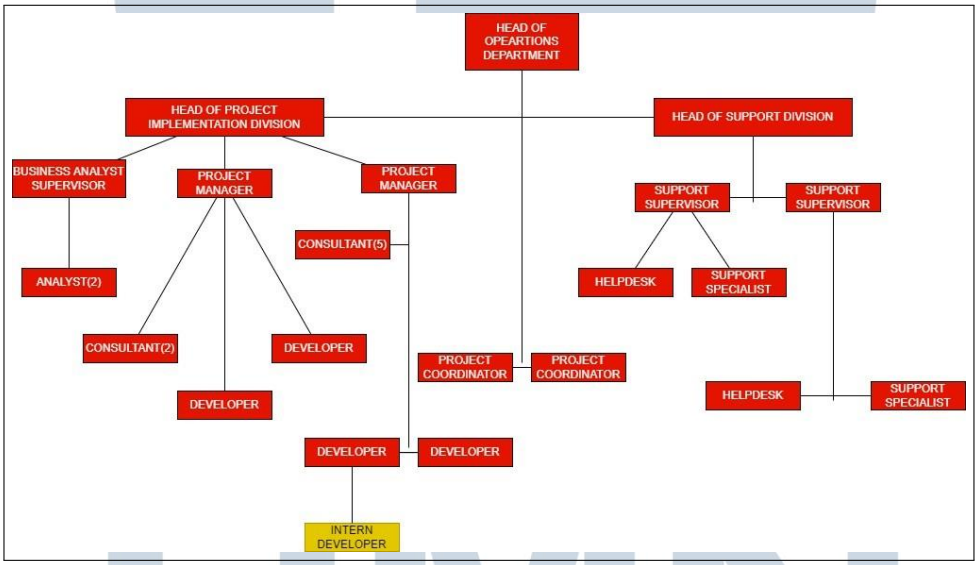
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Kedudukan Divisi *Software Programmer* pada perusahaan PT Hashmicro Solusi Indonesia berada di bawah Departemen *Agile*. Pada divisi *Software Programmer Intern* akan di bagi menjadi beberapa kelompok yang dipandu oleh salah satu *Senior Software Programmer ERP* dan diminta untuk memiliki pemahaman mengenai python, XML, CSS, Odoo, PostgreSQL beserta JavaScript.



Gambar 3.1. Struktur Departemen Agile pada perusahaan PT Hashmicro Solusi Indonesia

Sumber: [3]

Untuk koordinasi dalam pengerjaan tugas akan dibimbing oleh Bapak Arif Dharmawan selaku *Software Programmer ERP* pada PT Hashmicro Solusi Indonesia. Pemberian tugas dan koordinasi dilakukan melalui media *Skype* ataupun *Google Sheet* yang diberikan oleh *Project Manager* dan juga *Software Implementation Consultant*, lalu untuk setiap tugas akan dibagikan oleh *Software Programmer ERP* selaku mentor kepada setiap pemagang.

Lalu sebagai media komunikasi di dalam satu departemen *Agile* di PT. Hashmicro Solusi Indonesia untuk membagikan informasi, peraturan, dan pengumuman terkait pelaksanaan kerja magang akan dilakukan melalui 2 aplikasi media seluler yaitu *Telegram* dan juga *WhatsApp*.

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama melaksanakan kerja magang di PT. Hashmicro Solusi Indonesia adalah melakukan pengembangan aplikasi *Enterprise Resource Planning* pada Modul yang sudah dimodifikasi oleh Hashmicro untuk salah satu perusahaan *client* yang bergerak di bidang perminyakan dan juga perusahaan *client* yang bergerak di bidang sipil. *Enterprise Resource Planning* yang digunakan oleh Hashmicro berbasis Odoo dengan menggunakan bahasa pemrograman Python.

Dalam melakukan tugas sebagai *Software Programmer Intern*, pemagang diminta untuk melakukan pengawasan dan memperbaiki bug, berhubungan dengan *Software Implementation Consultant* untuk merancang fitur yang dibutuhkan oleh *client*. Lalu terdapat juga pembuatan tampilan *report* yang menggunakan *Extensible Markup Language* (XML) yang telah di *inherit* oleh *template* aslinya seperti *Invoice* yang akan di *print*.

3.3 Uraian Pelaksanaan Magang

Untuk pelaksanaan kerja magang di PT Hashmicro Solusi Indonesia Berlangsung selama 5 bulan, dimulai dari 1 Februari 2022 hingga 30 Juni 2022, di- mana pada 2 bulan pertama akan diberikan pembelajaran mengenai sistem ERP dan modul modifikasi yang dimiliki oleh Hashmicro yang menggunakan Odoo versi 14. Lalu untuk 3 bulan selanjutnya akan masuk ke dalam proyek yang sudah berjalan. Pelaksanaan kerja magang diuraikan pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Melakukan onboarding, adanya pengenalan perusahaan, lalu memperlihatkan struktur organisasi dan juga pengenalan mentor yang akan membantu dalam bimbingan kerja magang
2	Mempersiapkan pemagang mengenai apa saja yang dibutuhkan untuk pengerjaan kedepan seperti JAVASCRIPT, ODOO, PYTHON, POSTGRESQL, GIT, Extension yang digunakan pada VSC
3	Training mengenai 5 modul yaitu *Purchase, Inventory, Sales, Accounting, POS* dimana peserta magang dari berbagai divisi, Lalu juga dilanjutkan training untuk divisi Software Programmer, yang mengajarkan mengenai hal dasar bahasa pemrograman python, dan juga GIT
4	Mengikuti *training* mengenai module *Manufacture, Construction*, dan HRM. Lalu penjelasan Technical kepada para programmer dan consultant mengenai Framework odoo, Cara Backend Configuration, dan alur kerja
5	Mengikuti *training* untuk menggunakan Odoo seperti Pembuatan Modul sendiri, lalu dilakukannya pembuatan Model, lalu melakukan konfigurasi agar Model tersebut dapat di tampilkan dan penggunaan CRUD yang terhubung dengan postgre
6	Melatih pembuatan modul di odoo 14, lalu perkenalan dengan mentor, meeting pertama yang mempersiapkan kita sebelum masuk ke dalam project
7	Mengikuti pelatihan pembuatan modul Odoo 14
8	Mengikuti pelatihan pembuatan modul Odoo 14
9	Adanya pembagian task yang proyek yang sedang dikerjakan mentor sebagai pelatihan sebelum memegang proyek yang tidak di pegang oleh mentor dan melakukan setup HM3 modul ERP modifikasi Hashmicro.
10	Pembuatan report invoice without payment

Tabel 3.10. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (Lan-jutan)

11	Masuk ke dalam modul Purchase dan Sale. **Tugas 1**: Membuat fungsi dan tombol untuk memunculkan *popup* yang fungsinya untuk mengirimkan *email* dengan *template* yang sudah ada dengan membawa *attachment report*. **Tugas 2**: Membuat *Smart Button* yang menghitung ada berapa banyak *material request* yang sudah ada dan ketika di klik button tersebut akan memperlihatkan *material request* yang sudah dibuat. **Tugas 3**: Menghilangkan tombol *Create Purchase Tender*.
12	**Tugas 4**: Membuat fungsi dan tombol untuk memunculkan *popup* yang fungsinya untuk mengirimkan *WhatsApp* dengan *template* yang sudah ada dengan membawa *attachment report*.
13	**Tugas 5**: Membuat tipe transfer untuk *Stock Picking* lalu membuat *field Reserved* pada *QC inspection invisible*. **Tugas 6**:Mengganti *string* yang bertulisan costcode menjadi *internal category* di *job estimate* dan *job cost sheet*. **Tugas 7**:Menambahkan *selection field* pada *material request, purchase request, purchase order*. **Tugas 8**:Melakukan *Overide Purchase Order Line action* untuk mengubah *domain* yang telah digunakan. **Tugas 9**:Melakukan penambahan *field* yang dibutuhkan pada *material request, purchase request, purchase order*.
14	Libur Idul Fitri 1443H dan cuti bersama.
15	Untuk minggu ini masih membahas terlebih dahulu kode sebelum libur nasional lalu mengerjakan task yang diberikan untuk modul *accounting*. **Tugas 10**: Melanjutkan tugas minggu lalu dengan Menghubungkan *material request* ke *purchase request* lalu ke *purchase order*. Sebelumnya data tidak bisa langsung dikirim dan adanya konsultasi dengan mentor untuk melakukan penyelesaian penghubung.
16	**Tugas 11**: Membuat *Access Right* untuk pengguna agar tidak semua pengguna yang dapat melakukan *approving* pada modul *Accounting*. **Tugas 12**: Melakukan pengecekan dan *fixing* pada 3 *report* pada modul *accounting* yaitu bukti bank keluar, bukti bank terima, dan *invoice without payment*. **Tugas 13**: *Rename serial number* menjadi plat nomor pada *tree and form*. Melakukan

Tabel 3.11. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (Lan-jutan)

	pengubahan untuk menu *vehicle* karena, *form* dan *tree view* pada *vehicle* mengambil langsung dari menu asset sehingga perlu dibuatnya *overide button action* yang mengarahkan kepada *tree* baru.**Tugas 14**: Mengubah letak posisi *attachment* yang berada di *invoice* menjadi setelah *notebook* dengan *id other_tab*, Membuat modul baru untuk melakukan edit pada *Asset Management*, lalu menambahkan *field* baru yaitu *sertification date* dan mengubah *string* pada *tree* dan *form* yang dimiliki asset, yang sebelumnya serial no menjadi *activa number*.
17	**Tugas 15**: Membuat *log certificate assets* dengan tahapan memilih tombol lalu munculkan wizard saat di *confirm* akan menambahkan baris secara otomatis untuk tanggal buat dan tanggal *release* lalu juga menambahkan *Validation Error* ketika tanggal *release* lebih kecil daripada tanggal buat.
18	Melakukan pengecekan error yang terjadi pada project baru pada Odoo 10 dan project lama Odoo 14, sebelumnya melakukan installing terlebih dahulu di local untuk project baru Odoo 10, dan melakukan perbandingan pada project lama Odoo 14 yang sudah di update pada core nya Terutama pada modul modifikasi Hashmicro *Construction*
19	Melakukan trial fixing pada modul *construction* yang sudah di gabungkan dengan *update* modul modifikasi Hashmicro *Contruction*

3.3.1 Pengerjaan Tugas Kerja Magang

Proyek yang diberikan setelah pelatihan selesai adalah proyek salah satu *client* dari perusahaan Hashmicro yang bergerak pada bidang perminyakan, lalu selama berjalannya proyek pertama diberikan lagi proyek kedua yang merupakan salah satu *client* dari perusahaan Hashmicro yang bergerak di bidang sipil, dimana kedua proyek tersebut di pegang oleh bapak Samuel Soeharli sebagai *Project Manager*, *Software Implementation Consultant senior* Bapak Andhika, dan beberapa *Software Implementation Consultant intern*. Dengan sisa waktu yang diberikan yaitu 3 bulan, dilakukannya pengembangan pada proyek pertama yang berbasis

Odoo 14 dan untuk pengembangan proyek kedua menggunakan Odoo 10 sebagai basisnya.

Pengembangan yang dilakukan untuk kedua proyek tersebut dikerjakan berdasarkan tugas yang diberikan oleh Pak Samuel atau Pak Andhika dan *Software Implementation Consultant Intern*. Modul ERP yang dilakukan pengembangan ada *Accounting*, *Purchase*, *Assets*, *Manufacturing*, dan *Construction*. Berikut merupakan beberapa uraian penjelasan tugas apa saja yang dikerjakan pada masa kerja maganguntuk beberapa modul:

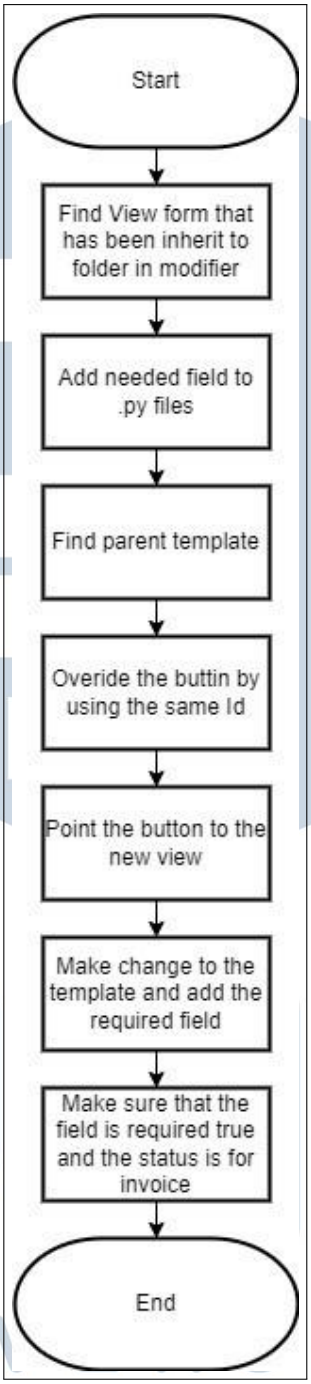
A. Modul Accounting

Modul ini digunakan untuk memonitor keuangan bisnis secara menyeluruh, membantu menghitung pendapatan dan biaya keluar secara tepat, membantu dalam membuat laporan laba rugi, arus kas, dan perubahan modal, dan melakukan *appor-val matrix* berdasarkan budget yang ditentukan.

A.1 Penambahan dan pembuatan Report Invoice Without Payment

Pada task pertama ini diminta untuk membuat Laporan *Invoice Without Payment* yang sesuai dengan contoh *printout* dengan *template* yang sudah ada tanpa mengubah *parent template*, karena *template* tersebut digunakan juga ke dalam laporan yang lain. Sehingga kebutuhan yang dilakukan selama pembuatan laporan *invoice without payment* adalah sebagai beriku:

- Menambahkan field yang diperlukan untuk dimunculkan ke dalam laporan
- Membuat laporan *required True* sehingga selama melakukan perubahan pada *Parent template* tidak mempengaruhi ke template lainnya



Gambar 3.2. Arus pengerjaan pembuatan laporan *Invoice Without Payment*

Tugas ini di awali dengan membuka menu *accounting* lalu memilih *invoice* dan memilih invoice yang ingin digunakan, hal tersebut dilakukan untuk melihat *view form* dan model yang digunakan, lalu melihat apakah sudah dibuatkannya modul baru untuk dilakukan memodifikasi modul *accounting invoice* yang sudah

ada. Selanjutnya melakukan inherit untuk modelnya, setelah dilakukannya *inherit* baru menambahkan field yang dibutuhkan untuk keperluan data pada laporan *invoice*. Contoh sebuah model odoo yang dilakukan *inherit* pada gambar 3.3

```python
from odoo import models, fields, _


class ModuleName1(models.Model):
    _inherit = 'model.contoh'

    binary = fields.Binary('binary')
    boolean = fields.Boolean('boolean')
    Character = fields.Char('Character')
    Date = fields.Date('Date')
    Date_time = fields.Datetime('field_name')
    float = fields.float('float')
    total = fields.Float('total', compute="_compute_total")
    Interger = fields.Integer('Interger')

    @api.depends("float")
    def _compute_total(self):
        for record in self:
            record.total = 2.0 * record.float

    testing_fields_ids = fields.One2many('modul.contoh.2',
    'testing_fields_id',
    string='testing_fields')

class ModuleName2(models.Model):
    _name = 'model.contoh.2'
    _description = 'New Description'

    testing_fields_id = fields.Many2one('modul.contoh',
    string='testing_fields')
```

Gambar 3.3. Contoh pembuatan model *inherit* untuk laporan *invoice*

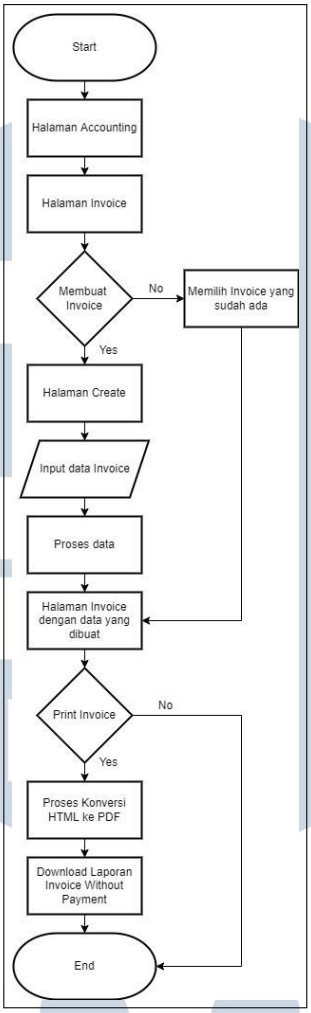
Pada fungsi *compute* untuk menghitung total bergantung pada *field* float, sehingga jika dilakukan perubahan pada *field* float maka *field* total akan ikut berubah bergantung dengan apa yang dilakukan di dalam *compute*.

Untuk relasi *One2many* isi dari *field* tersebut merupakan kumpulan dari banyak *field* yang berasal dari *comodel* yang mengarah ke *inverse comodel* nya yang memiliki isi yang sama dengan field *One2Many*. Sebagai contoh Budi memi-

liki banyak mobil, sehingga Aktor Budi sebagai *One* dan objek mobil sebagai *many*.

Relasi *Many2one* isi dari *field* tersebut hanya bisa 0 atau hanya satu saja. Sebagai contoh, suatu mobil dapat dimiliki oleh Budi, dan banyak mobil dapat dimiliki oleh Budi. Objek mobil disini sebagai *Many* dan aktor Budi disini sebagai *One*[5].

Selanjutnya dilakukan *overide button print* yang sudah ada dan mengarahkannya ke dalam view form yang ingin di modifikasi, dengan cara membuat *file* XML lalu menggunakan *record id* yang sama dan mengarahkannya ke *view form* yang baru. Setelah di arahkan ke *view form* yang baru, di tambahkan mode *primary*, mode tersebut digunakan agar *view form parent* yaitu *template invoice* yang di *inherit* tidak berubah sehingga *view* yang di *inherit* membentuk *view* yang baru. Lalu ditambahkan *field* yang diperlukan, menambahkan status untuk *invoice* agar laporan tersebut hanya dapat dilihat pada *submenu invoice*, dan melakukan modifikasi *layout* tampilan laporan, untuk modifikasi *layout* tampilan digunakan *inline style*.



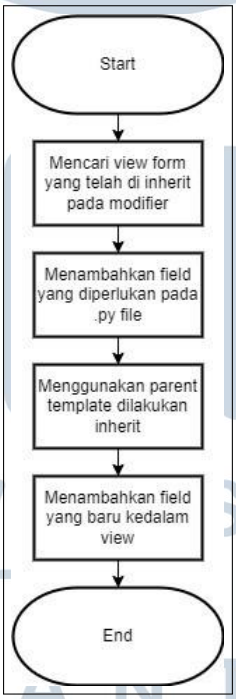
Gambar 3.4. Arus pengerjaan pembuatan laporan *Invoice Without Payment* untuk pengguna

Pengguna dapat melakukan mengunduh laporan *invoice without payment* dengan cara membuka menu *accounting*, lalu masuk ke dalam *submenu invoice*. Jika pengguna ingin membuat *invoice* dapat memilih *button create* lalu diarahkan ke halaman *create*, di dalam halaman tersebut *invoice* akan masuk ke dalam *state draft* dan pengguna diminta untuk memasukkan data untuk keperluan *invoicing*, setelah data sudah diisi maka dapat dilakukan penyimpanan, selanjutnya pengguna dapat melakukan pengunduhan laporan dengan memilih tombol *print* dan memilih tipe laporan *Invoice Without Payment*.

A.2 Pembuatan Notebook attachment Pada Invoice dan Vendor Bill

Pada tugas ini diminta untuk membuat tab *notebook* baru dengan nama *attachment* pada *invoice* dan *vendor bill*, dikarenakan *parent* untuk *view invoice* dan *vendor bill* sama, maka pada saat melakukan pengubahan untuk *parent template* tidak digunakan mode *primary* agar dapat mengubah langsung ke 2 *view* nya. Sehingga kebutuhan yang di lakukan dalam pengerjaan tugas tersebut adalah sebagai berikut:

- Menambahkan 3 *field* baru, untuk menyimpan nama *attachment* atau keterangan, satu lagi digunakan untuk menyimpan *file* dan yang satu lagi digunakan untuk menyimpan ke dua *field* tersebut.
- Membuat *field* tersebut *required true* sehingga pengguna harus memasukkan *attachment* baru setiap kali membuat *invoice* dan *vendor bill*.
- Membuat *pop up UserError* ketika pengguna mengeklik tombol menyimpan pada saat tidak memasukkan *attachment*.



Gambar 3.5. Arus pengerjaan pembuatan *Tab Notebook Attachment* pada *Invoice* dan *Vendor Bill*

Untuk tugas ini diawali dengan mencari *view form* nya agar dapat di *inherit* untuk di lakukan penambahan *field* baru. Setelah ditemukannya *view form* nya dilihat untuk model dan juga *external* data, setelah ditemukan modelnya dilakukan *inherit* ke dalam model tersebut agar dapat dilakukan penambahan *field*, begitu juga dengan *view* nya, perlu dibuatnya *view* baru yang *inherit* ke *external* data atau *view form* nya baru dilakukan xpath untuk melakukan penambahan *view* baru.

Perlu dibuat 1 *field* baru yang mengarah ke model baru yang berisi 2 *field* baru. Jadi 1 *field* baru merupakan *field one2many* yang akan dimunculkan di model utama, *field one2many* tersebut akan mengarah ke *comodel model* yang baru dan diberikan *inverse comodel* agar dapat melakukan pengambilan *field* pada model yang baru, yaitu *field* nama *attachment* dan *field* penyimpan *file* yang di *upload*.

Selanjutnya untuk penyempurnaan dilakukan *required true* untuk tiap *field* nya, hal tersebut dilakukan agar pada saat melakukan penginputan data, *field* tersebut akan ter *Highlight* dan jika pengguna masih belum mengisi *field* tersebut akan berikan *UserError* ketika pengguna ingin melakukan penyimpanan data.

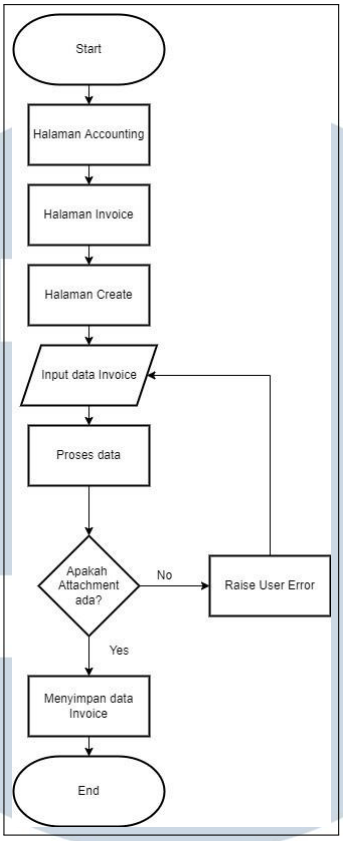
Berikut merupakan contoh untuk pengerjaan tugas pada gambar 3.6, contoh tersebut bukanlah kode asli, karena kode asli sudah dirahasiakan dan menjadi milik perusahaan Hashmicro. Berikutnya merupakan contoh pembuatan xml ke dalam *view form* yang sudah ada dan dilakukan modifikasi penambahan *tab notebook attachment* pada gambar 3.7.

```python
from odoo import models, fields, _
from odoo.exceptions import UserError

class ModuleName1(models.Model):
    _inherit = 'model.contoh'

    one2many_field_attach = fields.One2many('modul.contoh.2', 'inverse_comodel',
    string='Attachment', required = True)

    def check_line(self):
        if len(one2many_field_attach) == 0:
            raise UserError('Please fill attachment file')

class ModuleName2(models.Model):
    _name = 'model.contoh.2'
    _description = 'Field untuk one2many_field_attach'

    inverse_comodel = fields.Many2one('modul.contoh', string='inverse_comodel')
    name = fields.Char(string="Name", required = True)
    attach = fields.Binary(string="Attachment", required = True)
```

Gambar 3.6. Contoh pembuatan model untuk penambahan *tab attachment*

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <record id="inherit_view_id_inherit_module_name" model="ir.ui.view">
        <field name="name">model.name.view.form.inherit</field>
        <field name="model">model.contoh</field>
        <field name="inherit_id" ref="inherit_module_name.inherit_view_id"/>
        <field name="arch" type="xml">

            <xpath expr="//field_yangsudah_ada" position="after">
                <page >
                    <field name="one2many_field_attach" widget="one2many_list">
                        <tree string="Attachment" editable="bottom">
                            <field name="name"/>
                            <field name="attach"/>
                        </tree>
                    </field>
                </page>
            </xpath>

        </field>
    </record>
</odoo>
```

Gambar 3.7. Contoh pembuatan modifikasi *view* yang sudah ada untuk *attachment*



Gambar 3.8. Arus yang terjadi ketika sudah ditambahkan *tab notebook attachment* sebagai pengguna

Jadi setelah diterapkan tugas yang sudah diberikan maka pengguna pada saat masuk ke dalam halaman *accounting* lalu memilih *submenu invoice* dan melakukan *create invoice*, pada saat pengguna menginput data dan ingin menyimpan data tersebut akan mengecek apakah pada *tab attachment* sudah diisi jika belum akan menampikan *UserError* yang meminta untuk mengisi *attachment*, jika *attachment* sudah diisi makan data *invoice* tersebut berhasil disimpan. Hal ini juga berlaku untuk tahapan pembuatan *vendor bill*.

A.3 Penambahan field department pada pettycash yang autofill

Pada tugas ini diperlukan penambahan *field department* pada *submenu pettycash*. Karena *view form* untuk *pettycash* orisinil atau tidak melakukan *inherit template* yang sudah ada, maka tidak diperlukannya mode *primary*. Lalu untuk *field Department* tersebut akan melakukan *autofill* ketika *field Custodian* sudah terisi. Sehingga kebutuhan yang di lakukan dalam pengerjaan tugas tersebut adalah seba-

gai berikut:

- Mencari *form view* dan juga model yang digunakan
- Melakukan penambahan *field Department* pada model yang di *inherit*
- Membuat fungsi yang melakukan *autofill* untuk *field Department* yang bergantung dengan *field Custodian*

Tugas ini diawali dengan mencari *form view* yang menampilkan halaman *petty cash*, setelah ditemukannya *form view* dilakukan pengecekan apakah tampilan tersebut pernah ada modifikasi atau tidak, karena jika tidak perlu dibuatnya modul baru untuk memodifikasi tampilan tersebut, karena sudah ada tinggal menambahkan *field* baru ke dalam model yang di *inherit* dan tidak lupa untuk membuat fungsi untuk melakukan input ke dalam *field* departemen yang bergantung kepada *field Custodian*.

Selanjutnya untuk melakukan penambahan *field* yang sudah di buat ke dalam tampilan *form* yang sudah ada, dengan melakukan *inherit* ke dalam *id* tampilan *form*, lalu menentukan letak dimana ingin menambahkan *field* tersebut. Jika membuat *file XML* baru perlu dilakukannya konfigurasi ke dalam *manifest* untuk menambahkan *field* tersebut.

Untuk tugas ini dilakukan untuk memenuhi kelengkapan data pada saat pembuatan *pettycash Custodian* dan juga agar pengguna tidak perlu melakukan input kembali pada saat sudah memilih *Custodian*. Namun jika *Custodian* yang dipilih belum masuk kedalam departemen manapun, dapat dilakukan input manual dengan pilihan departemen yang sudah ada.

A.4 Membuat Access Right agar user dapat melakukan confirm pada saat status nya draft

Pada tugas ini diminta untuk membuat *access right* untuk menu *accounting* karena pada saat pembuatan *Invoice* dan *Vendor Bill* pada saat status *Invoice* ingin diubah ke status *posted*, untuk tombol *Confirm* tidak semua pengguna yang dapat menggunakannya sehingga perlu dibuatnya pengelompokan pengguna. Sehingga kebutuhan yang dilakukan dalam pengerjaan tugas tersebut adalah sebagai berikut:

- Mencari *form view* untuk tombol konfirmasi
- Membuat pengelompokan baru pada *file* xml

- Mengubah pengelompokan yang digunakan pada tombol konfirmasi menjadi pengelompokan yang baru dibuat

Pada awalnya dicari terlebih dahulu *view form* yang menampilkan tombol konfirmasi tersebut, lalu melihat apakah *view form* tersebut pernah dilakukan modifikasi, setelah dilihat pada *inherited views* ternyata pernah dilakukan modifikasi sehingga tidak perlunya di buat modul baru.

Selanjutnya membuat pengelompokan baru yang digunakan untuk mengakses tombol konfirmasi di dalam *File XML* baru. Untuk Contoh pembuatan pengelompokan dapat dilihat pada gambar 3.9. Selanjutnya melakukan pengubahan *attributes* pada tombol konfirmasi sehingga hanya kelompok tersebut yang dapat melakukan konfirmasi. Contoh kode pengubahan *attributes button* dapat dilihat pada gambar 3.10.

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>

    <data noupdate="0">

        <record id="group_1" model="res.groups">
            <field name="name">Your Group Name 1</field>
        </record>

        <record id="group_2" model="res.groups">
            <field name="name">Your Group Name 2</field>
        </record>

    </data>

</odoo>
```

Gambar 3.9. Contoh pembuatan *group* baru pada *file XML*

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <record id="inherit_model_contoh" model="ir.ui.view">
        <field name="name">model.name.view.form.inherit</field>
        <field name="model">model.name</field>
        <field name="inherit_id" ref="inherit_module_name.inherit_view_id"/>
        <field name="arch" type="xml">

            <xpath expr="//posisi_button_yang_diubah[1]" position="attributes">
                <attribute name="groups">nama_model.group_1</attribute>
            </xpath>

        </field>
    </record>
</odoo>
```

Gambar 3.10. Contoh pengubahan *attributes button confirm*

Setelah diterapkan tugas yang diberikan, pengguna perlu mendaftarkan atau memasukkan diri mereka ke dalam pengelompokan yang baru, jika pengguna tersebut diperlukan dalam melakukan konfirmasi *draft* pada modul *Accounting*. Karena jika tidak, tombol konfirmasi tersebut tidak akan muncul dan *draft* tersebut tidak akan berubah statusnya menjadi *post* sampai pengguna yang masuk ke dalam pengelompokan melakukan konfirmasi *Draft*.

B. Modul Puchase

Modul ini digunakan untuk melacak produk yang masuk ke gudang, manajemen *Backorder* sehingga pembelian barang dapat disesuaikan dengan jumlah barang yang diterima, adanya manajemen *budgeting* dan *cost center* per departemen atau proyek.

B.1 Melakukan perbaikan terhadap Purchase Order Line

Pada tugas ini hanya dilakukan pengecekan mengapa *submenu Purchase Order Line* pada menu *Purchase* ada dan berfungsi namun tidak memperlihatkan data yang sudah tersimpan. Untuk tugas ini dilakukan perbaikan dengan tahapan seperti berikut:

- Mencari *window action* pada saat *submenu Purchase Order Line* ingin dibuka.

- Melakukan pengecekan apa yang membuat *submenu Purchase Order Line* tidak memberikan data yang sudah.

Untuk tugas ini dilakukan pengecekan pada *window action* yang berjalan ketika *submenu Purchase Order Line* ingin dibuka. *Window action* digunakan untuk menampilkan visualisasi suatu model melalui *view* [6]. Jika *Window Action* sudah ditemukan selanjutnya dilakukan pencarian apa yang membuat data yang sudah ada ditampilkan.

setelah dilakukan pengecekan ternyata pada *Window Action* tersebut memiliki *domain*. *Domain* digunakan untuk melakukan filter terhadap data dengan kondisi yang dibarikan, *domain* memerlukan tiga input dalam membuat kondisi yaitu nama *Field*, operator, dan nilai. Sebagai contoh dalam model *Product* hanya ingin menampilkan data dengan unit *price* di atas 1000 makan pemberian Domain seperti pada gambar 3.11.

```
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <record id="action_model_act_window" model="ir.actions.act_window">
        <field name="type">ir.actions.act_window</field>
        <field name="name">Human readable name</field>
        <field name="res_model">model</field>
        <field name="view_mode">tree,form</field>
        <field name="view_type">form</field>
        <field name="target">current</field>
        <field name="domain">[('unit_price', '>', 1000)]</field>
        <field name="context">\{}</field>
        <field name="search_view_id" ref="view_model_search" />
        <field name="help" type="html">
            <p class="oe_view_nocontent_create">
                Click to add new Human readable name
            </p><p>
                Something about
            </p>
        </field>
    </record>
</odoo>
```

Gambar 3.11. Contoh penambahan *domain* pada *Action Window*

Karena permintaan tugas ini hanya untuk memperlihatkan semua data *Purchase Order Lines* sehingga tidak perlu adanya domain untuk *Action Windows* tersebut. Sehingga untuk perlu membuat *Action window* yang baru dengan *id* yang mengarah ke *Action window* yang lama (nama_modu.id_action_window_yang_lama),

yang hanya diubah hanya bagian *field domain* di kosongi.

C. Modul Assets Management

Modul ini digunakan untuk melakukan perhitungan secara otomatis depresiasi aset, ROI, dan serta biaya yang dikeluarkan, mengelola kontrak aset yang disewakan dan biayanya, dapat melakukan *asset tracking* dapat menyimpan informasi nama penanggung jawab, nilai, lokasi, biaya, dan dokumen kontrak aset, melakukan *asset stock taking* memastikan jumlah aset yang dimiliki sama dengan yang ada dalam sistem, dan membuat laporan nilai aset.

C.1 Membuat log certificate asset dengan memunculkan wizard dan akan mengisi secara otomatis

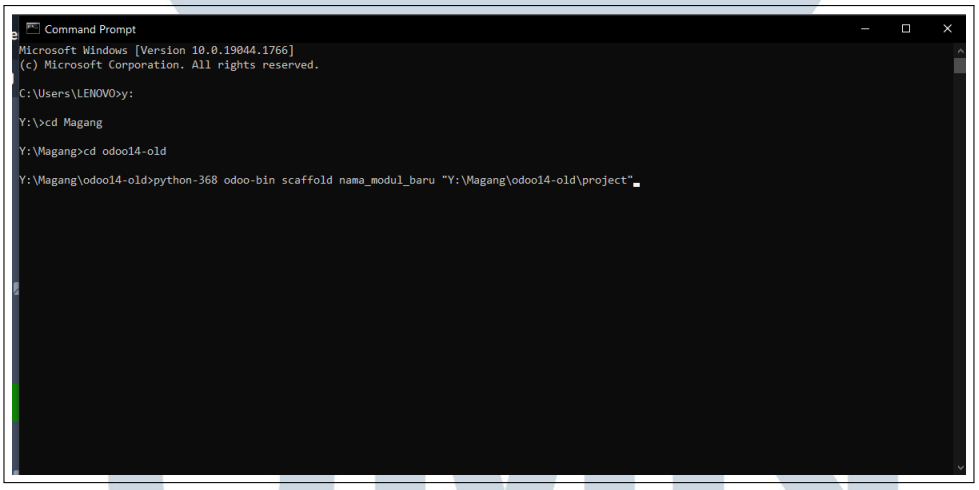
Pada tugas ini perlu dibuatnya *field* baru untuk menyimpan tanggal sertifikasi pada submenu *Asset*. *Field* tersebut akan digunakan pada saat melakukan *Certificate Asset* sehingga akan memunculkan *Wizard* yang memunculkan beberapa *field* yang perlu di isi oleh pengguna. Ketika sudah diisi data tersebut akan masuk ke dalam Tab *Notebook Certification Log*. Sehingga tahapan yang diperlukan dalam pengerjaan tugas adalah sebagai berikut:

- Mencari *View Form* yang digunakan untuk menu *Asset*, melakukan pengecekan apakah ada modul modifikasi yang mengarah ke model dan *view* tersebut.
- Membuat modul modifikasi baru dengan membuat konfigurasi juga untuk *manifest*.
- Membuat model baru yang melakukan *inherit* ke model yang digunakan di *asset* pada *file* Python.
- Membuat model baru untuk keperluan *Wizard*.
- Membuat fungsi pada saat *button Asset Certificate* di klik melakukan pengiriman data *Certification* Log Ke dalam *Wizard*.
- Membuat *view* baru untuk penambahan *field* baru, dan membuat *view* baru untuk bentuk *wizard* nya.

- Melakukan konfigurasi pada *manifest* yaitu menambahkan *view* yang ingin di tampilkan beserta *view wizard* nya.

Untuk tugas ini diawali dengan melakukan pencarian untuk *view form* yang digunakan pada submenu *Asset*, lalu dilakukan pengecekan apakah sudah adanya modul modifikasi yang sudah di buat mengarah ke dalam modul Asset dan *view form*, dikarenakan tidak ada perlu dibuatnya modul baru untuk melanjutkan tugas ini.

Selanjutnya membuat modul baru untuk melakukan modifikasi ke dalam modul *Asset*, hal tersebut dapat dilakukan melalui *command prompt* dengan mengarahkan dulu ke dalam folder odoo yang digunakan karena ingin menggunakan odoo-bin yang sudah ada, lalu memasukkan perintah python_yang_digunakan odoo-bin scaffold nama modul baru lokasi untuk modul baru. Contoh penggunaan perintah pembuatan modul baru pada gambar 3.12



Gambar 3.12. Perintah yang digunakan untuk membuat modul baru

Setelah dibuatnya modul baru dilakukan penambahan *field* dengan cara melakukan *inherit* ke dalam model yang digunakan pada modul *Asset* yang pertama *field date* untuk menyimpan tanggal sertifikasi, lalu *field* one2many yang akan menyimpan beberapa *field* seperti inverse_model lalu nomor, nama sertifikat, tanggal mulai sertifikat, tanggal berakhir sertifikat, dan deskripsi sertifikat. Contoh kode dapat dilihat pada gambar 3.13.

```python
from odoo import models, fields, _
from odoo.exceptions import UserError

class ModuleName1(models.Model):
    _inherit = 'model.contoh'

    one2many_field_sertif = fields.One2many('modul.contoh.2', 'inverse_comodel', string='Attachment', required = True)

    tanggal_sertif = fields.Date("Tanggal Sertifikasi")

class ModuleName2(models.Model):
    _name = 'model.contoh.2'
    _description = 'field untuk one2many_field_sertif'

    inverse_comodel = fields.Many2one('model.contoh', string='inverse_comodel')
    no = fields.Char('Number', readonly=True, default = 'New')
    name = fields.Char(string = 'Name', required=True)
    start = fields.Date(string = 'Start')
    end = fields.Date(string = 'End')
    desc = fields.Char(string = 'Description')
```

Gambar 3.13. Contoh pembuatan model untuk keperluan sertifikasi

Untuk *field number* dilakukan *readonly* agar pengguna tidak perlu menginput manual, jadi akan dibuatkan *sequence* yang akan bertambah setiap kali adanya penambahan pada sertifikasi log. Selanjutnya dengan *field* yang sudah ada *field* tersebut harus ditampilkan terlebih dahulu, yaitu dengan cara melakukan *view* yang mengarah ke *id view parent* yang digunakan oleh menu asset. Untuk contoh penerapan *field* baru kedalam *view* yang di *inherit* bisa di lihat pada gambar 3.14.

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <record id="inherit_view_modul_asset" model="ir.ui.view">
        <field name="name">inherit_view_modul_asset</field>
        <field name="model">model.asset</field>
        <field name="inherit_id" ref="modul_aset.view_modul_aset"/>
        <field name="arch" type="xml">

            <xpath expr="//arahin_posisi_create_date" position="attributes">
                <attribute name="string" >Acquisition Date</attribute>
            </xpath>

            <xpath expr="//arahin_ke_field" position="after">
                <field name="tanggal_sertif" />
            </xpath>

            <xpath expr="//posisi_notebook" position="inside">
                <page string="Certi Log" name="certi_page">
                    <field name="one2many_field_sertif" widget="one2many_list">
                        <tree string="Serti Log">
                            <field name="no" />
                            <field name="name" />
                            <field name="start" />
                            <field name="end" />
                            <field name="desc" />
                        </tree>
                    </field>
                </page>
            </xpath>

        </field>
    </record>
</odoo>
```

Gambar 3.14. Dilakukan *inherit* untuk *View Parent* pada modul Asset

Selanjutnya dibuat tombol untuk memunculkan *Wizards*. *Wizards* merupakan *extend* dari suatu *class* dan merupakan *Transient Model* bukan termasuk model. Disebut *Transient Model* karena *records data* yang berada pada *Wizards* akan secara otomatis terhapus oleh *database* dalam kurun waktu tertentu. *Wizards* dapat memberikan sesi interaktif(Kotak dialog) yang memperlihatkan formulir yang dinamis [7]. Untuk memunculkannya *Wizards*, dibuatnya suatu tombol yang memiliki fungsi untuk menampilkan *Wizards* tersebut, sehingga perlu juga dibuat modeluntuk *Wizards Certificate* untuk mengambil input dari pengguna.

Membuat model baru untuk menyimpan input yang diberikan oleh pengguna isi input akan disamakan dengan model.contoh.2 karena hasil input pada *Wizards* akan mengisi pada model.contoh.2, lalu dibuatkan fungsi untuk mengantarkan hasil

input tersebut ke *field* one2many pada model.contoh yaitu one2many_field_sertif untuk mempermudah melakukan pengantaran data. Dengan menggunakan *active_id* pengiriman data akan mengirim dengan model yang memiliki *active_id* yang sama. Untuk contoh pembuatan model dapat dilihat pada gambar 3.15.

```python
from odoo import models, fields, _
from odoo.exceptions import UserError

class WizardsModuleName2(models.Model):
    _name = 'model.contoh.wiz'
    _description = 'model Wizards'

    name = fields.Char(string="Name", required = True)
    start = fields.Date(string="Start")
    end = fields.Date(String = "End")
    desc = fields.Char(String = "Description")

    def send_data(self):
        get_active_id = self._context.get('active_id')
        get_model = self.env['modul.contoh'].browse(get_active_id)
        values = {
            'name': self.name,
            'start': self.start,
            'end': self.end,
            'desc': self.desc,
        }
        get_model.one2many_field_sertif = [(0,0, values)]
```

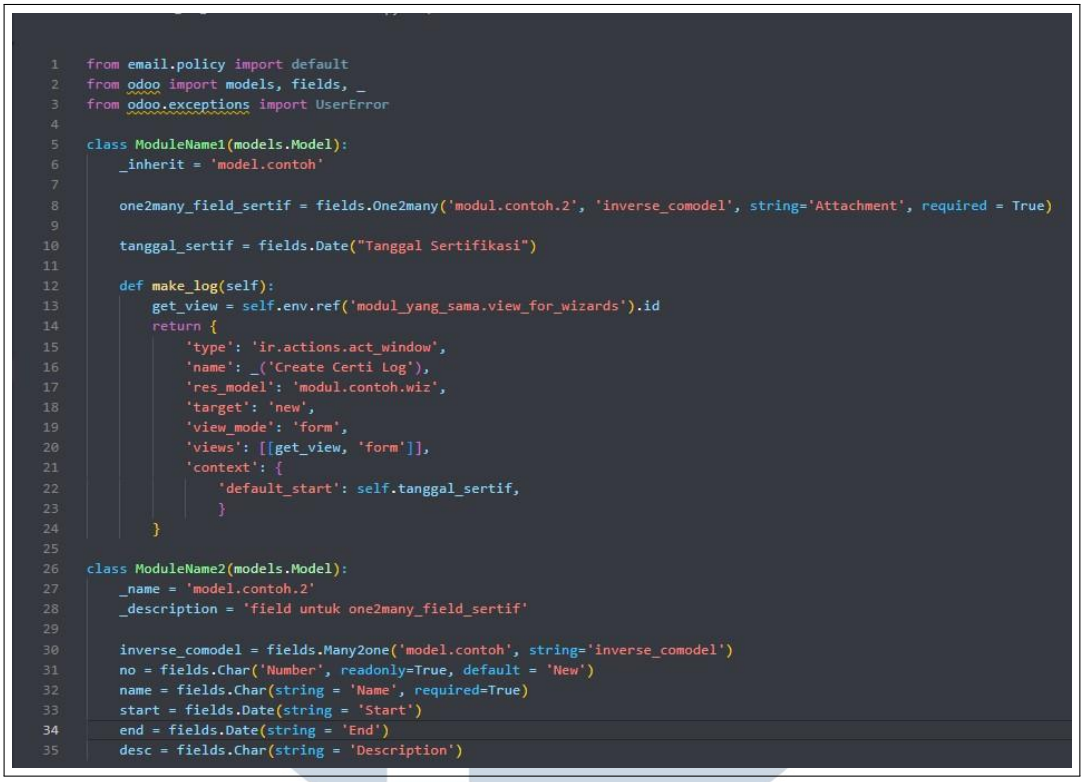
Gambar 3.15. Contoh pembuatan model untuk *Wizards*

Setelah modelnya sudah terbentuk selanjutnya dibuatkan *view* untuk *wizard* yang akan tampil dengan tambal tombol kirim data dengan *name = 'send_data'*, hal ini dilakukan agar tombol tersebut pada saat di klik akan menjalan fungsi yang telah dibuat. Pembuatan kode untuk *view Wizards* ada pada gambar 3.16.

```xml
<?xml version='1.0' encoding='utf-8'?>
<odoo>
    <record id="view_for_wizards" model="ir.ui.view">
        <field name="name">view_for_wizards</field>
        <field name="model">model.contoh.wiz</field>
        <field name="type">tree</field>
        <field name="arch" type="xml">
            <form string = "Create Certi Log">
                <sheet>
                    <group>
                        <field name="name"/>
                        <field name="desc"/>
                        <field name="start"/>
                        <field name="end"/>
                    </group>
                    <footer>
                        <button name="send_data" type="object" string="Create log" class="btn btn-primary"/>
                        <button special="cancel"/>
                    </footer>
                </sheet>
            </form>
        </field>
    </record>
</odoo>
```

Gambar 3.16. Pembuatan *Wizards View*

Selanjutnya setelah *Wizards* sudah selesai perlu dilakukan penambahan fungsi tombol pada model model.contoh untuk membuka *view Wizards* dan membawa data tanggal_sertif jika ada. Setelah fungsi dibuat, perlu dibuatnya tombol yang akan menjalankan fungsi tersebut pada *view* dengan menggunakan name yang sama dengan nama fungsinya. Berikut penerapan fungsi yang digunakan pada model.contoh pada gambar 3.17.



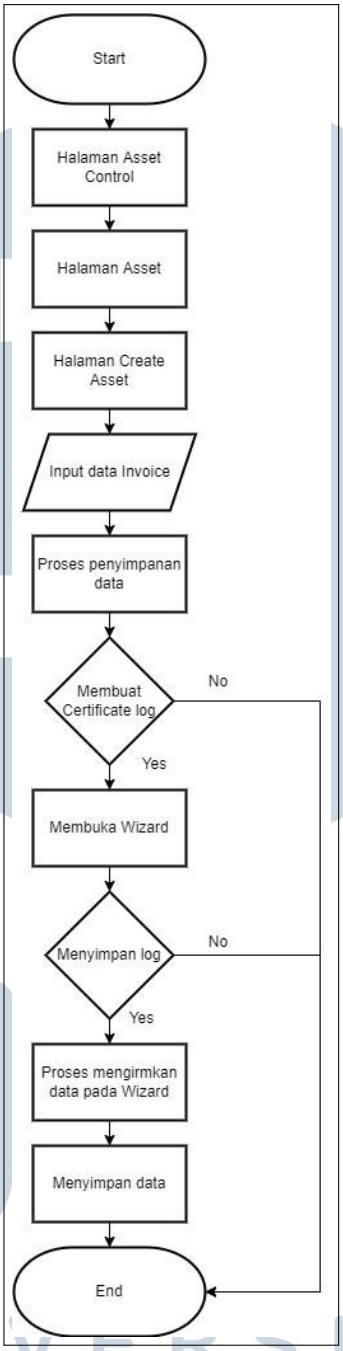
```python
from email.policy import default
from odoo import models, fields, _
from odoo.exceptions import UserError

class ModuleName1(models.Model):
    _inherit = 'model.contoh'

    one2many_field_sertif = fields.One2many('modul.contoh.2', 'inverse_comodel', string='Attachment', required = True)

    tanggal_sertif = fields.Date("Tanggal Sertifikasi")

    def make_log(self):
        get_view = self.env.ref('modul_yang_sama.view_for_wizards').id
        return {
            'type': 'ir.actions.act_window',
            'name': _('Create Certi Log'),
            'res_model': 'modul.contoh.wiz',
            'target': 'new',
            'view_mode': 'form',
            'views': [[get_view, 'form']],
            'context': {
                'default_start': self.tanggal_sertif,
                }
        }

class ModuleName2(models.Model):
    _name = 'model.contoh.2'
    _description = 'field untuk one2many_field_sertif'

    inverse_comodel = fields.Many2one('model.contoh', string='inverse_comodel')
    no = fields.Char('Number', readonly=True, default = 'New')
    name = fields.Char(string = 'Name', required=True)
    start = fields.Date(string = 'Start')
    end = fields.Date(string = 'End')
    desc = fields.Char(string = 'Description')
```

Gambar 3.17. Penerapan fungsi untuk tombol

Setelah dilakukan penambahan fungsi pada model model.contoh dilakukannya penambahan tombol pada *view* yang sudah dibuat, agar dapat menampilkan *Wizards* dan membawa data yang diperlukan. Berikut penambahan tombol untuk membuka *Wizards* pada gambar 3.18.

```python
from email.policy import default
from odoo import models, fields, _
from odoo.exceptions import UserError

class ModuleName1(models.Model):
    _inherit = 'model.contoh'

    one2many_field_sertif = fields.One2many('modul.contoh.2', 'inverse_comodel', string='Attachment', required = True)

    tanggal_sertif = fields.Date("Tanggal Sertifikasi")

    def make_log(self):
        get_view = self.env.ref('modul_yang_sama.view_for_wizards').id
        return {
            'type': 'ir.actions.act_window',
            'name': _('Create Certi Log'),
            'res_model': 'modul.contoh.wiz',
            'target': 'new',
            'view_mode': 'form',
            'views': [[get_view, 'form']],
            'context': {
                'default_start': self.tanggal_sertif,
                }
        }

class ModuleName2(models.Model):
    _name = 'model.contoh.2'
    _description = 'field untuk one2many_field_sertif'

    inverse_comodel = fields.Many2one('model.contoh', string='inverse_comodel')
    no = fields.Char('Number', readonly=True, default = 'New')
    name = fields.Char(string = 'Name', required=True)
    start = fields.Date(string = 'Start')
    end = fields.Date(string = 'End')
    desc = fields.Char(string = 'Description')
```

Gambar 3.18. Penambahan tombol untuk membuka *Wizards*



Gambar 3.19. Arus pengerjaan pembuatan *Asset* untuk pengguna

Setelah diterapkan tugas yang diberikan pengguna dapat melakukan pembuatan *asset* dengan masuk ke dalam halaman *Asset Control* lalu masuk ke dalam halaman *Asset*, selanjutnya membuat *Asset* baru dengan memilih tombol *Create* dan masuk ke dalam halaman *Create Asset*, berikutnya pengguna akan diminta untuk memasukkan input untuk kelengkapan data. Setelah data berhasil disimpan peng-

guna tidak diharuskan untuk membuat sertifikasi log, jika pengguna ingin membuat sertifikasi log pada *asset* tersebut pengguna dapat menekan tombol *Create certi log* lalu akan muncul *wizard*, yang didalam-nya pengguna diminta untuk memasukkan data sebagai kelengkapan sertifikasi log, setelah pengguna memilih untuk menyimpan, *Wizards* akan otomatis tertutup dan *Tab Notebook Certificate* akan otomatis terisi sesuai dengan input yang dilakukan pada *Wizard*.

3.4 Kendala

Kendala yang dialami selama menjalani kerja magang di PT. Hashmicro Solusi Indonesia sebagai beriku:

- Adanya kendala dalam melakukan *task* karena perlunya dilakukan setup Odoo(10) yang merupakan versi odoo yang berbeda dengan masa *training*(14).
- Adanya *module* Python yang tidak dapat digunakan atau hanya bisa digunakan jika menggunakan *Operating System Linux*.
- Kurangnya pemahaman Odoo pada saat masuk ke dalam pembagian proyek. Dikarenakan materi yang diberikan pada masa *training* masih sangat mendasar, sehingga terkadang memerlukan waktu yang lama untuk mengerjakan satu tugas.

3.5 Solusi

Berikut beberapa solusi yang ditemukan selama menjalani kerja magang di PT. Hashmicro Solusi Indonesia:

- Melakukan diskusi dengan mentor ataupun rekan kerja magang yang sudah berhasil melakukan setup proyek dengan versi Odoo yang berbeda.
- Melakukan *Dual Booting* jika diperlukannya menggunakan OS Linux karena jika menggunakan seperti *Virtual Machine* akan memakan banyak *Resource*.
- Melakukan pembelajaran mandiri seperti membaca dokumentasi versi Odoo yang digunakan pada situs resmi Odoo. Bisa juga mencari secara langsung masalah yang dialami pada forum Odoo.