BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama periode magang, mahasiswa ditempatkan pada Departemen *Enterprise Technology* yang berada di bawah naungan CITIS (*Corporate IT & IS*). Departemen ini dipimpin oleh Anwari Rahman selaku *Manager Enterprise Technology Department*. Dalam kegiatan magang, mahasiswa berperan sebagai *Junior Software Engineer* dan mendapatkan bimbingan langsung dari Herman Gusti Anugrah, yang menjabat sebagai *Software Engineer Lead*. Beliau berperan sebagai mentor sekaligus pengawas utama selama proses magang berlangsung. Setiap hasil pekerjaan dilaporkan kepada pembimbing tersebut sebelum akhirnya diteruskan kepada manajer departemen. Alur penugasan umumnya diberikan oleh manajer melalui supervisi kepada mahasiswa magang untuk memastikan koordinasi berjalan terarah.

Sistem penugasan dan distribusi proyek di lingkungan CITIS diawali dengan penyusunan daftar proyek oleh direktur CITIS. Selanjutnya, direktur melakukan koordinasi dengan *General Manager* dari divisi terkait guna menentukan penanggung jawab proyek. Setelah itu, *General Manager* menyampaikan proyek tersebut kepada manajer departemen yang akan menangani pengembangannya. Tahapan berikutnya, manajer departemen membahas dan membagi tugas kepada anggota tim dalam sebuah rapat koordinasi internal agar pelaksanaan proyek dapat berjalan sesuai rencana.

3.2 Tugas yang Dilakukan

Selama pelaksanaan magang di CITIS Kompas Gramedia dengan posisi sebagai *Junior Software Engineer*, kegiatan utama berfokus pada pengembangan sistem ERP melalui proses kustomisasi terhadap fitur yang telah ada maupun pembuatan fitur baru. Pengembangan dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan *framework* Odoo sebagai fondasi utama. Dalam pelaksanaannya, setiap tugas dikerjakan berdasarkan prinsip *Clean Code*, yaitu pendekatan penulisan kode yang menekankan keterbacaan, kemudahan pemeliharaan, serta efisiensi tanpa mengorbankan fungsi program. Prinsip ini penting diterapkan karena sebagian besar waktu pengembang dihabiskan untuk

memahami dan membaca kode yang sudah ada dibandingkan menulis kode baru [9]. Implementasi dari prinsip tersebut tercermin dalam berbagai aktivitas kustomisasi teknis yang dilakukan selama masa magang, antara lain sebagai berikut:

- **Modul Project**
  Melakukan pengembangan fitur pada proyek aset untuk menyesuaikan logika pengisian *branch_ids* agar tidak selalu bersifat wajib, khususnya pada proyek dengan kelas aset atau AUC. Selain itu, logika filter pada tab Settlement disesuaikan berdasarkan hak akses pengguna dan kelas proyek.

- **Modul Accounting**
  Mengembangkan fitur pada menu *Invoice Memo Batch* dengan menambahkan fungsi *Search Data*, validasi otomatis, penyesuaian tampilan form dan *tree view*, serta pembatasan akses berdasarkan peran pengguna. Penyesuaian filter *Cost Center* pada berbagai laporan keuangan juga dilakukan, serta penambahan fitur *Print Out Journal Entries* agar pengguna dapat mencetak jurnal transaksi secara langsung.

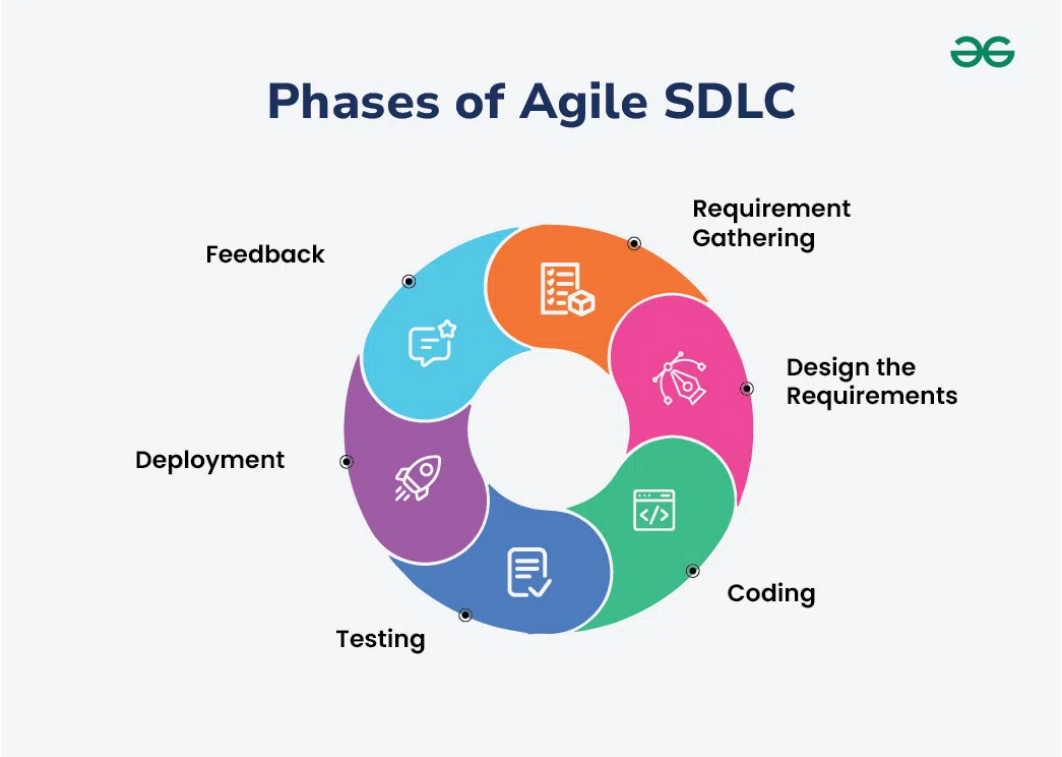
- **Modul Purchase**
  Menambahkan fitur *Billing Address* pada proses pembuatan RFQ atau Draft PO melalui *Purchase Request*, sehingga alamat penagihan dapat terisi otomatis pada dokumen *Purchase Order* dan meminimalkan kesalahan input manual.

- **Modul Inventory**
  Mengembangkan fitur cetak *Form Shipping* pada transaksi *Inventory Transaction* untuk menampilkan rincian barang dalam satu transaksi. Selain itu, menambahkan field *Procurement Lead Time* pada master produk yang digunakan untuk menghitung tanggal kebutuhan otomatis pada dokumen MSR, PR, dan PO.

3.2.1 Metode Pengembangan Sistem (SDLC Agile)

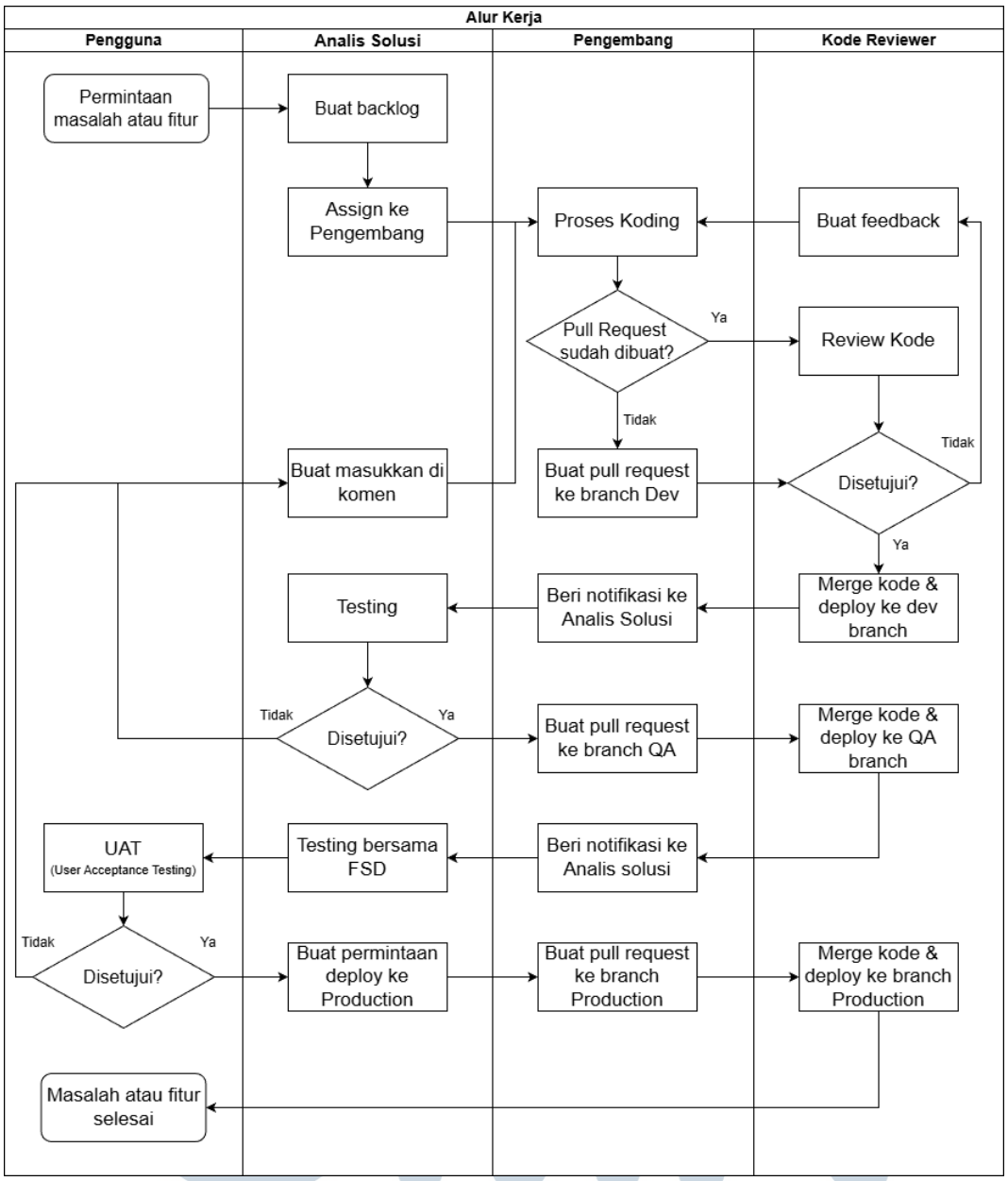
Selama pelaksanaan magang, proses pengembangan dan pemeliharaan sistem dilakukan dengan pendekatan *Agile*. Pendekatan ini menekankan pengembangan secara iteratif dan bertahap, sehingga perubahan kebutuhan dapat diakomodasi melalui siklus kerja yang lebih fleksibel. Secara umum, alur kerja dimulai dari penyusunan *backlog* dan pembagian tugas, dilanjutkan dengan proses

pengembangan, pengujian, serta evaluasi hasil sebelum diterapkan pada lingkungan yang digunakan pengguna. Alur pengembangan sistem yang digunakan selama magang dapat dilihat pada Gambar 3.1.



Gambar 3.1. SDLC Agile [1]

Dalam pelaksanaan tugas yang diterima, kegiatan magang dijalankan dengan mengikuti tahapan atau alur kerja sebagai berikut:



Gambar 3.2. Swimlane diagram alur proses kerja di lingkungan *Corporate IT & IS* Kompas Gramedia

Gambar 3.2 menampilkan alur pelaksanaan kegiatan magang secara keseluruhan. Penjelasan lebih lanjut mengenai alur tersebut dijabarkan pada bagian berikut.

1. **Pembuatan Backlog**

   (a) *Solution Analyst* berperan sebagai penghubung antara kebutuhan atau kendala dari *user* dengan tim *developer*. Proses ini dilakukan melalui

pembuatan backlog pada Azure DevOps Boards yang berfungsi sebagai sarana pendeskripsian kebutuhan agar dapat diterjemahkan dengan baik oleh tim pengembang.

(b) Untuk melaksanakan backlog yang telah dibuat, terdapat dua metode yang dapat digunakan oleh *developer*:

- *Solution Analyst* dapat menunjuk langsung *developer* yang dianggap paling sesuai berdasarkan bidang keahliannya dan hasil diskusi yang telah dilakukan sebelumnya.
- *Developer* juga dapat melakukan *self-assign* dengan mengambil backlog yang tersedia secara mandiri melalui Azure DevOps.

2. **Tahapan Pengembangan Kode**

(a) Setelah menerima atau memilih backlog, *developer* akan mulai mengimplementasikan solusi sesuai deskripsi dan panduan yang tercantum.

(b) Apabila terdapat bagian yang kurang jelas, *developer* dapat melakukan klarifikasi dengan *Solution Analyst* yang membuat backlog tersebut atau berdiskusi dengan sesama tim *developer*.

(c) Setelah kode selesai dibuat dan hasil pengujian di *local environment* sesuai dengan kebutuhan, *developer* akan melakukan *commit* serta *push* ke Azure DevOps, kemudian membuat *pull request* menuju branch *development* untuk direview oleh tim *reviewer*.

3. **Tahapan Review Kode**

(a) Setiap *pull request* yang diajukan akan melalui proses peninjauan oleh *reviewer* guna memastikan kesesuaian logika, efisiensi, serta penerapan prinsip *clean code*.

(b) Jika terdapat catatan atau perbaikan yang diminta oleh *reviewer*, maka *developer* wajib melakukan revisi hingga hasilnya sesuai dengan standar yang ditetapkan.

(c) Setelah disetujui, hasil pengembangan akan di-*deploy* ke *development server* untuk dilakukan pengujian oleh *Solution Analyst*.

4. **Proses Deployment**

(a) Setelah *Solution Analyst* menyelesaikan pengujian pada server *development*, ia akan mengajukan *request deployment* ke tahap QA untuk dilakukan pengujian bersama tim *Financial System Development* (FSD) serta pelaksanaan *User Acceptance Testing* (*UAT*) dengan pihak *user*.

(b) Apabila dari hasil pengujian ditemukan kebutuhan tambahan atau revisi, *Solution Analyst* akan menyampaikan umpan balik tersebut kepada *developer* melalui kolom komentar pada backlog terkait.

(c) Jika hasil pengujian telah sesuai dan mendapat persetujuan dari *user*, maka *Solution Analyst* akan mengajukan *request deployment* ke server *production* untuk diimplementasikan secara resmi.

3.3 Uraian Pelaksanaan Magang

Tabel 3.1 berikut menampilkan rangkuman kegiatan selama magang di Kompas Gramedia, dari minggu pertama hingga minggu ke-14. Ringkasan ini mencakup tugas yang dikerjakan, hal-hal yang dipelajari, serta hasil yang dicapai selama pelaksanaan magang.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu ke-	**Pekerjaan yang dilakukan**
1	Pngembangan fitur pada proyek aset untuk membuat field dari *Profit Center* tidak wajib diisi, untuk meningkatkan fleksibilitas pengelolaan proyek yang terkait dengan aset.
2	Masih melakukan pengembangan fitur proyek aset dengan melakukan diskusi terkait feedback dengan tim analis, terkait dengan fitur yang sudah di kembangkan.
3	Mengambil dan mengerjakan proyek pada menu *Invoice Memo Batch* di modul *Accounting*.
4	Melakukan pengembangan fitur pada menu *Invoice Memo Batch*, serta berdiskusi dengan tim analis dan manajer mengenai perubahan yang telah diunggah ke server *Develop*, agar disesuaikan dengan kebutuhan *User*.
	Lanjut ke halaman berikutnya

Tabel 3.1 – lanjutan dari halaman sebelumnya

Minggu ke-	Pekerjaan yang dilakukan
5	Mengembangkan fitur pada menu *Invoice Memo Batch* dengan melakukan penambahan *enhancement* pada beberapa *field*.
6	Melakukan *enhancement* pada menu *Invoice Memo Batch* dengan menambahkan akses serta melakukan beberapa perubahan pada *field-field* tertentu sesuai permintaan dari tim analis.
7	Melakukan *enhancement* pada beberapa menu di modul seperti *Accounting* dan *Purchase*, yang di dalamnya terdapat *Cost Center field*.
8	Melakukan *enhancement* pada *master data* untuk *Department (Cost Center)* sesuai dengan permintaan pengguna.
9	Melakukan *enhancement* pada laporan *mass matching (Pesanan Dana)* dengan mengubah nama laporan serta melakukan pemetaan *template* nama *approval* terhadap laporan tersebut.
10	Mengembangkan fitur *Print Out Journal Entries* untuk menampilkan rekaman *Journal Entries*.
11	Melanjutkan pengembangan fitur *Print Out Journal Entries* berdasarkan arahan dan saran dari senior serta masukan dari tim analis setelah dilakukan uji coba.
12	Melakukan *enhancement* pada modul *Purchase Request* dengan menambahkan *Billing Address field* pada *wizard* pembuatan *Request for Quotation (RFQ)* atau *Purchase Order*.
13	Mengembangkan fitur *printout shipping* pada model *stock picking* di modul *Inventory*.
14	Melakukan *enhancement* pada *master product* di menu *Inventory* dengan menambahkan *Procurement Lead Time field* serta mengatur alur *value schedule date* pada *Purchase Request (PR)*, *Material/Service Request (MSR)*, dan *Purchase Order*.

3.3.1 Pengembangan Fitur Proyek Aset

Pada proyek ini, dilakukan pengembangan fitur pada model `project.project` dengan tujuan untuk meningkatkan fleksibilitas pengelolaan proyek yang berkaitan dengan aset perusahaan. Sebelumnya, pada sistem yang berjalan (*as-is*), field `branch_ids` bersifat wajib diisi dalam semua kondisi. Selain itu, pada tab *Settlement*, field `branch_id` dan `department_id` pada setiap baris secara otomatis terfilter berdasarkan nilai `branch_ids` yang dipilih pada form proyek.

Melalui pengembangan ini (*to-be*), dilakukan perubahan logika agar sistem dapat menyesuaikan kondisi proyek berdasarkan jenis kelas proyek yang dipilih. Apabila field `project_class_id` memiliki atribut `is_asset` atau `is_AUC` dengan nilai `TRUE`, maka field `branch_ids` tidak lagi bersifat wajib diisi. Selain itu, jika pengguna tidak mengisi field `branch_ids`, maka pada tab *Settlement line*, filter untuk field `branch_id` dan `department_id` akan diambil berdasarkan *company project* dan disesuaikan dengan hak akses `allowed_department_id` yang dimiliki oleh pengguna pada model `res.users`.

Proses pengembangan dilakukan dengan melakukan penyesuaian pada struktur model Python serta logika tampilan XML. Perubahan kode difokuskan pada tiga *file* utama, yaitu `project_project.py`, `project_settlement.py`, dan `project_project_views.xml`. Koordinasi dilakukan dengan tim analis untuk memastikan perubahan yang dilakukan sesuai dengan kebutuhan pengguna, serta melakukan pengujian pada server *Develop* sebelum fitur diimplementasikan pada lingkungan *Production*.

Pada *file* `project_project.py`, dilakukan penambahan dua field baru, yaitu `is_auc_project_class` dan `is_asset_project_class`. Kedua field ini bersifat relasional terhadap atribut yang dimiliki oleh `project_class_id`, dan berfungsi untuk menentukan apakah proyek yang sedang dibuat merupakan proyek aset atau proyek *Asset Under Construction (AUC)*. Informasi ini kemudian menjadi acuan untuk menentukan sifat field `branch_ids` agar tidak selalu wajib diisi.

```
# File: project_project.py
project_class_id = fields.Many2one(
    'project.class',
    string='Project Class',
    track_visibility='onchange'
)

```

```
is_auc_project_class = fields.Boolean(
    related='project_class_id.is_auc',
    readonly=True
)
is_asset_project_class = fields.Boolean(
    related='project_class_id.is_asset',
    readonly=True
)
```

Kode 3.1: Penambahan field pada model `project.project`

Penyesuaian berikutnya dilakukan pada *file* `project_settlement.py`, dengan tujuan agar proses pemfilteran data pada tab *Settlement line* dapat menyesuaikan kondisi proyek. Jika proyek memiliki `branch_ids`, maka filter digunakan berdasarkan cabang tersebut. Sebaliknya, jika cabang tidak diisi karena proyek termasuk kategori aset, sistem akan menggunakan data `allowed_department_id` pengguna di model `res.users` untuk menentukan `branch_id` dan `department_id` yang boleh diakses.

```
# File: project_settlement.py
@api.depends('project_id', 'branch_id')
def _compute_domain_branch_id(self):
    for rec in self:
        company = rec.company_id.id
        domain_branch = [('company_id', '=', company)]
        domain_department = [('company_id', '=', company)]

        if rec.project_id.branch_ids:
            project_branch_ids = rec.project_id.branch_ids.ids
            domain_branch.append(('id', 'in', project_branch_ids))
            domain_department.append(('branch_id', 'in',
    project_branch_ids))
        elif rec.env.user.department_ids:
            allowed_branch_ids = rec.env.user.department_ids.
    filtered(
                lambda d: d.company_id.id == company
            ).mapped('branch_id').ids
            allowed_department_ids = rec.env.user.department_ids.
    filtered(
                lambda d: d.company_id.id == company
            ).ids
            domain_branch.append(('id', 'in', allowed_branch_ids))
            domain_department.append(('id', 'in',
    allowed_department_ids))
```

```
22
23          rec.domain_branch_id = json.dumps(domain_branch)
24          rec.domain_department_id = json.dumps(domain_department)
```

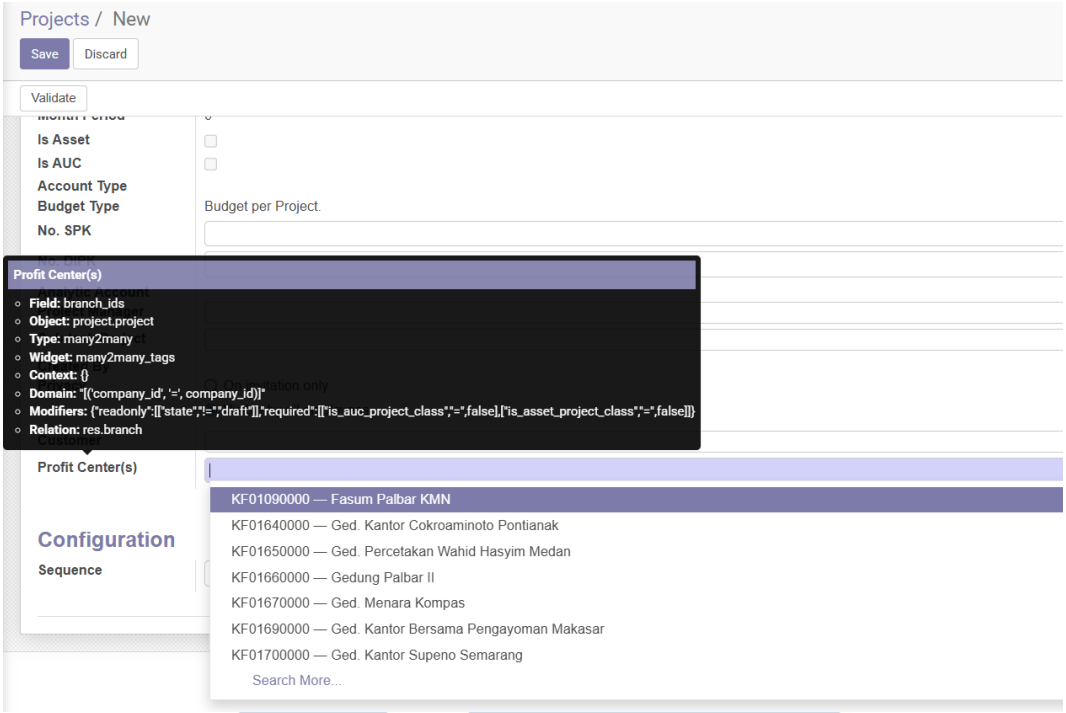
Kode 3.2: Logika filter pada model `project.settlement`

Terakhir, penyesuaian dilakukan pada *file* `project_project_views.xml` untuk mengatur tampilan form proyek. Pada *file* ini, field `branch_ids` tidak lagi memiliki atribut `required="1"` secara statik, namun dikendalikan oleh kondisi field `is_auc_project_class` dan `is_asset_project_class`. Dengan demikian, ketika salah satu kondisi bernilai `True`, sistem akan secara otomatis membuat field tersebut menjadi opsional.

```
1 <!-- File: project_project_views.xml -->
2 <field name="is_auc_project_class" invisible="1"/>
3 <field name="is_asset_project_class" invisible="1"/>
4 <field name="branch_ids"
5        widget="many2many_tags"
6        options="{'no_create': True}"
7        attrs="{'required': [('is_auc_project_class', '=', False),
8                             ('is_asset_project_class', '=', False)
    ]}"/>
```

Kode 3.3: Penyesuaian tampilan field `branch_ids` pada form proyek

Secara keseluruhan, perubahan pada tiga *file* tersebut berhasil meningkatkan fleksibilitas sistem dalam menangani proyek-proyek yang termasuk kategori aset atau AUC. Selain itu, logika yang diterapkan memastikan bahwa data cabang dan departemen dapat terfilter dengan benar berdasarkan hak akses pengguna, sehingga sistem menjadi lebih adaptif terhadap kebijakan perusahaan dan kebutuhan operasional. Untuk memberikan gambaran terkait hasil implementasi, berikut ditampilkan tampilan akhir dari salah satu fitur Proyek Aset setelah dilakukan penyesuaian terhadap field Profit Center.



Gambar 3.3. Tampilan field Profit Center setelah pengembangan

3.3.2 Pengembangan Fitur pada Menu *Invoice Memo Batch* (Modul Accounting)

Pada tahap pengembangan berikutnya, dilakukan melakukan modifikasi dan penambahan fitur pada menu *Invoice Memo Batch* di bawah modul *Accounting* (*Accounting →Adviser →Invoice Memo Batch*). Fitur ini digunakan untuk mengelola memo invoice yang berkaitan dengan transaksi keuangan perusahaan. Sebelum dilakukan pengembangan (*as-is*), menu *Invoice Memo Batch* hanya berfungsi untuk mencatat memo invoice secara manual tanpa adanya filter data, validasi otomatis. Kondisi tersebut membuat pengguna di bagian akuntansi kesulitan mencari data invoice secara spesifik dan melakukan kontrol data dengan efisien.

Melalui pengembangan yang dilakukan (*to-be*), ditambahkan beberapa fungsionalitas baru dan menyesuaikan logika bisnis pada modul *Accounting*. Langkah pertama adalah menyalin menu `account.invoice.memo.batch` dibwah menu *Accounting Mass Matching* agar dapat diakses langsung dari bagian *Adviser*. Selanjutnya, sistem diperluas dengan fitur pencarian data yang memiliki konsep serupa dengan menu *Accounting Mass Matching*. Pada proses pembuatan data (*Create*), ditambahkan label “Search Data” serta beberapa field baru seperti `Filter Account` dan `Field No`. Field `Filter Account` digunakan untuk memfilter akun

berdasarkan tipe transaksi (`payable` atau `receivable`), sedangkan field `Field No` digunakan untuk mengisi data memo invoice secara otomatis berdasarkan nomor faktur yang sesuai.

Selain itu, dilakukan beberapa penyesuaian tambahan berdasarkan masukan dari tim analis. Penyesuaian tersebut mencakup aktivasi field `trx_type` (yang sebelumnya *readonly*, perubahan status wajib pada field `memo_variant_id` dan `coa_memo_id`, serta penataan ulang kolom pada tampilan *tree view*. Beberapa kolom seperti `Memo Variant`, `COA Memo`, `SSP No`, `Tanggal SSP`, dan `Amount Invoice Due` diatur ulang posisinya serta disesuaikan atribut *readonly* dan *mandatory*-nya sesuai hak akses pengguna. Kolom seperti `inv_amount_total`, `ssp_ppn`, dan `adjustment` dihapus untuk menyederhanakan tampilan. Validasi baru juga ditambahkan, seperti memastikan nilai "Amount Memo" tidak melebihi "Amount Inv Residual" serta mewajibkan pengisian `coa_memo_id` jika modul yang digunakan bukan dari Accounting.

Dari sisi pengguna, sistem kini memiliki beberapa mekanisme perlindungan tambahan, seperti peringatan saat tombol `Approve/Post` ditekan tanpa mengisi `Memo Variant`, serta logika otomatis untuk mereset field tertentu jika terjadi perubahan pada `company_id`. Setelah data disimpan (*Save*), field `company_id` akan menjadi *readonly* untuk menjaga konsistensi data.

Dengan perubahan tersebut, *Invoice Memo Batch* terdapat fitur baru yaitu proses pencarian dan pengelolaan memo invoice menjadi lebih cepat, validasi data lebih ketat, dan akses pengguna lebih terkontrol. Selain itu, sistem juga mendukung otomatisasi penyesuaian tipe transaksi berdasarkan `trx_type` dan menampilkan pesan peringatan bila data tidak ditemukan. Hanya pengguna dengan peran tertentu, seperti *Accounting Manager* atau *General Manager*, yang dapat melakukan proses `Approve/Post`. Berikut cuplikan kode utama yang merepresentasikan perubahan tersebut.

```
@api.multi
def action_search_data(self):
    if not self.field_no:
        raise UserError(_('Please input Field No to search data!')
    )
    invoice_domain = [
        ('company_id', '=', self.company_id.id),
        ('number', 'in', self.field_no.split('\n')),
        ('state', 'in', ('open', 'paid'))
    ]
```

```
    if self.filter_account_id:
        invoice_domain.append(('account_id', '=', self.
    filter_account_id.id))
    invoices = self.env['account.invoice'].search(invoice_domain)
    if not invoices:
        raise UserError(_('No invoices found matching the criteria
    .'))
    ...
```

Kode 3.4: Fungsi pencarian invoice berdasarkan filter dan nomor faktur

```
@api.multi
def post_memo_batch(self):
    if not self.env.user.has_group('kg_bank.
    group_accounting_manager'):
        raise UserError(_('Access denied. Contact your Accounting
    Manager.'))
    ...

@api.onchange('cm_dm_amount')
def _onchange_cm_dm_amount(self):
    if self.cm_dm_amount > self.inv_residual:
        raise ValidationError(_('Amount Memo Can\'t be Greater
    Than Amount Inv Residual'))
```

Kode 3.5: Validasi akses pengguna dan batas nilai Amount Memo

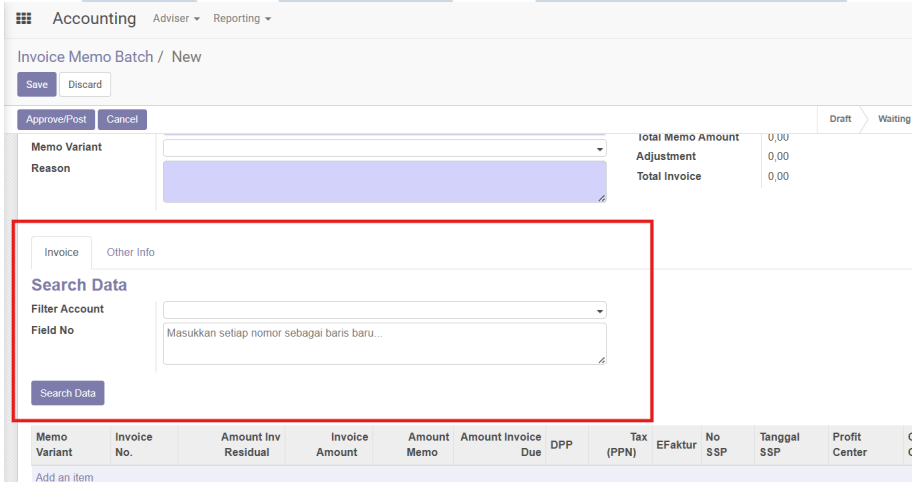
```
<page string="Search Data">
    <group>
        <field name="filter_account_id" domain="[('deprecated
    ','=',False)]"/>
        <field name="field_no" placeholder="Masukkan nomor invoice
    "/>
    </group>
    <button name="action_search_data" type="object" string="Search
     Data" class="btn-primary"/>
    ...
</page>
```

Kode 3.6: Penambahan section Search Data pada form Invoice Memo Batch

Setelah seluruh proses pengembangan dan pengujian diselesaikan, fitur tambahan pada modul *Invoice Memo Batch* berhasil diimplementasikan secara stabil pada server *Develop* dan siap untuk diterapkan pada server *Production*. Melalui pembaruan ini, sistem menjadi lebih efisien dalam pengelolaan memo

invoice. Pengguna kini dapat melakukan pencarian serta pemfilteran data dengan lebih mudah melalui mekanisme *Search Data*, sekaligus memperoleh validasi otomatis sesuai logika yang telah ditetapkan. Penerapan validasi nilai "Amount Memo", penyesuaian tampilan kolom pada *tree view*, serta modularisasi konfigurasi `trx_type` dan `coa_memo_id` menjadikan fitur ini lebih fleksibel digunakan lintas departemen dan sesuai dengan kebijakan perusahaan.

Untuk melengkapi penjelasan terkait pengembangan pada menu Invoice Memo Batch, berikut disajikan tampilan akhir fitur setelah penambahan fungsi Search Data, penyesuaian validasi, serta pembaruan struktur form dan tree view.



Gambar 3.4. Tampilan akhir fitur Invoice Memo Batch setelah pengembangan

3.3.3 Pengembangan dan Penyesuaian *Cost Center*

Pada tahap ini, dilakukan penyesuaian fitur pada beberapa menu laporan dan modul yang menggunakan filter *Cost Center*. Penyesuaian ini bertujuan untuk memisahkan antara data departemen yang digunakan sebagai unit kerja administratif dan yang berfungsi sebagai pusat biaya (*cost center*). Sebelum dilakukan pengembangan (*as-is*), seluruh data `hr.department` muncul tanpa pembeda sehingga membuat proses pelaporan menjadi kurang akurat.

Melalui pembaruan (*to-be*), sistem diperbaiki agar hanya menampilkan data departemen dengan tag tertentu. Untuk menu *Accounting→Reporting→GL Balance, Journal List, Monitoring PO, dan PO Outstanding*, filter `Cost Center` kini hanya menampilkan data `hr.department` dengan atribut `is_section = False`. Selain itu, berdasarkan masukan dari tim analis, dilakukan penyesuaian tambahan pada menu *Accounting→Adviser*, yaitu pada form *Journal Entries*

dan *Invoice Memo Batch*. Pada form tersebut, field `department_id` dan `cost_center_id` diperbarui agar hanya menampilkan data departemen yang memiliki atribut `is_cost_center = True`.

Penyesuaian kode dilakukan dengan menambahkan logika domain filter pada masing-masing wizard di file Python seperti `gl_balance.py`, `wizard_kg_report_journal_list.py`, `wizard_kg_report_monitoring_po.py`, dan `wizard_kg_report_po_outstanding.py`. Berikut contoh potongan kode implementasi domain filter yang digunakan.

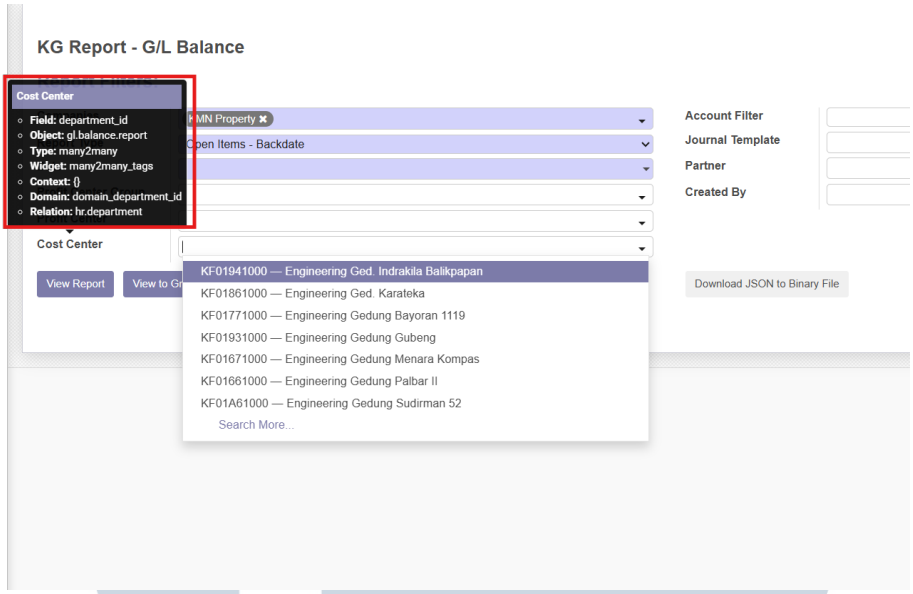
```
@api.depends('company_id', 'branch_id')
def _compute_domain_department_id(self):
    for rec in self:
        domain = [('is_section', '=', False)]
        if rec.company_id:
            domain.append(('company_id', '=', rec.company_id.id))
        if rec.branch_id:
            domain.append(('branch_id', 'in', rec.branch_id.ids))
        rec.domain_department_id = json.dumps(domain)
    ...
```

Kode 3.7: Penyesuaian domain filter untuk Cost Center

```
@api.depends('branch_ids', 'company_id')
def _compute_domain_department_ids(self):
    for rec in self:
        domain = [('is_cost_center', '=', True)]
        if rec.company_id:
            domain.append(('company_id', '=', rec.company_id.id))
        rec.domain_department_ids = json.dumps(domain)
    ...
```

Kode 3.8: Penerapan filter is_cost_center pada modul Accounting Adviser

Dengan pengembangan ini, sistem dapat membedakan antara departemen fungsional dan pusat biaya dengan lebih presisi. Pengguna kini hanya melihat data sesuai konteks laporan dan modul yang diakses, sehingga hasil pelaporan keuangan menjadi lebih akurat dan mudah dianalisis. Selain itu, penyesuaian ini juga menyederhanakan tampilan filter dalam laporan serta mengurangi risiko kesalahan pemilihan *Cost Center* pada proses pelaporan. Untuk memberikan gambaran mengenai hasil implementasi pada penyesuaian domain pada field Cost Center, berikut disajikan tampilan akhir fitur setelah perubahan diterapkan pada berbagai menu laporan dan form terkait.



Gambar 3.5. Tampilan field Cost Center pada salah satu menu *Reporting*

3.3.4 Pengembangan Laporan *Mass Matching* dan Fitur *Print Out Journal Entries*

Pada tahap ini, dilakukan pengembangan laporan dan fitur cetak yang berkaitan dengan proses akuntansi di modul *Accounting* dan *Cash Bank*. Pengembangan meliputi dua bagian utama, yaitu penyesuaian laporan *Mass Matching (Pesanan Dana)* dan pembuatan fitur *Print Out Journal Entries*.

Pada bagian pertama, dilakukan pengembangan laporan *Mass Matching* untuk menambahkan *wizard* baru sebelum proses pencetakan. *Wizard* ini menampilkan field opsional seperti `Direktur Keuangan`, `Direktur Kelompok`, `AP Supervisor`, dan `Kasir/AP Payment` yang akan digunakan sebagai kolom tanda tangan pada hasil laporan. Tombol cetak diubah dari label "Print" menjadi "Pesanan Dana (Payment Voucher)" dan dipindahkan ke dalam menu *Action*, sehingga pengguna dapat melengkapi data tanda tangan sebelum mencetak laporan. Selain itu, nama laporan juga diperbarui pada file MRT agar sesuai dengan perubahan label.

```
class MappingApprovalKGReportMassMatching(models.TransientModel):
    _name = 'wizard.mapping.approval.kg.report.mass.matching'
    _inherit = 'wizard.kg.report.base'

    direktur_keuangan = fields.Char(string='Dir. Keuangan', ...)
    direktur_kel = fields.Char(string='Dir. Kel', ...)
```

```
    ap_supervisor = fields.Char(string='AP Supervisor', ...)
    ap_payment = fields.Char(string='Kasir/AP Payment', ...)

    @api.multi
    def _get_data(self):
        active_ids = self.env.context.get('active_ids', False)
        mass_matching_ids = self.env['account.mass.matching'].
    browse(active_ids)
        params = {...}
        return self.env['report.report_mass_matching'].get_data(
            doc_ids=active_ids, records=mass_matching_ids, data=
    params)
```

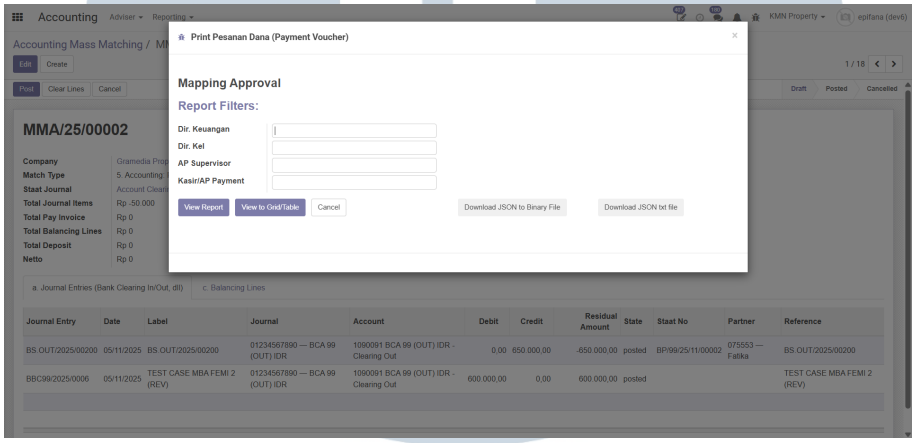
Kode 3.9: Wizard input tanda tangan sebelum proses cetak laporan

```
<record id="wizard_mapping_approval_report_mass_matching" model="
    ir.ui.view">
    <field name="name">Mapping Approval</field>
    <field name="model">wizard.mapping.approval.kg.report.mass.
    matching</field>
    <field name="inherit_id" ref="kg_report.base_wizard_view"/>
    <field name="arch" type="xml">
        <field name="working_date" position="after">
            <field name="direktur_keuangan"/>
            <field name="direktur_kel"/>
            <field name="ap_supervisor"/>
            <field name="ap_payment"/>
        </field>
    </field>
</record>

<record id="action_wizard_mapping_approval_mass_matching" model="
    ir.actions.act_window">
    <field name="name">Print Pesanan Dana (Payment Voucher)</field
    >
    <field name="res_model">wizard.mapping.approval.kg.report.mass
    .matching</field>
    <field name="view_mode">form</field>
    <field name="target">new</field>
</record>
...
```

Kode 3.10: Konfigurasi tampilan wizard dan aksi cetak di menu Action

Dengan penambahan *wizard* tersebut, proses pencetakan dokumen *Pesanan Dana* kini menjadi lebih fleksibel dan dinamis. Pengguna dapat menentukan langsung siapa yang akan dicantumkan dalam kolom tanda tangan tanpa perlu mengubah template laporan secara manual. Perubahan ini juga membantu mengurangi kesalahan input dan mempercepat proses administrasi di modul *Cash Bank*. Untuk memberikan gambaran mengenai hasil pengembangan pada laporan Mass Matching, berikut ditampilkan tampilan akhir wizard Pesanan Dana setelah penambahan input tanda tangan dan penyesuaian proses cetak.



Gambar 3.6. Tampilan akhir wizard Pesanan Dana (Payment Voucher) setelah pengembangan

Selain pengembangan pada laporan *Mass Matching (Pesanan Dana)*, dilakukan juga penambahan fitur *Print Out Journal Entries* pada menu *Invoicing→Adviser→Journal Entries*. Sebelum dilakukan pengembangan (*as-is*), sistem belum memiliki fitur pencetakan jurnal transaksi secara langsung, dan masukkan dari analis di proses development ketika pengguna memilih lebih dari satu transaksi, cetakan hanya menampilkan data pada entri terakhir.

Melalui pengembangan yang dilakukan (*to-be*), ditambahkan tombol `Print Journal Entries` yang memungkinkan pengguna mencetak laporan jurnal dalam format A4. Perubahan dilakukan pada model `account.move` dengan menambahkan *report action* baru untuk menghasilkan file laporan `report_journal_entries.mrt`. Selain itu, dilakukan penyesuaian pada tampilan laporan, yaitu memperlebar kolom `Debit`, memperkecil kolom `Amount Curr`, dan memastikan bahwa ketika lebih dari satu jurnal dipilih, setiap `account.move` akan dicetak pada halaman terpisah.

```
1 class KGReportJournalEntries(models.TransientModel):
```

```
    _inherit = 'report.kg_report_action_abstract'
    _name = 'report.report_journal_entries'

    @api.multi
    def _get_data(self, doc_ids, data, records, **kwargs):
        docs = []
        for rec in records:
            query = f"""
                SELECT concat(aa.code,' - ',aa.name) AS account,
                       aml.name AS label,
                       concat(rp.ref,' - ',rp.name) AS partner,
                       concat(rb.code,' - ',rb.name) AS
    profit_center,
                       concat(hr.code,' - ',hr.name) AS
    cost_center,
                       concat(aac.code,' - ',aac.name) AS project,
                       aml.debit, aml.credit, aml.amount_currency
                FROM account_move am
                LEFT JOIN account_move_line aml ON aml.move_id =
    am.id
                ...
                WHERE am.id = {rec.id}
            """
            rec.env.cr.execute(query)
            journal_entries = rec.env.cr.dictfetchall()
            docs.append({...})
        return {"data": docs}

    @api.model
    def _define_report_name(self, doc_ids, data, records, **kwargs
    ):
        report_name = 'report_journal_entries.mrt'
        return f'/kg_account/static/rpt/{report_name}'
```

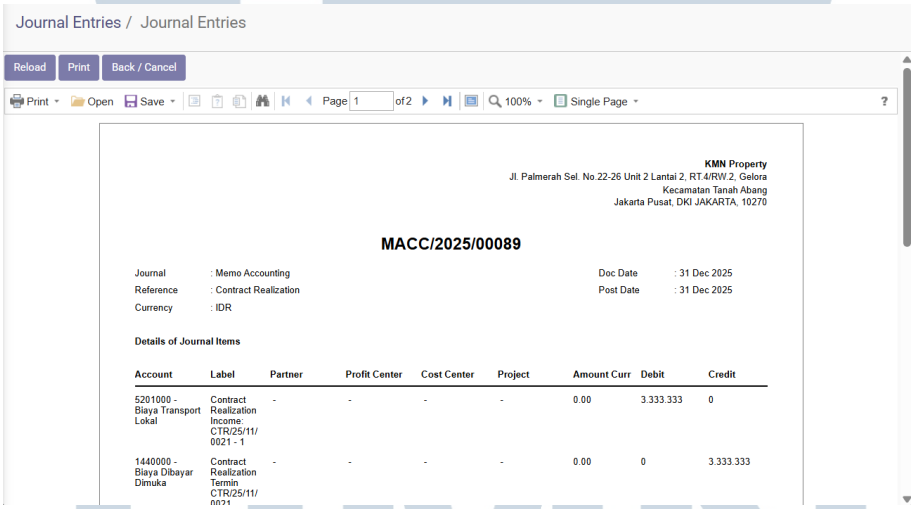
Kode 3.11: Logika pengambilan data untuk laporan Journal Entries

```
<report
    string="Journal Entries"
    id="action_print_journal_entries"
    model="account.move"
    report_type="mrt"
    name="report.report_journal_entries"
    file="report.report_journal_entries"
```

```
/>
```

Kode 3.12: Konfigurasi report action pada menu Journal Entries

Dengan pengembangan ini, pengguna kini dapat mencetak jurnal transaksi langsung dari sistem dengan format yang rapi dan sesuai kebutuhan akuntansi. Fitur *Print Out Journal Entries* juga mempercepat proses audit internal karena setiap transaksi dapat ditampilkan secara terpisah dalam satu hasil cetak. Selain itu, penataan ulang kolom memperbaiki keterbacaan laporan dan membantu pengguna memahami komponen debit dan kredit dengan lebih jelas. Untuk memberikan gambaran mengenai penjelasan dan hasil dari pengembangan, berikut disajikan contoh tampilan akhir laporan yang dihasilkan setelah penyesuaian dilakukan pada sistem.



Gambar 3.7. Tampilan laporan Print Out Journal Entries

3.3.5 Pengembangan Fitur *Billing Address* pada Menu Purchase Request

Pada tahap ini, dilakukan pengembangan pada menu *Purchase Request→Purchase Request Line→Action: Create RFQ/Draft PO*. Tujuan dari pengembangan ini adalah menambahkan opsi pengisian alamat penagihan (*Billing Address*) agar proses pembuatan dokumen *Purchase Order (PO)* menjadi lebih otomatis dan valid. Sebelum dilakukan pengembangan (*as-is*), proses pembuatan PO belum menyediakan field khusus untuk alamat penagihan, sehingga data tersebut harus diinput secara manual setelah PO dibuat.

Melalui pengembangan yang dilakukan (*to-be*), ditambahkan field baru bernama `bill_address_id` pada wizard `make_purchase_order`. Field ini

bersumber dari model `shipping.address` dengan domain yang disesuaikan seperti field `bill_address_id` pada `purchase.order`. Field bersifat *mandatory*, disembunyikan opsi *create/edit*, dan secara default mengambil data dari konfigurasi perusahaan (`res_company_setting.default_billing_address`) berdasarkan perusahaan yang aktif pada form `purchase.request`. Posisi field ditempatkan di bawah `currency_id`.

Ketika pengguna menekan tombol `Make Purchase Order`, sistem secara otomatis akan mengisi kolom alamat penagihan (`bill_address_id`) pada `purchase.order` beserta detail alamat seperti `street`, `city`, dan `zip`, sesuai data yang dipilih pada wizard. Jika tidak ada data yang diisi oleh pengguna, sistem akan mengambil nilai default dari konfigurasi perusahaan.

```
class PurchaseRequestLineMakePurchaseOrder(models.TransientModel):
    _inherit = 'purchase.request.line.make.purchase.order'

    bill_address_id = fields.Many2one(
        'shipping.address', string='Billing Address',
        domain="[('line_ids.shipping_type', 'in', ('billing','
    shipping_and_billing'))]",
        required=True
    )

    @api.model
    def prepare_purchase_order(self, picking_type, group_id,
    company, origin, purchase_type=False):
        ...
        if self.order_type == 'confirm_po' and self.
    bill_address_id:
            po_bill_address_id = self.bill_address_id.id
            po_bill_street = self.bill_address_id.street
            po_bill_city = self.bill_address_id.city
            po_bill_country_id = self.bill_address_id.country_id.
    id
            address_line = self.bill_address_id.line_ids.filtered(
                lambda l: l.shipping_type in ['billing','
    shipping_and_billing'] and
                          (not l.company_ids or self.company_id in
     l.company_ids)
            )[:1]
            if address_line:
                po_bill_contact_id = address_line.
    contact_person_id.id
```

```
        elif self.company_id.get_setting('default_billing_address'):
            ....

    @api.model
    def default_get(self, fields):
        ...
        if company_ids:
            res['company_id'] = company_ids[0]
            company = self.env['res.company'].browse(company_ids[0])
            default_billing_address = company.get_setting('default_billing_address')
            if default_billing_address:
                res['bill_address_id'] = default_billing_address[0]
```

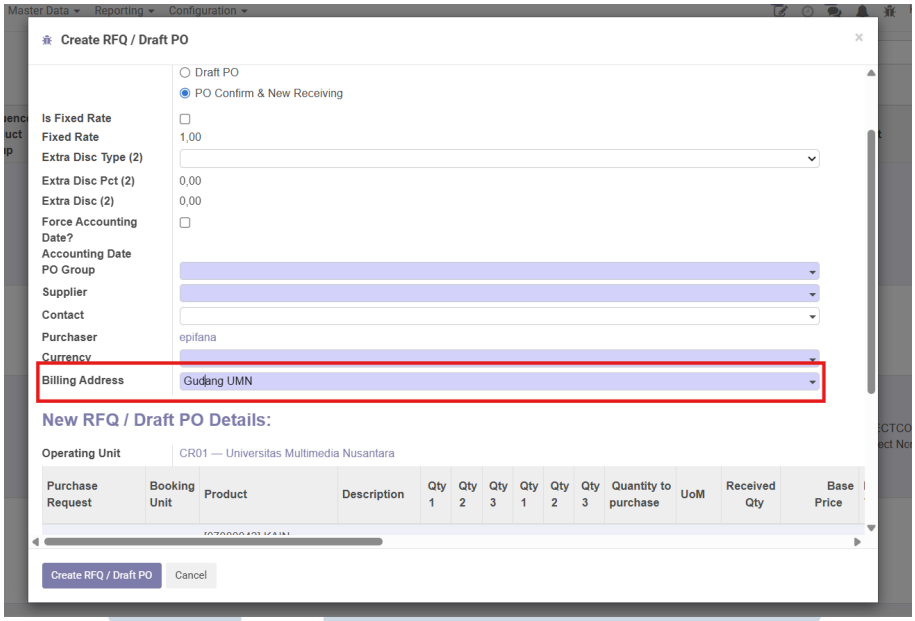
Kode 3.13: Penambahan field Billing Address dan pengisian otomatis Purchase Order

```
<field name="currency_id" required="1" options="{'no_create': True}"/>
<field name="bill_address_id"
       string="Billing Address"
       required="1"
       attrs="{'invisible':[('order_type','!=','confirm_po')]}"
       domain="[('line_ids.shipping_type','in',('billing','shipping_and_billing'))]"
       options="{'no_create': True, 'no_open': True}"/>
...
```

Kode 3.14: Penambahan field Billing Address pada wizard Create RFQ atau Draft PO

Dengan penambahan field *Billing Address* ini, proses pembuatan *Purchase Order* menjadi lebih efisien karena alamat penagihan langsung terisi sesuai data perusahaan atau input pengguna. Fitur ini juga mengurangi potensi kesalahan input manual dan meningkatkan konsistensi data antar dokumen pembelian.

Berikut hasil implementasi pada fitur Billing Address di menu Purchase Request, yaitu tampilan akhir wizard pembuatan Purchase Order dari menu Purchase Request



Gambar 3.8. Tampilan akhir wizard pembuatan Purchase Order dengan fitur Billing Address

3.3.6 Pengembangan Fitur *Print Out Shipping* pada Modul Inventory

Pada tahap ini, dilakukan pengembangan fitur pada menu *Inventory*→*Operations*→*Inventory Transaction* (model `stock.picking`). Pengembangan dilakukan untuk menambahkan opsi cetak `Form Shipping` yang berfungsi menampilkan daftar produk yang diproses dalam satu transaksi *stock picking*. Sebelum dilakukan pengembangan (*as-is*), pengguna belum memiliki akses untuk mencetak daftar produk dari form *Inventory Transaction*, meskipun fitur serupa telah tersedia pada form *Stock Request Order* dengan nama `Form Issuing`.

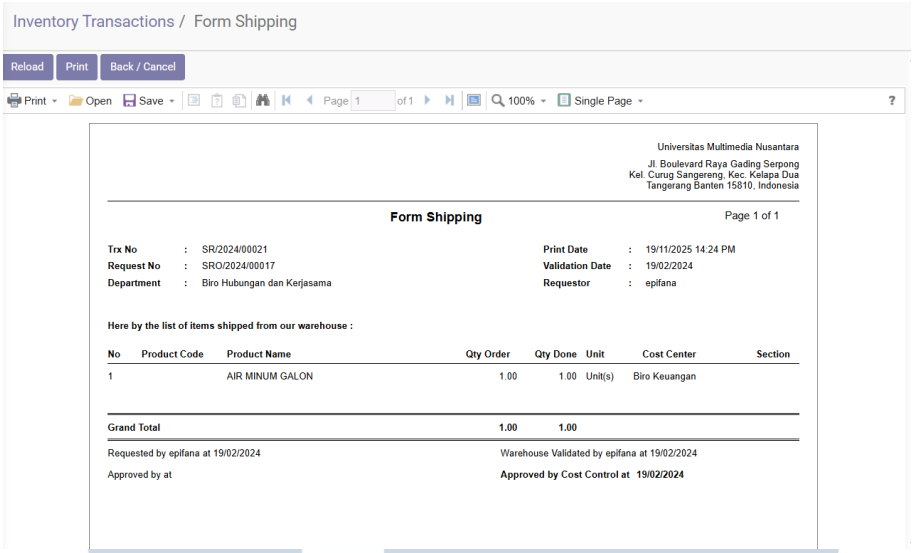
Melalui pengembangan yang dilakukan (*to-be*), ditambahkan tombol `Print: Form Shipping` pada form *Inventory Transaction*. Saat tombol tersebut diklik, sistem akan menjalankan laporan baru dengan template `form_shipping.mrt` yang menampilkan rincian produk, seperti kode, nama produk, kuantitas, satuan, lokasi, serta departemen pemohon barang. Struktur laporan ini mengacu pada format *Form Issuing* yang ada pada modul *Stock Request*, dengan beberapa penyesuaian informasi tambahan seperti *warehouse name*, *cost center*, dan *requestor*.

```
class KGReportFormShipping(models.TransientModel):
    _inherit = 'report.kg_report_action_abstract'
    _name = 'report.report_form_shipping'
```

```
4
5       @api.multi
6       def _get_data(self, doc_ids, data, records, **kwargs):
7           form_shipping = []
8           for rec in records:
9               query = f"""
10                  SELECT sp.origin, sp.name, pt.default_code AS
    product_code,
11                      pt.name AS product_name, sm.product_uom_qty
    ,
12                      pu.name AS uom_name, hr.name AS department,
13                      rp.name AS requestor, loc.locations AS
    location
14                  FROM stock_move sm
15                  LEFT JOIN stock_picking sp ON sp.id = sm.
    picking_id
16                  ...
17                  WHERE sp.id = {rec.id} AND spt.code = 'outgoing'
18              """
19              rec.env.cr.execute(query)
20              form_shipping = rec.env.cr.dictfetchall()
21              config = {"company_name": rec.company_id.name, ...}
22              data = {"config": config, "data": form_shipping}
23          return data
24
25      @api.model
26      def _define_report_name(self, doc_ids, data, records, **kwargs
    ):
27          report_name = 'form_shipping.mrt'
28          return f'/kg_stock/static/rpt/{report_name}'
```

Kode 3.15: Logika pengambilan data untuk laporan Form Shipping

Dengan adanya pengembangan ini, pengguna dapat mencetak dokumen *Form Shipping* secara langsung dari setiap transaksi inventori. Fitur ini memudahkan proses dokumentasi pengiriman barang dan memastikan informasi pengeluaran barang terdokumentasi dengan rapi. Laporan yang dihasilkan juga membantu bagian logistik dan departemen terkait dalam melakukan verifikasi barang keluar sesuai permintaan. Berikut tampilan akhir dan output Form Shipping yang dihasilkan.



Gambar 3.9. Tampilan laporan Form Shipping

3.3.7 Pengembangan Fitur *Procurement Lead Time* pada Master Produk

Pada tahap ini, dilakukan pengembangan fitur pada master produk yang terdapat pada beberapa menu, yaitu *Sales→Catalog→Product*, *Purchases→Master Data→Product*, serta *Inventory→Master Data→Product*. Sebelum dilakukan pengembangan (*as-is*), sistem belum memiliki data waktu pengadaan produk (*Procurement Lead Time*) yang dapat mempengaruhi jadwal kebutuhan di dokumen permintaan atau pembelian. Akibatnya, pengguna harus mengatur tanggal pengiriman secara manual pada setiap transaksi.

Melalui pengembangan yang dilakukan (*to-be*), ditambahkan field baru `procurement_lead_time` bertipe *integer* pada tab *Inventory* di form produk. Field ini bersifat opsional (*non-mandatory*) dan digunakan untuk menghitung tanggal kebutuhan (*schedule date*) secara otomatis pada dokumen *Material Service Request (MSR)*, *Purchase Request (PR)*, dan *Purchase Order (PO)*. Nilai default pada field `schedule date` di setiap dokumen dihitung berdasarkan tanggal hari ini ditambah dengan durasi *lead time* dari produk terkait (`today + procurement_lead_time`). Perubahan ini berlaku hanya untuk transaksi yang tidak memiliki referensi dari dokumen sebelumnya, sehingga ketika data diturunkan dari MSR ke PR atau dari PR ke PO, jadwal mengikuti dokumen asal.

```
1 class ProductTemplate(models.Model):
2     _inherit = 'product.template'
3
```

```
    procurement_lead_time = fields.Integer(
        string='Procurement Lead Time', default=0,
        help='Jumlah hari estimasi pengadaan barang')
```

Kode 3.16: Penambahan field Procurement Lead Time pada produk

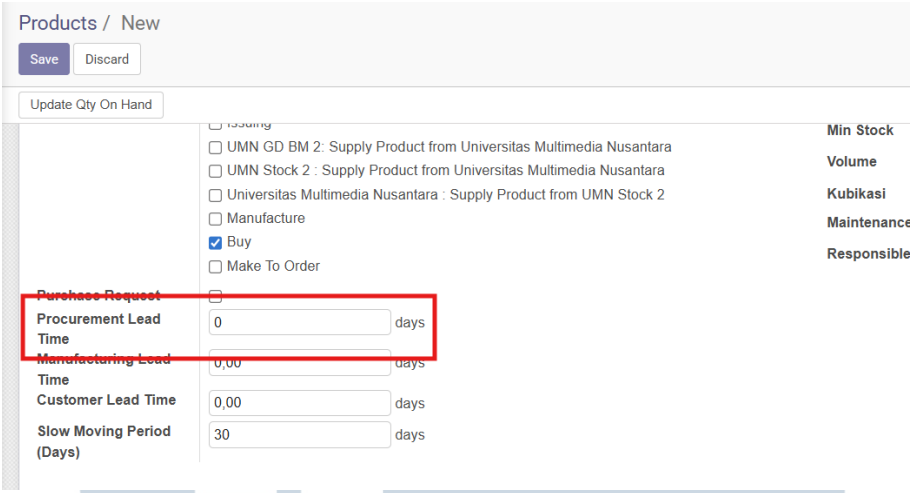
```
# Contoh pada model Material Service Request Line
@api.onchange('product_id')
def _onchange_product(self):
    for rec in self:
        lead_time = rec.product_id.procurement_lead_time
        base_date = fields.Date.context_today(rec)
        if isinstance(base_date, str):
            base_date = fields.Date.from_string(base_date)
        suggested_date = base_date + timedelta(days=lead_time)
        rec.date_required = suggested_date
    ...
```

Kode 3.17: Perhitungan otomatis schedule date pada MSR, PR, dan PO

```
<xpath expr="//field[@name='route_ids']" position="after">
    <div>
        <label for="procurement_lead_time"/>
        <field name="procurement_lead_time" class="oe_inline"/>
    </div>
</xpath>
```

Kode 3.18: Penambahan field Procurement Lead Time pada tab Inventory

Dengan adanya pengembangan ini, sistem kini dapat secara otomatis menetapkan tanggal kebutuhan (*schedule date*) sesuai *lead time* produk. Fitur ini membantu tim pengadaan dalam mengestimasi waktu yang diperlukan sejak pengajuan permintaan hingga pengiriman barang diterima. Pengaturan `Procurement Lead Time` juga meningkatkan konsistensi jadwal antar transaksi serta meminimalkan risiko keterlambatan dalam proses pengadaan. Berikut tampilan akhir field Procurement Lead Time pada *master* data Produk.



Gambar 3.10. Tampilan field Procurement Lead Time pada master produk

3.4 Kendala dan Solusi yang Ditemukan

Selama menjalani kegiatan magang di Kompas Gramedia, ditemui beberapa kendala meskipun sudah memiliki pengalaman dari periode magang sebelumnya. Kendala tersebut muncul baik dari segi teknis, seperti penggunaan teknologi dan bahasa pemrograman, maupun dari sisi non-teknis, seperti komunikasi dan penyesuaian terhadap lingkungan kerja. Namun, dari setiap kendala yang ditemui, upaya dilakukan untuk mencari solusi agar pekerjaan tetap dapat diselesaikan dengan baik.

3.4.1 Kendala yang Dihadapi

Beberapa kendala yang dihadapi selama magang antara lain sebagai berikut:

1. **Masih perlu beradaptasi dengan framework Odoo.** Walaupun sudah pernah mengenal Odoo pada magang sebelumnya, kesulitan masih dialami karena masih ada fitur atau modul yang belum dipahami dengan baik.
2. **Pemahaman proses bisnis ERP.** Setiap divisi memiliki alur kerja yang berbeda, sehingga diperlukan waktu untuk memahami keterkaitan antara sistem dan kegiatan bisnis yang berjalan.
3. **Hambatan komunikasi dengan tim analis.** Terkadang terdapat perbedaan pemahaman mengenai tugas (*backlog*) yang diberikan, sehingga perlu dilakukan diskusi tambahan dengan *Solution Analyst*.

4. **Kurangnya penguasaan bahasa Python.** Dalam beberapa tugas pengembangan, pemahaman perlu diperdalam mengenai sintaks dan logika Python agar dapat menyesuaikan dengan struktur kode pada Odoo.

3.4.2 Solusi yang Ditemukan

Untuk mengatasi kendala-kendala tersebut, dilakukan beberapa langkah sebagai berikut:

1. **Belajar mandiri melalui dokumentasi resmi.** dengan memanfaatkan dokumentasi Odoo dan sumber belajar daring untuk memahami cara kerja framework serta mencoba membuat eksperimen kecil agar lebih terbiasa.
2. **Diskusi dengan tim pengembang dan analis.** Dengan berdiskusi langsung dengan *Senior Developer* dan *Solution Analyst*, penjelasan tambahan didapatkan yang dapat membantu dalam memahami tugas.
3. **Meningkatkan komunikasi rutin.** dengan berusaha lebih aktif dalam komunikasi antar tim melalui rapat mingguan maupun platform seperti *Microsoft Teams* agar tidak terjadi salah tafsir terkait pekerjaan.
4. **Memperdalam kemampuan Python.** dengan mengikuti *tutorial online* dan mempraktikkannya langsung dengan membuat program sederhana yang relevan dengan Odoo.

Dengan langkah-langkah tersebut, berbagai hambatan yang muncul selama pelaksanaan magang dapat diatasi, dan tugas yang diselesaikan dapat berupa hasil yang lebih baik dibandingkan periode sebelumnya.