

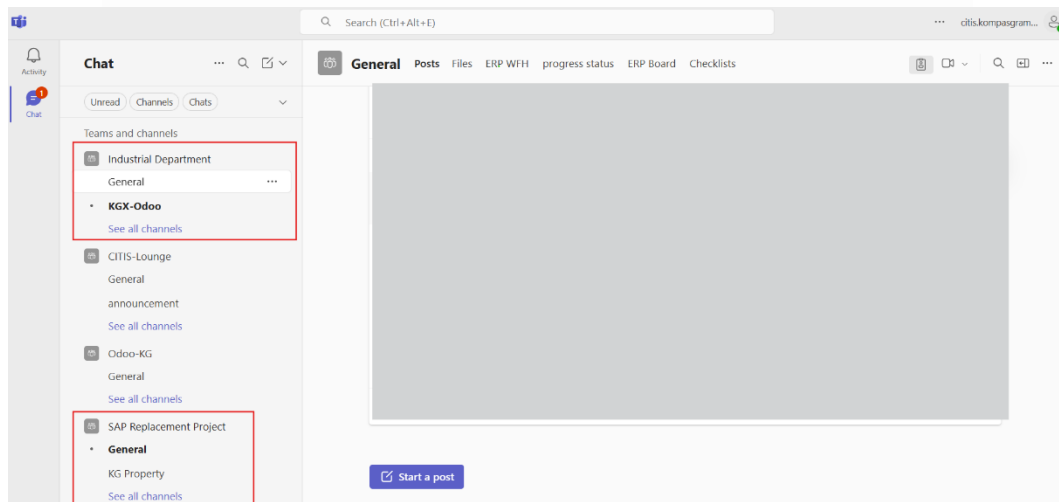
BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam pelaksanaan magang di Kompas Gramedia, mahasiswa magang ditempatkan di *Departemen Enterprise Technology CITIS (Corporate IT & IS)* sebagai *Intern Junior Software Engineer*. Selama masa magang, mahasiswa mendapatkan bimbingan dan supervisi langsung dari *Senior Software Engineer*, yang bertindak sebagai mentor serta pengawas dalam setiap tugas yang diberikan. Dalam menjalankan pekerjaannya, mahasiswa magang bertanggung jawab kepada supervisi, yang selanjutnya akan melaporkan hasil kerja kepada manajer departemen. Kedudukan ini memungkinkan mahasiswa untuk mendapatkan arahan langsung dalam pengembangan sistem.

Koordinasi dalam pelaksanaan tugas dilakukan melalui dua metode, yaitu tatap muka di kantor Kompas Gramedia Palmerah Selatan dan secara online melalui *platform Microsoft Teams*. Selain itu, setiap dua minggu sekali pada hari Kamis, mahasiswa magang mengikuti *sprint meeting industrial*, yang menjadi forum untuk mengevaluasi capaian dan perencanaan backlog selanjutnya.



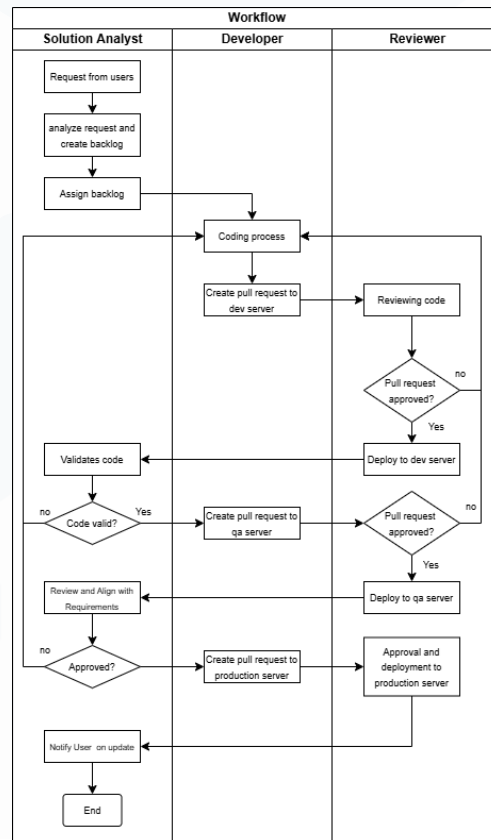
Gambar 3. 1 Tampilan Microsoft Teams

Pada *platform Microsoft Teams*, mahasiswa magang berkoordinasi dengan Analyst yang bertugas memberikan panduan terkait kebutuhan sistem serta memastikan bahwa pengembangan fitur sesuai dengan spesifikasi yang telah ditetapkan. Peserta

magang telah diberikan akses oleh pihak perusahaan untuk bergabung ke dalam *Microsoft Teams*, sebagaimana ditunjukkan pada Gambar 3.1

3.2 Tugas dan Uraian Kerja Magang

subbab ini menguraikan tugas-tugas yang telah dilaksanakan serta deskripsi kerja yang mencakup tanggung jawab, aktivitas harian, dan kontribusi terhadap operasional. Selama menjalani program magang di Kompas Gramedia sebagai *software engineer*, pekerjaan yang dilakukan adalah mengkostumisasi fitur-fitur pada sistem Odoo, sebuah *platform Enterprise Resource Planning (ERP)* yang digunakan untuk mengelola berbagai proses bisnis. Pekerjaan ini melibatkan pengembangan dan penyesuaian fitur sesuai kebutuhan perusahaan menggunakan bahasa pemrograman Python, yang merupakan bahasa utama untuk pengembangan modul Odoo. Proses kerja selama magang dilaksanakan mengikuti alur sebagai berikut.



Gambar 3. 2 Alur Kerja dalam Corporate IT & IS Kompas Gramedia

Gambar 3.2 menunjukkan *flowchart* alur kerja di divisi *Corporate IT & IS* Kompas Gramedia. Proses dimulai ketika pengguna (*user*) mengajukan permintaan kepada analis untuk melakukan perbaikan atau kustomisasi fitur sesuai dengan kebutuhan mereka. Kemudian, analis akan menganalisis permintaan tersebut, menyusun *backlog*, dan menugaskan *backlog* tersebut kepada *developer*. Selanjutnya, *developer* akan mengerjakan permintaan tersebut. Setelah selesai, *developer* akan membuat *pull request* ke *server* pengembangan (*development server*). Kode tersebut kemudian akan ditinjau oleh *reviewer*. Jika kode dianggap tidak aman, *pull request* akan ditolak, dan *developer* harus memperbaiki kesalahan yang ada. Sebaliknya, jika kode dinilai aman, *reviewer* akan melakukan *deployment* ke *server dev*.

Selanjutnya, kode yang telah diunggah ke *server dev* akan diuji oleh analis untuk memastikan apakah kode tersebut berfungsi sesuai dengan permintaan pengguna. Jika tidak sesuai, analis akan memberikan umpan balik kepada *developer*, dan *developer* harus memperbaiki kesalahan yang ada. Sebaliknya, jika kode sesuai dengan kebutuhan, analis akan meminta *developer* untuk membuat *pull request* ke *server QA*. *Pull request* tersebut kemudian akan ditinjau kembali oleh *reviewer*. Jika *pull request* ditolak, proses akan kembali ke *developer* untuk melakukan perbaikan. Namun, jika *pull request* dinilai sesuai, *reviewer* akan melakukan *deployment* ke *server QA*. Selanjutnya, analis akan kembali menguji kode bersama *Financial System Developer (FSD)* untuk memastikan bahwa kode memenuhi standar etik dan kebijakan perusahaan. Jika kode tidak sesuai, proses akan dikembalikan kepada *developer* untuk diperbaiki. Sebaliknya, jika kode telah sesuai, analis akan meminta *developer* untuk membuat *pull request* ke *server produksi*. *Reviewer* kemudian akan meninjau dan melakukan *deployment* ke *server produksi*. Setelah itu, analis akan memberi tahu pengguna bahwa permintaan mereka telah selesai.

Berdasarkan alur kerja yang telah dijelaskan sebelumnya, Tabel 3.1 merincikan tugas yang secara umum telah dikerjakan oleh penulis selama masa magang.

Tabel 3. 1 Kegiatan kerja magang

Tanggal	Kegiatan yang dilakukan	Kategori	PIC
28 – 31 januari	Konfigurasi dan <i>set up</i> Odoo	Pembelajaran	Supervisor
03 – 14 Februari	Mempelajari Python, Odoo, dan alur Git	Eksplorasi	Supervisor
17 – 28 Februari	Penamaan label, urutan filter, sequence, dan menu pada modul accounting & invoicing	Pengembangan Perangkat Lunak, Maintenance	Enterprise Technology Manager
03 – 07 Maret	Ubah field partner_id, revisi backlog, sprint meeting, pengerjaan fitur ratio project	Maintenance, Pengembangan, Kolaborasi Tim	Enterprise Technology Manager
10 – 18 Maret	Modifikasi tombol dan akses, report, field baru, form project/budget	Pengembangan, Maintenance	Enterprise Technology Manager
24 – 27 Maret	Ubah formview project dan opex, revisi form change budget	Maintenance dan Optimalisasi Sistem	Enterprise Technology Manager
28 Maret – 25 April	Cuti Lebaran	-	-
07 – 11 April	Tambah kolom payment order line, field no_open, sprint meeting	Pengembangan, Kolaborasi Tim	Enterprise Technology Manager
14 – 17 April	Mengerjakan Feedback form change budget, tambah field relasi di project budget	Maintenance, Pengembangan	Enterprise Technology Manager
21 April – 02 Mei	Field use_budget readonly, struktur budgeting, pengosongan fund center, pengaturan project ID & tombol	Maintenance, Pembelajaran	Enterprise Technology Manager
05 – 16 Mei	Hide create/edit deposit request, sprint meeting, pengosongan fund center, belajar alur Git dan struktur PO/PR	Maintenance, Kolaborasi Tim, Pembelajaran	Enterprise Technology Manager
22 Mei – 05 Juni	Tambah akses grup user sales, viewer inventory, hide create/edit di budget opex	Pengembangan, Maintenance	Enterprise Technology Manager

Mengacu pada Tabel 3.1, pekerjaan-pekerjaan yang telah diuraikan sebelumnya dapat diklasifikasikan ke dalam beberapa kelompok berikut:

3.2.1 Pembelajaran

Pada tahap awal pelaksanaan magang, khususnya pada minggu kedua hingga minggu ketiga, difokuskan untuk melakukan proses pembelajaran terhadap sistem dan lingkungan kerja yang digunakan di Kompas Gramedia, khususnya pada sistem ERP Odoo. Proses pembelajaran ini mencakup pemahaman terhadap alur kerja internal tim, struktur kode pada proyek yang digunakan, serta tools yang digunakan. Meskipun fokus utama pembelajaran berlangsung di dua minggu awal, proses belajar tidak berhenti di situ. Di minggu-minggu berikutnya, penulis tetap melakukan pembelajaran tambahan secara paralel dengan pengerjaan *backlog*, terutama ketika menemui kasus baru atau fitur yang belum pernah dikerjakan sebelumnya. Berikut secara detail pembelajaran yang dilakukan saat periode magang

3.2.1.1 Pengenalan *Framework* Odoo ERP

Selama satu bulan pertama masa magang, difokuskan untuk mempelajari *framework* Odoo ERP guna memahami dasar-dasar sistem yang akan digunakan dalam pekerjaan pengembangan. Sebagai bagian dari proses ini, berbagai materi pembelajaran diberikan berupa video tutorial dan dokumentasi resmi yang menjelaskan struktur dasar Odoo, arsitektur modularnya, serta alur pengembangan fitur menggunakan bahasa pemrograman *Python* dan *XML*. Materi pembelajaran ini mencakup pemahaman tentang model, *View*, *Action*, dan juga tutorial tutorial lain sesuai kebutuhan bisnis.

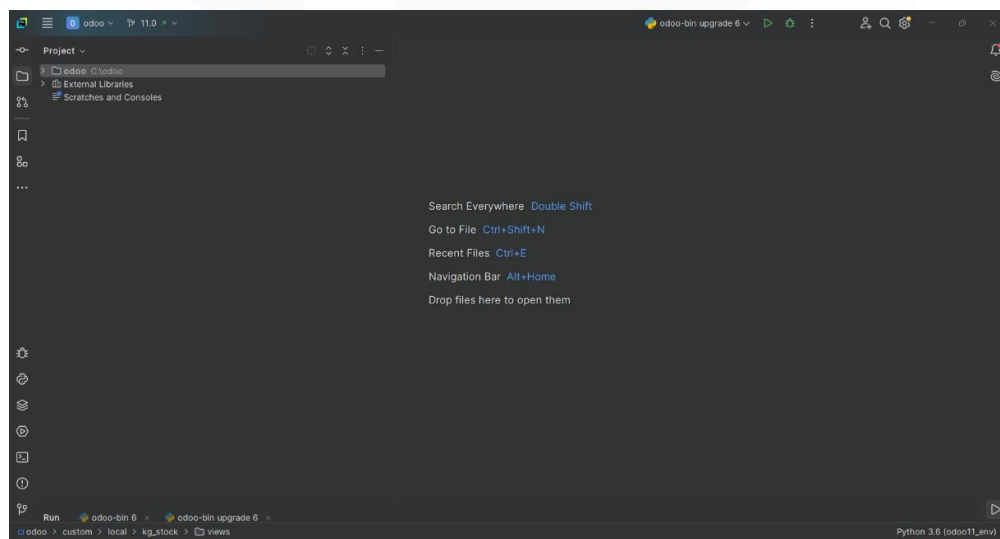
3.2.1.2 Alur kerja di dalam CITIS

Di dalam divisi *Corporate IT & IS* (CITIS) Kompas Gramedia, proses pengembangan proyek dilakukan dengan menerapkan metodologi *Agile Scrum* berbasis sprint. Setiap pagi dilakukan sesi *daily stand-up meeting* untuk menyampaikan progres harian, hambatan yang dihadapi, serta rencana kerja yang akan dilakukan. Informasi ini juga dicatat melalui *platform microsoft teams* yang diisi secara rutin oleh seluruh anggota tim.

Selain itu, setiap dua minggu sekali dilakukan *sprint meeting* secara terjadwal dalam lingkup industrial untuk melakukan evaluasi hasil pekerjaan dalam satu siklus *sprint*, serta merencanakan *backlog* yang akan dikerjakan pada *sprint* berikutnya. Semua tugas dikumpulkan dalam bentuk *backlog*, yang dibuat dan dikelola oleh analis. Pengembang kemudian mengambil *backlog* sesuai penugasan dan fokus pada penyelesaiannya dalam periode yang sudah ditentukan.

3.2.1.3 Tools yang digunakan

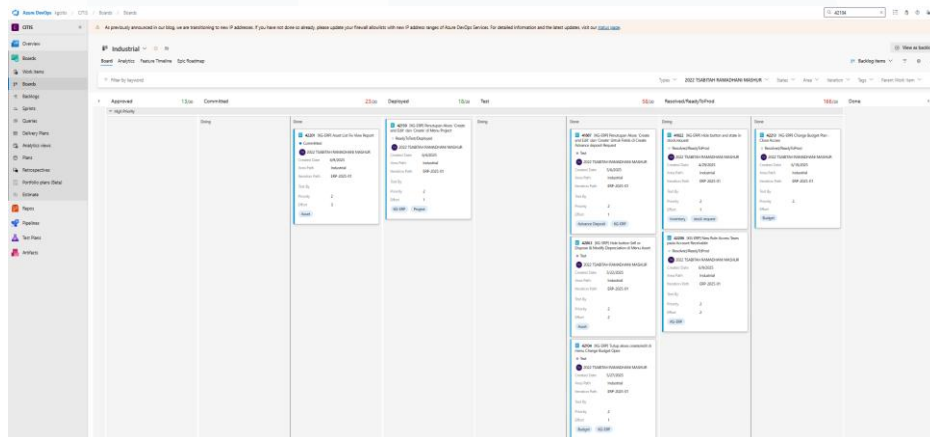
Dalam mendukung proses pengembangan selama masa magang, digunakan beberapa *tools* utama yang berperan penting dalam kegiatan teknis dan kolaboratif. Untuk keperluan pemrograman, digunakan PyCharm sebagai *integrated development environment* yang mendukung bahasa Python, bahasa utama dalam pengembangan modul dan fitur pada *framework* Odoo. Berikut merupakan tampilan antarmuka dari pyCharm



Gambar 3. 3 Tampilan Antarmuka PyCharm

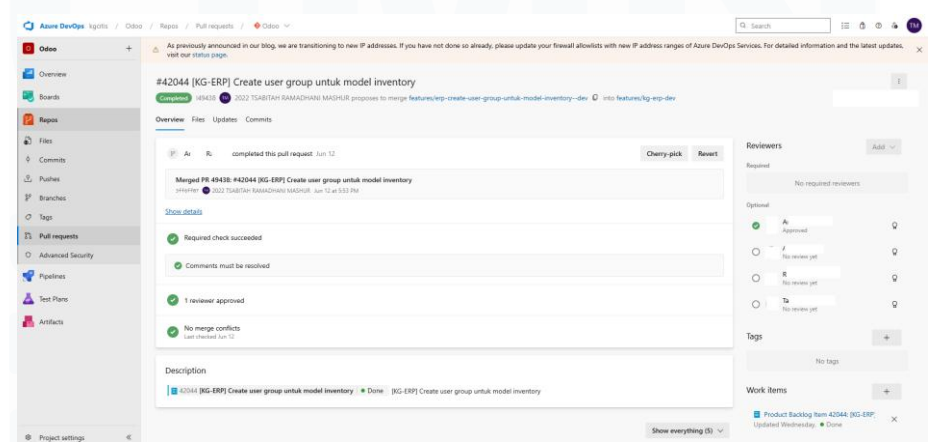
Selain itu, sistem manajemen proyek dan kolaborasi tim dijalankan menggunakan Azure DevOps, yang terdiri dari beberapa fitur utama seperti *Azure Boards*, *Azure Repos*, dan *Azure Pipelines*.

Azure Boards digunakan sebagai tempat pengelolaan *backlog* secara terstruktur. Pada platform ini, analis menempatkan *backlog* yang berisi *task* ke dalam kolom *Approved*. *Backlog* yang telah disetujui ini kemudian dapat diambil dan dikerjakan oleh *developer*. Setelah *task* ditugaskan kepada *developer*, *backlog* tersebut akan dipindahkan ke kolom *Committed (Doing)* sebagai penanda *backlog* sedang dikerjakan.



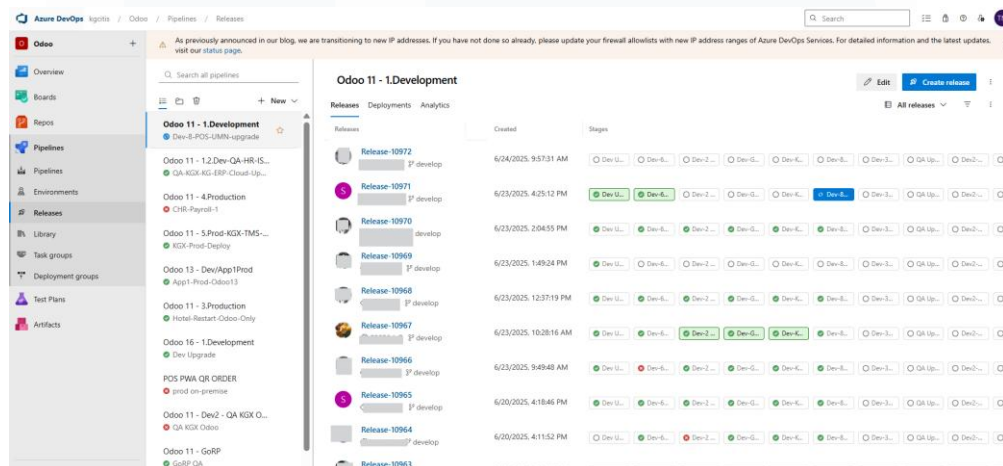
Gambar 3. 4 Tampilan Antarmuka Azure Boards

Ketika *backlog* telah selesai dikerjakan, maka *developer* akan memindahkannya ke kolom *Committed (Done)*. Selanjutnya, setelah *task* tersebut berhasil dideploy ke *server*, *backlog* kembali dipindahkan ke kolom *Deployed* oleh *developer*. Setelah mencapai tahap ini, proses selanjutnya berada di bawah tanggung jawab analis.



Gambar 3. 5 Tampilan Antarmuka Azure Repos

Azure Repos digunakan untuk melakukan *pull request*. Setelah *developer* menyelesaikan mengerjakan *backlog*, mereka membuat *pull request* melalui Azure Repos. Setiap perubahan kemudian ditinjau terlebih dahulu oleh *manager* atau senior (*reviewer*) sebelum disetujui dan digabungkan ke *branch* utama. Proses ini memastikan bahwa setiap pembaruan kode telah diperiksa dengan cermat demi menjaga kualitas sistem.



Gambar 3. 6 Tampilan Antarmuka Azure Pipelines

Sementara itu, Azure Pipelines untuk proses *deployment*. Setelah perubahan kode disetujui, pipeline akan dijalankan secara otomatis untuk melakukan proses build dan *deployment* ke *server*. Penggunaan Azure Pipelines mempercepat dan mempermudah proses integrasi serta pengujian, sehingga hasil dari *backlog* dapat segera diimplementasikan dan diuji.

3.2.1 Pengerjaan *backlog*

Selama masa pelaksanaan magang, salah satu tanggung jawab utama yang diberikan adalah mengerjakan *backlog* yang telah disusun oleh tim analis sebagai bagian dari proses pengembangan dan pemeliharaan sistem erp di Kompas Gramedia. *Backlog* merupakan daftar pekerjaan atau fitur yang perlu dikerjakan, baik yang bersifat perbaikan *bug*, pengembangan fitur baru, maupun penyesuaian tampilan atau alur sistem sesuai kebutuhan pengguna.

Backlog-backlog yang dikerjakan bervariasi dari segi tingkat kompleksitas, ruang lingkup pekerjaan, serta modul atau menu sistem yang terlibat di dalamnya. Beberapa *backlog* bersifat sederhana seperti penggantian label, penyesuaian *field*, atau perbaikan minor pada tampilan, namun ada pula *backlog* yang menuntut pemahaman lebih mendalam terhadap logika bisnis, alur data, serta integrasi antar modul dalam sistem ERP.

Pengerjaan *backlog* tersebar di berbagai modul penting dalam sistem, di antaranya mencakup menu *Account, Receivable, Accounting, Transportation, Invoicing, Inventory, Project, Budgeting, Stock Request, Purchases, Sales*, hingga *Assets*. Setiap modul memiliki karakteristik dan tantangan tersendiri, sehingga pengerjaan *backlog* tidak hanya menuntut keterampilan teknis dalam pemrograman, tetapi juga kemampuan dalam memahami proses bisnis yang berjalan di perusahaan.

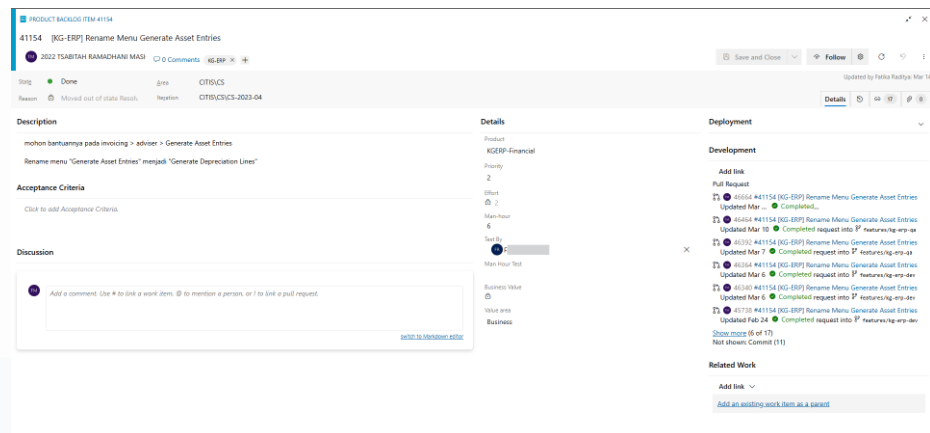
Dalam proses pengerjaannya, setiap *backlog* diawali dengan pemahaman terhadap deskripsi tugas yang tertera pada sistem manajemen proyek, yang dalam hal ini menggunakan Azure DevOps (DevAzure). Setelah memahami kebutuhan dan ruang lingkup dari setiap *backlog*, proses selanjutnya adalah eksplorasi terhadap struktur kode yang ada, pengembangan solusi, hingga proses pengujian sebelum akhirnya *backlog* dinyatakan selesai dan siap untuk di review oleh tim QA (*Quality Assurance*). Berikut *backlog backlog* yang telah dikerjakan selama periode magang berlangsung

3.2.1.1 Backlog Yang Dikerjakan Pada Menu *Invoicing*

3.2.1.1.1 Mengubah Nama Menu *Generate Asset Entries*

Pada *backlog* dengan judul “*Rename Menu Generate Asset Entries*”, dilakukan penyesuaian terhadap salah satu nama sub-menu dalam menu

Invoicing, tepatnya pada menu Generate Asset Entries.



Gambar 3. 7 Backlog Rename Menu Generate Asset Entries

Gambar 3.8 Menunjukkan Permintaan dalam *backlog* adalah untuk mengubah nama menu tersebut menjadi *Generate Depreciation Lines* agar lebih sesuai dengan fungsi yang dijalankan oleh menu tersebut, yaitu untuk menghasilkan baris depresiasi atas aset tetap.

```
<menuitem id="menu_generate_asset_entries"
  parent="menu_adviser_generate_entries"
  action="account_asset.action_asset_depreciation_confirmation_wizard"
  name="Generate Assets Entries"
  sequence="0"
  groups="kg_bank.group_accounting_officer"/>
```

Gambar 3. 8 Cuplikan Kode Menu Generate Asset Entries sebelum perubahan

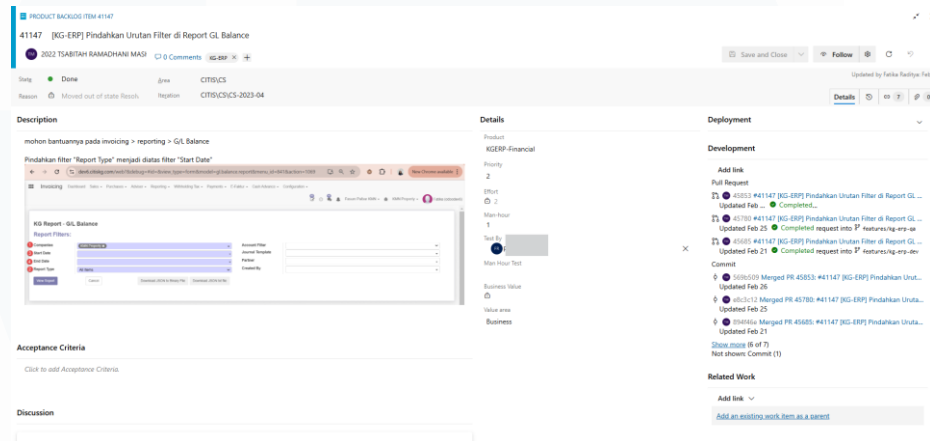
```
<menuitem id="menu_generate_asset_entries"
  parent="menu_adviser_generate_entries"
  action="account_asset.action_asset_depreciation_confirmation_wizard"
  name="Generate Depreciation Lines"
  sequence="0"
  groups="kg_bank.group_accounting_officer"/>
```

Gambar 3. 9 Cuplikan Kode Menu Generate Asset Entries setelah perubahan

Perubahan ini dilakukan dengan memodifikasi elemen `menuitem` pada *file XML* yang terkait. Sebelumnya, nama menu menggunakan nilai "Generate Asset Entries" seperti pada gambar 3.9, kemudian diubah menjadi "Generate Depreciation Lines" seperti pada gambar 3.10

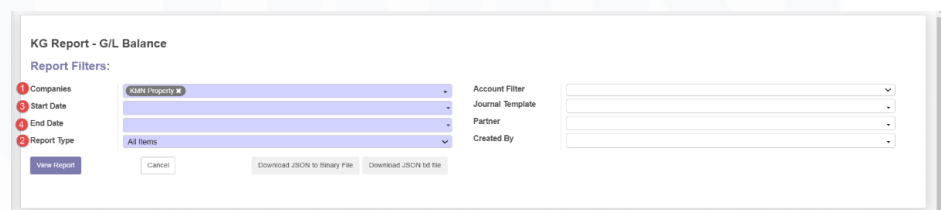
3.2.1.1.2 Memindahkan urutan filter di report *General Ledger Balance*

backlog dengan judul “Rename Menu Generate Asset Entries” ini meminta untuk melakukan penyesuaian pada tampilan filter di menu *Invoicing > Reporting > G/L Balance* pada sistem Odoo.



Gambar 3. 10 Backlog Pindahkan urutan filter di Report General Ledger Balance

Gambar 3.11 Menunjukkan Permintaan dalam *backlog* untuk melakukan Penyesuaian tersebut berupa pemindahan posisi filter "Report Type" agar diletakkan di atas filter "Start Date", sesuai dengan permintaan dari tim pengguna. *backlog* ini bertujuan untuk meningkatkan kenyamanan pengguna dalam menyaring laporan berdasarkan jenis laporan terlebih dahulu sebelum menentukan rentang tanggal yang diinginkan.



Gambar 3. 11 Tampilan G/L Balance Perubahan di Tetapkan

Penyesuaian filter "Report Type" ini dilakukan dengan memindahkan letak baris kode yang sebelumnya berada di bawah filter "Start Date" menjadi di atasnya.

```

<field name="working_date" position="after">
    <field name="company_id" />
    <field name="company_ids" />
    <field name="start_date" />
    <field name="end_date"/>
    <field name="end_date"/>
    <field name="report_type"/>
</field>

```

Gambar 3. 12 Cuplikan kode General Ledger Balance sebelum perubahan

```

<field name="working_date" position="after">
    <field name="company_id" />
    <field name="company_ids" />
    <field name="report_type"/>
    <field name="start_date" />
    <field name="end_date"/>
    <field name="end_date"/>
</field>

```

Gambar 3. 13 Cuplikan kode General Ledger Balance sebelum perubahan

Perubahan ini bersifat sederhana, tanpa penambahan logika baru, dan hanya mengatur ulang urutan tampilan filter agar sesuai dengan permintaan pengguna pada menu *Invoicing > Reporting > G/L Balance* di sistem Odoo.

KG Report - G/L Balance

Report Filters:

Companies	KMN Property ✕
Report Type	Open Items - Backdate
Date	
Profit Center Group	
Profit Center	
Cost Center	

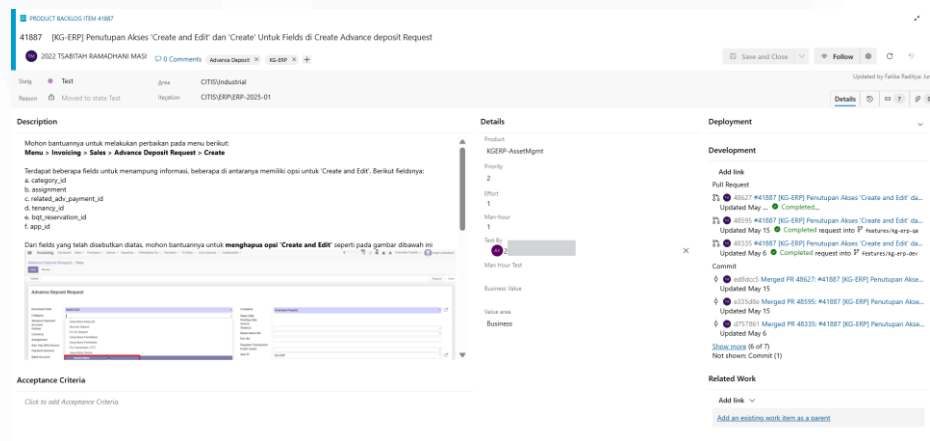
View Report **Cancel**

Gambar 3. 14 Hasil akhir perubahan Report General ledger Balance

Hasil dari perubahan tersebut seperti pada gambar 3.15, filter *report type* yang awal nya dibawah *date* menjadi diatas *date*.

3.2.1.1.3 Penutupan akses 'Create and Edit' Untuk Fields di Create Advance Deposit Request

backlog dengan judul "Penutupan Akses 'Create and Edit' dan 'Create' Untuk Fields di Create Advance deposit Request" ini meminta untuk melakukan penyesuaian *field* di menu *Invoicing > Sales > Advance Deposit Request* pada sistem Odoo



Gambar 3. 15 Penutupan Akses 'Create and Edit' dan 'Create' Untuk Fields di Create Advance deposit Request

Gambar 3.16 menunjukkan tujuan utama dari implementasi *backlog* ini yaitu untuk membatasi pengguna agar tidak dapat menambah *item* baru pada *field-field* tertentu. Pembatasan ini dilakukan karena penambahan *item* secara manual dapat mengganggu stabilitas dan integritas sistem secara keseluruhan. Dengan demikian, pengguna hanya dapat memilih dan menggunakan *item-item* yang sudah tersedia dan disiapkan dalam sistem. Hal ini dikarenakan setiap *item* memiliki keterkaitan dan hierarki dengan data atau modul lain yang sudah ada, sehingga konsistensi data dan kelancaran alur kerja sistem dapat terjaga.

Gambar 3. 16 Field-field yang akses 'Create and Edit' nya perlu ditutup

Beberapa *field* yang dihilangkan dalam menu *invoicing*, sub menu *Advanced Deposit Request* adalah 'category_id', 'assignment', 'related_adv_payment_id', 'tenancy_id', 'bqt_reservation_id', dan 'app_id'.

```
1 <field name="app_id" options="{
  'no_create_edit': True, 'no_create': True}"/>
```

Gambar 3. 17 Kode penambahan atribut 'no_create_edit'

Dalam pengerjaan *backlog* ini, perubahan diterapkan pada kode program dengan menambahkan atribut spesifik pada *field* yang relevan. Gambar 3.18 Merupakan contoh implementasi kode yang digunakan adalah sebagai berikut

opsi `options="{ 'no_create_edit': True, 'no_open': True}"` menjadi inti dari pembatasan akses data. Opsi `'no_create_edit': True` secara langsung menghilangkan kemampuan pengguna untuk membuat atau mengedit item baru melalui *field* tersebut, sehingga opsi penambahan atau perubahan pilihan tidak akan muncul. Selanjutnya, opsi `'no_create': True` mencegah pengguna untuk membuat *item* baru juga di *field* tersebut. Kombinasi opsi ini memperketat kontrol, memastikan bahwa pengguna hanya dapat memilih dari daftar yang tersedia tanpa bisa mengubah detailnya atau pun menambah *item* baru.

3.2.1.2 Backlog Yang Dikerjakan Pada Menu *Inventory*

3.2.1.2.1 *Readonly vendor in receiving from purchase order*

backlog ini bertujuan untuk meningkatkan konsistensi data pada modul *Inventory* sub menu *Inventory Transaction*. Fokus utama dari *backlog* ini adalah pada *field* *partner_id* yang terdapat dalam dokumen *stock.picking*, yaitu dokumen yang tercipta secara otomatis saat tombol "*Confirm*" (*button_approve*) pada *purchase.order* ditekan.

Sebelumnya, *field* *partner_id* ini masih dapat diedit setelah dokumen *stock.picking* terbentuk. Kondisi ini berpotensi menimbulkan ketidaksesuaian data jika ada perubahan pada *partner* setelah transaksi pembelian dikonfirmasi, yang bisa mengganggu konsistensi informasi antara *purchase order* dan *inventory transaction*. Oleh karena itu, tujuan dari *backlog* ini adalah untuk menjadikan *field* *partner_id* tersebut hanya bisa dibaca (*readonly*) setelah proses konfirmasi *purchase order*, sehingga data *partner* yang tercatat pada *stock.picking* akan selalu sesuai dengan data saat *purchase order* dikonfirmasi.

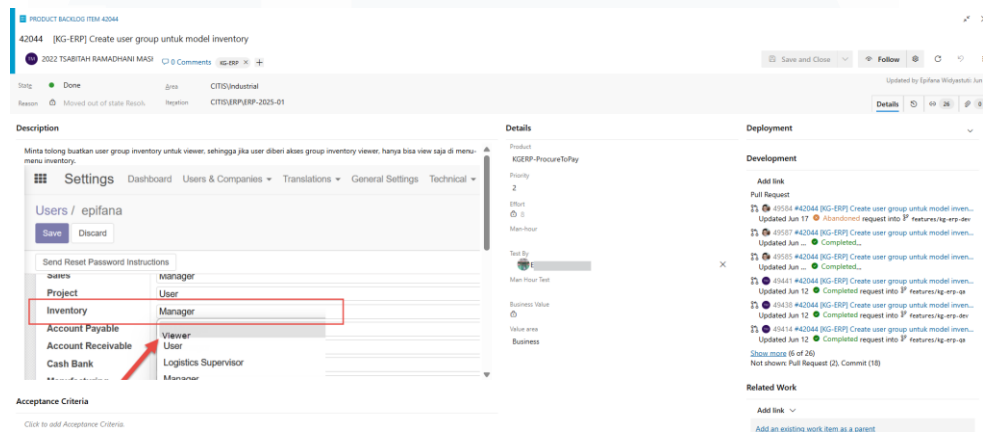
```
1 <field name="partner_id" position="replace">
2   <field name="partner_id" attrs="{ 'readonly': [('state', '!=', 'draft')] }"/>
3 </field>
```

Gambar 3. 18 Kode untuk mengubah *field* menjadi *read only*

Dalam menyelesaikan *backlog* ini, perubahan dilakukan dengan cara menambahkan atribut pada *field* yang dibutuhkan di dalam kode program. atribut *attrs="{ 'readonly': [('state', '!=', 'draft')] }* ditambahkan pada definisi *field* *partner_id* seperti pada gambar 3.19. Atribut ini berfungsi untuk mengatur kondisi kapan *field* tersebut akan menjadi *readonly*. Dalam kasus ini, *partner_id* akan menjadi hanya bisa dibaca (*readonly*) ketika *state* (status) dari dokumen *stock.picking* tidak sama dengan 'draft'. Artinya, begitu dokumen *stock.picking* berubah status dari 'draft' *field* *partner_id* tidak akan bisa lagi diedit oleh pengguna.

3.2.1.2.2 Create user group untuk model Inventory

backlog yang di kerjakan dalam menu *Inventory* yang berjudul "Create user group untuk model inventory". *Backlog* ini bertujuan untuk membuat grup baru bernama "viewer" di *inventory* seperti yang ditunjukkan pada gambar 3.20.



Gambar 3. 19 Backlog Create user group untuk model inventory

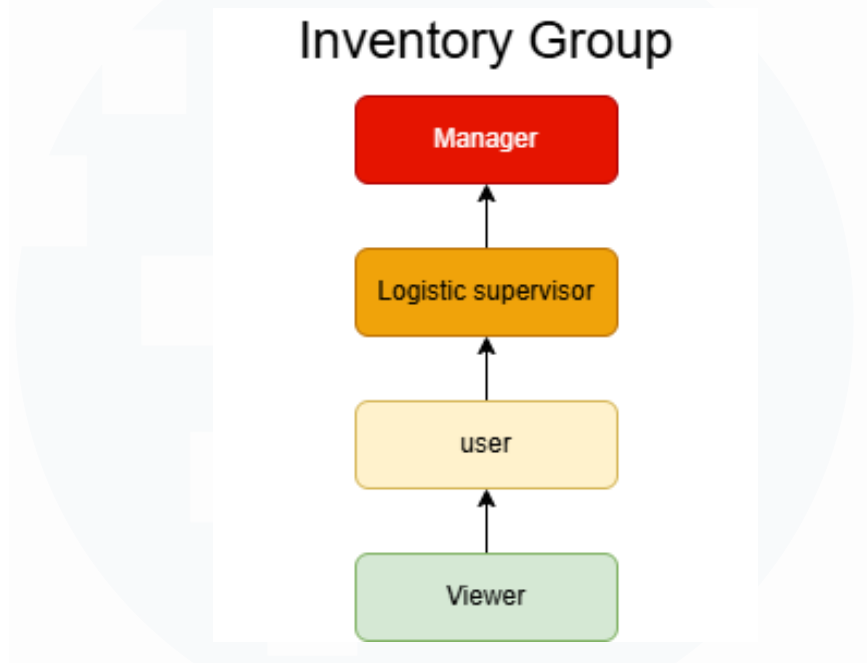
Tujuan dari pembuatan grup ini adalah untuk memberikan hak akses terbatas kepada pengguna yang hanya perlu melihat data tanpa melakukan perubahan apa pun. Grup *Viewer* hanya diberikan akses ke menu *Operations* dan *Master Data* di modul *Inventory*, itu pun tidak semua sub menu dapat diakses. Selain itu, seluruh tampilan dan data yang diakses oleh pengguna dalam grup ini disetting sebagai *read-only*, sehingga mereka tidak dapat melakukan perubahan, penambahan, atau penghapusan data.

```
<record id="group_stock_viewer" model="res.groups">
  <field name="name">Viewer</field>
  <field name="category_id" ref="base.module_category_warehouse_management"/>
  <field name="implied_ids" eval="[(4, ref('base.group_user'))]"/>
</record>
```

Gambar 3. 20 Kode Group Viewer

Dalam pengerjaan backlog ini, langkah pertama yang dilakukan adalah membuat grup baru bernama "*Viewer*" di dalam modul *Inventory*. Grup ini dibuat dengan mendefinisikan *record* baru pada model *res.groups* dan di beri nama grup sebagai *Viewer*, menempatkannya di bawah kategori

Warehouse Management, serta menyertakan `implied_ids` agar grup ini otomatis mewarisi hak akses dasar dari `base.group_user`. Gambar 3.21 merupakan potongan kode XML yang digunakan.



Gambar 3. 21 Hierarki Group Inventory

Gambar 3.22 menunjukkan struktur dari hierarki *inventory*. Sebelumnya, struktur grup yang ada terdiri dari *User*, *Logistic Supervisor*, dan *Manager*. Untuk menyusun hierarki dengan benar, grup *Viewer* ditempatkan berada di posisi paling bawah. Dengan demikian, *Viewer* hanya akan memiliki akses yang benar-benar terbatas dan tidak mewarisi hak akses dari grup-grup yang lebih tinggi.

```

1 access_stock_picking_viewer,kg_stock.stock_picking_viewer,stock.model_s
  tock_picking,kg_stock.group_stock_viewer,1,0,0,0
  
```

Gambar 3. 22 Kode pemberian hak akses read only kepada viewer

Setelah hierarki tersusun, selanjutnya mengatur hak akses terhadap model yang diperbolehkan untuk dilihat oleh grup *Viewer*. Pengaturan dilakukan melalui file CSV (`ir.model.access.csv`) dengan memberikan hak akses hanya untuk membaca (*read-only*), tanpa hak untuk *write*, *create*, atau *unlink*.

```

1 <record id="stock.menu_stock_root" model="ir.ui.menu">
2   <field name="groups_id" eval="[(4, ref('kg_stock.group_stock_viewer'))]" />
3 </record>

```

Gambar 3. 23 Kode inherit group viewer

Selanjutnya, agar grup *Viewer* hanya dapat melihat menu tertentu saja, *override* dilakukan pada beberapa menu melalui mekanisme inherit di Odoo. Gambar 3.24 merupakan contoh kode grup *Viewer* yang ditambahkan pada menu utama *Inventory* (stock.menu_stock_root).

Selain *menu root* nya dilakukan juga penyesuaian pada beberapa submenu yang relevan, jika sebuah menu tidak memiliki pengaturan grup secara langsung, maka aksesnya akan mengikuti grup dari menu induknya (parent menu). Oleh karena itu, perlu dipastikan semua akses menu yang diinginkan tersedia sesuai dengan kebutuhan grup *Viewer*.

Dalam proses pengerjaan *backlog* ini, terdapat beberapa kendala. Salah satunya adalah adanya miskomunikasi dengan tim analis terkait ruang lingkup menu yang boleh diakses oleh grup *Viewer*. Permasalahan ini berhasil diselesaikan setelah dilakukan klarifikasi dan diskusi secara langsung dengan manajer tim, sehingga kebutuhan dan ekspektasi bisa diselaraskan. Selain itu juga terdapat masalah teknis saat melakukan update pada menu, di mana perubahan yang dikerjakan tidak muncul meskipun kode sudah benar.

	☒ noupdate	123 id	123 create_uid	create_date	write_date	123 write_uid
1	[v]	68,408	1	2025-06-05 03:30:15.183	2025-06-09 04:27:29.501	1

Gambar 3. 24 Menu yang dituju memiliki property *no_update = True*

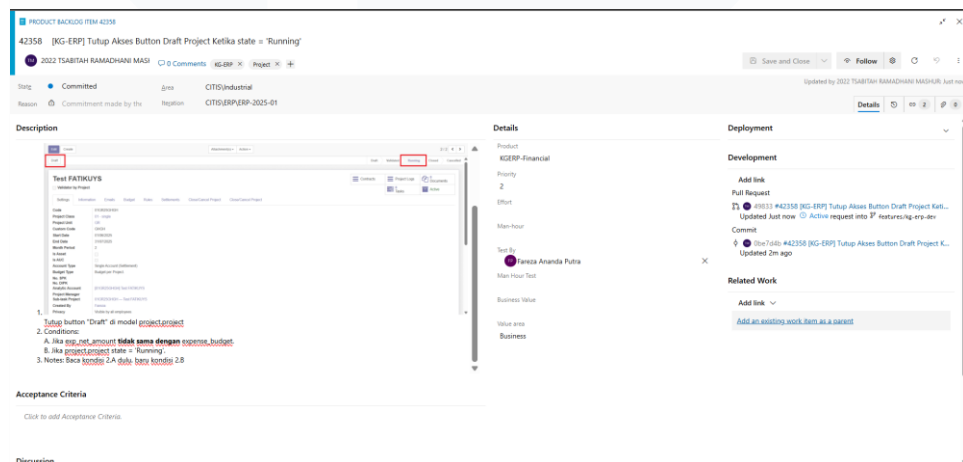
Setelah ditelusuri melalui database seperti pada gambar 3.25, terdapat properti *no_update=True* yang menyebabkan data pada menu tersebut tidak

bisa diubah melalui *update* module. Hal ini tidak terlihat langsung di kode Odoo, namun ternyata *status* *no_update* tersebut telah diubah langsung di database sebelumnya. Setelah menghapus flag *no_update* dari entri tersebut di database, barulah perubahan yang dilakukan dapat diterapkan dengan semestinya.

3.2.1.3 Backlog Yang Dikerjakan Pada Menu *Project*

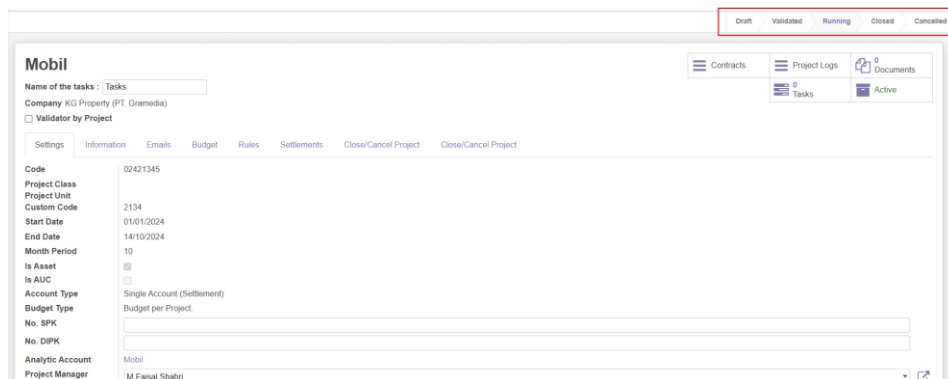
3.2.1.3.1 Menutup Akses *Button Draft Project* Ketika *State Running*

backlog dengan judul "*Tutup Akses Button Draft Project Ketika State = 'Running'*" bertujuan untuk menyesuaikan tampilan antarmuka sistem agar tombol "*Draft Project*" tidak muncul saat kondisi tertentu terpenuhi, sebagai bagian dari upaya menjaga konsistensi proses bisnis dan mencegah perubahan data yang tidak sesuai alur, seperti yang ditunjukkan pada gambar 3. 26.



Gambar 3. 25 Backlog Tutup Akses Button Draft Project Ketika state = 'Running'

Sebelumnya, tombol "*Draft Project*" sudah diatur agar disembunyikan ketika *state* berada dalam kondisi '*draft*'. Pada pengerjaan *backlog* ini, logika sistem diperluas, yakni tombol juga harus disembunyikan apabila *state* berada di kondisi '*running*' dan juga *field* *exp_net_amount* dan *expense budget* memiliki nilai yang tidak sama.



Gambar 3. 26 State pada project

terdapat lima status utama (*state*) yang digunakan dalam proses bisnis ini, yaitu: *draft*, *validated*, *running*, *closed*, dan *cancelled*. Kelima status tersebut mengatur alur dari suatu project, dan dapat dilihat pada gambar 3.27 yang menampilkan representasi visual dari status tersebut.

```
<button name="back2draft" string="Draft" type="object"
  attrs="{ 'invisible': ['|', ('state', '=', 'draft'),
    '&', ('state', '=', 'running'),
    ('exp_net_amount', '!=', 'expense_budget')] }"/>
```

Gambar 3. 27 Kode penambahan atribut untuk menyembunyikan tombol draft

Pada tahap awal pengerjaan *backlog* ini, implementasi logika untuk menyembunyikan tombol "*Draft*" dilakukan langsung melalui atribut *attrs* di XML dengan menggunakan kode seperti pada gambar 3.28

Namun, setelah dilakukan penelusuran lebih mendalam terhadap mekanisme evaluasi atribut *attrs* dalam Odoo, ditemukan bahwa *attrs* hanya dapat memproses ekspresi dengan pembandingan eksplisit terhadap nilai literal, seperti *field = True*, *field = False*, atau *field = value*. Atribut *attrs* tidak mendukung perbandingan antar dua field secara langsung contohnya *field = field*. Hal ini menyebabkan logika yang dibuat tidak dapat berjalan. Sebagai solusi, dibuat field baru bertipe *computed boolean* di Python.

```

show_back2draft = fields.Boolean(
    string='Show Back to Draft',
    compute='_compute_show_back2draft'
)

```

Gambar 3. 28 Inialisasi field `show_back2draft`

Berdasarkan gambar 3.29 Field baru ini diberi nama `show_back2draft`, yang berfungsi untuk menentukan apakah tombol "Draft" perlu ditampilkan atau disembunyikan, berdasarkan kombinasi nilai dari beberapa *field* lainnya.

```

@api.depends('state', 'exp_net_amount', 'expense_budget')
def _compute_show_back2draft(self):
    for record in self:
        record.show_back2draft = (
            record.state == 'draft' or
            (record.state == 'running' and
             record.exp_net_amount != record.expense_budget)
        )

```

Gambar 3. 29 Kode method `_compute_show_back2draft`

Nilai dari *field* `show_back2draft` dihitung melalui metode `_compute_show_back2draft`, dengan logika tombol hanya ditampilkan jika status project adalah 'draft', atau jika statusnya 'running' dan terdapat ketidaksesuaian antara nilai `exp_net_amount` dengan anggaran `expense_budget`.

```

<field name="show_back2draft" invisible="1"/>
<button name="back2draft" string="Draft" type="object"
    attrs="{ 'invisible': [('show_back2draft', '=', True)] }"/>

```

Gambar 3. 30 Kode inialisasi field `show_back2draft` di XML

Setelah itu, field `show_back2draft` diinisialisasi di dalam file XML sebagai *field invisible* agar nilainya tetap bisa diakses dan dimanfaatkan. Tombol "Draft" kemudian dikonfigurasi agar tidak muncul apabila nilai `show_back2draft` adalah *True*.

3.2.1.3.2 Menambah *Field Budget Ratio* pada Menu *Project*

Salah satu tugas yang dikerjakan pada menu *project* adalah melakukan penambahan *field* baru pada menu *project.project*, tepatnya pada *tab Budget*, khusus untuk jenis proyek bertipe 1 (*budget per project*) dan 3 (*budget per project detail*). Field yang ditambahkan diberi nama “*Budget Ratio*” dan diletakkan di atas field *Income* dan *Expense*.

Budget Ratio = $\text{income_budget} - \text{expense_budget} / (\text{expense_budget} / \text{income_budget} * 100\%)$

Gambar 3. 31 Rumus perhitungan budget ratio

Gambar 3.32 menunjukkan rumus dari *Field Budget Ratio*. *Field* ini berfungsi untuk menampilkan data berupa selisih antara anggaran pemasukan dan pengeluaran, serta persentase perbandingan di antara keduanya.

Tujuan *backlog* ini adalah agar pengguna dapat lebih mudah melihat perbandingan anggaran secara langsung tanpa perlu melakukan perhitungan manual, sehingga dapat mengambil keputusan dengan cepat. Dalam mengerjakan backlog ini, dilakukan beberapa penambahan pada sisi Python dan juga view XML nya.

```
budget_ratio = fields.Char(string="Budget Ratio", compute="_compute_budget_ratio")
```

Gambar 3. 32 Inialisasi field budget_ratio

Pada gambar 3.33, dibuat sebuah field baru bernama *budget_ratio* bertipe Char dengan label "Budget Ratio". *Field* ini menggunakan *parameter compute*, yang berarti nilainya tidak diisi secara manual, melainkan dihitung secara otomatis melalui fungsi *_compute_budget_ratio*.

```

@api.depends("income_budget", "expense_budget", "budget_type")
def _compute_budget_ratio(self):
    for rec in self:
        if rec.budget_type in [1, 3] and rec.income_budget and rec.expense_budget:
            difference = rec.income_budget - rec.expense_budget
            percentage = (rec.expense_budget / rec.income_budget) * 100 if rec.income_budget != 0
            rec.budget_ratio = f"{difference:,.0f} / {percentage:.2f}%"
        else:
            rec.budget_ratio = "N/A"

```

Gambar 3. 33 Fungsi `_compute_budget_ratio`

Fungsi `_compute_budget_ratio` pada gambar 3.34 dirancang untuk menghitung nilai *Budget Ratio* berdasarkan data *income* dan *expense* yang tersedia pada masing-masing proyek. Perhitungan ini hanya dijalankan apabila proyek memiliki *budget type* 1 atau 3 dan kedua nilai *income* maupun *expense* memiliki data yang valid. Di dalam fungsi, sistem akan menghitung selisih antara *income* dan *expense* terlebih dahulu, kemudian menghitung persentasenya berdasarkan rumus $(\text{expense_budget} / \text{income_budget}) * 100$. Hasil perhitungan akan ditampilkan dalam satu string gabungan yang memuat selisih nilai anggaran serta persentasenya. Jika data tidak memenuhi kriteria, maka *field* `budget_ratio` akan otomatis menampilkan nilai "N/A".

```

<group>
  <group colspan="4">
    <label string="Budget Ratio:" class="oe_bold"/>
    <field name="budget_ratio"
      nolabel="1"
      readonly="1"
      class="oe_bold"
      attrs="{ 'invisible': [('budget_type', 'not in', [1, 3])]}"/>
  </group>

  <group string="Expense" colspan="4">
    ...
  </group>

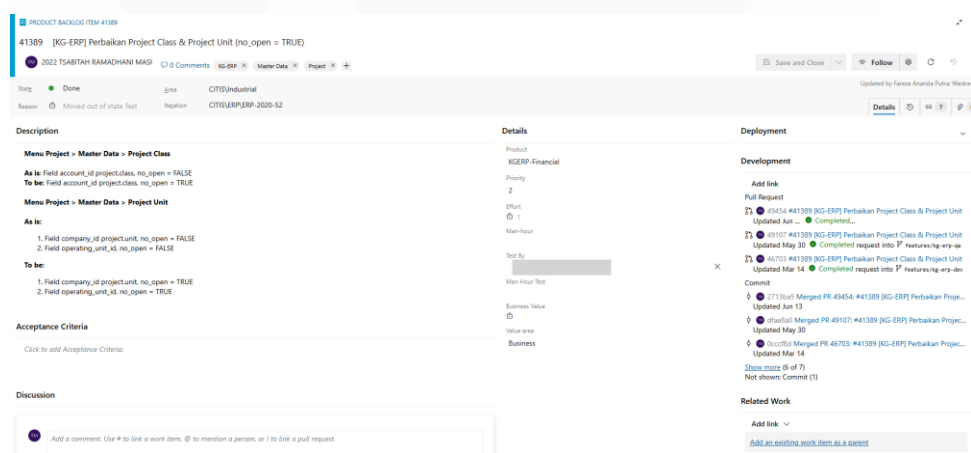
  <group string="Income" colspan="4">
    ...
  </group>
</group>

```

Gambar 3. 34 Peletakan *field* baru pada tampilan antarmuka pengguna

Pada bagian tampilan antarmuka pengguna atau XML view nya, *field* *budget_ratio* diletakkan tepat di atas *field* *Income* dan *Expense*. Ditambahkan juga label “Budget Ratio” sebagai judul. *Field* ini diset menjadi *readonly* agar tidak bisa diedit secara manual oleh user, karena seluruh nilainya dihasilkan melalui fungsi komputasi yang sudah dijelaskan sebelumnya. Selain itu, digunakan atribut kondisi tampilan agar *field* ini hanya muncul apabila proyek bertipe 1 atau 3. Dengan cara ini, pengguna hanya akan melihat *Budget Ratio* jika memang relevan, dan tidak akan terganggu dengan *field* yang tidak diperlukan.

3.2.1.3.3 Perbaikan *project class* dan *project unit* backlog dengan judul “Perbaikan Project Class & Project Unit (*no_open* = *TRUE*)” . ini meminta untuk melakukan perbaikan pada sub menu dari *project*, yaitu *project class* dan juga *project unit*.



Gambar 3. 35 Perbaikan Project Class & Project Unit (*no_open* = *TRUE*)

Pada gambar 3.36 backlog ini meminta perbaikan pada *Project Class*, field *account_id* diminta untuk diubah dengan menambahkan atribut *no_open=True*. Pengaturan ini bertujuan agar saat pengguna mengakses *field* tersebut, tidak ada opsi untuk membuka atau melihat detail dari data akun terkait secara langsung. begitu juga pada submenu *Project Unit*, di mana *field* *company_id* dan *operating_unit_id* juga diatur menggunakan properti *no_open=True*, agar tidak dapat dibuka oleh *user*. Perubahan yang

dilakukan pada *backlog* ini bersifat sederhana, yaitu dengan menambahkan atribut `no_open="1"`. Penyesuaian ini diterapkan pada *field* `account_id` di *submenu Project Class*, serta pada *field* `company_id` dan `operating_unit_id` di *submenu Project Unit*. Penambahan atribut tersebut dilakukan langsung pada definisi *field* di dalam view XML, tanpa mengubah struktur atau logika program lainnya.



Gambar 3. 36 Contoh ikon "open" pada *field*

Tujuan dari penggunaan `no_open="1"` adalah untuk menonaktifkan fitur *pop-up* yang biasanya muncul ketika pengguna mengklik ikon "open" di sebelah kanan *field* relasi.

```
<field name="account_id"
      options="{ 'no_create_edit': True, 'no_open': True}"
      attrs="{ 'readonly': [('account_type', '=', 'multiple')] }"/>
```

Gambar 3. 37 Kode penambahan atribut 'no_open=True' pada *field* `account_id`

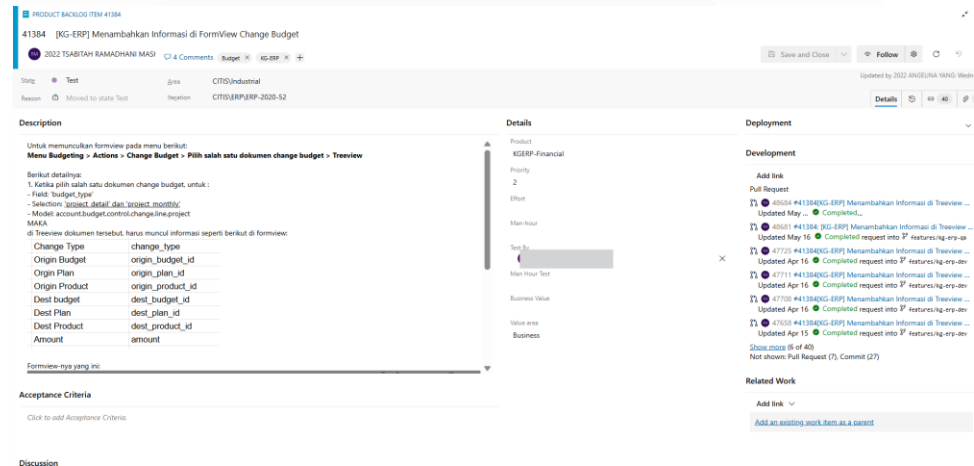
Gambar 3.38 merupakan contoh kode yang ditambahkan untuk mengatur properti tampilan *field* sesuai kebutuhan pada *submenu Project Class* dan *Project Unit*.

3.2.1.4 Backlog Yang Dikerjakan Pada Menu *Budgeting*

3.2.1.4.1 Menambahkan informasi di *treeview change budget*

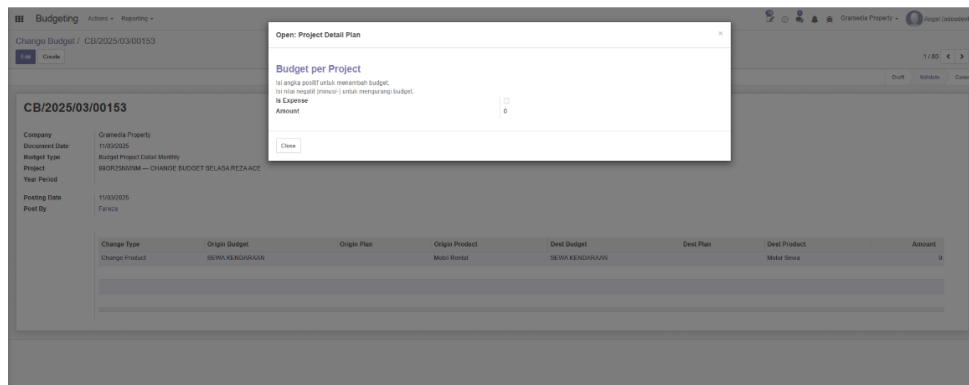
backlog dengan judul 'Menambahkan Informasi di *FormView Change Budget* '. yang dikerjakan pada menu *budgeting* diminta untuk melakukan

penyesuaian tampilan pada salah satu fitur di menu *Budgeting* pada sub menu *Change budget*, seperti yang ditunjukkan pada gambar 3. 39.



Gambar 3. 38 Backlog Menambahkan Informasi di FormView Change Budget

Penyesuaian ini bertujuan untuk menampilkan *form view* dari data detail perubahan anggaran ketika pengguna memilih salah satu dokumen pada tampilan *treeview*. ketika pengguna memilih dokumen *Change Budget* dan dengan tipe *budget project detail* atau *project monthly*, maka sistem harus menampilkan data dalam bentuk *form view*. Pada kondisi tersebut, *field-field* seperti *change_type*, *origin_budget_id*, *origin_plan_id*, *origin_product_id*, *dest_budget_id*, *dest_plan_id*, *dest_product_id*, serta *amount* ditampilkan dalam *form view*. *Backlog* ini bertujuan untuk memberikan informasi lebih detail untuk pengguna saat melakukan pengelolaan data perubahan anggaran proyek. Berikut merupakan kondisi sebelum *backlog* dikerjakan.



Gambar 3. 39 Change budget sebelum perbaikan dilakukan

Dapat dilihat dari gambar 3. 40 bahwa informasi yang ditampilkan masih sangat terbatas dan tidak memberikan detail yang cukup bagi pengguna. Untuk mendukung tampilan *form view* yang sesuai dengan jenis perubahan anggaran, terlebih dahulu ditambahkan sebuah *field* baru bernama *change_budget_type*.

```
change_budget_type = fields.Selection(related='budget_id.budget_type', readonly=True)
```

Gambar 3. 40 Kode inisialisasi field *change_budget_type*

Field *change_budget_type* yang di buat bertipe *Selection* dan berhubungan dengan field *budget_type* yang terdapat pada *budget_id*.

```
budget_id = fields.Many2one('account.budget.control.change')
```

Gambar 3. 41 Kode fields *budget_id*

Gambar 3.42 merupakan kode dari *budget_id*, dimana *budget_id* memiliki hubungan *many to one* ke model *account.budget.control.change*, field ini seperti penghubung antara model *account.budget.control.change.line.project* dengan parent nya *account.budget.control.change*.

```

budget_type = fields.Selection([
    ('opex', 'Budget Opex'),
    ('project', 'Budget Project'),
    ('project_detail', 'Budget Project Detail'),
    ('project_monthly', 'Budget Project Detail Monthly')
], required=True)

```

Gambar 3. 42 Kode fields budget_type

Gambar 3. 43 menunjukkan kode budget_type, dimana kode ini berada di dalam model `account.budget.control.change` *field* ini nantinya akan digunakan untuk menjadi penanda kondisi tampilan.

```

<group name="budget_project_detail"
    string="Budget Project Detail"
    attrs="{ 'invisible': [('change_budget_type', '!=', 'project_detail')] }">
    <field name="change_type"/>
    <field name="origin_budget_id"/>
    <field name="origin_product_id"/>
    <field name="dest_budget_id"
        context="{ 'form_view_ref': 'kg_project.project_budget_change_budget_form' }"/>
    <field name="dest_product_id"/>
    <field name="amount"/>
</group>

```

Gambar 3. 43 Kode group project_detail

Selanjutnya, menambahkan group yang dibuat dinamis berdasarkan nilai dari `change_budget_type`. Jika nilai `change_budget_type` adalah `project_detail`, maka akan ditampilkan *group* dengan *field* seperti `change_type`, `origin_budget_id`, `origin_product_id`, `dest_budget_id`, `dest_product_id`, dan `amount`.

```

<group name="budget_project_detail_monthly"
    string="Budget Project Detail Monthly"
    attrs="{ 'invisible': [('change_budget_type', '!=', 'project_monthly')] }">
    <field name="change_type"/>
    <field name="origin_budget_id"/>
    <field name="origin_plan_id"/>
    <field name="origin_product_id"/>
    <field name="dest_budget_id"
        context="{ 'form_view_ref': 'kg_project.project_budget_change_budget_form' }"/>
    <field name="dest_plan_id"/>
    <field name="dest_product_id"/>
    <field name="amount"/>
</group>

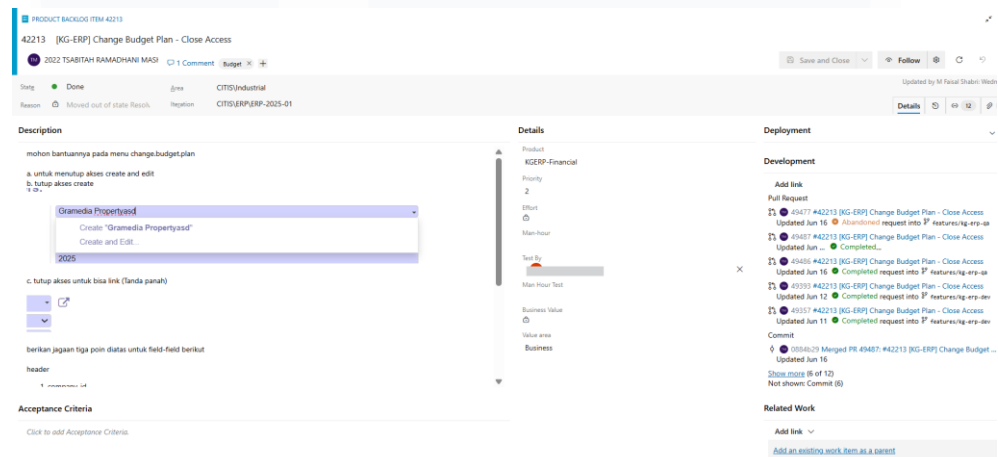
```

Gambar 3. 44 Kode group project_detail_monthly

Sedangkan jika nilainya adalah `project_monthly`, maka akan ditampilkan *group* berbeda yang memuat *field* tambahan, yaitu `origin_plan_id` dan `dest_plan_id`.

3.2.1.4.2 Menutup Akses *Change Budget Plan*

backlog yang dikerjakan pada menu *budgeting* ini dengan judul “*Change Budget Plan - Close Access* “. *Backlog* ini merupakan permintaan untuk melakukan pengaturan terhadap *field-field* tertentu pada model `change.budget.plan`, seperti yang ditunjukkan pada gambar 3. 46.



Gambar 3. 45 *Change Budget Plan - Close Access*

Terdapat 3 poin utama permintaan dalam *backlog* ini, Pertama, menyembunyikan tombol *Create* dan *Edit* untuk mencegah pengguna melakukan penambahan atau pengubahan data pada *field* yang telah ditentukan. Kedua, membatasi akses hanya pada fitur *Create* tanpa mengubah hak edit. Ketiga, menonaktifkan fungsi *link* (ikon tanda panah) yang biasanya digunakan untuk membuka relasi antar data pada *field* tertentu.

Sebagai bagian dari penyelesaian *backlog*, dilakukan pengaturan pada *field* tertentu dalam model `change.budget.plan` guna membatasi akses pengguna terhadap fungsi *create*, *edit*, dan *link*. Pembatasan ini diterapkan melalui

atribut *options* pada elemen *field* di XML, yang digunakan untuk mengontrol tampilan dan interaksi pengguna di antarmuka Odoo.

```
<field name="company_id"
      options="{ 'no_create_edit': True, 'no_create': True, 'no_open': True }"/>
<field name="operating_unit_id"
      options="{ 'no_create_edit': True, 'no_create': True, 'no_open': True }"/>
<field name="department_id"
      options="{ 'no_create_edit': True, 'no_create': True, 'no_open': True }"/>
<field name="account_id"
      options="{ 'no_create_edit': True, 'no_create': True, 'no_open': True }"/>
<field name="branch_id"
      options="{ 'no_create_edit': True, 'no_create': True, 'no_open': True }"/>
```

Gambar 3. 46 Kode penambahan atribut dengan 3 parameter

Atribut *options* yang diterapkan terdiri dari tiga parameter utama. Yang pertama, *'no_create_edit': True* berfungsi untuk menyembunyikan opsi *"Create and Edit..."* yang biasanya muncul ketika pengguna ingin menambahkan data baru langsung dari *dropdown*. Kedua, *'no_create': True* mencegah munculnya opsi *"Create"* saat pengguna mengetikkan nilai yang belum tersedia dalam database. Ketiga, *'no_open': True* menonaktifkan ikon panah yang biasanya digunakan untuk membuka tampilan detail dari *record* yang dipilih. Contoh kode dari ketiga pengaturan atribut *option* ini dapat dilihat secara keseluruhan pada gambar 3.47.

Dalam proses pengerjaan *backlog* ini, sempat ditemukan kendala saat melakukan penyesuaian akses pada *field-field* di bagian *Line Item* model *change.budget.plan*. Kendala tersebut disebabkan oleh tidak ditemukannya kode *form view* yang digunakan untuk menampilkan data dari model *change.budget.plan.lines*. Setelah ditelusuri lebih lanjut, ternyata *form view* untuk model tersebut memang belum tersedia dalam sistem. Oleh karena itu, sebelum dapat melanjutkan pengaturan akses seperti yang diminta, terlebih dahulu dibuat *form view* baru secara manual untuk model *change.budget.plan.lines*.

```

<!-- Change Budget Plan: Form View -->
<record id="view_change_budget_plan_line_form" model="ir.ui.view">
  <field name="name">change.budget.plan.line.form</field>
  <field name="model">change.budget.plan.lines</field>
  <field name="arch" type="xml">
    <form string="Change Budget Plan Line">
      <sheet>
        <group>
          ...
        </group>
      </sheet>
    </form>
  </field>
</record>

```

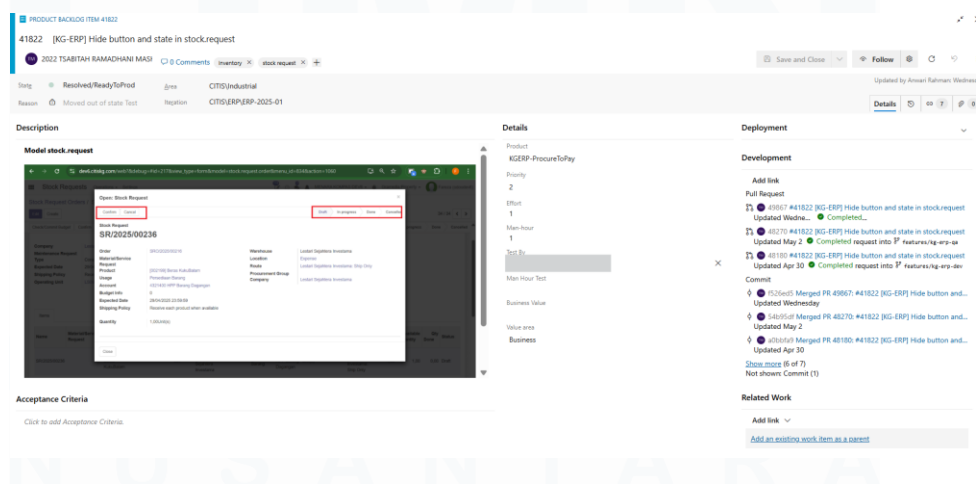
Gambar 3. 47 Kode form view change budget plan

Kode *form view* pada gambar 3. 48 bertujuan agar *field-field* pada bagian *Line Item* dapat ditampilkan dan dikontrol melalui antarmuka Odoo. Setelah *form view* berhasil dibuat, konfigurasi *options* digunakan untuk menonaktifkan fungsi *create*, *edit*, dan *link* pada masing-masing *field*.

3.2.1.5 Backlog Yang Dikerjakan Pada Menu *Stock Request*

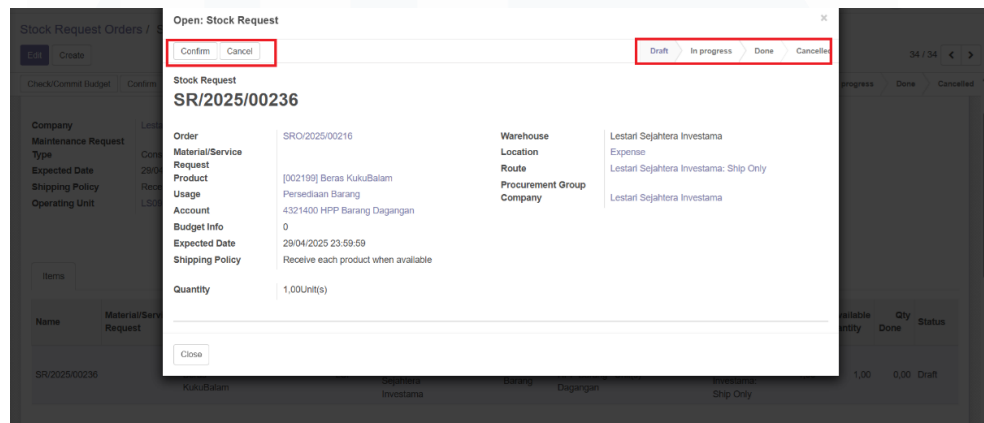
3.2.1.5.1 Menyembunyikan *button* dan *state* di model *stock.request*

backlog dengan judul “*Hide button and state in stock.request*” yang dikerjakan pada *menu stock request* meminta untuk melakukan penyesuaian pada tampilan antarmuka modul *Stock.Request*, seperti yang ditunjukkan pada gambar nomor 3. 49.



Gambar 3. 48 Backlog Hide button and state in stock.request

Tujuan dari backlog ini adalah menyederhanakan proses penggunaan sistem oleh pengguna. Penyesuaian tersebut berupa permintaan untuk menyembunyikan tombol `action_confirm` dan `action_cancel`, serta menyembunyikan informasi status atau *state* pada objek `stock.request`.



Gambar 3. 49 Tampilan model `stock.request`

Implementasi perubahan ini dilakukan melalui pengaturan pada View XML dengan cara melakukan inherit terhadap *field* aslinya dan menambahkan beberapa aturan visibilitas menggunakan `xpath`. Melalui metode ini, beberapa elemen pada tampilan seperti tombol dan *field* status dapat disesuaikan tanpa mengubah struktur *view* aslinya secara langsung.

```
<xpath expr="//button[@name='action_done']" position="attributes">
  <attribute name="invisible">1</attribute>
</xpath>
<xpath expr="//button[@name='action_cancel']" position="attributes">
  <attribute name="invisible">1</attribute>
</xpath>
<xpath expr="//button[@name='action_confirm']" position="attributes">
  <attribute name="invisible">1</attribute>
</xpath>
<xpath expr="//button[@name='action_draft']" position="attributes">
  <attribute name="invisible">1</attribute>
</xpath>
<xpath expr="//field[@name='state']" position="attributes">
  <attribute name="invisible">1</attribute>
</xpath>
```

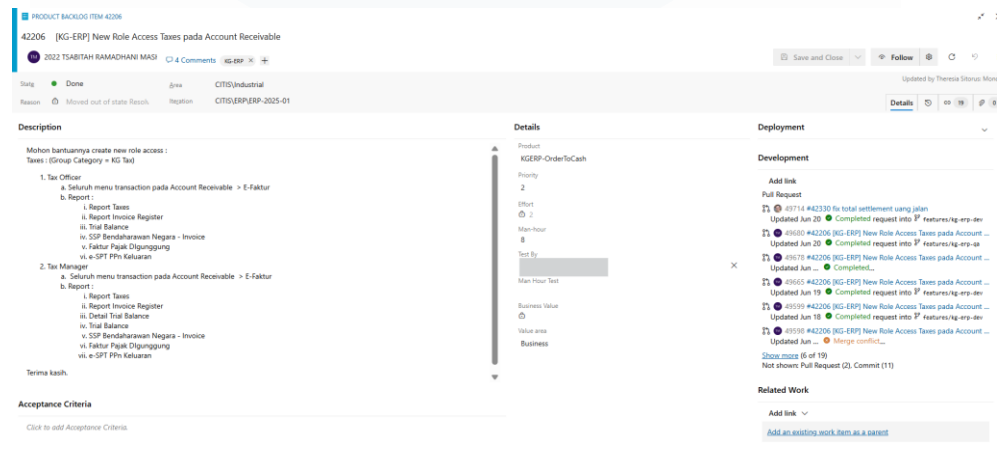
Gambar 3. 50 Kode inherit *field* untuk menambah atribut baru

Gambar 3.51 merupakan kode inherit *button* yang tujuannya untuk menambah atribut baru. Kode yang digunakan melakukan inherit *button* (*action_done*, *action_cancel*, *action_confirm*, dan *action_draft*) serta *field state*, lalu menambahkan atribut *invisible="1"* pada elemen-elemen tersebut bertujuan untuk menyembunyikan *action button* nya dan juga status dokumen dari tampilan pengguna.

3.2.1.6 Backlog Yang Dikerjakan Pada Menu *Account Receivable*

3.2.1.6.1 Membuat *Group Structure Taxes*

backlog dengan judul “*New Role Access Taxes pada Account Receivable* “. yang dikerjakan pada menu *account receivable* diminta untuk membuat struktur grup baru dengan nama *Taxes* yang terdiri dari dua grup akses, yaitu *group_tax_officer* dan *group_tax_manager*, seperti yang ditunjukkan pada gambar 3. 52.



Gambar 3. 51 New Role Access Taxes pada Account Receivable

Tujuan dari pembuatan grup ini adalah untuk mengatur hak akses pengguna berdasarkan peran yang berkaitan dengan pengelolaan perpajakan di sistem. *Grup Tax Officer* diberikan hak akses terhadap seluruh menu yang terdapat dalam modul *Transaction* dan *e-Faktur*, serta beberapa menu *reports* yang relevan. Sub menu *reports* yang dapat diakses oleh *Tax Officer* antara lain:

Report Taxes, Report Invoice Register, Trial Balance, SSP Bendaharawan Negara - Invoice, Faktur Pajak Digunggung, dan e-SPT PPN Keluaran.

Sementara itu, grup *Tax Manager* diberikan hak akses yang serupa namun dengan cakupan *reports* yang lebih lengkap. Selain dapat mengakses seluruh menu pada modul *Transaction* dan *e-Faktur*, pengguna dalam grup ini juga memiliki akses ke reports tambahan, seperti *Detail Trial Balance*, di samping laporan-laporan yang juga dapat diakses oleh *Tax Officer*.

```
<record model="ir.module.category" id="module_kg_tax">
  <field name="name">KG Taxes</field>
  <field name="sequence">5</field>
</record>
```

Gambar 3. 52 Kode kategori group KG Taxes

Dalam pengerjaan *backlog* ini dilakukan penambahan struktur grup baru untuk kebutuhan pengelolaan akses data perpajakan di sistem, dimulai dengan membuat kategori grup baru bernama *KG Taxes*. Kategori ini dibuat menggunakan model *ir.module.category* seperti ditunjukkan pada gambar 3. 53

```
<record model="res.groups" id="group_tax_officer">
  <field name="name">Tax Officer</field>
  <field name="category_id" ref="module_kg_tax"/>
  <field name="implied_ids" eval="[(4, ref('base.group_user'))]"/>
</record>
<record model="res.groups" id="group_tax_manager">
  <field name="name">Tax Manager</field>
  <field name="category_id" ref="module_kg_tax"/>
  <field name="implied_ids" eval="[(4, ref('kg_bank.group_tax_officer'))]"/>
</record>
```

Gambar 3. 53 Kode group tax officer dan tax manager

Setelah itu, dibuat dua grup pengguna baru dengan model *res.groups*, yaitu *group_tax_officer* dan *group_tax_manager*, yang dikategorikan ke dalam grup *KG Taxes*. *Grup Tax Officer* diberikan hak akses dasar sebagai pengguna sistem (*base.group_user*), sedangkan grup *Tax Manager* memiliki hak akses turunan dari *Tax Officer*.

```
access_account_invoice_tax_group_tax_officer,account_invoice_tax_group_tax_officer,
account.model_account_invoice_tax,kg_bank.group_tax_officer,1,1,1,1
```

Gambar 3. 54 Kode akses kontrol

Untuk melengkapi pengaturan hak akses yang telah dibuat sebelumnya melalui groups pada menu, dilakukan juga penambahan akses kontrol ke model tertentu menggunakan file `ir.model.access.csv`. Salah satu contohnya adalah pemberian akses terhadap model `account.invoice.tax` kepada grup *Tax Officer*. Kode ini dituliskan dalam format CSV.

```
<menuitem
  id="main_ar"
  name="Account Receivable"
  web_icon="kg_bank,static/description/icon_ar.png"
  groups="kg_bank.group_billing_ar_officer,kg_bank.group_tax_officer,kg_bank.group_tax_manager"
  sequence="80"/>
```

Gambar 3. 55 Kode penambahan atribut group

Selanjutnya, untuk membatasi hak akses ke menu tertentu berdasarkan grup yang telah dibuat, atribut *groups* ditambahkan pada masing-masing menu. Sebagai contoh, menu *Account Receivable* diatur agar hanya dapat diakses oleh pengguna yang tergabung dalam grup *Billing AR Officer*, *Tax Officer*, dan *Tax Manager*, seperti ditunjukkan pada gambar 3.56.

```
<menuitem
  id="menu_report_customer"
  name="KG Report Customer"
  sequence="13"
  parent="kg_bank.menu_ar_sub_report2"
  action="kg_bank.action_report_customer"
  groups="kg_bank.group_ar_officer"/>
```

Gambar 3. 56 Kode penambahan atribut group pada menu

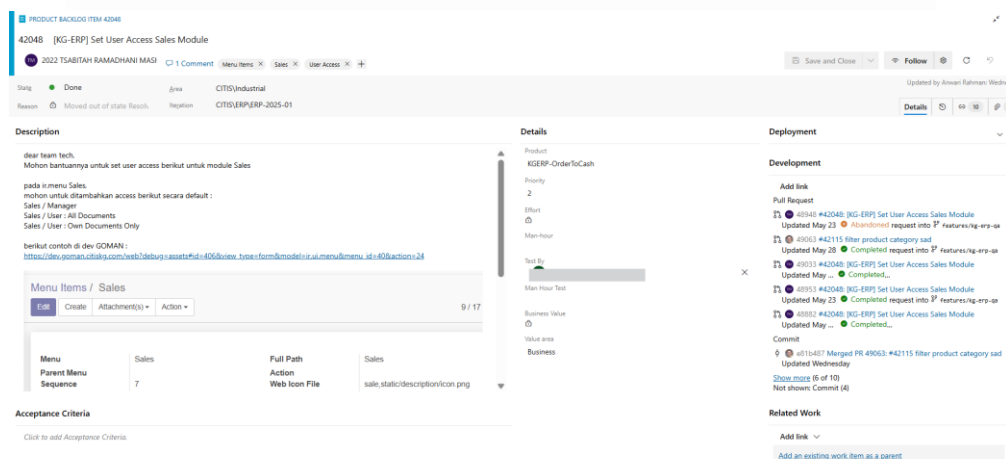
Namun, tidak semua menu sebelumnya memiliki *access right* secara eksplisit. Dalam kasus seperti ini, akses menu akan mengikuti pengaturan grup dari menu induknya. Oleh karena itu, untuk mencegah *Tax Officer* dan *Tax Manager* mengakses menu yang tidak sesuai dengan perannya,

dilakukan penyesuaian dengan cara menambahkan atribut groups secara eksplisit pada menu tersebut, dan hanya menetapkan grup yang memang diperbolehkan. Contohnya pada menu report customer, untuk mencegah group *Tax Officer* dan *Tax Manager* mengakses menu mengakses menu tersebut, ditambahkan atribut *group ar officer* saja seperti gambar 3.57. sehingga *group* selain *ar officer* tidak dapat mengakses menu ini.

3.2.1.7 Backlog Yang Dikerjakan Pada Menu Sales

3.2.1.7.1 Mengatur Akses user pada Modul Sales

Backlog dengan judul “*Set User Access Sales Module*” pada gambar 3.58 berisi permintaan untuk mengatur akses *user* secara default pada menu utama modul Sales. Tepatnya, pada menu Sales yang terdapat dalam model *ir.menu*, *backlog* ini meminta agar akses diberikan kepada tiga *Group user*, yaitu *Sales / Manager*, *Sales / User : All Documents*, dan *Sales / User : Own Documents Only*.



Gambar 3. 57 Set User Access Sales Module

Tujuannya adalah untuk memastikan bahwa ketiga peran pengguna tersebut dapat secara langsung mengakses menu *Sales* tanpa perlu pengaturan akses tambahan secara manual.

```

<menuitem id="sale.sale_menu_root"
          name="Sales"
          web_icon="sale,static/description/icon.png"
          active="True"
          sequence="7"
          groups="sales_team.group_sale_salesman_all_leads,
                sales_team.group_sale_salesman,
                sales_team.group_sale_manager"/>

```

Gambar 3. 58 Kode penambahan atribut *groups*

Untuk memenuhi *backlog* ini, dilakukan penyesuaian pada menuitem seperti pada gambar 3. 59, penyesuaian ini dengan cara menambahkan atribut *groups* yang berisi ketiga kelompok user yaitu *group_sale_saleman_all_leads*, *group_sale_salesman*, dan *group_sale_manager*.

3.3 Kendala yang Ditemukan

Selama proses pelaksanaan kegiatan magang, terdapat berbagai kendala dan tantangan yang muncul, baik dari sisi teknis maupun non-teknis. Berikut ini beberapa kendala yang ditemukan selama masa magang:

3.3.1 Ketidaktahuan terhadap *Tools* dan *Software* yang Digunakan

Salah satu kendala paling awal yang dihadapi adalah kurangnya pemahaman terhadap berbagai *software* dan *tools* yang digunakan dalam proses kerja. Sebagian besar perangkat lunak seperti PyCharm, Odoo, DBeaver, dan Azure DevOps (DevAzure) merupakan hal yang benar-benar baru dan belum pernah diperkenalkan secara langsung dalam pembelajaran di bangku perkuliahan. Hal ini menjadi tantangan tersendiri, mengingat proses kerja sangat bergantung pada penguasaan *tools* tersebut, baik untuk keperluan pengembangan kode program, manajemen basis data, pengelolaan proyek, hingga integrasi dan deployment aplikasi. Ketidaktahuan terhadap *tools* tersebut sempat

menimbulkan kebingungan dan menyebabkan proses adaptasi menjadi lebih lambat, khususnya pada minggu-minggu awal kegiatan magang.

3.3.2 Kurangnya Pemahaman Terhadap Alur Kerja *Git*

implementasi alur kerja *Git* di perusahaan sangat kompleks dibandingkan pemahaman dasar yang dimiliki sebelumnya. Penggunaan *branch* seperti *dev*, *qa*, dan *master*, serta aturan *merge* dan *pull request*, memunculkan kebingungan pada awalnya. Selain itu, pengelolaan konflik kode, pemahaman tentang *staging*, hingga penggunaan *command-line* untuk *Git* menjadi kendala tersendiri yang memerlukan adaptasi lagi.

3.3.3 Bahasa Pemrograman yang Tidak Familiar

Meskipun pernah mendapatkan pengenalan terhadap bahasa Python selama masa kuliah, namun penggunaannya lebih terfokus pada konteks *data analysis* dengan *tools* seperti Jupyter Notebook. Dalam proyek magang, Python digunakan sebagai bahasa *backend* dalam konteks pengembangan sistem, yang memiliki struktur, pendekatan, dan penggunaan *library* yang sangat berbeda. Di sisi lain, XML memang telah dipelajari sebelumnya saat mata kuliah *mobile applicataion development* pada masa perkuliahan, namun penggunaannya dalam konteks *framework* Odoo atau sistem *enterprise* memerlukan pemahaman mendalam terkait struktur dan relasi antar komponen. Keterbatasan pemahaman ini menjadi salah satu penghambat dalam memahami dan mengembangkan modul yang ada.

3.3.4 Miss Komunikasi dengan Analis Sistem

Dalam proses pengerjaan *backlog* sering kali terjadi perbedaan pemahaman atau penafsiran terhadap kebutuhan sistem yang dijelaskan oleh analis. Miss komunikasi ini menyebabkan beberapa implementasi fitur tidak sesuai dengan ekspektasi awal atau harus mengalami revisi yang cukup besar. Hal ini terjadi karena banyak faktor.

3.3.5 Ketidaktahuan terhadap Konsep ERP dan Istilah-Istilah Teknis di Dalamnya

Kendala lainnya adalah kurangnya pemahaman mengenai konsep ERP (*Enterprise Resource Planning*) dan berbagai istilah yang umum digunakan dalam sistem tersebut. Di dalam ERP terdapat banyak istilah teknis dan alur kerja yang cukup rumit dan saling berkaitan satu sama lain. Selama masa perkuliahan tidak pernah terdapat mata kuliah khusus yang membahas secara mendalam mengenai ERP maupun implementasinya di dunia nyata. Pemahaman yang diperoleh sebelumnya hanya terbatas pada pengantar teori manajemen sistem informasi, tanpa adanya praktik langsung terhadap sistem ERP seperti Odoo yang digunakan di perusahaan tempat magang.

Hal ini menjadi tantangan tersendiri, khususnya ketika mengikuti rapat-rapat di awal bulan pertama magang. Dalam beberapa pertemuan, pembahasan sudah cukup teknis dan langsung merujuk pada istilah-istilah ERP seperti *sales order*, *purchase requisition*, *stock picking*, *invoice*, dan sebagainya. Karena tidak memiliki dasar yang kuat, sulit untuk mengikuti jalannya diskusi, memahami konteks yang dibahas, maupun memberikan tanggapan yang relevan. Kondisi ini membuat proses pembelajaran di tahap awal terasa cukup berat dan memerlukan waktu ekstra untuk menyesuaikan diri.

3.4 Solusi atas Kendala yang Ditemukan

Untuk mengatasi berbagai kendala yang telah diuraikan sebelumnya, Berikut adalah beberapa solusi yang diterapkan untuk mengatasi kendala-kendala tersebut

3.4.1 Belajar Mandiri Melalui *Video*, Dokumentasi, dan Diskusi

Untuk mengatasi ketidaktahuan terhadap *software* dan *tools* yang digunakan, pertama tama yang dilakukan adalah dengan memanfaatkan berbagai sumber belajar yang telah disediakan oleh perusahaan. Beberapa *video* tutorial internal sangat membantu dalam memberikan gambaran umum mengenai cara kerja *tools* tersebut. Selain itu, dokumentasi dijadikan sebagai referensi dalam

memahami fungsi-fungsi tertentu. Jika masih mengalami kesulitan, diskusi langsung dengan rekan kerja atau senior.

3.4.2 Bertanya dan Berlatih Langsung Mengenai Alur *Git*

Menghadapi kompleksitas alur kerja *Git*, solusi yang diambil adalah dengan bertanya secara langsung kepada senior. Penjelasan yang diberikan mengenai fungsi masing-masing *branch* (*dev*, *qa*, *master*) serta cara kerja dalam siklus pengembangan sangat membantu dalam memperjelas pemahaman. Selain itu, latihan langsung melalui simulasi *commit*, *push*, dan *merge* dalam repositori internal perusahaan memberikan pengalaman konkret yang sangat berguna dalam memahami alur tersebut.

3.4.3 Mengeksplorasi Kode dan Mempelajari Logika Program Secara Mandiri

Keterbatasan dalam memahami bahasa pemrograman Python diatasi dengan eksplorasi langsung terhadap kode yang sudah ada. Melalui proses membaca kode, mengikuti alur logika program, serta memahami struktur fungsi dan kelas, pemahaman secara bertahap mulai terbentuk. Namun terkadang, ditemukan beberapa bagian kode yang kompleks. Dalam situasi seperti ini langkah yang diambil adalah dengan berkonsultasi langsung kepada senior atau anggota tim yang lebih berpengalaman.

3.4.4 Menjalinkan Komunikasi Intensif dan Kolaboratif dengan Analis

Untuk mengurangi miss komunikasi dengan analis sistem, dilakukan lebih banyak berdiskusi secara langsung. Setiap penjelasan dari analis ditanggapi dengan pertanyaan klarifikasi untuk memastikan bahwa kedua belah pihak memiliki pemahaman yang sama. Jika diskusi dua arah tidak membuahkan hasil yang memuaskan, maka langkah selanjutnya adalah mengadakan pertemuan bersama dengan atasan atau manajer sebagai penengah agar menemukan jalan tengah.

3.4.5 Bertanya Langsung kepada Analis Mengenai Alur ERP dan Istilahnya

Untuk mengatasi keterbatasan dalam memahami konsep ERP serta berbagai istilah teknis yang digunakan dalam proyek, solusi yang diterapkan adalah dengan aktif bertanya langsung kepada analis. Analis diposisikan sebagai pihak yang memiliki pemahaman paling mendalam terhadap kebutuhan dan proses bisnis perusahaan, sehingga menjadi sumber informasi yang sangat berharga. Dengan mengajukan pertanyaan-pertanyaan secara bertahap, terutama mengenai alur proses kerja, dependensi antar modul, serta fungsi dari setiap istilah yang digunakan, pemahaman secara perlahan mulai terbentuk.