Supervisor* menekan tombol *New Space*, kemudian sistem menampilkan formulir pembuatan *workspace* beserta konfigurasi awal seperti status dan prioritas *Kanban*. Untuk menjaga isolasi data antar-divisi (*tenant boundary*), sistem secara otomatis mengisi dan mengunci kolom divisi berdasarkan identitas *Supervisor* yang sedang *login*.

Selanjutnya, *Supervisor* mengisi nama *workspace* dan melakukan penyesuaian status maupun prioritas sesuai kebutuhan divisi. Sistem juga melakukan validasi *real-time* terhadap nama *workspace* untuk mencegah duplikasi data. Setelah tombol *submit* ditekan, sistem memeriksa kelengkapan data yang dimasukkan.

Apabila terdapat data yang belum lengkap, sistem akan menampilkan peringatan untuk dilakukan perbaikan. Sebaliknya, jika seluruh data valid, sistem menyimpan konfigurasi *workspace* ke dalam basis data MongoDB dan mengarahkan pengguna ke halaman *workspace* yang baru dibuat.

Secara umum, diagram ini menunjukkan mekanisme pembuatan *workspace* yang mendukung isolasi tenant, validasi data, serta konfigurasi awal *Kanban Board* secara terstruktur.

5) Activity Diagram Pengajuan Task



Gambar 3. 9 Activity Diagram pengajuan task

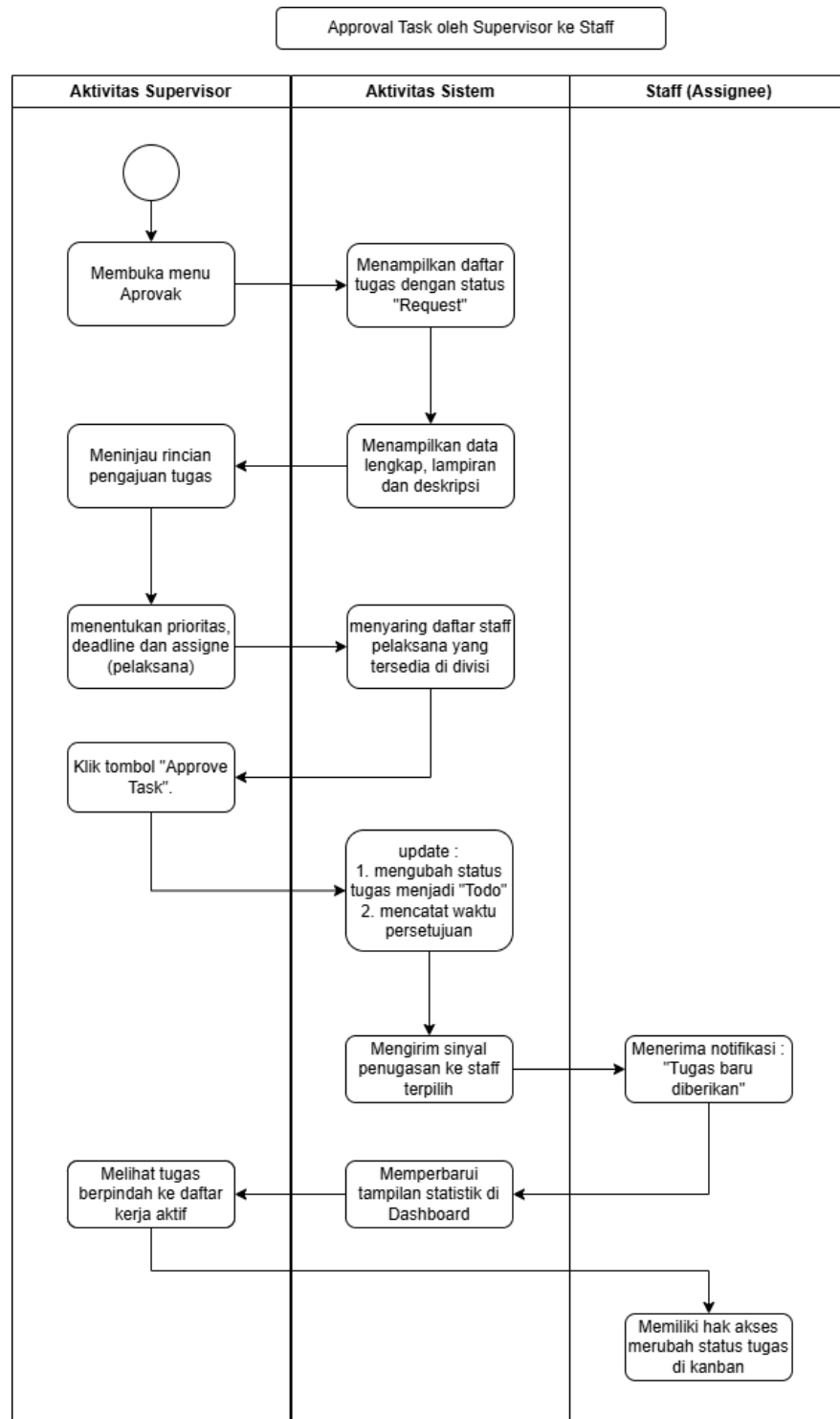
Gambar di atas menunjukkan *Activity Diagram* proses pembuatan pengajuan *task* oleh pengguna, yang dalam skenario ini diperankan oleh aktor *Staff*. Diagram tersebut menggambarkan interaksi antara pengguna dan sistem dalam memproses pengajuan tugas baru ke dalam basis data MongoDB.

Proses dimulai ketika *Staff* membuka halaman *dashboard* dan memilih tombol *New Task*. Sistem kemudian menampilkan pilihan *workspace* agar tugas ditempatkan pada ruang kerja yang sesuai. Setelah *workspace* dipilih, sistem menampilkan formulir pengajuan *task* yang digunakan untuk mengisi detail tugas, memilih *Supervisor*, serta menambahkan lampiran pendukung.

Setelah pengguna menekan tombol *Submit Request*, sistem melakukan validasi terhadap kelengkapan data input. Jika terdapat data yang belum lengkap, sistem akan menampilkan peringatan dan meminta pengguna melengkapi data tersebut. Sebaliknya, apabila seluruh data valid, sistem akan menyimpan dokumen *task* baru ke dalam basis data MongoDB dengan status awal *Request*.

Setelah proses penyimpanan berhasil, sistem secara otomatis mengirimkan notifikasi *real-time* melalui *WebSocket* kepada pengguna terkait, kemudian mengarahkan pengguna kembali ke halaman *dashboard*.

6) *Activitas Diagram Approval Task*



Gambar 3. 10 Activity Diagram approval task

Gambar di atas menunjukkan *Activity Diagram* proses *approval* tugas oleh *Supervisor* kepada *Staff*. Diagram tersebut menggambarkan interaksi antara *Supervisor*, sistem, dan *Staff* dalam proses persetujuan serta penugasan tugas secara terstruktur.

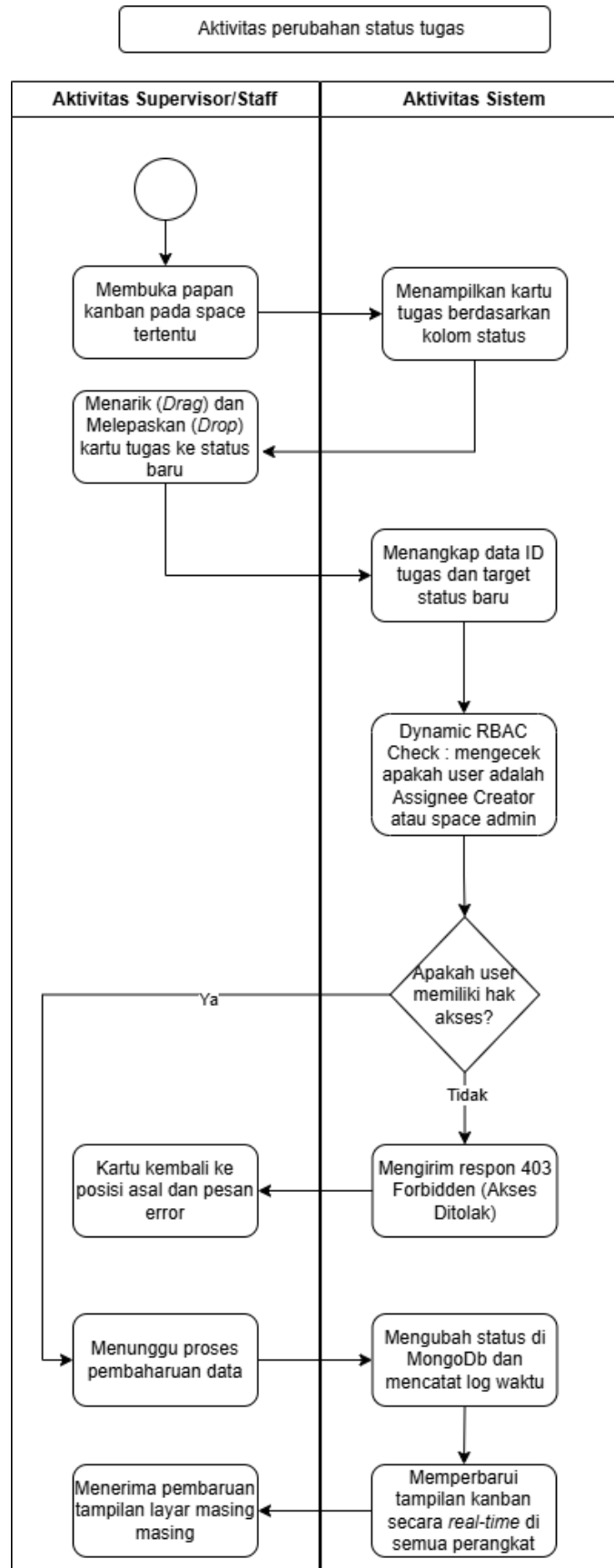
Proses dimulai ketika *Supervisor* membuka menu *Approval*, kemudian sistem menampilkan daftar tugas dengan status *Request*. Selanjutnya, sistem menampilkan detail tugas beserta lampiran dan deskripsinya agar *Supervisor* dapat melakukan peninjauan pengajuan tugas.

Pada tahap berikutnya, *Supervisor* menentukan prioritas, *deadline*, dan *assignee* yang akan menerima tugas. Sistem secara otomatis membatasi daftar *Staff* berdasarkan divisi terkait untuk menjaga isolasi antar-divisi (*tenant boundary*). Setelah tombol *Approve Task* ditekan, sistem memperbarui status tugas menjadi *Todo* serta mencatat aktivitas persetujuan ke dalam *audit trail*.

Selanjutnya, sistem mengirimkan notifikasi penugasan kepada *Staff* yang dipilih dan memperbarui data statistik pada *dashboard*. Setelah proses persetujuan selesai, *Staff* yang ditugaskan memperoleh hak akses untuk mengelola dan memperbarui status tugas pada *Kanban Board*.

Secara umum, diagram ini menunjukkan mekanisme persetujuan tugas yang terintegrasi dengan validasi divisi, pencatatan aktivitas, notifikasi *real-time*, dan pengelolaan hak akses berbasis konteks tugas

7) Activity Diagram Perubahan Status Tugas



Gambar 3. 11 Activity Diagram perubahan status task

Gambar di atas menunjukkan *Activity Diagram* proses perubahan status tugas menggunakan metode *drag and drop* pada papan *Kanban*. Diagram tersebut menggambarkan interaksi antara pengguna (*Supervisor* atau *Staff*) dan sistem dalam melakukan pembaruan status tugas secara *real-time*.

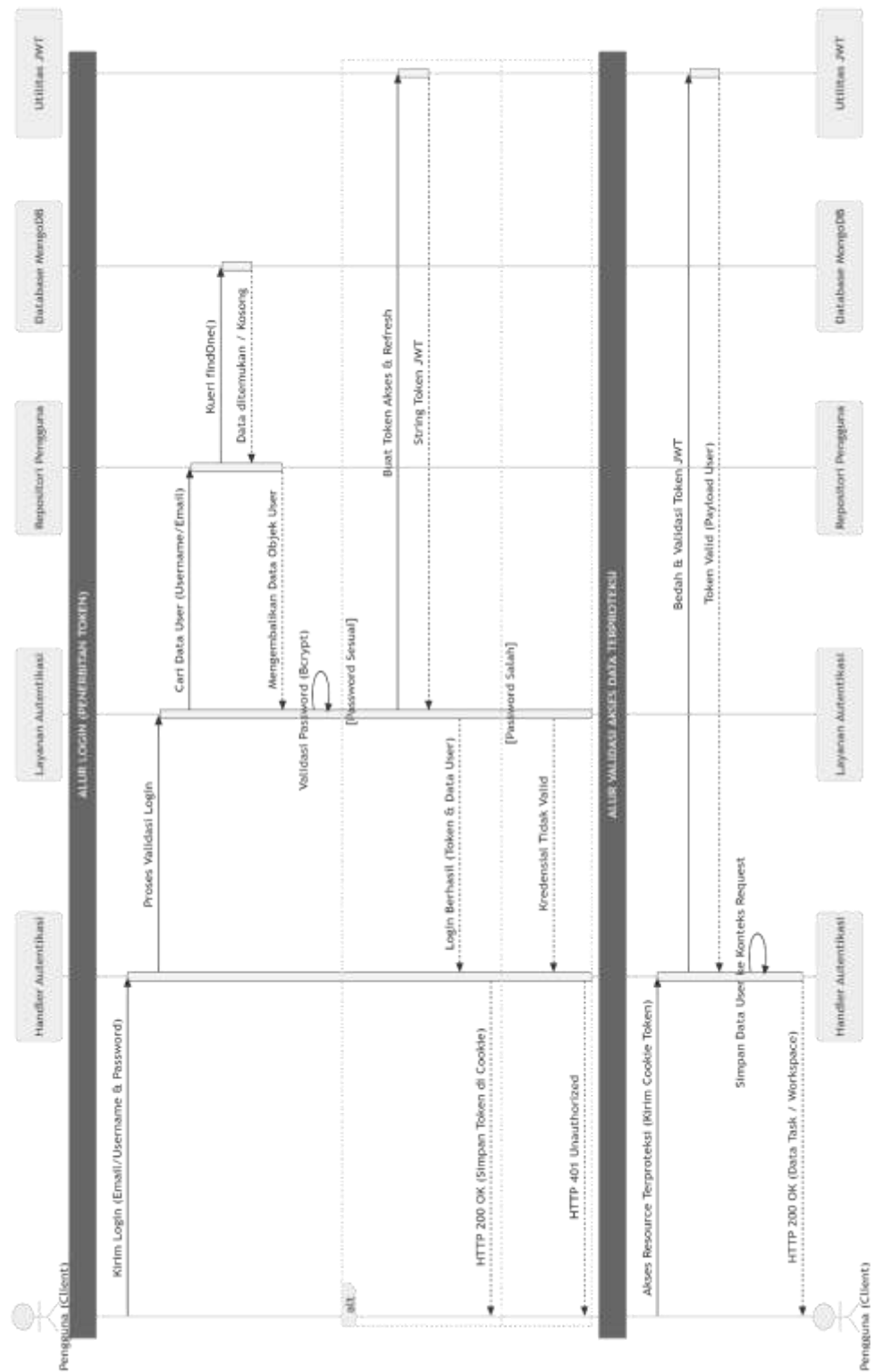
Proses dimulai ketika pengguna membuka papan *Kanban* pada *workspace* tertentu, kemudian sistem menampilkan daftar kartu tugas berdasarkan status masing-masing. Selanjutnya, pengguna melakukan perpindahan kartu tugas dengan mekanisme *drag and drop* ke kolom status baru. Sistem kemudian menerima data perpindahan berupa identitas tugas dan target status yang dipilih.

Sebelum perubahan disimpan, sistem melakukan proses *Dynamic RBAC Check* untuk memvalidasi hak akses pengguna berdasarkan konteks hubungan terhadap tugas, seperti *Assignee*, *Creator*, atau *Admin Space*. Jika pengguna tidak memiliki hak akses, sistem akan menolak permintaan dengan respons *403 Forbidden*, mengembalikan kartu tugas ke posisi awal, serta menampilkan pesan kesalahan.

Sebaliknya, apabila validasi berhasil, sistem memperbarui status tugas pada basis data MongoDB dan mencatat aktivitas perubahan ke dalam *audit log*. Setelah itu, sistem mengirimkan pembaruan data secara *real-time* melalui *WebSocket* sehingga seluruh pengguna pada *workspace* yang sama dapat menerima perubahan tampilan *Kanban Board* secara langsung.

Secara umum, diagram ini menunjukkan mekanisme validasi akses dinamis, pengelolaan workflow tugas, serta sinkronisasi data *real-time* pada sistem manajemen tugas multi-tenant.

iii. Sequence Diagram
1) Login dan Validasi JWT



Gambar 3. 12 Sequence diagram login dan validasi token JWT

Gambar di atas menunjukkan urutan interaksi antar komponen sistem pada proses autentikasi dan validasi akses pengguna menggunakan mekanisme JSON Web Token (JWT). Sequence diagram tersebut terdiri atas dua proses utama, yaitu proses login (penerbitan token) dan proses validasi akses terhadap resource terproteksi.

a) *Alur Login (Penerbitan Token)*

Proses dimulai ketika pengguna (*client*) mengirimkan kredensial berupa email/username dan password melalui *handler autentikasi*. Permintaan tersebut diteruskan ke layanan Autentikasi untuk dilakukan proses validasi data pengguna dengan mengakses database MongoDB melalui Repositori Pengguna. Setelah data pengguna ditemukan, sistem melakukan proses validasi password menggunakan metode *enkripsi Bcrypt*.

Pada kondisi *password* valid, sistem akan menghasilkan *Access Token* dan *Refresh Token* melalui Utilitas JWT. Token yang berhasil dibuat kemudian dikirimkan kembali kepada pengguna dan disimpan pada *cookie browser* sebagai mekanisme *autentikasi* sesi dengan respons HTTP 200 OK.

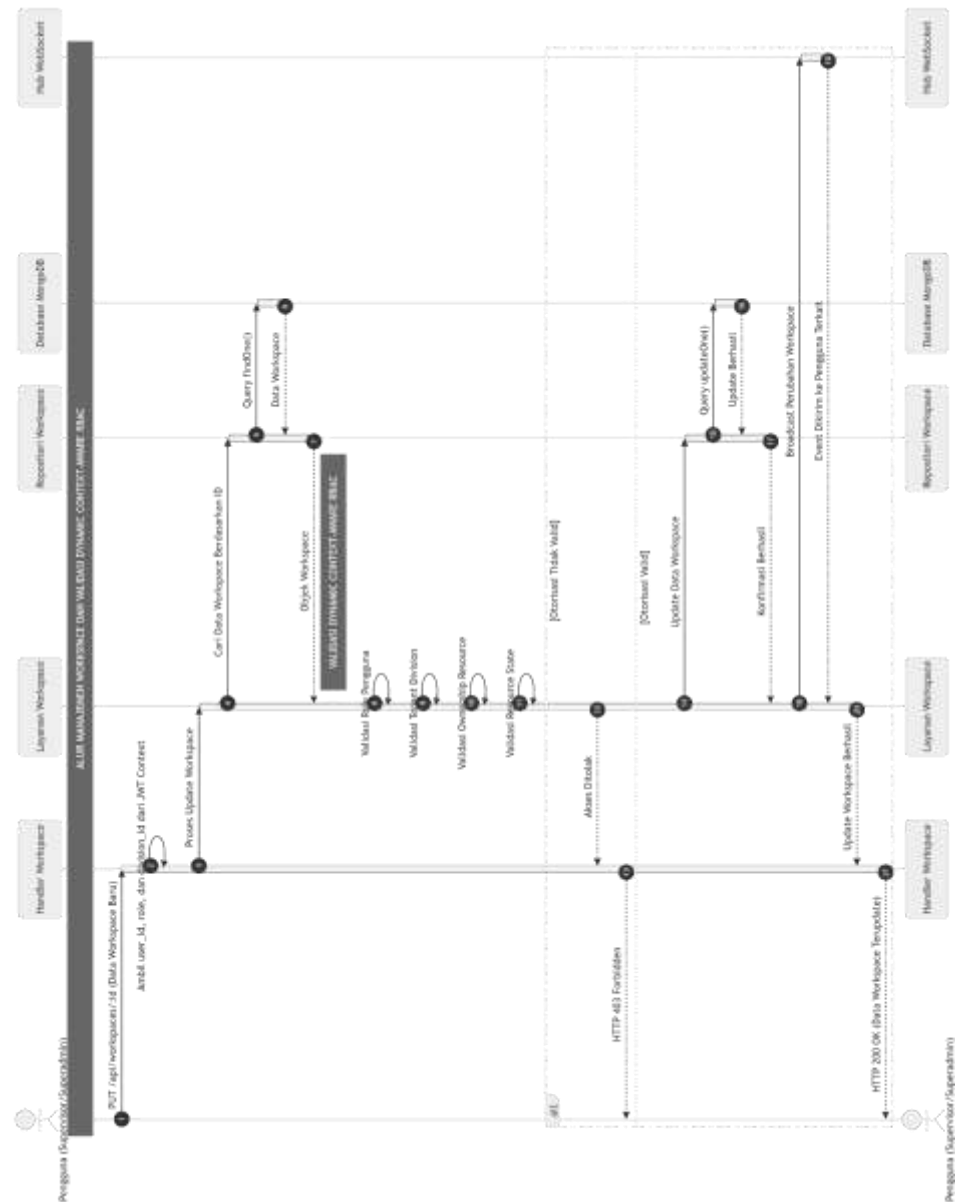
Sebaliknya, apabila *password* tidak sesuai, sistem akan menghentikan proses autentikasi dan mengembalikan respons HTTP 401 *Unauthorized* sebagai bentuk penolakan akses.

b) *Alur Validasi Akses Resource Terproteksi*

Ketika pengguna mengakses fitur atau resource yang memerlukan otorisasi, sistem akan mengambil token yang tersimpan pada *cookie browser* untuk dilakukan proses validasi menggunakan Utilitas JWT. Jika token dinyatakan valid, sistem akan mengekstraksi informasi identitas pengguna seperti *user_id*, *role*, dan *division_id* untuk disimpan ke dalam konteks request sistem.

Informasi tersebut selanjutnya digunakan sebagai dasar proses otorisasi berbasis *Dynamic Context-Aware RBAC*, termasuk validasi role pengguna, *tenant isolation* antar-divisi, dan validasi hak akses terhadap *resource* sistem. Setelah proses validasi berhasil dilakukan, sistem akan memberikan respons HTTP 200 OK beserta data yang diminta oleh pengguna.

2) Sequence Diagram Manajemen Space dengan Validasi RBAC (Context Aware)



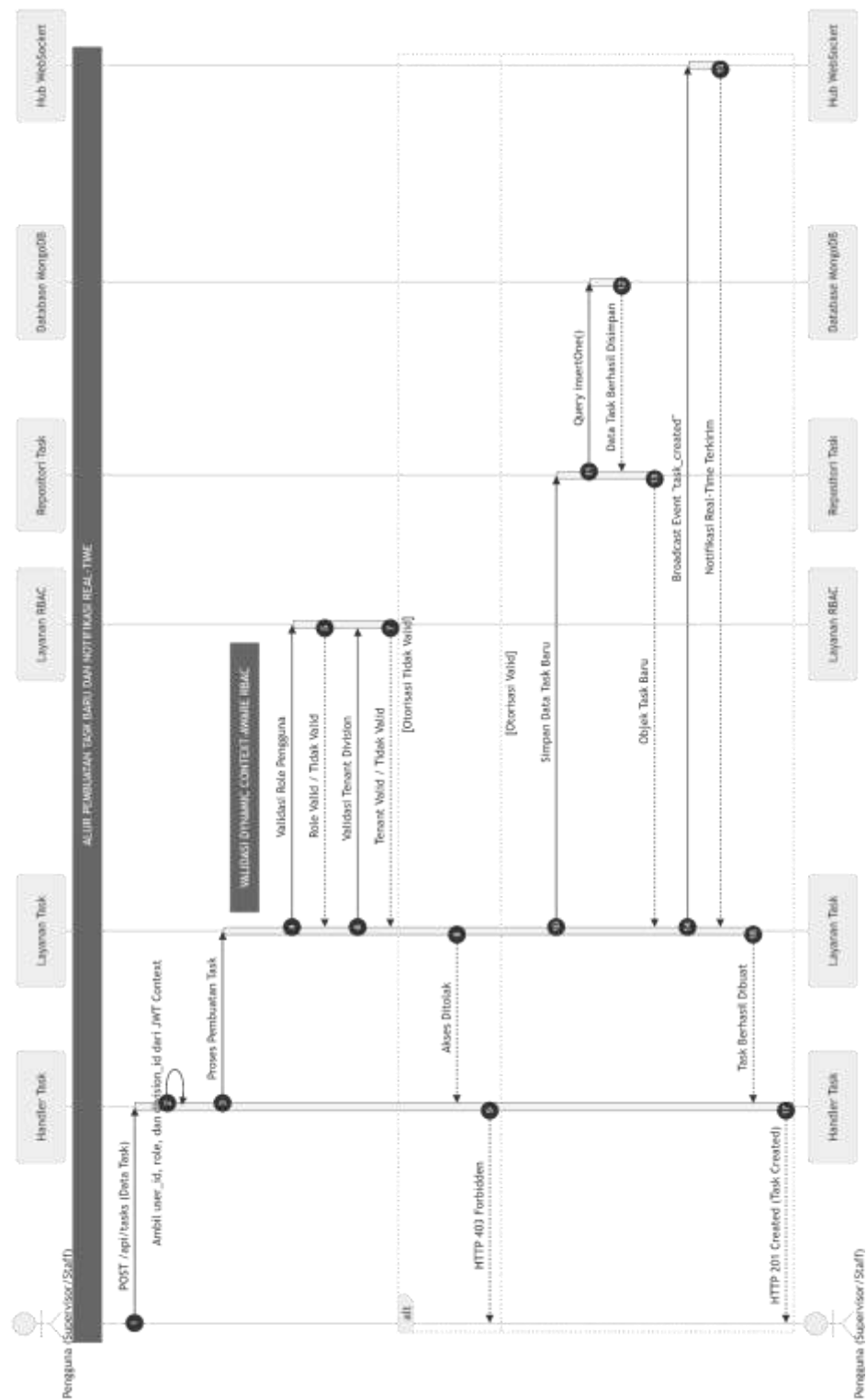
Gambar 3. 13 Sequence diagram manajemen space dengan validasi RBAC context aware

Gambar di atas menggambarkan proses manajemen *workspace* yang menerapkan mekanisme *Dynamic Context-Aware* RBAC pada sistem manajemen tugas multi-tenant. Proses dimulai ketika pengguna dengan role supervisor atau superadmin mengirimkan permintaan pembaruan data *workspace* melalui *Handler Workspace*. Sistem kemudian mengambil informasi identitas pengguna dari *JWT context*, seperti *user_id*, *role*, dan *division_id* sebagai dasar proses validasi akses.

Selanjutnya, layanan *workspace* mengambil data *workspace* dari database MongoDB melalui *repository workspace* untuk dilakukan serangkaian validasi otorisasi berbasis konteks. Validasi tersebut mencakup validasi role pengguna, *tenant division*, *ownership resource*, dan *resource state* guna memastikan bahwa pengguna memiliki hak akses yang sesuai terhadap *resource* yang dikelola.

Apabila salah satu proses validasi tidak terpenuhi, sistem akan menolak akses dan mengembalikan respons *HTTP 403 Forbidden*. Sebaliknya, apabila seluruh validasi berhasil dilakukan, sistem akan memperbarui data *workspace* pada database dan mengirimkan notifikasi perubahan secara real-time melalui *Hub WebSocket* kepada pengguna terkait. Setelah proses berhasil diselesaikan, sistem memberikan respons *HTTP 200 OK* sebagai indikasi bahwa pembaruan data *workspace* berhasil dilakukan.

3) Sequence Diagram Membuat Task Baru dengan Real-time Notification



Gambar 3. 14 Sequence Diagram Membuat Task Baru dengan Real-time Notification

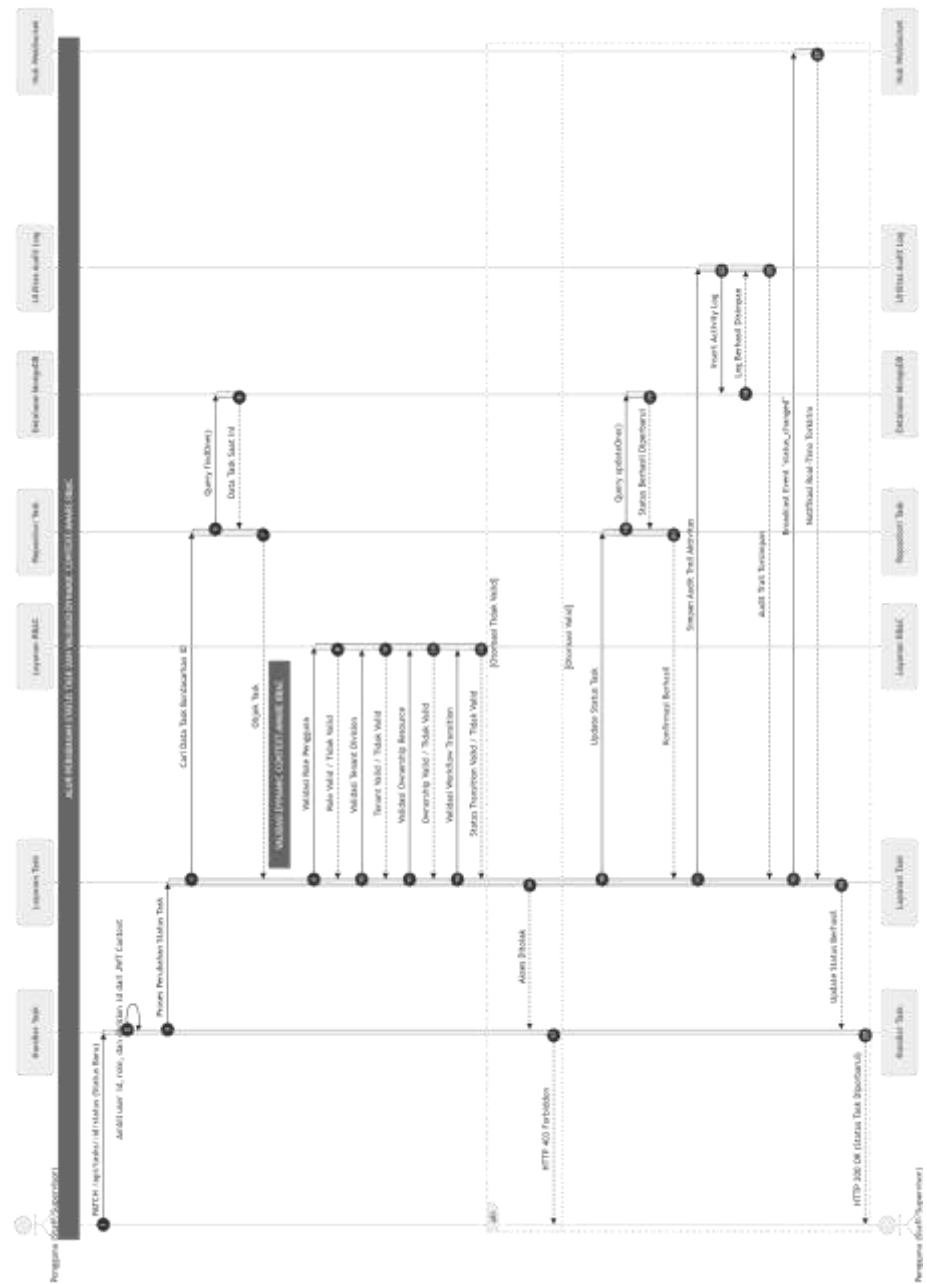
Gambar di atas menggambarkan proses pembuatan task baru yang menerapkan mekanisme *Dynamic Context-Aware* RBAC serta notifikasi *real-time* pada sistem manajemen tugas multi-tenant. Proses dimulai ketika pengguna mengirimkan permintaan pembuatan task melalui *handler task*. Sistem kemudian mengambil informasi identitas pengguna dari *JWT context*, seperti *user_id*, *role*, dan *division_id* sebagai dasar proses validasi akses.

Selanjutnya, Layanan Task melakukan proses validasi otorisasi berbasis konteks melalui Layanan RBAC. Validasi tersebut mencakup validasi role pengguna dan tenant division untuk memastikan bahwa pengguna memiliki hak akses yang sesuai dalam membuat task pada divisi terkait.

Apabila proses validasi gagal, sistem akan menolak permintaan dan mengembalikan respons *HTTP 403 Forbidden*. Sebaliknya, apabila validasi berhasil dilakukan, sistem akan menyimpan data task baru ke dalam database MongoDB melalui repositori task. Setelah data berhasil disimpan, sistem akan mengirimkan notifikasi perubahan secara *real-time* melalui Hub *WebSocket* dengan *event task_created* kepada pengguna terkait.

Setelah seluruh proses berhasil diselesaikan, sistem mengembalikan respons *HTTP 201 Created* sebagai indikasi bahwa task baru berhasil dibuat.

4) Sequence Diagram Perubahan Status Task



Gambar 3. 15 Sequence Diagram Perubahan Status Task

Gambar di atas menggambarkan proses perubahan status task pada sistem manajemen tugas multi-tenant yang menerapkan mekanisme *Dynamic Context-Aware* RBAC. Proses dimulai ketika pengguna mengirimkan permintaan perubahan status task melalui *handler task*. Sistem kemudian mengambil informasi identitas pengguna dari *JWT context*, seperti *user_id*, *role*, dan *division_id* sebagai dasar proses validasi akses.

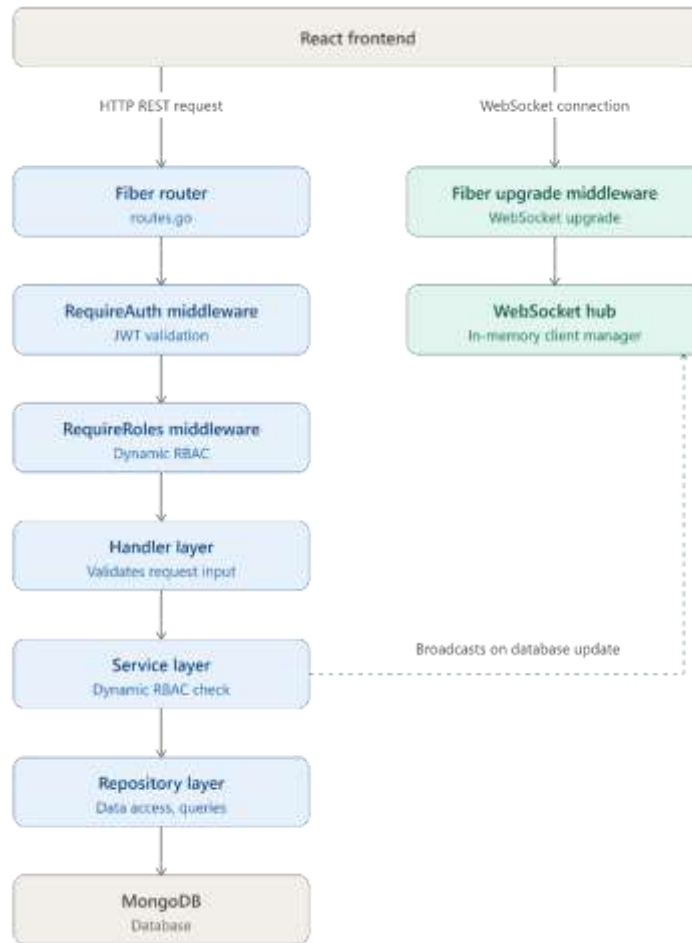
Selanjutnya, Layanan Task mengambil data *task* dari *database* MongoDB melalui *repository task* untuk dilakukan proses validasi otorisasi berbasis konteks melalui Layanan RBAC. Validasi tersebut mencakup validasi *role* pengguna, *tenant division*, *ownership resource*, serta validasi *workflow transition* untuk memastikan bahwa pengguna memiliki hak akses yang sesuai terhadap task dan perpindahan status yang dilakukan.

Apabila salah satu proses validasi tidak terpenuhi, sistem akan menolak permintaan dan mengembalikan respons *HTTP 403 Forbidden*. Sebaliknya, apabila seluruh validasi berhasil dilakukan, sistem akan memperbarui status task pada database dan menyimpan riwayat aktivitas ke dalam *audit trail system* sebagai bentuk pencatatan aktivitas pengguna.

Setelah proses pembaruan berhasil dilakukan, sistem akan mengirimkan *notifikasi* perubahan status secara *real-time* melalui *hub webSocket* kepada pengguna terkait. Sistem kemudian mengembalikan respons *HTTP 200 OK* sebagai indikasi bahwa perubahan status task berhasil dilakukan.

iv. Perancangan Arsitektur Sistem

Perancangan arsitektur sistem bertujuan untuk menggambarkan hubungan antar komponen utama yang digunakan dalam pengembangan aplikasi manajemen tugas berbasis *multi-tenant*. Arsitektur dirancang menggunakan prinsip *Clean Architecture* untuk memisahkan tanggung jawab setiap lapisan sistem. Selain itu, mekanisme *Dynamic Context-Aware Role-Based Access Control (RBAC)* diintegrasikan pada proses otorisasi sebagai kontribusi utama penelitian.



Gambar 3. 16 Perancangn Arsitektur Sistem

Gambar 3.16 menunjukkan arsitektur sistem yang terdiri atas *React Frontend*, *Fiber Router*, *WebSocket Hub*, dan *MongoDB*. Sistem menggunakan *HTTP REST API* untuk proses pertukaran data serta *WebSocket* untuk sinkronisasi data secara *real-time*.

Pada jalur *REST API*, setiap permintaan pengguna diproses melalui *RequireAuth Middleware* untuk autentikasi menggunakan *JWT*, kemudian dilanjutkan ke *RequireRoles Middleware* untuk melakukan validasi awal terhadap *role* pengguna yang berhak mengakses *endpoint*.

Selanjutnya, permintaan diteruskan ke *Handler Layer*, *Service Layer*, dan *Repository Layer*. Pada *Service Layer*, diterapkan mekanisme *Dynamic Context-Aware RBAC* yang melakukan pengambilan keputusan otorisasi berdasarkan empat konteks utama, yaitu *role*, *ownership*, *resource state*, dan *tenant boundary*, sebelum sistem memberikan izin akses terhadap suatu operasi.

Selain itu, perubahan data yang berhasil diproses akan didistribusikan melalui *WebSocket Hub* sehingga pengguna yang memiliki hak akses memperoleh pembaruan data secara *real-time* tanpa melakukan *refresh* halaman.

v. Perancangan Basis Data

Perancangan basis data menggunakan pendekatan *Document-Oriented Database* pada MongoDB. Berbeda dengan basis data relasional, MongoDB tidak mengharuskan skema yang kaku, namun struktur data dirancang secara konsisten untuk menjamin integritas data dan mendukung mekanisme *multi-tenancy* serta *Dynamic Context-Aware RBAC*. Sistem menggunakan 5 collection utama dengan dua pola penyimpanan. *Refrencing* untuk relasi antar collection utama dan *embedding* untuk data yang selalu diakses bersama dokumen induknya.

1) Collection 1 : *Users*

Collection users menyimpan seluruh data pengguna sistem beserta identitas peran dan divisi. Field role dan division menjadi klaim utama JSON Web Token (JWT) dan dasar validasi isolasi akses pada setiap permintaan.

Tabel 3. 2 *Collection Users*

No	Field	Tipe Data	Keterangan
1	_id	ObjectId	Primary key unik yang digenerate otomatis
2	Namalengkap	String	Nama lengkap pengguna
3	Username	String	Username unik untuk autentikasi login
4	Email	String	Alamat email unik pengguna
5	Password	String	Hash bcrypt dari password pengguna
6	Division_id	ObjectId	ID Divisi (FK ke Collection Divisions)
7	Position	String	Jabatan pengguna dalam divisi

No	Field	Tipe Data	Keterangan
8	Role	String	Peran pengguna: superadmin / supervisor / member
9	Phone	String	Nomor telepon pengguna (bersifat optional)
10	createdAt	Date	Waktu akun dibuat
11	updatedAt	Date	Waktu akun terakhir diperbarui

2) Collection 2 : spaces

Collection spaces menyimpan ruang kerja (workspace) yang menjadi unit isolasi multi-tenant. Setiap Space memiliki daftar divisi yang berhak mengaksesnya (divisions), tahapan alur kerja (status), dan tingkatan prioritas (priority) yang disimpan secara embedded.

Tabel 3. 3 Collection spaces

NO	Field	Tipe Data	Keterangan
1	_id	Object.id	Primary key unik
2	Name	String	Nama ruang kerja
3	Description	String	Deskripsi ruang kerja
4	owner_id	ObjectId	Referensi ke users.id pemilik space
5	divisions[]	Array<object>	Daftar divisi yang berhak mengakses space
6	divisions[].id	String	ID unik divisi
7	divisions[].name	String	Nama divisi
8	divisions[].color	String	Kode warna
9	status[]	Array<object>	Tahapan Kanban board yang disimpan di embedded
10	status[].id	String	Id unik tahapan
11	status[].title	String	Label tahapan (todo, Doing dll)

NO	Field	Tipe Data	Keterangan
12	status[].order	Integer	Urutan tampil tahapan
13	priority[]	Array<object>	Tingkatan prioritas tugas per space
14	priority[].id	ObjectId	ID unik prioritas
15	priority[].name	String	Nama prioritas (contoh: High, Medium, Low)
16	priority.description	String	Deskripsi tingkat prioritas
17	priority[].color	String	Kode warna hex prioritas
18	createdAt	Date	Waktu Space dibuat
19	updatedAt	Date	Waktu Space terakhir diperbarui

3) Struktur Collection tasks

Collection tasks menyimpan data tugas beserta seluruh konteks akses dinamisnya.

Tabel 3. 4 *Collection Tasks*

NO	Field	Tipe Data	Keterangan
1	_id	ObjectId	Primary key unik tugas
2	requestor_name	String	Nama pemohon/pelapor
3	requestor_division	String	Divisi asal pemohon
4	Phone	String	Nomor telepon pelapor
5	Keluhan	String	Judul keluhan/masalah
6	Description	String	Detail deskripsi masalah
7	space_id	ObjectId	Referensi ke spaces.id

NO	Field	Tipe Data	Keterangan
8	priority_id	ObjectId	Referensi ke ID di dalam spaces.priority
9	assign_to[]	Array<string>	Daftar User ID yang ditugaskan
10	start_date	Date	Waktu mulai pengerjaan
11	due_date	Date	Batas waktu pengerjaan
12	Status	String	Status saat ini (Request, Todo, dll)
13	todo_approval	Date	Waktu persetujuan supervisor
14	done_approval	Date	Waktu konfirmasi selesai
15	created_by	ObjectId	Referensi ke users.id pembuat tugas
16	spv_req	ObjectId	Referensi ke users.id supervisor terkait
17	assigned_division	String	Divisi target yang ditugasi menyelesaikan tugas lintas divisi.
18	assignment_requester	ObjectId	Referensi ke <i>users._id</i> pengaju penugasan lintas divisi
19	assignment_requester_d_at	Date	Waktu pengajuan tugas lintas divisi

NO	Field	Tipe Data	Keterangan
20	attachments[]	Array<object>	Daftar lampiran file/link
21	attachments[].id	ObjectId	ID unik lampiran
22	attachments[].name	String	Nama file/link
23	attachments[].path	String	Lokasi file di server atau URL link
24	attachments[].type	String	Tipe lampiran (file / link)
25	attachments.mime_type	String	Mime Type berkas (contoh: "image/png", "application/pdf")
26	attachments[].size	Integer	Ukuran file dalam bytes
27	attachments[].uploaded_by	ObjectId	Referensi ke users.id pengunggah
28	attachments.uploader_type	String	Peran pengunggah (contoh: "staff", "supervisor")
29	comments[]	Array<object>	Daftar diskusi/komentar
30	comments[].id	ObjectId	ID unik komentar
31	comments[].user_id	ObjectId	Referensi ke users.id pengirim
32	comments[].role	String	Role pengirim saat mengirim komentar
33	comments[].comment	String	Isi teks komentar

NO	Field	Tipe Data	Keterangan
34	comments[].created_at	Date	Waktu komentar dikirim
35	createdAt	Date	Waktu tugas dibuat
36	updatedAt	Date	Waktu tugas terakhir diperbarui

4) Struktur Collection notifications

Collection notifications menyimpan notifikasi yang dikirimkan kepada pengguna. Setiap notifikasi terikat pada satu pengguna dan memiliki tipe yang membedakan jenis peristiwa yang memicunya.

Tabel 3. 5 Collection Tasks

No	Field	Tipe Data	Keterangan
1	_id	ObjectId	Primary key unik notifikasi
2	user_id	ObjectId	Referensi ke users.id (Penerima notifikasi)
3	actor_id	ObjectId	Referensi ke users.id (Pemicu/Aktor peristiwa)
4	task_id	ObjectId	Referensi ke tasks.id yang terkait
5	space_id	ObjectId	Referensi ke spaces.id tempat peristiwa terjadi
6	Type	String	Jenis notifikasi (assignment, status_change, dll)
7	Message	String	Isi pesan notifikasi yang akan ditampilkan
8	is_read	Boolean	Status apakah notifikasi sudah dibaca (True/False)
9	Meta	Object	Data tambahan dinamis (misal: status lama & baru)
10	createdAt	Date	Waktu notifikasi dibuat

5) Struktur collection audits

Collection audits mencatat jejak aktivitas (audit trail) setiap perubahan kritis pada sistem sebagai implementasi prinsip akuntabilitas dalam mekanisme Dynamic Context-Aware RBAC. Setiap aksi yang melibatkan perubahan status atau hak akses tugas direkam secara otomatis.

Tabel 3. 6 Collection Audits

No	Field	Type Data	Keterangan
1	_id	ObjectId	Primary key unik dokumen audit
2	task_id	ObjectId	Referensi ke tasks.id yang dimodifikasi
3	Action	String	Jenis tindakan (misal: status_change, task_approved)
4	actor_id	String	ID Pengguna yang melakukan tindakan (Aktor)
5	Detail	Object	Detail teknis perubahan (misal: old_status & new_status)
6	createdAt	Date	Waktu pencatatan jejak audit

2. Tahap Pengembangan Artefak

Pengembangan artefak dilakukan sebagai tahap implementasi dari hasil perancangan sistem yang telah dibuat sebelumnya. Artefak yang dikembangkan berupa sistem manajemen tugas multi-tenant berbasis *backend* service yang menerapkan mekanisme *Dynamic Context-Aware* RBAC untuk mengontrol hak akses pengguna secara dinamis berdasarkan *role*, *tenant division*, *ownership resource*, dan *workflow transition*.

a) Implementasi Backend API

Sistem dikembangkan menggunakan *framework Go Fiber* dengan pendekatan *REST API* untuk mendukung komunikasi data antara client dan server. Arsitektur backend dirancang menggunakan pendekatan modular yang terdiri atas *handler*, *service*, *repository*, dan *database layer* guna mempermudah proses pengelolaan logika bisnis dan validasi akses sistem.

b) Implementasi JWT Authentication

Sistem menerapkan mekanisme autentikasi berbasis *JSON Web Token* (JWT) untuk menjaga keamanan sesi pengguna. Informasi identitas pengguna seperti *user_id*, *role*, dan *division_id* disimpan pada token dan digunakan sebagai dasar proses validasi akses pada setiap *request* yang masuk ke sistem.

c) Implementasi *Dynamic Context-Aware* RBAC

Implementasi *Dynamic Context-Aware Role-Based Access Control* (RBAC) dilakukan pada *Service Layer* sebagai mekanisme utama dalam proses pengambilan keputusan otorisasi terhadap setiap permintaan akses. Berbeda dengan *Role-Based Access Control* (RBAC) konvensional yang hanya mempertimbangkan peran pengguna, mekanisme yang dikembangkan pada penelitian ini mengintegrasikan empat komponen validasi sebagai berikut.

i. *Role Validation*

Validasi *role* dilakukan untuk memastikan bahwa pengguna memiliki hak akses sesuai dengan perannya pada sistem sebelum suatu operasi dijalankan.

ii. *Ownership Validation*

Validasi *ownership* dilakukan untuk memverifikasi hubungan operasional pengguna terhadap sumber daya yang diakses. Pengguna hanya diizinkan melakukan operasi apabila memiliki hubungan yang sah terhadap *task*.

iii. *Tenant Boundary Validation*

Validasi *tenant boundary* dilakukan untuk menjaga isolasi data antar-divisi pada lingkungan *multi-tenant*. Mekanisme ini memastikan bahwa pengguna hanya dapat mengakses *task* maupun *workspace* yang berada dalam ruang lingkup *tenant* yang sesuai atau memiliki hubungan operasional yang sah.

iv. *Resource State Validation*

Validasi *resource state* dilakukan untuk memastikan bahwa setiap operasi terhadap *task* mengikuti kondisi sumber daya yang berlaku melalui mekanisme *workflow transition*. Perubahan status hanya diizinkan sesuai urutan *Request*, *Todo*, *On Going*, *Testing*, *Done*, dan *Archive*, serta berdasarkan kewenangan pengguna pada setiap tahapan *workflow*.

Keempat proses validasi tersebut dijalankan secara berurutan pada *Service Layer* sebelum sistem memberikan keputusan otorisasi. Apabila seluruh proses validasi terpenuhi, sistem memberikan izin akses

terhadap operasi yang diminta. Sebaliknya, apabila salah satu proses validasi tidak terpenuhi, sistem akan menolak permintaan akses sehingga keamanan, integritas data, dan isolasi antar-tenant tetap terjaga.

d) Implementasi Audit Trail

Sistem menyediakan mekanisme *audit trail* untuk mencatat seluruh aktivitas penting pengguna, seperti perubahan status task dan proses pembaruan data. Pencatatan aktivitas ini digunakan untuk mendukung *traceability* dan monitoring aktivitas sistem secara terstruktur.

e) Implementasi *Real-Time Notification*

Sistem menerapkan mekanisme *real-time notification* menggunakan *WebSocket* untuk mendukung sinkronisasi data secara langsung antar pengguna. Setiap perubahan data task maupun *workspace* akan dikirimkan secara otomatis kepada pengguna terkait sehingga informasi sistem dapat diperbarui secara real-time.

d. Tahap Demonstrasi Artefak

Tahap demonstrasi dilakukan untuk menunjukkan bahwa artefak *backend* yang dikembangkan mampu menjalankan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) sesuai rancangan penelitian. Demonstrasi dilakukan melalui serangkaian skenario penggunaan sistem yang merepresentasikan proses bisnis pada sistem manajemen tugas *multi-tenant* sebelum dilakukan tahap evaluasi menggunakan metode *black-box testing*.

Rencana demonstrasi artefak dilakukan melalui beberapa skenario utama sebagai berikut.

i. Demonstrasi *Autentikasi Pengguna*

Demonstrasi dilakukan dengan mensimulasikan proses autentikasi pengguna melalui *endpoint login* menggunakan mekanisme *JSON Web Token* (JWT). *Backend* diharapkan mampu memverifikasi kredensial pengguna, menghasilkan identitas berupa *user_id*, *role*, dan *division*, serta membentuk *access token* yang digunakan sebagai konteks awal pada mekanisme *Dynamic Context-Aware RBAC*.

ii. Demonstrasi *Tenant Boundary*

Demonstrasi dilakukan dengan mensimulasikan akses pengguna terhadap *workspace* milik divisi lain, termasuk skenario kolaborasi melalui *outgoing task*. *Backend* diharapkan mampu menjaga isolasi data antar-divisi dengan hanya menampilkan tugas yang memiliki hubungan operasional terhadap pengguna, sedangkan tugas internal yang tidak berkaitan tetap disembunyikan.

iii. Demonstrasi *Ownership*

Demonstrasi dilakukan dengan mensimulasikan perubahan data tugas oleh pengguna yang memiliki maupun tidak memiliki hubungan operasional terhadap tugas tersebut. *Backend* diharapkan hanya mengizinkan operasi kepada pengguna yang memiliki kepemilikan sesuai konteks operasional, sedangkan pengguna lain memperoleh penolakan akses sesuai mekanisme *ownership* yang diterapkan.

iv. Demonstrasi *Resource State*

Demonstrasi dilakukan dengan mensimulasikan perubahan status tugas sesuai alur *workflow* yang diterapkan pada *backend*. *Backend* diharapkan memvalidasi setiap transisi status berdasarkan kondisi sumber daya (*resource state*) sehingga perubahan status hanya dapat dilakukan sesuai urutan *workflow* dan kewenangan pengguna.

v. Demonstrasi *Audit Trail*

Demonstrasi dilakukan dengan mensimulasikan berbagai aktivitas penting pada sistem, seperti perubahan status, penugasan, persetujuan, dan penolakan tugas. *Backend* diharapkan secara otomatis mencatat

setiap aktivitas tersebut ke dalam koleksi *audit* sehingga seluruh riwayat perubahan dapat ditelusuri kembali sebagai bentuk *traceability* dan *accountability*.

vi. Demonstrasi *Real-Time Notification*

Demonstrasi dilakukan dengan mensimulasikan perubahan data tugas pada *backend*. Setelah proses bisnis berhasil dijalankan, backend diharapkan mengirimkan *event* perubahan melalui *WebSocket* kepada pengguna yang terhubung sehingga informasi terbaru dapat diterima secara langsung tanpa memerlukan proses *refresh* halaman.

e. Tahap Pengujian Artefak

Tahap pengujian artefak dilakukan untuk memverifikasi bahwa *backend* yang dikembangkan mampu mengimplementasikan mekanisme *Dynamic Context Aware Role Based Access Control* (RBAC) sesuai kebutuhan penelitian pada lingkungan *multi-tenant*. Pengujian menggunakan *black box testing*, yaitu metode pengujian yang berfokus pada evaluasi fungsionalitas sistem berdasarkan keluaran (*output*) yang dihasilkan terhadap masukan (*input*) tanpa memperhatikan implementasi kode program [24].

Dalam penelitian ini, metode *black box testing* diterapkan melalui serangkaian skenario pengujian yang dirancang berdasarkan mekanisme utama *Dynamic Context-Aware RBAC*, yaitu *authentication*, *tenant boundary*, *ownership*, *resource state*, *audit trail*, dan *real-time notification*. Setiap skenario merepresentasikan aspek yang berbeda dalam proses pengambilan keputusan otorisasi sehingga secara kolektif digunakan untuk mengevaluasi kesesuaian implementasi mekanisme *Dynamic Context-Aware RBAC* terhadap tujuan penelitian.

Pengujian dilakukan menggunakan beberapa akun pengguna dengan peran superadmin, supervisor, dan staff untuk memverifikasi setiap mekanisme tersebut. Hasil pengujian dianalisis secara deskriptif dengan membandingkan keluaran sistem terhadap perilaku yang diharapkan berdasarkan aturan otorisasi yang telah dirancang.

Berdasarkan mekanisme utama *Dynamic Context-Aware RBAC* yang diimplementasikan pada artefak, pengujian disusun secara bertingkat untuk memudahkan proses evaluasi. Tahap pertama menyajikan rencana pengujian umum yang menggambarkan mekanisme utama yang akan dievaluasi beserta target dan indikator keberhasilannya. Selanjutnya, setiap mekanisme dijabarkan ke dalam skenario pengujian spesifik yang memuat aktor, langkah pengujian, dan hasil yang diharapkan. Dengan penyusunan tersebut, setiap mekanisme dapat dievaluasi secara sistematis dan terstruktur sesuai tujuan penelitian.

i. Rencana Pengujian Umum

Tabel 3. 7 Rencana Pengujian Umum

No	Skenario Pengujian	Target Pengujian	Indikator Keberhasilan
1	Pengujian <i>Authentication</i>	Memverifikasi proses autentikasi pengguna menggunakan JSON Web Token (JWT).	Pengguna dengan kredensial yang valid berhasil login dan memperoleh JWT, sedangkan kredensial tidak valid ditolak oleh sistem.
2	Pengujian <i>Tenant Boundary</i>	Memverifikasi mekanisme isolasi data antar-tenant serta akses lintas divisi berdasarkan konteks penugasan.	Pengguna hanya dapat mengakses workspace dan data tugas sesuai tenant atau hubungan operasional yang sah.
3	Pengujian <i>Ownership</i>	Memverifikasi bahwa operasi terhadap task hanya dapat dilakukan oleh pengguna yang memiliki hubungan operasional atau administratif terhadap tugas.	<i>Backend</i> hanya mengizinkan perubahan data kepada pengguna yang memiliki <i>ownership</i> sesuai aturan sistem.
4	Pengujian <i>Resource State</i>	Memverifikasi validasi perubahan status task berdasarkan kondisi sumber daya (<i>resource state</i>) melalui <i>workflow transition</i> .	Perubahan status hanya dapat dilakukan sesuai urutan <i>workflow</i> dan kewenangan pengguna.
5	Pengujian <i>Audit Trail</i>	Memverifikasi bahwa setiap aktivitas yang mempengaruhi perubahan data tercatat secara otomatis sebagai bagian dari mekanisme akuntabilitas sistem.	Aktivitas pengguna berhasil dicatat secara otomatis dan dapat ditelusuri kembali melalui <i>audit log</i> .
6	Pengujian <i>Real-Time Notification</i>	Memverifikasi bahwa perubahan data telah berhasil diproses	Perubahan data diterima oleh pengguna terkait secara <i>real-time</i> tanpa

		<i>backend</i> disinkronkan kepada pengguna terkait secara <i>real-time</i> melalui <i>Websocket</i> .	melakukan <i>refresh</i> halaman.
--	--	--	-----------------------------------

ii. Detail Skenario Pengujian Spesifik

Tabel 3. 8 Skenario Pengujian Spesifik

ID Test Case	Skenario Pengujian	Role Pengguna	Data Uji	Langkah Pengujian	Hasil yang Diharapkan
<i>Authentication</i>					
TC-AU TH-01	Login menggunakan akun valid	Supervisor / Staff	Email dan password valid	Memasukkan email dan password yang benar, kemudian menekan tombol Login .	Sistem berhasil melakukan autentikasi, mengembalikan HTTP 200 OK, dan mengarahkan pengguna ke dashboard sesuai hak aksesnya
TC-AU TH-02	Login menggunakan kredensial yang tidak valid	Supervisor / Staff	Email valid dan password tidak valid	Memasukkan email yang benar dan password yang salah, kemudian menekan tombol <i>Login</i> .	Sistem menolak autentikasi, tidak menghasilkan JWT, dan mengembalikan <i>HTTP 401 Unauthorized</i> .
<i>Tenant Boundary</i>					
TC-TB-01	Mengakses workspace divisi lain tanpa hubungan operasional	Supervisor Divisi A	Workspace divisi lain tanpa relasi operasional	Supervisor mencoba mengakses workspace Divisi B yang tidak memiliki hubungan operasional.	Sistem menolak akses, mengembalikan <i>HTTP 403 Forbidden</i> , serta tidak menampilkan data tenant lain.

ID Test Case	Skenario Pengujian	Role Pengguna	Data Uji	Langkah Pengujian	Hasil yang Diharapkan
TC-TB-02	Supervisor mengakses incoming task dari divisi lain	Supervisor Divisi B	Incoming task dengan relasi operasional yang sah	Backend mengizinkan akses ke workspace terkait dan hanya menampilkan outgoing task yang diterima.	Supervisor membuka incoming task yang diteruskan dari divisi lain.
TC-TB-03	Staff mengakses incoming task sesuai assignment	Staff Divisi B	Incoming task dengan assignment yang valid	Staff membuka incoming task yang ditugaskan kepadanya.	Sistem hanya menampilkan dan mengizinkan pengelolaan task yang sesuai dengan assignment dan tenant boundary.
<i>Ownership</i>					
TC-OWN-01	Mengubah task yang menjadi tanggung jawab sendiri	Staff	Task dengan assignment milik sendiri	Staff mengubah data atau status task yang ditugaskan kepadanya.	Sistem mengizinkan perubahan data dan mengembalikan <i>HTTP 200 OK</i> .
TC-OWN-02	Mengubah task yang bukan menjadi tanggung jawab	Staff	Task milik pengguna lain	Staff mencoba mengubah task yang tidak ditugaskan kepadanya.	Sistem menolak perubahan, dan tidak melakukan perubahan data.
TC-OWN-03	Mengubah outgoing task yang telah diteruskan	Supervisor Pengaju	Outgoing task yang telah diteruskan	Supervisor mencoba mengubah outgoing task yang telah	Sistem menolak perubahan dan hanya memberikan hak akses

ID Test Case	Skenario Pengujian	Role Pengguna	Data Uji	Langkah Pengujian	Hasil yang Diharapkan
				diteruskan ke divisi tujuan.	<i>view-only</i> terhadap task tersebut.
<i>Resource State</i>					
TC-RS-01	Mengubah status task sesuai urutan workflow	Staff (Assignee)	Task dengan status <i>Todo</i>	Staff mengubah status task dari <i>Todo</i> → <i>On Going</i> → <i>Testing</i> .	Sistem mengizinkan perubahan status sesuai aturan workflow transition.
TC-RS-02	Mengubah status dengan melompati workflow	Staff (Assignee)	Task dengan status <i>Todo</i>	Staff mencoba mengubah status <i>Todo</i> → <i>Done</i> atau <i>Testing</i> → <i>Todo</i> .	Sistem menolak perubahan karena tidak sesuai aturan workflow transition.
TC-RS-03	Mengubah task yang telah berstatus Archive	Staff	Task dengan status <i>Archive</i>	Staff mencoba mengubah task yang telah berada pada status <i>Archive</i> .	Sistem menolak perubahan karena task berada pada status terminal (<i>read-only</i>).
TC-RS-04	Staff melakukan transisi status yang menjadi kewenangan Supervisor	Staff	Task dengan status <i>Testing</i>	Staff mencoba mengubah status <i>Testing</i> → <i>Done</i> atau <i>Done</i> → <i>Archive</i> .	Sistem menolak perubahan karena tidak memenuhi aturan kewenangan pengguna.
TC-RS-05	Supervisor melakukan transisi status akhir	Supervisor	Task dengan status <i>Testing</i>	Supervisor mengubah status <i>Testing</i> → <i>Done</i> atau <i>Done</i> → <i>Archive</i> .	Sistem mengizinkan perubahan sesuai workflow transition dan kewenangan pengguna.

ID Test Case	Skenario Pengujian	Role Pengguna	Data Uji	Langkah Pengujian	Hasil yang Diharapkan
<i>Audit Trail</i>					
TC-AT-01	Pencatatan perubahan status tugas	Staff / Supervisor	Perubahan status task	Mengubah status task sesuai workflow.	Sistem mencatat aktivitas perubahan status pada audit trail secara otomatis.
TC-AT-02	Pencatatan penugasan task	Supervisor	Penugasan task	Menugaskan task kepada staff.	Sistem mencatat aktivitas penugasan pada audit trail.
<i>Real-Time Notification</i>					
TC-RT-01	Notifikasi pembuatan task	Supervisor	Task baru	Membuat task baru.	Sistem mengirim notifikasi real-time kepada pengguna terkait melalui WebSocket.
TC-RT-02	Notifikasi perubahan status task	Staff / Supervisor	Perubahan status task	Mengubah status task sesuai workflow.	Sistem mengirim notifikasi perubahan status secara real-time kepada pengguna terkait.

f. Tahap Evaluasi

Tahap evaluasi dilakukan setelah seluruh skenario pengujian artefak selesai dilaksanakan. Evaluasi bertujuan untuk menilai kesesuaian implementasi backend terhadap tujuan penelitian serta memverifikasi bahwa mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) telah berfungsi sesuai rancangan pada lingkungan sistem manajemen tugas multi-tenant.

Proses evaluasi dilakukan dengan membandingkan hasil pengujian terhadap indikator keberhasilan yang telah ditetapkan pada setiap skenario

Black Box Testing. Evaluasi mencakup mekanisme *authentication*, *tenant boundary*, *ownership*, *resource state*, *audit trail*, dan *real-time notification*. Setiap mekanisme dinyatakan memenuhi tujuan penelitian apabila perilaku sistem yang dihasilkan sesuai dengan aturan fungsional dan kebijakan otorisasi yang telah dirancang

D. Jadwal Penelitian

Jadwal penelitian disusun untuk mengalokasikan waktu ke setiap tahapan penelitian selama 6 bulan (Februari–Juli 2026).

Tabel 3. 9 Tabel Penelitian

JADWAL PENELITIAN							
NO	KEGIATAN	PELAKSANAAN					
		Februari 2026	Maret 2026	April 2026	Mei 2026	Juni 2026	Juli 2026
Tahap Persiapan							
1	Studi Literatur						
2	Identifikasi & Rumusan Masalah						
3	Penetapan Metode Penelitian						
4	Penyusunan & Revisi Proposal						
5	Seminar Proposal						
Tahap Penelitian							
1	Pengumpulan Data						
2	Perancangan Sistem						
3	Implementasi Backend						
4	Pengujian API						
5	Pengujian & Testing						
6	Analisis & Evaluasi						
Tahap Penulisan & Akhir							
1	Penulisan Skripsi (Bab I-V)						
2	Revisi Bimbingan Skripsi						
3	Sidang Skripsi						
4	Revisi Pasca Sidang						

BAB IV HASIL DAN PEMBAHASAN

A. Hasil

1. Hasil Pengembangan Artefak

Penelitian ini menghasilkan artefak berupa *backend* sistem manajemen tugas yang mengimplementasikan mekanisme *Dynamic Context-Aware Role-Based Acces Control* (RBAC). Hasil pengembangan artefak disajikan berdasarkan komponen *backend* yang mendukung implementasi mekanisme tersebut sebelum dilakukan tahap demonstrasi dan evaluasi.

a) Pengembangan Infrastruktur Backend



Gambar 4. 1 Struktur Folder *Backend*

Gambar 4.1 menunjukkan implementasi struktur folder *backend* yang mengikuti pendekatan *Clean Architecture*. Struktur proyek memisahkan komponen utama ke dalam folder *routes*, *middlewares*, *internal*, *websocket*, *config*, dan *utils*. Folder *internal* menjadi inti implementasi yang terdiri atas layer *domain*, *handler*, *service*, dan *repository*, sedangkan folder lainnya berfungsi sebagai komponen pendukung untuk konfigurasi aplikasi, pengelolaan endpoint, autentikasi, dan komunikasi *real-time*.

Selanjutnya, fungsi setiap komponen utama pada infrastruktur backend dijelaskan pada Tabel berikut :

Tabel 4. 1 Komponen Infrastruktur *Backend*

NO	KOMPONEN	FUNGSI
1	<i>Routes</i>	Mendefinisikan endpoint REST API dan mengarahkan request ke handler yang sesuai.
2	<i>Middlewares</i>	Melakukan autentikasi pengguna serta validasi awal sebelum request diproses.
3	<i>Handler</i>	Menerima request, memvalidasi data masukan, dan meneruskan proses ke layer service.
4	<i>Service</i>	Mengimplementasikan logika bisnis serta mengoordinasikan proses utama sistem.
5	<i>Repository</i>	Mengelola proses penyimpanan, pengambilan, dan pembaruan data pada MongoDB.
6	<i>Websocket</i>	Mengelola koneksi serta mendistribusikan pembaruan data kepada client secara <i>real-time</i> .

Berdasarkan arsitektur dan struktur proyek yang telah dikembangkan, backend sistem manajemen tugas berhasil dibangun menggunakan pendekatan *Clean Architecture* dengan pemisahan tanggung jawab pada setiap lapisan. Infrastruktur tersebut menjadi dasar bagi pengembangan modul-modul sistem yang dibahas pada subbab berikutnya.

b) Pengembangan Modul *Workspace*

Workspace Multi-Tenant diimplementasikan sebagai representasi tenant yang menjadi dasar pengelompokan data operasional pada *Bharata Task Manager*. Setiap *workspace* memiliki identitas, pemilik (*owner*), konfigurasi *workflow*, daftar divisi, serta skala prioritas yang menjadi konteks utama dalam pengelolaan tugas. Dengan pendekatan tersebut, seluruh aktivitas sistem dijalankan di dalam ruang lingkup *workspace* sehingga mendukung pemisahan data dan kolaborasi antar.

1) Implementasi Pembentukan *Workspace*



Gambar 4. 2 Implementasi Pembentukan *Workspace*

Gambar 4.2 menunjukkan implementasi logika pembuatan *workspace* baru pada fungsi *Create()* berkas *space_service.go*. Proses inisialisasi ini mencakup tiga bagian utama :

- Validasi owner ID — memverifikasi format pengenalan unik pemilik *workspace* sebelum disimpan.
- Pembentukan tahap status/*workflow* — menyusun tahapan alur kerja tugas default (dari *Request* hingga *Archive*) jika tidak diatur manual.
- Pembentukan skala prioritas — menetapkan tingkat urgensi tugas default (*High*, *Medium*, *Low*) beserta kode warna penandanya.

Ketiga tahapan ini memastikan setiap *workspace* baru memiliki konfigurasi dasar sehingga dapat langsung digunakan sebagai ruang kerja operasional

2) Implementasi Model Data *Workspace*

```
package domain

import (
    "context"
    "time"
    "go.mongodb.org/mongo-driver/bson/primitive"
)

// --- Structs (Entities) ---

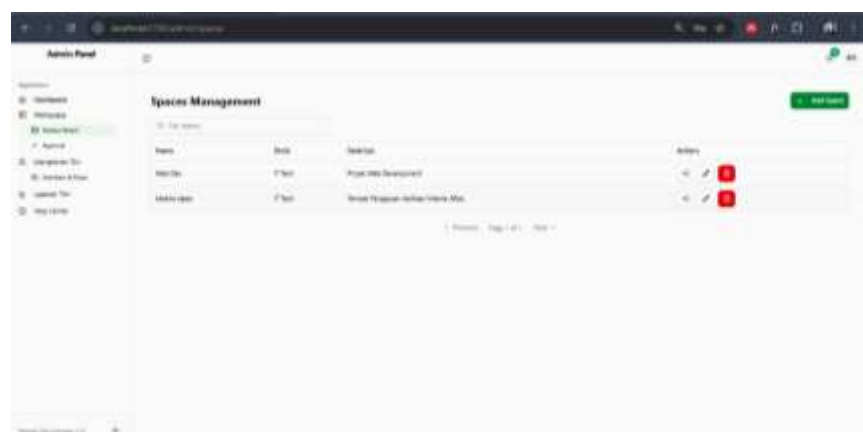
type Space struct {
    ID          primitive.ObjectID `bson:"_id,omitempty" json:"id"`
    Name        string              `bson:"name" json:"name"`
    Divisions   []Division          `bson:"divisions" json:"divisions"`
    OwnerID     primitive.ObjectID `bson:"owner_id,omitempty" json:"owner_id,omitempty"`
    Description string              `bson:"description" json:"description"`
    Status      []Stage             `bson:"status" json:"status"`
    Priority     []Priority           `bson:"priority" json:"priority"`
    CreatedAt   time.Time           `bson:"createdAt" json:"createdAt"`
    UpdatedAt   time.Time           `bson:"updatedAt" json:"updatedAt"`
}

type Division struct {
    ID      string `bson:"id" json:"id" // e.g. "div_1"`
    Name    string `bson:"name" json:"name" // e.g. "Marketing"`
    Color   string `bson:"color" json:"color" // Hex color`
}
```

Gambar 4. 3 Implementasi Model Data *Workspace*

Gambar 4.3 menunjukkan implementasi model data entitas ruang kerja pada berkas *space.go*. Struktur ini terdiri atas *Space struct* sebagai cetak biru utama *workspace* yang menyimpan data ID, nama, pemilik, status *workflow*, dan prioritas tugas, serta *Division struct* yang mengaitkan *workspace* dengan divisi tertentu untuk kebutuhan isolasi data operasional

3) Tampilan Halaman *Workspace*



Gambar 4. 4 Tampilan Halaman *Workspace*

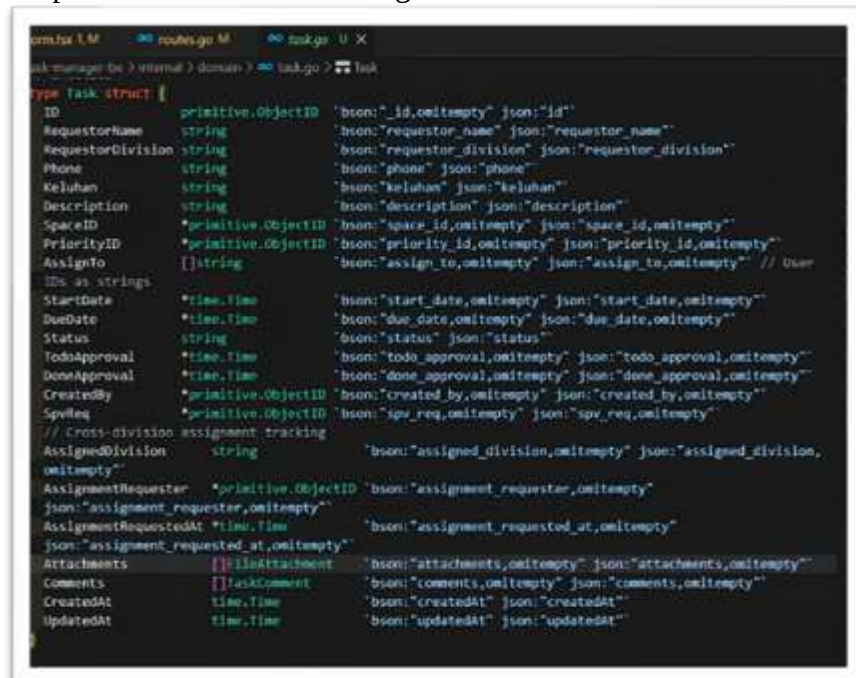
Gambar 4.4 menunjukkan antarmuka pengelolaan *workspace* sebagai hasil implementasi modul *Workspace*. Halaman ini

menampilkan daftar *workspace* beserta informasi nama, divisi, dan deskripsi, serta menyediakan fungsi penambahan, pencarian, masuk ke *workspace*, perubahan data, dan penghapusan *workspace*. Seluruh informasi yang ditampilkan berasal dari layanan backend yang telah diimplementasikan pada modul *Workspace*.

c) Implementasi Mekanisme Pengelolaan Tugas

Modul pengelolaan tugas dikembangkan sebagai inti proses bisnis pada Bharata Task Manager yang mendukung pencatatan, pendistribusian, pemantauan, dan penyelesaian tugas dalam setiap *workspace*. Setiap tugas dibuat berdasarkan konteks *workspace* sehingga seluruh aktivitas pengelolaan tugas dilakukan dalam ruang lingkup tenant yang sesuai.

1) Implementasi Model Data Tugas



Gambar 4. 5 Implementasi Model Data Task

Gambar 4.5 menunjukkan implementasi entitas Task sebagai representasi utama data tugas pada *backend*. Entitas ini menyimpan informasi seperti identitas tugas, *workspace*, pemohon, divisi tujuan, *assignee*, dan status *workflow* yang digunakan sebagai konteks pada mekanisme *Dynamic Context-Aware RBAC*.

2) Implementasi Mekanisme *Workflow* Tugas



Gambar 4. 6 Implementasi Mekanisme *Workflow* Tugas

Gambar 4.6 menunjukkan implementasi logika transisi status tugas (*workflow*) pada berkas `task_service.go`. Mekanisme ini divalidasi melalui tiga aturan utama :

- Pemetaan alur transisi linear yaitu mendefinisikan urutan perpindahan status tugas yang valid secara bertahap dan satu arah.
- Validasi alur untuk memblokir permintaan perubahan status apabila terdeteksi melompat atau mundur dari urutan FSM (*Finite State Machine*).
- Batasan hak akses dalam mengunci otorisasi perubahan ke status akhir (*Done* dan *Archive*) agar hanya dapat dilakukan oleh peran supervisor atau administrator.

3) Implementasi Mekanisme Penugasan Antar Divisi



Gambar 4. 7 Model Data Tugas

Gambar 4.7 menunjukkan implementasi mekanisme penugasan antar divisi pada *backend* yang terdiri atas proses *routing* tugas dan pembatasan akses setelah tugas dialihkan.

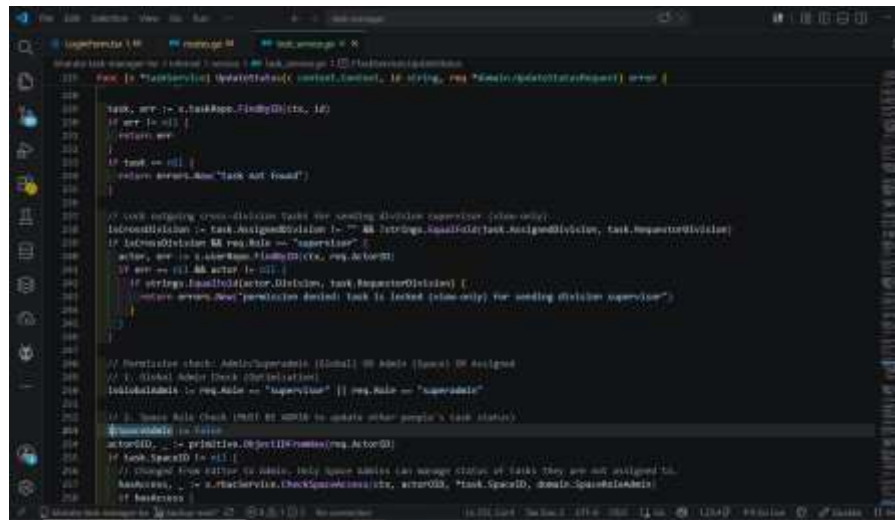
i. Logika Routing Penugasan Antar divisi

Backend memeriksa apakah divisi tujuan berbeda dengan divisi pengaju. Jika berbeda, sistem mencari supervisor pada divisi tujuan dan menetapkannya sebagai penerima tugas dengan status *PendingSupervisorApproval*. Apabila supervisor tidak ditemukan, tugas dialihkan dengan status *UnassignedCrossDivision*.

ii. Isolasi Akses Divisi Pengirim

Setelah tugas dialihkan, *backend* membatasi hak akses divisi pengirim menjadi *view-only*. Setiap permintaan perubahan terhadap tugas akan ditolak sehingga hanya divisi tujuan yang dapat melanjutkan proses penyelesaian tugas.

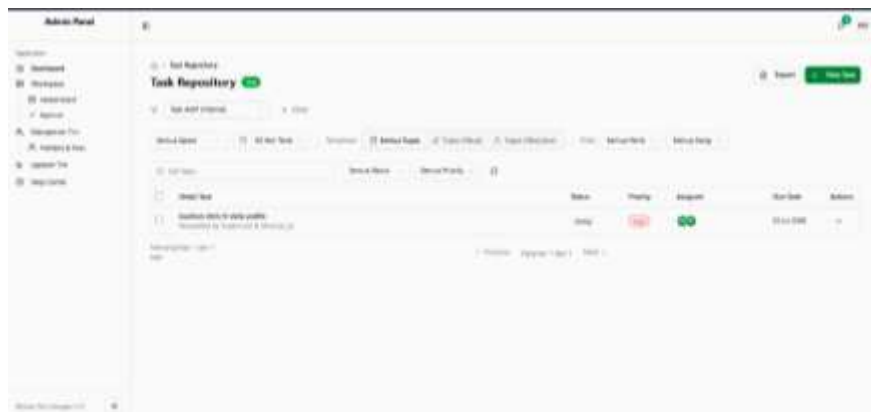
4) Implementasi Logika Bisnis Pengelolaan Tugas



Gambar 4. 8 Implementasi Logika Bisnis Pengelolaan Tugas

Gambar 4.8 menunjukkan implementasi logika bisnis pada modul pengelolaan tugas. *Backend* menangani proses pembuatan, perubahan, pembaruan status, serta penyelesaian tugas melalui layer *service* yang berfungsi mengoordinasikan validasi data, pemrosesan aturan bisnis, dan penyimpanan informasi ke basis data. Pendekatan tersebut memastikan setiap perubahan data dilakukan secara konsisten sesuai proses bisnis yang diterapkan..

5) Tampilan Modul Pengelolaan Tugas



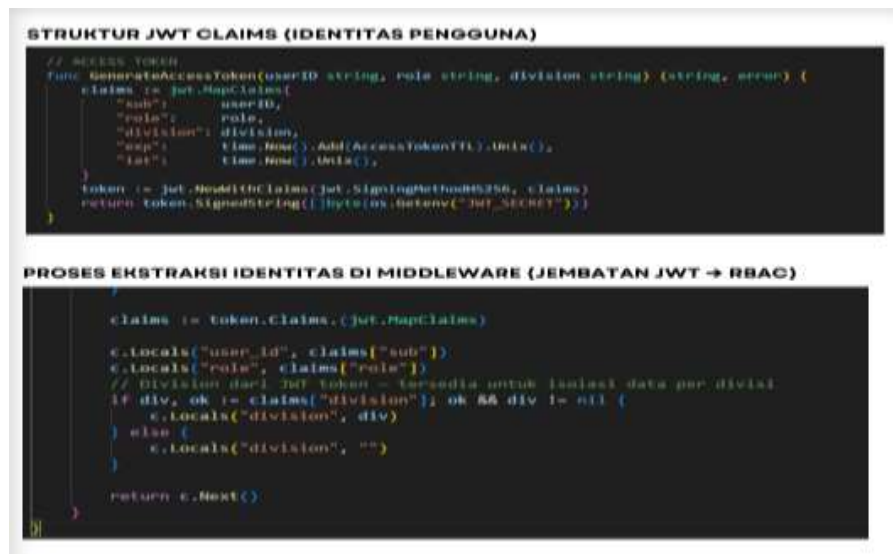
Gambar 4. 9 Tampilan Pengelolaan Tugas

Gambar 4.9 menunjukkan hasil implementasi modul pengelolaan tugas pada *Bharata Task Manager*. Halaman *Task Repository* menampilkan daftar tugas beserta informasi status, prioritas, penanggung jawab, dan tenggat waktu. Selain itu, halaman ini menyediakan fungsi pencarian, penyaringan data, ekspor, serta pembuatan tugas baru yang memanfaatkan layanan *backend* pada modul pengelolaan tugas.

d) Pengembangan Mekanisme Keamanan

Modul keamanan dikembangkan untuk memastikan setiap layanan backend hanya dapat diakses oleh pengguna yang memiliki identitas dan hak akses yang sesuai. Mekanisme keamanan pada Bharata Task Manager terdiri atas *Authentication* menggunakan *JSON Web Token* (JWT) dan *Dynamic Context-Aware Role-Based Access Control* (RBAC) sebagai mekanisme otorisasi berdasarkan konteks *workspace*, kepemilikan data, dan kondisi sumber daya.

1) Implementasi *Authentication*



Gambar 4. 10 Implementasi *Authentication*

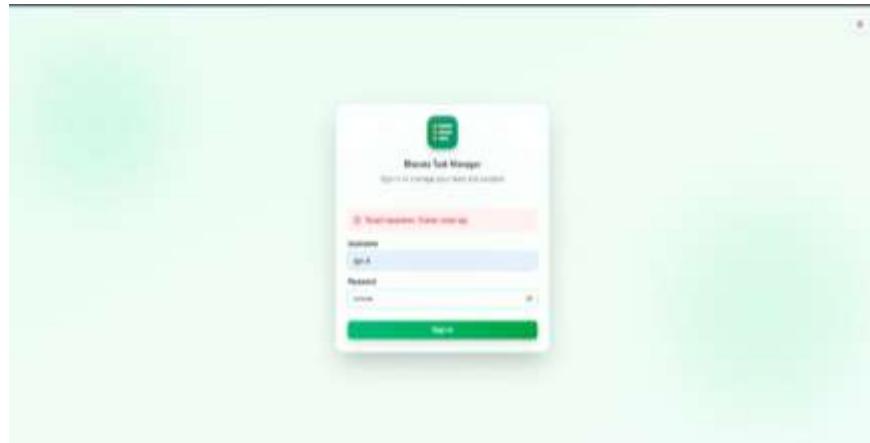
Gambar 4.10 menunjukkan implementasi pembentukan identitas pengguna menggunakan JWT. Fungsi *GenerateAccessToken()* menghasilkan *access token* yang memuat klaim *sub*, *role*, *division*, *exp*, dan *iat*, sedangkan *middleware RequireAuth()* mengekstraksi informasi tersebut ke dalam konteks *request* untuk digunakan pada proses otorisasi. Dengan mekanisme tersebut, setiap permintaan yang diterima backend telah memiliki identitas pengguna yang tervalidasi.

Tabel 4. 2 Hierarki Role Sistem

NO	Peran	Implementasi
1	Superadmin	Diinisialisasi oleh sistem dan bertanggung jawab mengelola akun Supervisor serta seluruh sumber daya sistem.
2	Supervisor	Dibuat oleh Superadmin dan bertugas mengelola akun Staff sesuai divisinya serta mengawasi pengelolaan tugas.

NO	Peran	Implementasi
3	Staff	Dibuat oleh Supervisor untuk menjalankan aktivitas operasional sesuai hak akses yang diberikan.

Berdasarkan hierarki tersebut, setiap pengguna memiliki identitas berupa *user_id*, *role*, dan *division* yang digunakan sebagai konteks awal pada proses otorisasi.



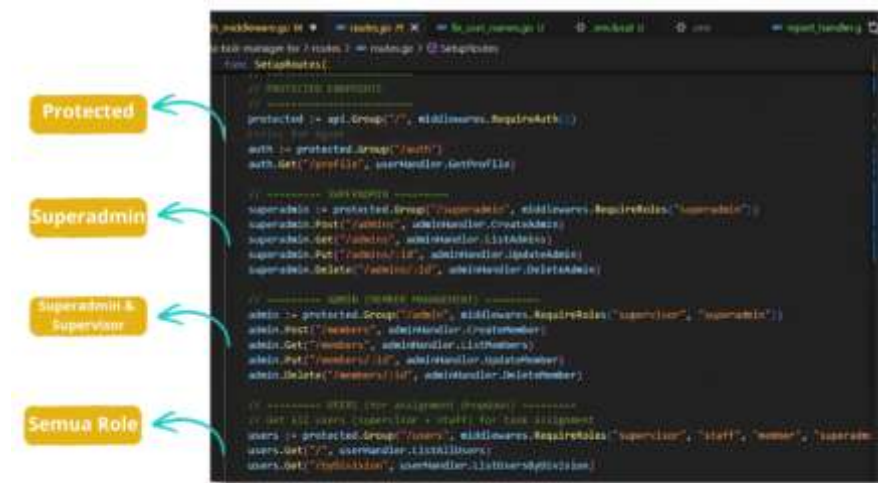
Gambar 4. 11 Tampilan Halaman *Authentication*

Gambar 4.12 menunjukkan hasil implementasi mekanisme *authentication* pada Bharata Task Manager. Ketika pengguna memasukkan kombinasi *username* atau *password* yang tidak valid, backend menolak proses autentikasi dan mengembalikan respons kegagalan yang ditampilkan dalam bentuk pesan kesalahan pada halaman login. Mekanisme ini memastikan hanya pengguna yang memiliki kredensial yang valid yang dapat memperoleh *access token* dan mengakses layanan *backend*.

2) Implementasi *Dynamic Context-Aware Role-Based Access Control* (RBAC)

Dynamic Context-Aware Role-Based Control (RBAC) merupakan artefak utama yang dikembangkan pada penelitian. Mekanisme ini diimplementasikan pada lapisan *backend* sebagai pusat pengambilan keputusan otorisasi dengan mengombinasikan validasi peran pengguna (*role*) dan konteks operasional sistem. Berbeda dengan mekanisme RBAC konvensional yang hanya memanfaatkan informasi peran pengguna, implementasi pada penelitian ini juga mempertimbangkan hubungan pengguna terhadap sumber daya yang diakses. Kondisi alur kerja (*workflow*), serta batas ruang kerja (*tenant boundary*) sebelum keputusan otorisasi diberikan.

i. Implementasi Otorisasi Tingkat Route (*Route Level Authorization*)



Gambar 4. 12 Implementasi Route Level Authorization

Gambar 4.12 menunjukkan konfigurasi tingkat rule pada berkas `routes.go`. Setiap kelompok *endpoint* dilindungi oleh *middleware RequireAuth()* untuk validasi JWT dan *RequireRoles()* untuk pemeriksaan peran. *Endpoint /superadmin* hanya dapat diakses oleh superadmin, *endpoint /admin* oleh supervisor dan superadmin, sedangkan *endpoint /users* dapat diakses oleh seluruh peran. Dengan konfigurasi ini, setiap permintaan dipastikan telah tervalidasi identitas dan perannya sebelum diteruskan ke lapisan berikutnya.

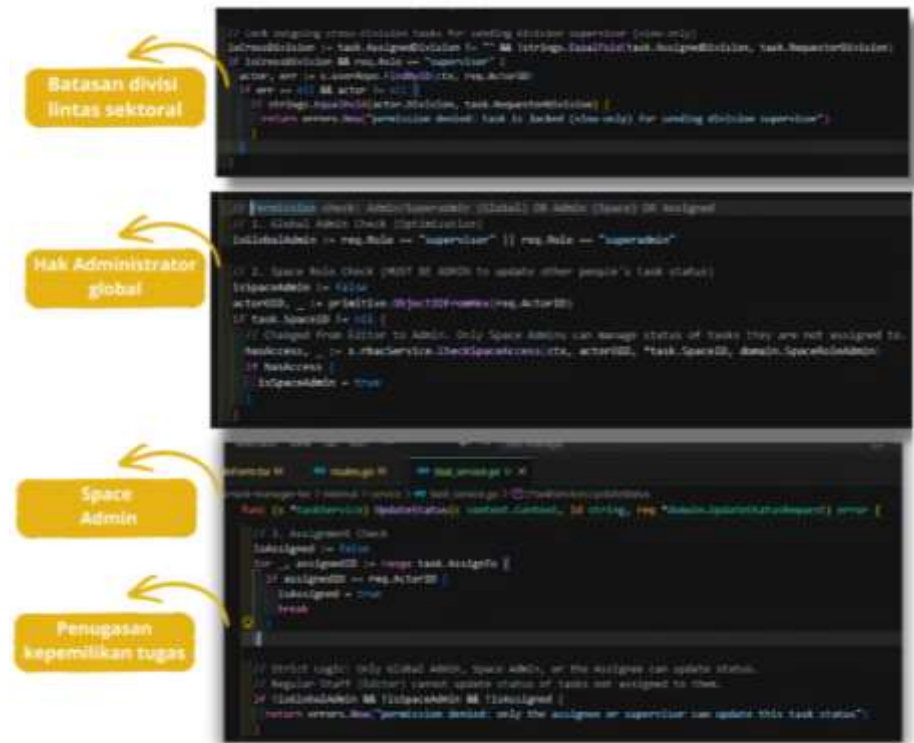
ii. Implementasi Otorisasi Tingkat *Workspace* (*Space Level Authorization*)



Gambar 4. 13 *Space Level Authorization*

Gambar 4.13 menunjukkan implementasi Sistem akan menjalankan tiga aturan *fallback* secara berurutan: (1) memberikan akses penuh jika pengguna adalah pembuat *workspace* (*owner*), (2) memberikan akses penuh bagi pengguna dengan peran global superadmin dan (3) menyesuaikan peran berdasarkan divisi pengguna terhadap divisi *workspace* terkait—di mana supervisor divisi otomatis mendapat hak akses Admin dan staff divisi mendapat hak akses Editor

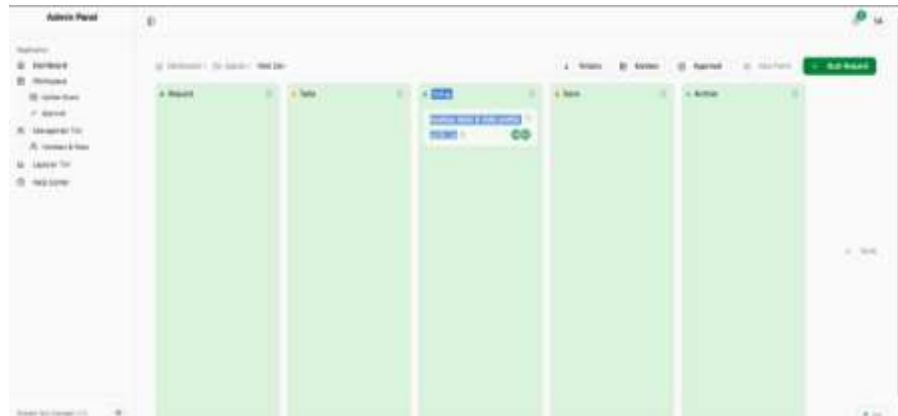
iii. Implementasi Otorisasi Tingkat Operasi (*Operation Level Authorization*)



Gambar 4. 14 *Operation Level Authorization*

Gambar 4.14 menunjukkan implementasi otorisasi tingkat operasi pada berkas `task_service.go` di dalam fungsi `UpdateStatus`. Proses otorisasi dievaluasi berdasarkan empat kriteria utama:

- Batasan divisi lintas sektoral — mengunci tugas agar bersifat *view-only* bagi supervisor dari divisi pengirim apabila tugas didelegasikan ke divisi lain.
- Hak Administrator global — mendeteksi pengguna dengan peran global supervisor atau superadmin.
- Space Admin — memvalidasi apakah pengguna memiliki hak akses sebagai administrator pada ruang kerja (*space*) terkait
- Penugasan kepemilikan tugas — memeriksa apakah pengguna terdaftar sebagai penerima tugas (*assignee*).



Gambar 4. 15 Implementasi Isolasi Akses Pengguna Antar Divisi

Gambar 4.15 menunjukkan implementasi *Dynamic Context-Aware RBAC* pada halaman Kanban. Tugas yang telah dialihkan ke divisi lain hanya dapat dilihat (*view-only*) oleh divisi pengirim, sedangkan perubahan status dan perpindahan tugas hanya dapat dilakukan oleh pengguna pada divisi tujuan sesuai hak akses yang dimiliki.

Berdasarkan implementasi tersebut, mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) dikembangkan melalui tiga lapisan otorisasi yang saling terintegrasi, yaitu *Route-Level Authorization*, *Space-Level Authorization*, dan *Operation-Level Authorization*. Pendekatan tersebut memungkinkan proses otorisasi tidak hanya mempertimbangkan peran pengguna, tetapi juga mengintegrasikan konteks ruang kerja dan logika bisnis sebelum suatu operasi dijalankan. Implementasi inilah yang menjadi dasar bagi proses evaluasi untuk membuktikan bahwa mekanisme tersebut mampu mengelola otorisasi dan menjaga isolasi data sesuai kebutuhan sistem manajemen tugas berbasis *multi-tenant*.

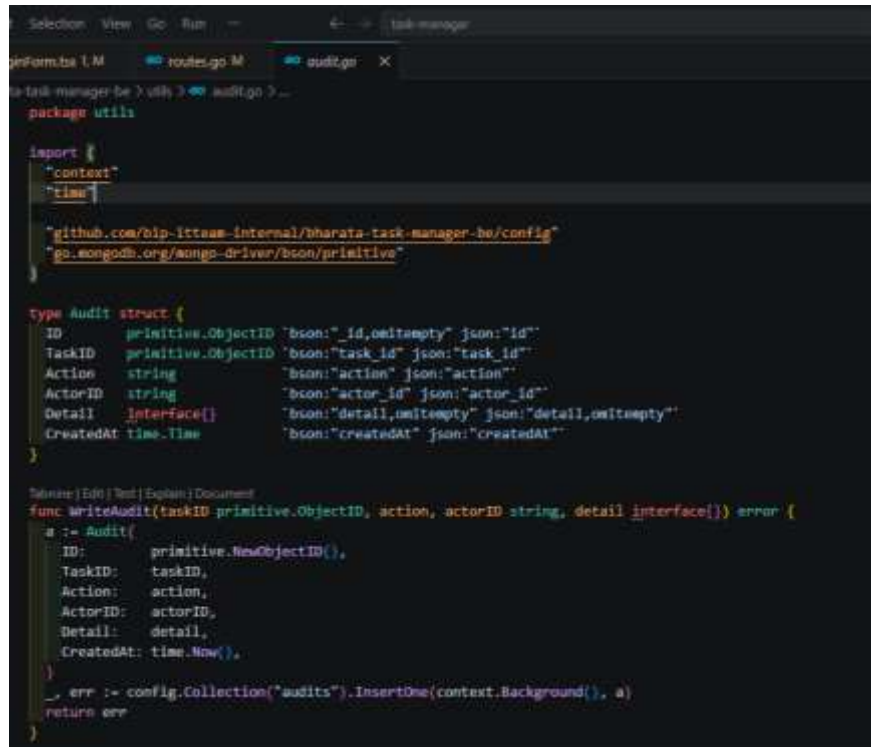
e) Pengembangan Modul Pendukung

Modul pendukung dikembangkan untuk mendukung proses operasional sistem melalui pencatatan aktivitas pengguna dan penyampaian informasi secara *real-time*. Pada Bharata Task Manager, modul pendukung terdiri atas *Audit Trail* sebagai mekanisme pencatatan aktivitas serta *Notifikasi Real-Time* untuk menyampaikan perubahan informasi kepada pengguna tanpa memerlukan proses penyegaran (*refresh*) halaman.

1) Implementasi *Audit Trail*

Audit Trail diimplementasikan sebagai mekanisme *traceability* pada *backend* untuk mencatat setiap perubahan penting yang terjadi pada data tugas. Pencatatan dilakukan setelah proses bisnis berhasil

dijalankan sehingga setiap aktivitas yang tersimpan merepresentasikan perubahan yang valid pada sistem



```
Selection View Go Run task-manager
perForm.ta L.M routes.go M audit.go X
task-manager-be > cd > go run audit.go > ...
package util

import {
    "context"
    "time"
    "github.com/bip-ittas-internal/bharata-task-manager-be/config"
    "go.mongodb.org/mongo-driver/bson/primitive"
}

type Audit struct {
    ID          primitive.ObjectID `bson:"_id,omitempty" json:"id"`
    TaskID      primitive.ObjectID `bson:"task_id" json:"task_id"`
    Action      string              `bson:"action" json:"action"`
    ActorID     string              `bson:"actor_id" json:"actor_id"`
    Detail      interface{}          `bson:"detail,omitempty" json:"detail,omitempty"`
    CreatedAt   time.Time           `bson:"createdAt" json:"createdAt"`
}

func WriteAudit(taskID primitive.ObjectID, action, actorID string, detail interface{}) error {
    a := Audit{
        ID:          primitive.NewObjectID(),
        TaskID:      taskID,
        Action:      action,
        ActorID:     actorID,
        Detail:      detail,
        CreatedAt:   time.Now(),
    }
    err := config.Collection("audits").InsertOne(context.Background(), a)
    return err
}
```

Gambar 4. 16 Implementasi *Audit Trail*

Gambar 4.16 menunjukkan implementasi *Audit* pada berkas *audit.go*. *Backend* membentuk objek audit dan menyimpannya ke koleksi *audits* melalui fungsi *WriteAudit()*.



Gambar 4. 17 *Audit Trail*

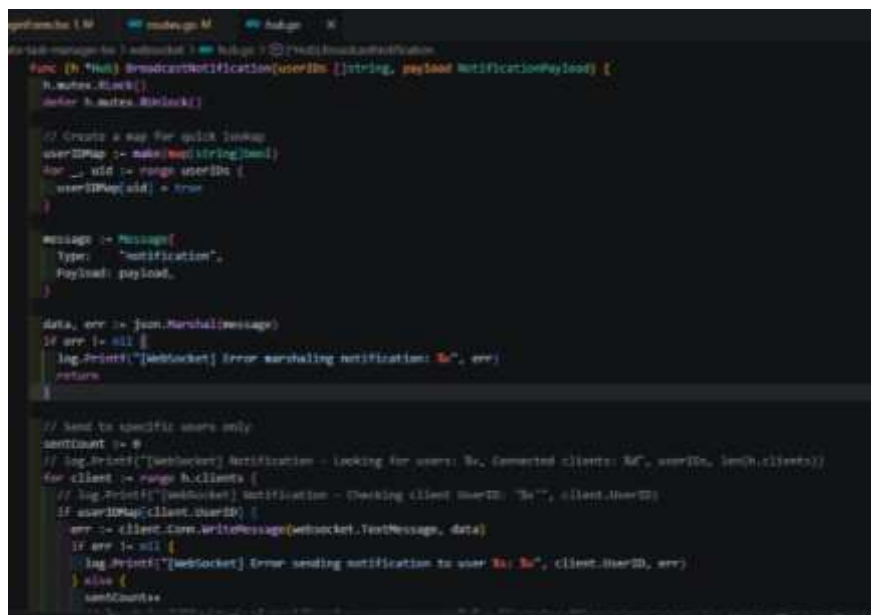
Gambar 4.17 menampilkan implementasi antarmuka *Audit Trail* pada halaman detail tugas di papan Kanban (*Workspace*). Informasi yang ditampilkan meliputi:

- Jumlah Log, yang menunjukkan banyaknya riwayat aktivitas yang tercatat pada suatu tugas.
- Identitas Aktor dan Waktu, yang menampilkan nama pengguna beserta waktu pelaksanaan aktivitas.

- c) Jenis Aktivitas, yang ditampilkan dalam bentuk *badge* untuk menunjukkan operasi yang dilakukan, seperti Persetujuan (*Approved*).
 - d) Peran Pengguna, yang menampilkan peran pengguna, seperti Supervisor, sebagai informasi pihak yang melakukan aktivitas.
- Berdasarkan implementasi tersebut, setiap aktivitas penting yang dicatat melalui fungsi *WriteAudit()* pada *backend* dapat ditampilkan kembali pada antarmuka secara kronologis.

2) Implementasi Notifikasi *Real-Time*

Sistem notifikasi diimplementasikan sebagai mekanisme penyampaian informasi setelah proses bisnis pada *backend* berhasil dijalankan. Implementasi menggunakan pendekatan *Hybrid Dual-Channel*, yaitu menyimpan notifikasi ke basis data sekaligus mengirimkan informasi secara *real-time* melalui *websocket* sehingga pengguna memperoleh pembaruan tanpa melakukan pemuatan ulang halaman.



```

func (h *Hub) BroadcastNotification(userID string, payload NotificationPayload) {
    h.mu.Lock()
    defer h.mu.Unlock()

    // Create a map for quick lookup
    userIDMap := make(map[string]bool)
    for _, uid := range userID {
        userIDMap[uid] = true
    }

    message := Message{
        Type: "notification",
        Payload: payload,
    }

    data, err := json.Marshal(message)
    if err != nil {
        log.Printf("[websocket] Error marshaling notification: %v", err)
        return
    }

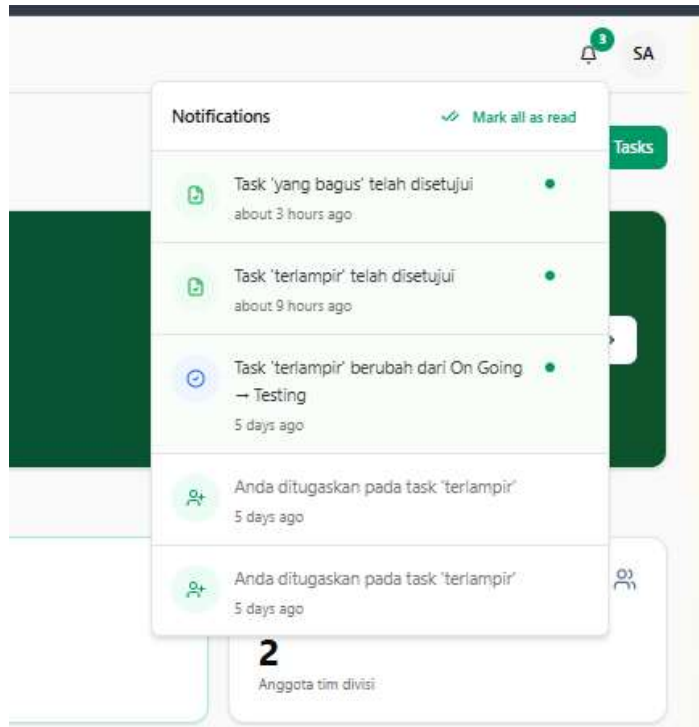
    // Send to specific users only
    notification := 0
    // log.Printf("[websocket] Notification - Looking for users: %v, Connected clients: %v, userID: %v", userID, len(h.clients))
    for client := range h.clients {
        // log.Printf("[websocket] Notification - Checking client userID: %v", client.userID)
        if userIDMap[client.userID] {
            err := client.Conn.WriteMessage(websocket.TextMessage, data)
            if err != nil {
                log.Printf("[websocket] Error sending notification to user %v: %v", client.userID, err)
            } else {
                notification++
            }
        }
    }
}

```

Gambar 4. 18 Implementasi *Websocket Service*

Gambar 4.18 menunjukkan implementasi mekanisme pengiriman notifikasi pada *WebSocket Hub* melalui fungsi *BroadcastNotification()*. Backend membentuk *payload* notifikasi, mengubahnya ke format JSON, kemudian mengirimkan notifikasi kepada pengguna yang dituju berdasarkan identitas *user* yang terhubung melalui koneksi *WebSocke* secara *real time*.

3) Tampilan Notifikasi



Gambar 4. 19 Implementasi Notifikasi

Gambar 4.19 menunjukkan antarmuka notifikasi real-time berbasis in-app popover. Fitur ini berfungsi memberikan informasi pembaruan tugas secara instan kepada pengguna terkait.

2. Hasil Pengujian Mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC)

Setelah artefak *backend* berhasil dikembangkan, tahap selanjutnya adalah melakukan evaluasi terhadap mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) menggunakan metode *Black Box Testing*. Evaluasi dilakukan untuk memverifikasi bahwa setiap mekanisme otorisasi yang dikembangkan telah berjalan sesuai kebutuhan sistem dengan mempertimbangkan identitas pengguna, batas ruang kerja (*tenant boundary*), kepemilikan sumber daya (*ownership*), dan kondisi sumber daya (*resource state*). Seluruh pengujian dilakukan pada *backend* yang telah diimplementasikan di PT Bharata Internasional Pharmaceutical menggunakan data operasional sistem.

a) Pengujian *Authentication*

Pengujian *Authentication* dilakukan untuk memverifikasi kemampuan sistem dalam melakukan autentikasi berdasarkan kredensial pengguna. Pengujian terdiri atas dua skenario, yaitu login menggunakan akun valid dan login menggunakan kredensial yang tidak valid..

Tabel 4. 3 Hasil Pengujian *Authentication*

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-AUTH-01	Login dengan akun valid	Sistem berhasil melakukan autentikasi, pengguna ke dashboard sesuai hak aksesnya.	Sistem berhasil melakukan autentikasi, dan mengarahkan pengguna ke dashboard sesuai hak aksesnya.	Valid
TC-AUTH-02	Login menggunakan akun tidak valid	Sistem menolak autentikasi.	Sistem menolak autentikasi.	Valid

Berdasarkan hasil pengujian pada Tabel 4.3, TC-AUTH-01 dan TC-AUTH-02 menghasilkan keluaran yang sesuai dengan hasil yang diharapkan. Dengan demikian, kedua skenario *Authentication* dinyatakan Valid.

Hasil pengujian menunjukkan bahwa mekanisme *Authentication* mampu membedakan kredensial yang valid dan tidak valid. Kredensial yang valid dapat digunakan untuk memperoleh akses ke sistem, sedangkan kredensial yang tidak valid ditolak oleh sistem.

Dokumentasi lengkap pelaksanaan pengujian TC-AUTH-01 dan TC-AUTH-02 disajikan pada Lampiran.

b) Pengujian *Tenant Boundary*

Pengujian *tenant boundary* dilakukan untuk memverifikasi bahwa backend membatasi akses data pengguna berdasarkan konteks workspace dan hubungan operasional terhadap suatu tugas. Pengujian menggunakan data uji yang disusun sesuai struktur organisasi dan alur proses bisnis PT Bharata Internasional Pharmaceutical.

Tabel 4. 4 Hasil Pengujian *Tenant Boundary*

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-TB-01	Supervisor mengakses <i>incoming task</i> dari divisi lain	Sistem mengizinkan akses ke <i>workspace</i>	Supervisor dapat mengakses <i>workspace</i> terkait dan <i>incoming task</i>	Valid

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
	dengan relasi operasional yang sah.	terkait dan hanya menampilkan <i>incoming task</i> yang diterima.	yang diterima sesuai hubungan operasional.	
TC-TB-02	Staff mengakses <i>incoming task</i> sesuai <i>assignment</i> .	Sistem hanya menampilkan dan mengizinkan pengelolaan <i>task</i> yang sesuai dengan <i>assignment</i> dan <i>tenant boundary</i> .	Staff hanya dapat mengakses dan mengelola <i>incoming task</i> yang ditugaskan kepadanya sesuai <i>assignment</i> dan <i>tenant boundary</i> .	Valid

c) Pengujian *Ownership*

Pengujian *ownership* dilakukan untuk memverifikasi bahwa backend hanya mengizinkan operasi terhadap tugas yang memiliki hubungan operasional dengan pengguna. Pengujian menggunakan data uji yang disusun sesuai struktur organisasi dan alur proses bisnis PT Bharata Internasional Pharmaceutical

Tabel 4. 5 Hasil Pengujian *Ownership*

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-OWN-01	Staff mengubah tugas yang menjadi tanggung jawabnya.	Sistem mengizinkan perubahan data dan mengembalikan <i>HTTP 200 OK</i> .	Staff dapat mengubah tugas yang menjadi tanggung jawabnya dan perubahan berhasil diproses.	Valid
TC-OWN-02	Staff mengubah tugas yang bukan menjadi	Sistem menolak perubahan, dan tidak	Perubahan terhadap tugas yang bukan menjadi tanggung	Valid

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
	tanggung jawabnya.	melakukan perubahan data.	jawab Staff ditolak dan data tidak berubah.	
TC-OWN-03	Supervisor pengaju mengubah <i>outgoing task</i> yang telah diteruskan ke divisi tujuan.	Sistem menolak perubahan dan memberikan hak akses <i>view-only</i> terhadap tugas tersebut.	<i>Outgoing task</i> yang telah diteruskan tidak dapat diubah oleh Supervisor pengaju dan hanya dapat dilihat (<i>view-only</i>).	Valid

d) Pengujian *Resource State*

Pengujian *Resource State* dilakukan untuk memverifikasi bahwa *backend* memvalidasi setiap perubahan status tugas berdasarkan kondisi (*state*) dan aturan transisi *workflow* yang diterapkan. Pengujian terdiri atas lima skenario.

Tabel 4. 6 Pengujian *Resource State*

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-RS-01	Staff mengubah status tugas sesuai urutan <i>workflow</i> (<i>Todo</i> → <i>Doing</i> → <i>Testing</i>).	<i>Backend</i> mengizinkan perubahan status sesuai aturan <i>workflow</i> .	<i>Backend</i> mengizinkan perubahan status dari <i>Todo</i> → <i>On Going</i> → <i>Testing</i> sesuai aturan <i>workflow</i> .	Valid
TC-RS-02	Staff mencoba mengubah status tugas dengan melompati tahapan <i>workflow</i> (<i>Todo</i> → <i>Done</i> atau <i>Testing</i> → <i>Todo</i>).	<i>Backend</i> menolak perubahan karena tidak sesuai dengan aturan transisi <i>workflow</i> .	<i>Backend</i> menolak perubahan status yang melompati atau membalik tahapan <i>workflow</i> .	Valid

No	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-RS-03	Pengguna mencoba mengubah status tugas yang telah berada pada status <i>Archive</i> .	<i>Backend</i> menolak perubahan karena tugas telah berada pada status akhir (<i>read-only</i>).	<i>Backend</i> menolak perubahan terhadap tugas yang telah berstatus <i>Archive</i> .	Valid
TC-RS-04	Staff mencoba mengubah status tugas dari <i>Testing</i> → <i>Done</i> atau <i>Done</i> → <i>Archive</i> .	<i>Backend</i> menolak perubahan karena transisi tersebut hanya dapat dilakukan oleh Supervisor sesuai aturan <i>workflow</i> .	<i>Backend</i> menolak perubahan status yang dilakukan oleh Staff karena tidak memiliki kewenangan terhadap transisi tersebut.	Valid
TC-RS-05	Supervisor mengubah status tugas dari <i>Testing</i> → <i>Done</i> atau <i>Done</i> → <i>Archive</i> sesuai kewenangannya.	<i>Backend</i> mengizinkan perubahan status sesuai aturan <i>workflow</i> yang berlaku.	<i>Backend</i> mengizinkan Supervisor melakukan transisi status sesuai kewenangan dan aturan <i>workflow</i> .	Valid

e) Pengujian *Audit Trail*

Pengujian *Audit Trail* dilakukan untuk memverifikasi bahwa setiap aktivitas penting pada sistem dicatat secara otomatis sebagai riwayat aktivitas

Tabel 4. 7 Pengujian *Audit Trail*

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-AT-01	Pencatatan perubahan status tugas	Sistem mencatat aktivitas perubahan status pada <i>audit trail</i> secara otomatis.	Aktivitas perubahan status tugas tercatat pada <i>audit trail</i> secara otomatis.	Valid

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-AT-02	Pencatatan penugasan tugas	Sistem mencatat aktivitas penugasan pada <i>audit trail</i> .	Aktivitas penugasan tugas tercatat pada <i>audit trail</i> .	Valid

f) Pengujian *Real-Time Notification*

Pengujian *Real-Time Notification* dilakukan untuk memverifikasi bahwa sistem dapat menyampaikan informasi perubahan data kepada pengguna terkait secara *real-time* melalui *WebSocket*.

Tabel 4. 8 Pengujian *Real-Time Notification*

ID Test Case	Skenario Pengujian	Hasil yang Diharapkan	Hasil Aktual	Status
TC-RT-01	Notifikasi pembuatan tugas	Sistem mengirim notifikasi <i>real-time</i> kepada pengguna terkait melalui <i>WebSocket</i> .	Notifikasi pembuatan tugas diterima oleh pengguna terkait secara <i>real-time</i> melalui <i>WebSocket</i> .	Valid
TC-RT-02	Notifikasi perubahan status tugas	Sistem mengirim notifikasi perubahan status secara <i>real-time</i> kepada pengguna terkait.	Notifikasi perubahan status tugas diterima oleh pengguna terkait secara <i>real-time</i> .	Valid

g) Rekapitulasi Hasil Pengujian *Black Box*

Setelah seluruh skenario pengujian dilaksanakan menggunakan metode *Black Box Testing*, dilakukan rekapitulasi untuk memberikan gambaran keseluruhan hasil pengujian pada setiap mekanisme yang dikembangkan. Rekapitulasi didasarkan pada kesesuaian antara hasil aktual dan hasil yang diharapkan dari setiap *test case*.

Tabel 4. 9 Rekapitulasi Hasil Pengujian

No	Mekanisme Pengujian	Jumlah Skenario	Valid	Tidak Valid	Persentase Validitas
1	<i>Authentication</i>	2	2	0	100%
2	<i>Tenant Boundary</i>	2	2	0	100%
3	<i>Ownership</i>	3	3	0	100%
4	<i>Resource State</i>	5	5	0	100%
5	<i>Audit Trail</i>	2	2	0	100%
6	<i>Real-Time Notification</i>	2	2	0	100%
Total		16	16	0	100%

Berdasarkan Tabel 4.7, terdapat 16 skenario pengujian yang mencakup *Authentication*, *Tenant Boundary*, *Ownership*, *Resource State*, *Audit Trail*, dan *Real-Time Notification*. Seluruh skenario menghasilkan status valid karena hasil aktual sesuai dengan hasil yang diharapkan. Dengan demikian, diperoleh 16 skenario valid dari 16 skenario yang diuji atau sebesar 100%, tanpa terdapat skenario dengan status Tidak Valid.

Hasil rekapitulasi mengonfirmasi bahwa seluruh *test case* yang telah direncanakan pada tahap pengujian dapat ditelusuri ke hasil pelaksanaan pengujian dan menghasilkan status Valid. Rekapitulasi tersebut mencakup seluruh mekanisme yang diuji dan memberikan gambaran keseluruhan terhadap hasil *Black Box Testing*.

h) Sintesis Hasil Evaluasi

Berdasarkan hasil evaluasi menggunakan metode *Black Box Testing*, seluruh skenario pengujian pada mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) menghasilkan keluaran yang sesuai dengan hasil yang diharapkan. Mekanisme *Authentication* memverifikasi identitas pengguna sebelum proses otorisasi dilakukan, *Tenant Boundary* membatasi akses berdasarkan hubungan operasional antar-*workspace*, *Ownership* mengendalikan hak operasi berdasarkan hubungan pengguna terhadap tugas, sedangkan *Resource State* memvalidasi perubahan status berdasarkan kondisi tugas dan aturan *workflow* yang berlaku.

Secara keseluruhan, hasil evaluasi menunjukkan bahwa keputusan otorisasi pada skenario yang diuji tidak hanya mempertimbangkan *role* pengguna, tetapi juga konteks akses yang meliputi identitas pengguna, hubungan antar-*workspace*, kepemilikan tugas, dan kondisi tugas. Hasil tersebut menunjukkan bahwa mekanisme *Dynamic Context-Aware RBAC* yang dikembangkan telah memenuhi kebutuhan pengujian yang ditetapkan dalam penelitian.

B. Pembahasan

1) Pembahasan Hasil Pengembangan *Backend* Sistem Manajemen Tugas *Multi-Tenan*

Backend yang dikembangkan pada penelitian ini ditujukan untuk mendukung pengelolaan tugas pada lingkungan *multi-tenant* melalui pemisahan fungsi autentikasi, pengelolaan *workspace*, manajemen tugas, pengendalian hak akses, serta modul pendukung berupa *audit trail* dan notifikasi *real-time*.

Hasil pengembangan menunjukkan bahwa backend menyediakan *REST API* sebagai mekanisme pertukaran data antara *client* dan server, sedangkan *WebSocket* digunakan untuk menyampaikan pembaruan informasi secara *real-time*. Pemisahan fungsi tersebut memungkinkan proses pengelolaan data dilakukan melalui API, sementara perubahan data tugas, penugasan, dan status dapat disampaikan kepada pengguna terkait tanpa memerlukan pemuatan ulang halaman.

Selain itu, implementasi *audit trail* menyediakan rekam aktivitas terhadap operasi penting pada tugas, seperti perubahan status, penugasan, persetujuan, dan penolakan. Dengan demikian, hasil pengembangan tidak hanya mendukung pengelolaan tugas, tetapi juga menyediakan mekanisme pencatatan aktivitas dan penyampaian informasi yang diperlukan dalam proses kolaborasi antar-*workspace*.

2) Pembahasan Hasil Penerapan *Dynamic Context-Aware Role-Based Access Control* RBAC

Hasil pengembangan menunjukkan bahwa mekanisme otorisasi pada *backend* tidak hanya menggunakan *role* sebagai dasar keputusan akses, tetapi juga mempertimbangkan konteks sumber daya yang diakses. Konteks tersebut meliputi *ownership*, *tenant boundary*, dan *resource state*. Penerapan dimensi tersebut memungkinkan keputusan otorisasi disesuaikan dengan kondisi pengguna dan sumber daya pada saat permintaan dilakukan.

Tabel 4. 10 Keunggulan Penerapan *Dynamic Context Aware* RBAC

No	Keunggulan	Pembahasan
1	Keputusan otorisasi berbasis konteks	Keputusan akses mempertimbangkan kombinasi <i>role</i> , <i>ownership</i> , <i>tenant boundary</i> , dan <i>resource state</i> sehingga hak akses disesuaikan dengan konteks operasional pengguna.

No	Keunggulan	Pembahasan
2	Isolasi data antar <i>tenant</i>	Akses terhadap <i>workspace</i> dan <i>task</i> dibatasi berdasarkan <i>tenant boundary</i> dan hubungan operasional.
3	Pengendalian <i>workflow</i>	Perubahan status tugas divalidasi berdasarkan <i>resource state</i> dan aturan <i>workflow</i> sehingga setiap proses mengikuti tahapan yang telah ditetapkan.
4	Adaptasi hak akses	Pengguna dengan <i>role</i> yang sama dapat memperoleh keputusan akses berbeda berdasarkan konteks sumber daya.

Berdasarkan Tabel 4.9, hasil pengujian, *tenant boundary* membatasi akses pengguna terhadap *workspace* yang tidak memiliki hubungan operasional. Pada aspek *ownership*, pengguna dapat melakukan operasi terhadap tugas yang menjadi tanggung jawabnya, sedangkan operasi terhadap tugas yang tidak menjadi tanggung jawabnya ditolak. Sementara itu, *resource state* membatasi perubahan status berdasarkan urutan *workflow* dan kewenangan pengguna.

Hasil tersebut menunjukkan bahwa pada skenario yang diuji, keputusan otorisasi tidak hanya ditentukan berdasarkan *role*, tetapi juga mempertimbangkan hubungan pengguna dengan sumber daya, batas *tenant*, dan kondisi tugas. Dengan demikian, mekanisme yang dikembangkan memberikan keputusan akses yang bersifat kontekstual sesuai dengan kondisi operasional yang telah ditetapkan.

- 3) Kontribusi Penelitian terhadap Sistem Manajemen Tugas *Multi-Tenant*.
Berdasarkan hasil pengembangan dan evaluasi, penelitian ini memberikan kontribusi berupa penerapan mekanisme kontrol akses berbasis konteks pada backend sistem manajemen tugas *multi-tenant*. Kontribusi tersebut ditunjukkan melalui integrasi *role*, *ownership*, *tenant boundary*, dan *resource state* dalam proses pengambilan keputusan otorisasi.

Tabel 4. 11 Kontribusi Penelitian terhadap Sistem Manajemen Tugas *Multi-Tenant*

No	Aspek Penerapan	Hasil Penelitian
1	Otorisasi berbasis konteks	Mengintegrasikan <i>role</i> , <i>ownership</i> , <i>tenant boundary</i> , dan <i>resource state</i> dalam keputusan otorisasi.

No	Aspek Penerapan	Hasil Penelitian
2	Isolasi data antar <i>tenant</i>	Membatasi akses terhadap <i>workspace</i> dan <i>task</i> berdasarkan batas <i>tenant</i> serta hubungan operasional.
3	Konsistensi proses bisnis	Mengendalikan perubahan status tugas berdasarkan <i>resource state</i> dan aturan <i>workflow</i> .
4	Penerapan artefak	Menghasilkan backend yang dapat digunakan untuk mendukung pengelolaan tugas lintas divisi pada lingkungan PT Bharata Internasional Pharmaceutical.

Berdasarkan Tabel 4.10, kontribusi utama penelitian terletak pada penerapan mekanisme otorisasi yang mempertimbangkan konteks operasional pengguna dan sumber daya. Mekanisme tersebut memungkinkan pembatasan akses antar-*tenant*, pengendalian operasi berdasarkan *ownership*, serta validasi perubahan status berdasarkan *resource state* dan aturan *workflow*.

Hasil *Black Box Testing* terhadap 16 skenario menunjukkan bahwa seluruh skenario menghasilkan status valid, sehingga hasil aktual pada seluruh skenario yang diuji sesuai dengan hasil yang diharapkan. Hasil tersebut mendukung bahwa mekanisme yang dikembangkan telah memenuhi kebutuhan pengujian yang ditetapkan dalam penelitian.

Secara keseluruhan, penelitian menghasilkan backend sistem manajemen tugas *multi-tenant* yang menerapkan kontrol akses berbasis konteks melalui kombinasi *role*, *ownership*, *tenant boundary*, dan *resource state*. Penerapan tersebut memberikan mekanisme pengendalian akses yang disesuaikan dengan konteks operasional pada skenario yang diuji, sekaligus mendukung pengelolaan tugas lintas divisi.

BAB V KESIMPULAN

A. Kesimpulan

Berdasarkan hasil implementasi, pengujian dan pembahasan, diperoleh kesimpulan sebagai berikut.

1. Penelitian ini berhasil mengembangkan *backend* sistem manajemen tugas berbasis *multi-tenant* dengan menerapkan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC). Mekanisme tersebut dikembangkan dengan mengintegrasikan *role*, *ownership*, *tenant boundary*, dan *resource state* sebagai dasar pengambilan keputusan otorisasi. Implementasi mekanisme tersebut didukung oleh *JWT Authentication* untuk autentikasi pengguna, *REST API* untuk pertukaran data, validasi *workflow* untuk pengendalian perubahan status tugas, *audit trail* untuk pencatatan aktivitas, serta *WebSocket* untuk penyampaian informasi secara *real-time*.
2. Pengujian artefak menggunakan metode *Black Box Testing* dilakukan terhadap 16 skenario, yang terdiri atas *Authentication* sebanyak 2 skenario, *Tenant Boundary* sebanyak 2 skenario, *Ownership* sebanyak 3 skenario, *Resource State* sebanyak 5 skenario, *Audit Trail* sebanyak 2 skenario, dan *Real-Time Notification* sebanyak 2 skenario. Hasil pengujian menunjukkan bahwa seluruh 18 skenario berstatus Valid, dengan 0 skenario Tidak Valid, sehingga diperoleh tingkat validitas sebesar 100%. Hasil tersebut menunjukkan bahwa hasil aktual pada seluruh skenario yang diuji sesuai dengan hasil yang diharapkan.

B. Saran

Adapun saran untuk penelitian berikutnya yaitu :

1. Penelitian selanjutnya dapat mengembangkan mekanisme *Dynamic Context-Aware Role-Based Access Control* (RBAC) dengan menambahkan dimensi konteks otorisasi lain, seperti waktu akses, lokasi pengguna, atau tingkat risiko akses, sehingga keputusan otorisasi dapat mempertimbangkan konteks yang lebih luas.
2. Penelitian selanjutnya dapat mengembangkan fitur *attachment versioning* untuk menyimpan dan mengelola beberapa versi dokumen lampiran pada setiap tugas, sehingga perubahan dokumen dapat ditelusuri dan pengguna dapat mengakses riwayat versi dokumen yang berkaitan dengan tugas.

DAFTAR PUSTAKA

- [1] A. Santhosh *et al.*, “International Research Journal on Advanced Engineering and Management SaaS (Software as a Service) and its Impact on Business Scalability”, doi: 10.47392/IRJAEM.2024.527.
- [2] S. Jaiswal, S. Desai, K. Gupta, A. Londhe, and Dr. R. Nanda, “AI Saas Web Application,” *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*, vol. 09, no. 04, pp. 1–9, Apr. 2025, doi: 10.55041/ijsrem43875.
- [3] M. Gupta, D. Gupta, and P. Rai, “Exploring the Impact of Software as a Service (SaaS) on Human Life,” *EAI Endorsed Transactions on Internet of Things*, vol. 10, 2024, doi: 10.4108/eetiot.4821.
- [4] Rishi Kumar Sharma, “Multi-Tenant Architectures in Modern Cloud Computing: A Technical Deep Dive,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 11, no. 1, pp. 307–317, Jan. 2025, doi: 10.32628/cseit25111236.
- [5] A. R. Reagan Panguraj, “Systematic Approach to Security Testing in Multi-Tenant Cloud Systems,” *International Journal of Multidisciplinary Research and Growth Evaluation.*, vol. 6, no. 1, pp. 2139–2142, 2025, doi: 10.54660/ijmrge.2025.6.1.2139-2142.
- [6] M Sahyudi and Erliyan Redy Susanto, “Analisis Implementasi Sistem Keamanan Basis Data Berbasis Role-Based Access Control (RBAC) pada Aplikasi Enterprise Resource Planning,” *SATESI: Jurnal Sains Teknologi dan Sistem Informasi*, vol. 5, no. 1, pp. 105–116, Apr. 2025, doi: 10.54259/satesi.v5i1.3997.
- [7] W. Chen, D. Han, L. Zhang, Q. Xiao, Q. Li, and H. Xiang, “A Model Study on Hierarchical Assisted Exploration of RBAC,” *International Journal of Digital Crime and Forensics*, vol. 14, no. 2, 2022, doi: 10.4018/IJDCF.302871.
- [8] R. Hariharan, “Zero Trust Security in Multi-Tenant Cloud Environments,” 2024. [Online]. Available: <https://www.jisem-journal.com/>
- [9] A. A. Zikra, A. Darnilasari, R. Yanuary, and K. Sari, “Design of Web-Based Project Management System with Multi-Level Role-Based Access Control,” *Journal of Computer Science and Informatics Engineering*, vol. 4, no. 3, pp. 201–215, Jul. 2025, doi: 10.55537/cosie.v4i3.1179.
- [10] J. Syafa Kamila and M. Falah Marzuq, “*International Journal On Informatic Visualization* journal homepage : www.joiv.org/index.php/joiv *International Journal On Informatic Visualization* Asana and Trello: A Comparative Assessment of Project Management Capabilities.” [Online]. Available: <https://planyway.com/blog/trello-kanban->

- [11] P. Pawłowski and M. Plechawska-Wójcik, “A comparative analysis of tools dedicated to project management Analiza porównawcza narzędzi dedykowanych zarządzaniu projektami,” 2022.
- [12] R. Ghosh, “Redefining CRM Infrastructure: What Businesses Gain by Shifting from Proprietary Suites to Unix-Based Open Architectures,” *Ritika Ghosh*, vol. 11, no. 6, 2023.
- [13] Liliana, D. Soesanto, B. Prijambodo, and J. Ohanna, “Exploration of Software as a Service (SaaS) as a Project Management Tools,” *Teknika*, vol. 13, no. 2, pp. 309–317, Jul. 2024, doi: 10.34148/teknika.v13i2.933.
- [14] Y. A. Prasetia and D. Manongga, “JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika) Journal homepage: <https://jurnal.stkippgritulungagung.ac.id/index.php/jipi> Role-Based Access Control (RBAC) Untuk Sistem Otorisasi Terpusat Berbasis Flask Studi Kasus PT. XYZ,” vol. 9, no. 4, pp. 1768–1778, 2024, doi: 10.29100/jipi.v4i1.5403.
- [15] L. Abdelrazek, L. Karaçay, and M. Orlic, “Authorization of Knowledge-base Agents in an Intent-based Management Function,” Oct. 2025, [Online]. Available: <http://arxiv.org/abs/2510.19324>
- [16] F. Krismahendra, B. Pangestu, D. Ramdani, K. Khanigara, M. I. Nurjamal, and Y. R. Wijanarko, “Pengembangan Model Kontrol Akses Berbasis Atribut (ABAC) Yang Selaras Dengan NIST Untuk Keamanan Rekam Medis Elektronik Pada Lingkungan *Hybrid Cloud*,” 1012.
- [17] A. C. Laksono and W. Sari, “Pengembangan Framework Tata Kelola Akses *Multi-Tenant* Untuk Mitigasi Ancaman Insider di *Cloud Publik*,” *Jurnal Informatika Teknologi dan Sains*.
- [18] F. Pamungkas and H. Setiaji, “Implementasi *Clean Architecture* Pada Pembuatan API Menggunakan Golang”.
- [19] S. Deochake and V. Channapattan, “Identity and Access Management Framework for Multi-tenant Resources in Hybrid Cloud Computing,” Mar. 2022, [Online]. Available: <http://arxiv.org/abs/2203.11463>
- [20] A. Morozov and K. J. Donovan, “International Journal of Modern Computer Science and IT Innovations (IJMCSII) The Transformative Impact of Containerization on Modern Web Development: An In-depth Analysis of Docker and Kubernetes Ecosystems,” vol. 10, pp. 27–39, 2025, [Online]. Available: <https://aimjournals.com/index.php/ijmcsit>
- [21] A. Anand B., D. B. C. Darunya, A. B. S. Anisha, and S. G. Raghavendra Prasad, “A survey on docker container and its use cases,” *International Journal of Advance Research, Ideas and Innovations in Technology*, vol. 7, no. 3, 2021.

- [22] H. Amin, L. Ana, A. Suhendi, K. Kunci, E. Operasional, and L. Terisolasi, "Otomatisasi Deployment dan Manajemen Multi-Kontainer Docker untuk Skalabilitas dan Efisiensi Lingkungan Komputansi Terisolasi," *Teknik Komputer dan Teknologi Pendidikan (JUSTIKPEN)*, vol. 5, 2025, doi: 10.55338/justikpen.v5i1.261.
- [23] S. J. Putri, D. Galih, P. Putri, W. Hayuhardhika, and N. Putra, "Analisis Komparasi pada Teknik Black Box Testing (Studi Kasus: Website Lars)," *Journal of Internet and Software Engineering*, vol. 5, no. 1, 2024.
- [24] P. Antunes, N. H. Thuan, and D. Johnstone, "Nature and purpose of visual artifacts in design science research," *Information Systems and e-Business Management*, vol. 20, no. 3, pp. 515–550, 2022, doi: 10.1007/s10257-022-00559-2

LAMPIRAN

Lampiran I Observasi Lapangan

Observasi Lapangan dan Crosscheck sistem yang berjalan di perusahaan untuk menyatakan bahwa aplikasi sudah di implementasikan di perusahaan.



Domain : <https://task.bharatainternasional.com>

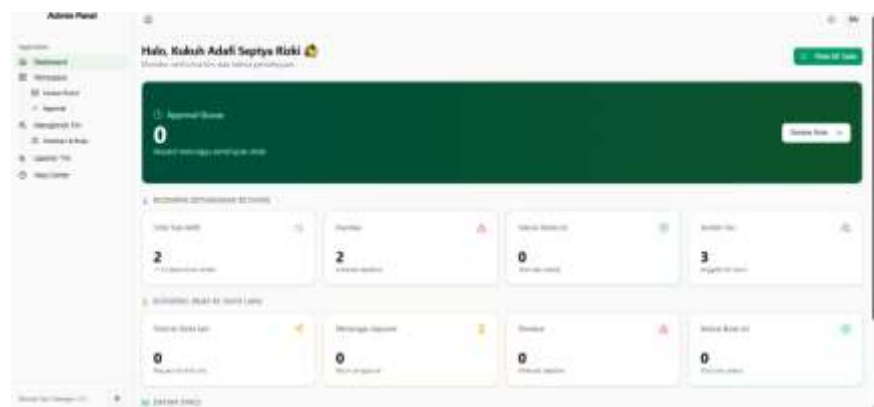
Link Github : (<https://github.com/aldikagagah/task-management-system.git>)

Lampiran II Pengujian Authentication

1. TC-AUTH-01

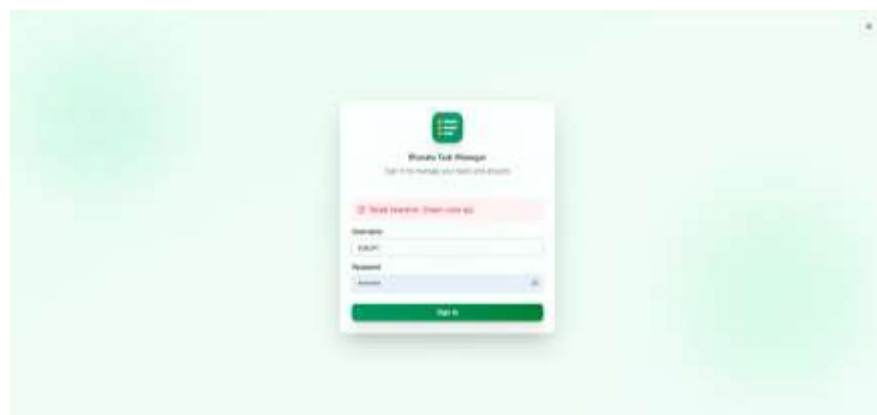


Gambar Lampiran 1 Login Dashboard



Gambar Lampiran 2 Tampilan Setelah Login

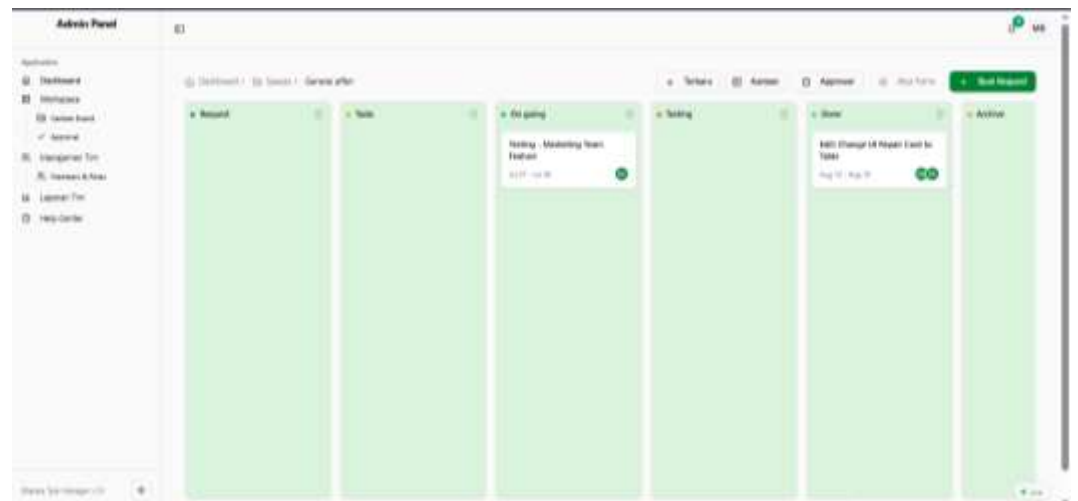
2. TC-AUTH-02



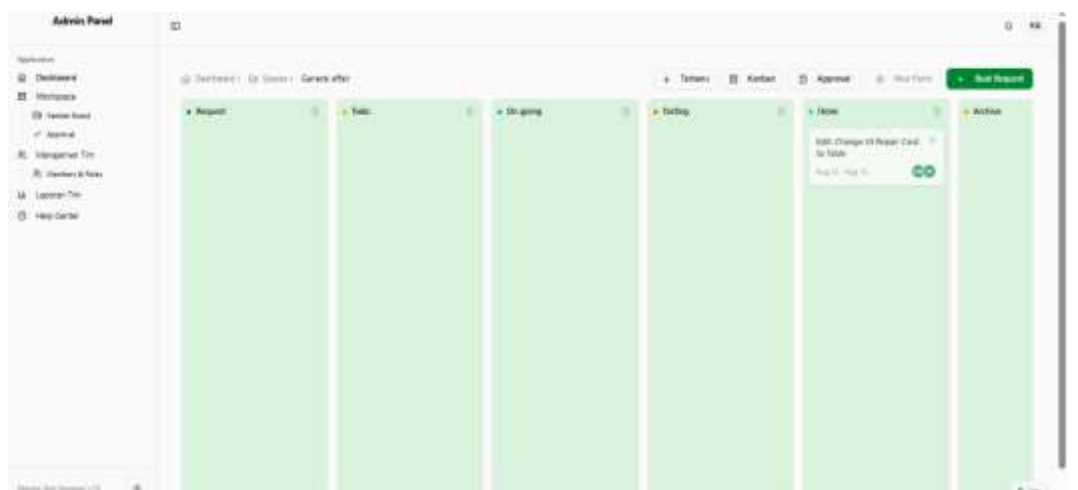
Gambar Lampiran 3 Akun Tidak Valid

Lampiran III Pengujian *Tenant Boundary*

1. TC-TB-01

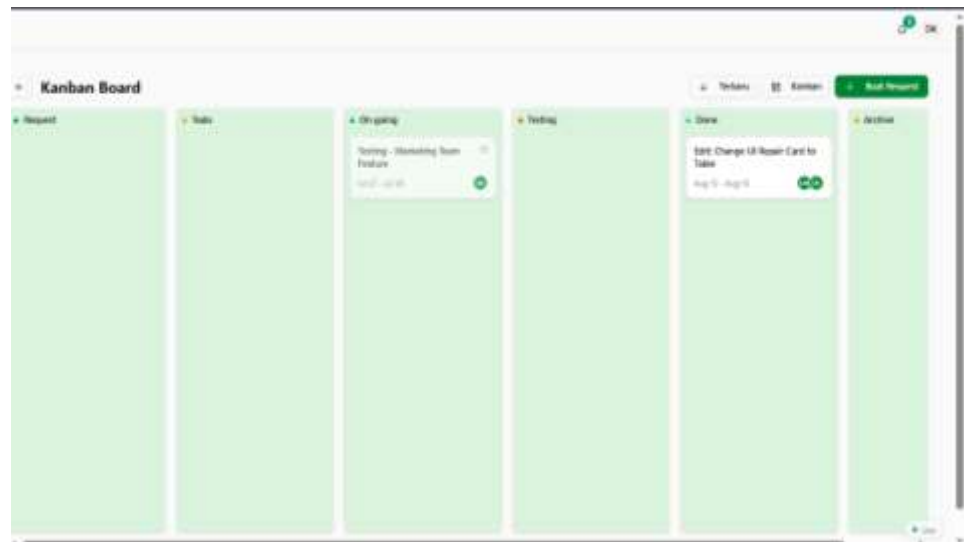


Gambar Lampiran 4 Data Task Incoming Di Divisi Penerima Task OutGoing



Gambar Lampiran 5 Kanban Hanya Menampilkan Task Yang Memiliki Korelasi Tanpa Memperlihatkan Data Task Internal Divisi Tersebut

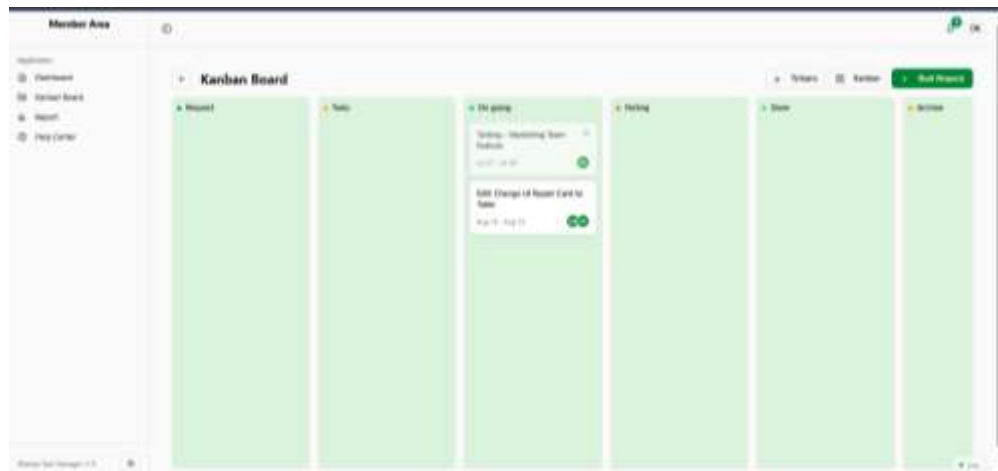
2. TC-TB-02



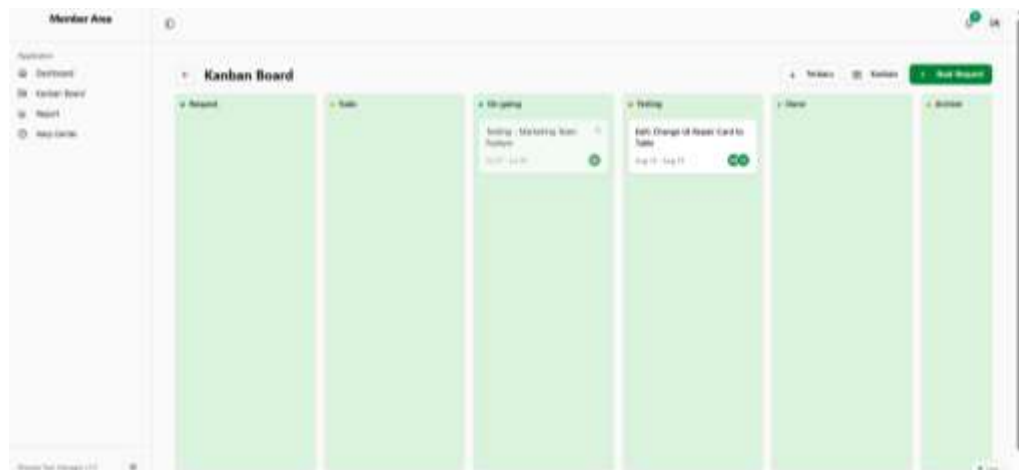
Gambar Lampiran 6 Staf Hanya Bisa Mengakses *Assignment Task* Yang Diberikan kepadanya

Lampiran IV Pengujian Ownership

1. TC-OWN-01

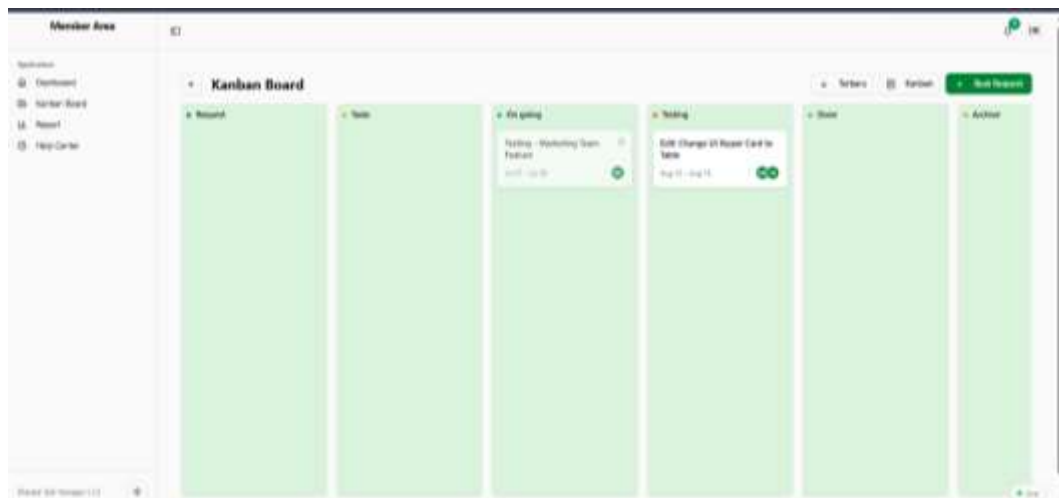


Gambar Lampiran 7 Status Tugas Sebelum Dipindah



Gambar Lampiran 8 Setelah Melakukan Perubahan Task

2. TC-OWN-02



Gambar Lampiran 9 Menunjukkan Aturan Hak Akses Yang membuat Staff tidak bisa merubah status task

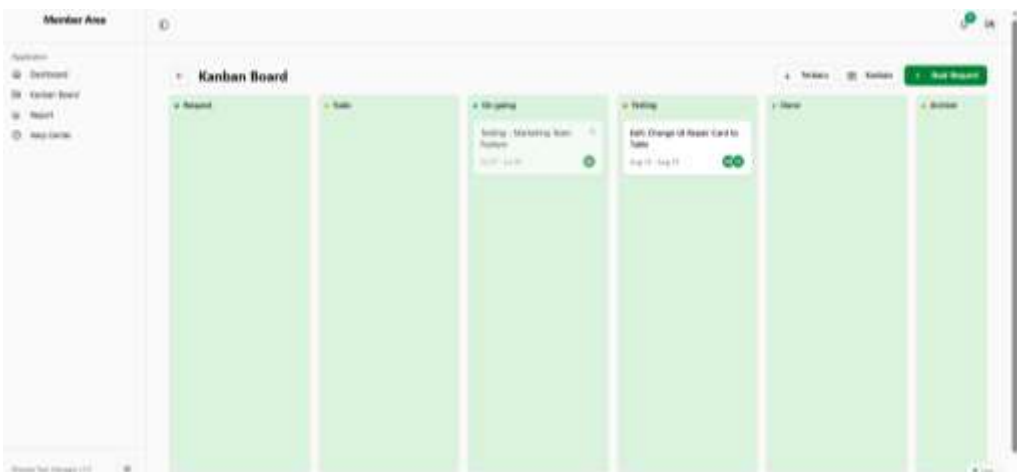
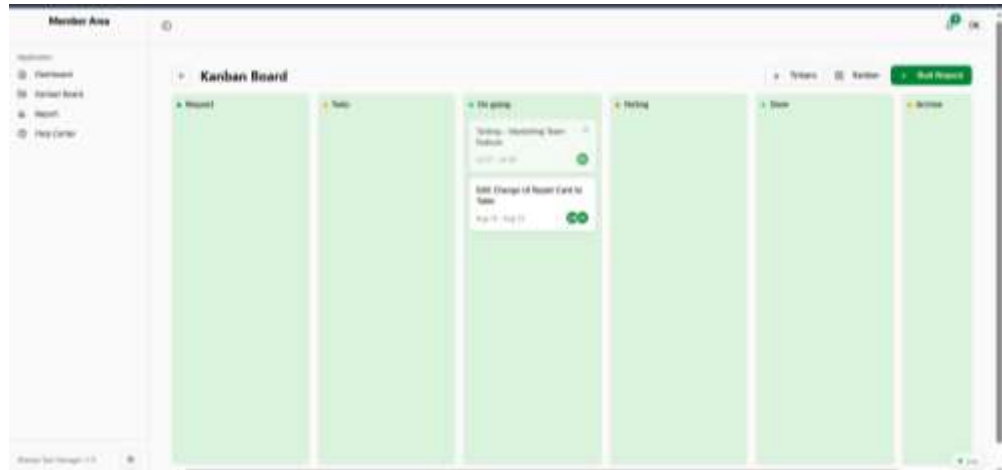
3. TC-OWN-03



Gambar Lampiran 10 Supervisor Pengaju Out Going Task Hanya Bisa Melihat task divisinya

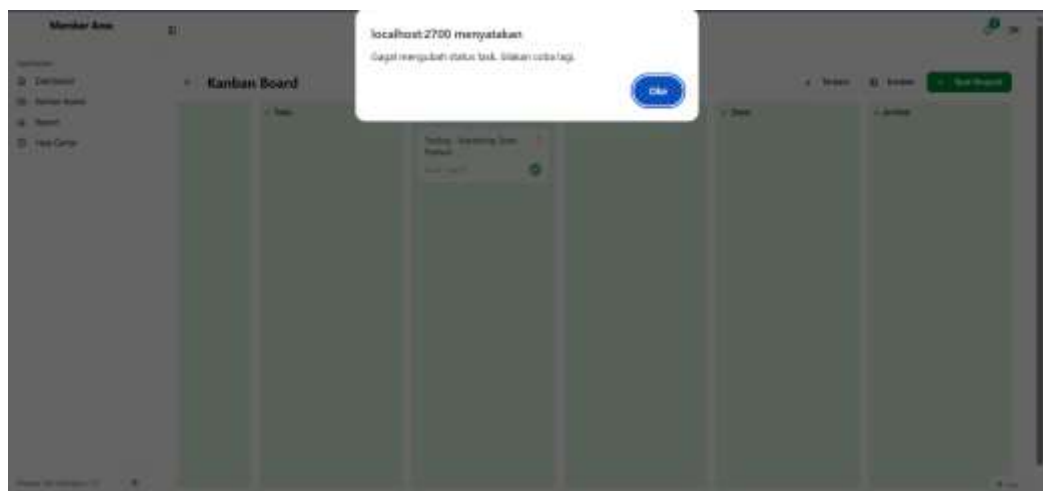
Lampiran V Pengujian Resource State

1. TC-RS-01



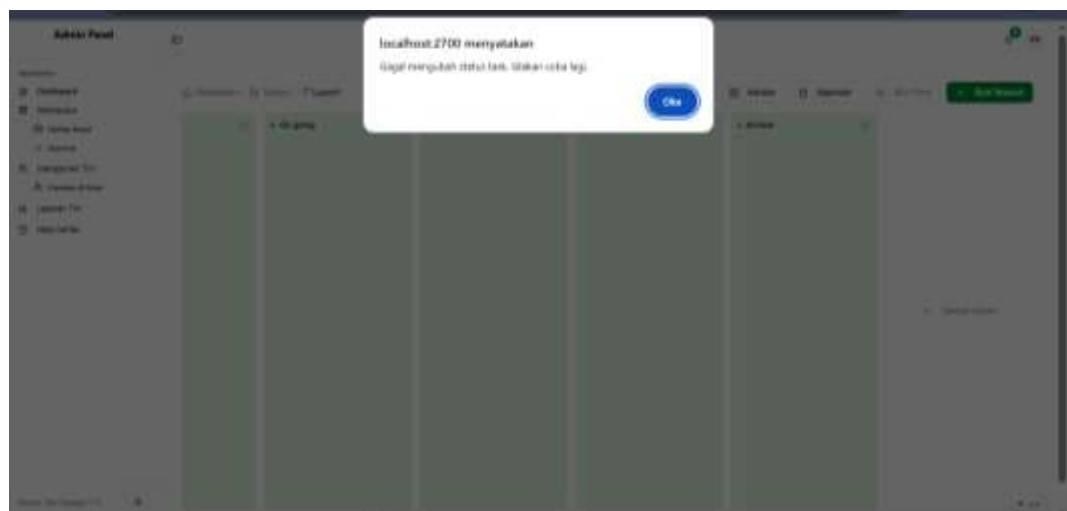
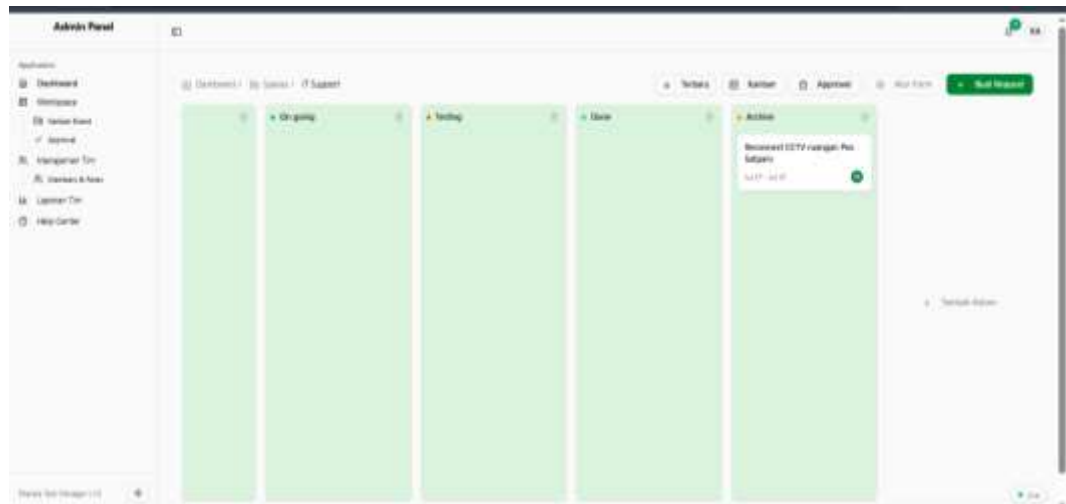
Gambar Lampiran 11 Staf Melakukan Perubahan Status Task Sesuai Aturan

2. TC-RS-02



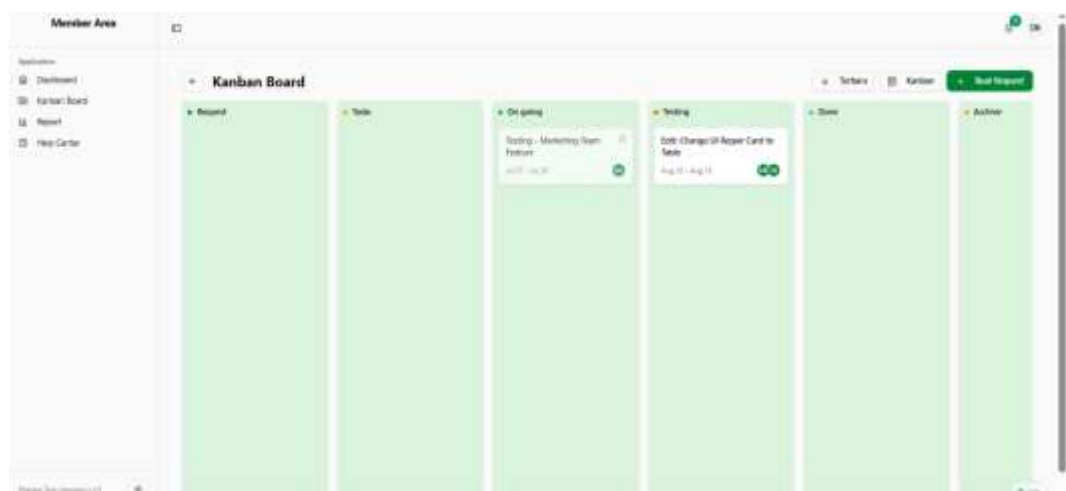
Gambar Lampiran 12 Sistem menolak Perubahan Status Yang tidak sesuai aturan workflow

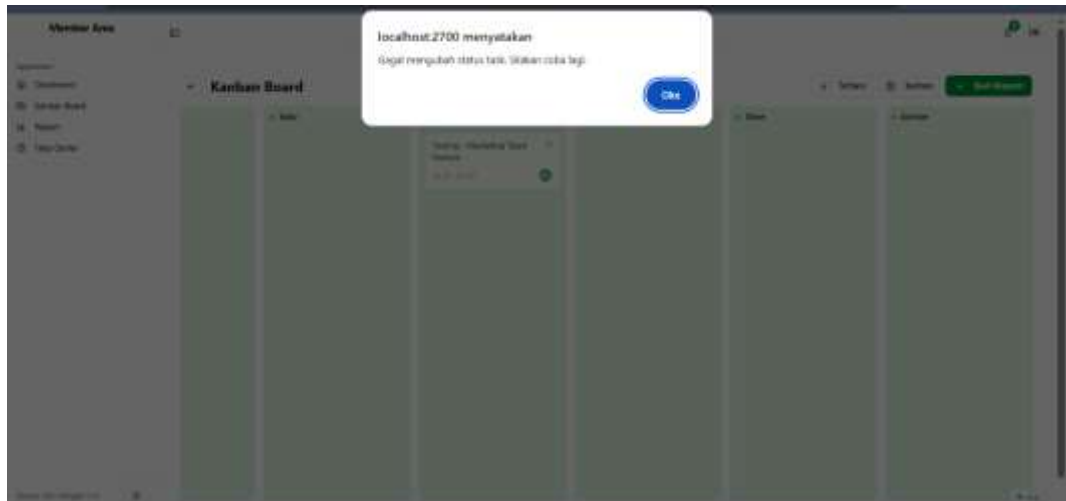
3. TC-RS-03



Gambar Lampiran 13 Status Task Yang Sudah Archive Tidak Bisa Dikembalikan Ke Status Lain

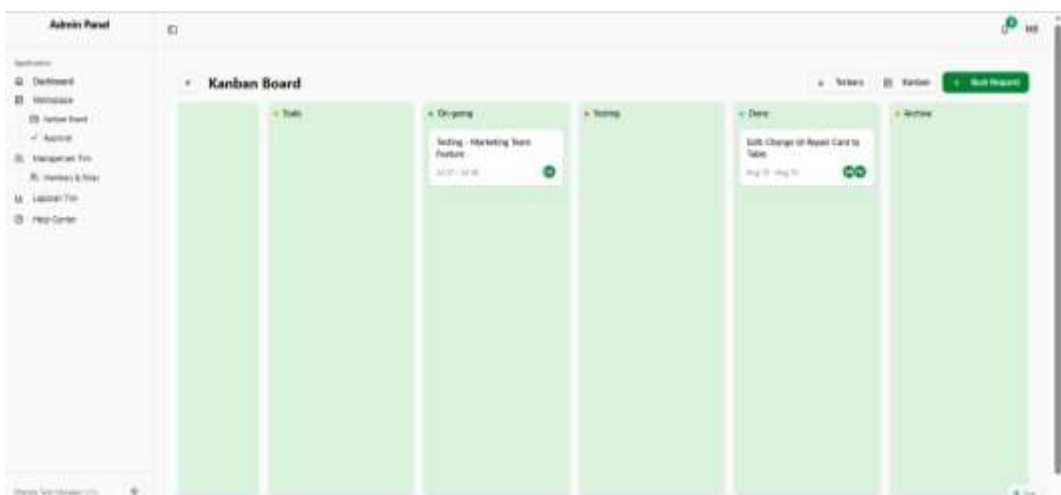
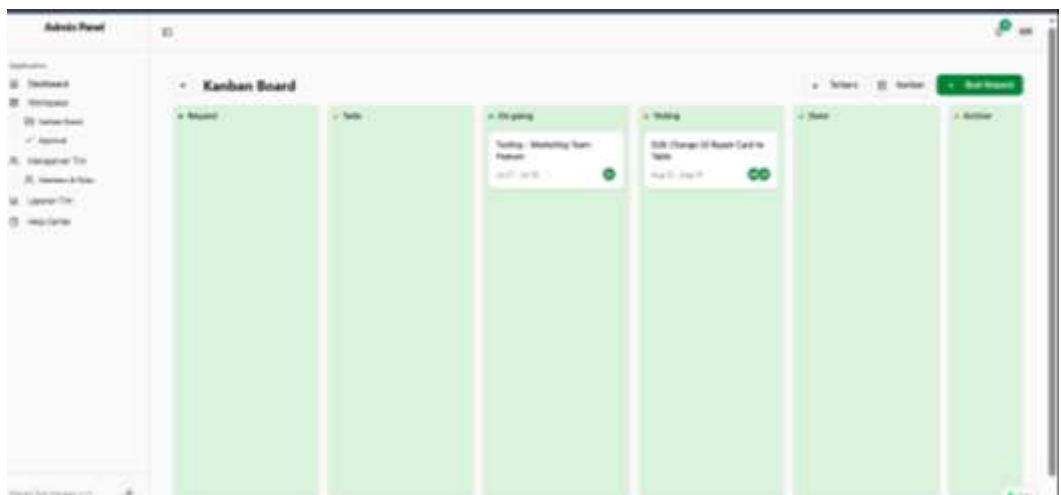
4. TC-RS-04





Gambar Lampiran 14 User Staff Tidak Bisa Melakukan Perubahan Status Ke Done Dan Itu Hanya Hak Akses Supervisor

5. TC-RS-05



Gambar Lampiran 15 Hak *Workflow Done* Hanya Supervisor

Lampiran VI Pengujian *Audit Trail*

1. TC-AT-01



Gambar Lampiran 16 Pencatatan Perubahan Status Tugas

2. TC-AT-02



Gambar Lampiran 17 Pencatatan Penugasan Tugas

Lampiran VII Pengujian Notifikasi

1. TC-RT-01



Gambar Lampiran 18 Notifikasi Pembuatan Task

2. TC-RT-02



Gambar Lampiran 19 Notifikasi Perubahan Status Tugas