BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Program kerja magang pada PT Sumber Inovasi Informatika dilakukan dalam sebuah divisi tim “Marvel”. Sebagai *software developer* pada tim tersebut, mahasiswa dituntut untuk melakukan *coding* untuk implementasi seperti sistem ERP maupun *Web Application* demi memenuhi kebutuhan klien.

Pekerjaan yang dilakukan oleh *software developer* adalah mengembangkan suatu aplikasi yang berfungsi untuk memenuhi kebutuhan para klien yang bersangkutan. Para *software developer* juga di wajibkan untuk melakukan kegiatan *sprint planning* selama dua minggu sekali untuk memastikan kebutuhan bisnis yang di inginkan oleh para klien.

Selama program kerja magang berlangsung, mahasiswa di bimbing dan di koordinasikan oleh Arvy Pradipta Budiarto selaku Direktur utama PT Sumber Inovasi Informatika. Perusahaan juga tidak memberikan perlakuan khusus bagi karyawan tetap maupun karyawan magang. Pekerjaan yang diberikan dari perusahaan sesuai dengan kebutuhan para klien.

3.2 Tugas yang Dilakukan

Pada PT Sumber Inovasi Informatika, mahasiswa memiliki beberapa tugas yang harus diselesaikan pada program kerja magang tersebut, yakni:

Tabel 3.1 Timeline Kegiatan Program Kerja Magang

No	Kegiatan	Waktu Pelaksanaan (Minggu)								
		1	2	3	4	5	6	7	8	9
1	Pembelajaran dasar ERPNext serta bahasa pemrograman python dan javascript.									
2	Pengembangan aplikasi ERP untuk Toko Cat Utama									
3	Pengembangan aplikasi ERP untuk Coway International									

Berdasarkan tabel 3.1 tersebut, berikut penjelasan singkat mengenai program kerja magang *software developer* di PT Sumber Inovasi Informatika yang mahasiswa lakukan yaitu:

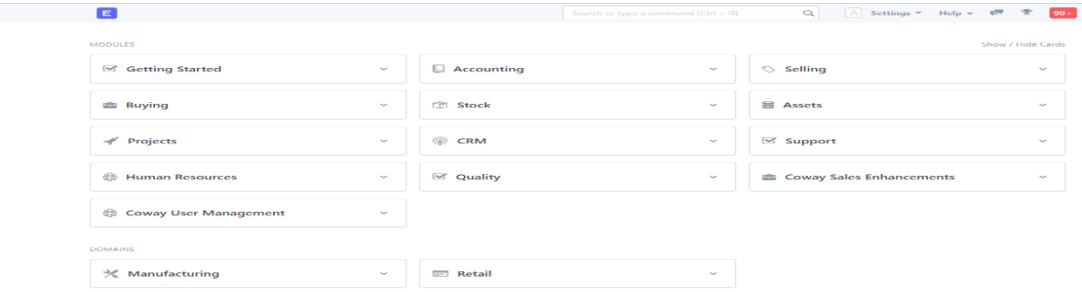
1. Pembelajaran dasar ERPNext serta bahasa pemrograman python dan javascript. Diperlukan pemahaman terdahulu mengenai apa itu sistem ERP sekaligus cara kerjanya. Serta dibutuhkan pengetahuan *coding* dengan bahasa pemrograman python dan Javascript.
2. Mengembangkan atau melakukan *custom* pada sistem ERP untuk memenuhi kebutuhan klien. Dalam hal ini klien yang dimaksud adalah Toko Cat Utama
3. Mengembangkan atau melakukan *custom* pada sistem ERP untuk memenuhi kebutuhan klien. Dalam hal ini klien yang dimaksud adalah Coway International

3.3 Uraian Kegiatan Magang

3.3.1 Pembelajaran dasar ERPNext serta bahasa pemrograman Python dan Javascript.

ERP adalah sistem informasi yang diperuntukkan bagi perusahaan manufaktur maupun jasa yang berperan mengintegrasikan dan mengotomasikan proses bisnis. Pada PT Sumber Inovasi Informatika, salah satu jasa yang ditawarkan dari perusahaan yaitu pembuatan sistem ERP untuk memenuhi kebutuhan klien.Sebelum memasuki pembuatan sistem ERP, dibutuhkan suatu pemahaman dasar akan ERP itu sendiri serta pemahaman tentang bahasa pemrograman yang dipakai. *Software* ERP yang dimaksud adalah ERPNext.Mahasiswa diminta dari perusahaan untuk melakukan pembelajaran secara individual sebelum memasuki tahapan *development*. Pembelajaran yang dilakukan mahasiswa selama 2 minggu yakni:

1. Modul ERPNext



Gambar 3. 1 Modul ERPNext

Dari gambar 3.1 diatas, Ada beberapa modul dalam sistem ERP tersebut seperti *Accounting, Selling, Buying, Stock, Human Resources*, dan masih banyak lagi. Mahasiswa diminta untuk mempelajari *flow* dasar yang ada pada sistem ERPNext itu sendiri. Mahasiswa juga harus mengerti tentang modul-modul yang berada pada ERPNext serta mengerti struktur ERPNext serta fungsi dari modul-modul yang terdapat pada ERP tersebut.

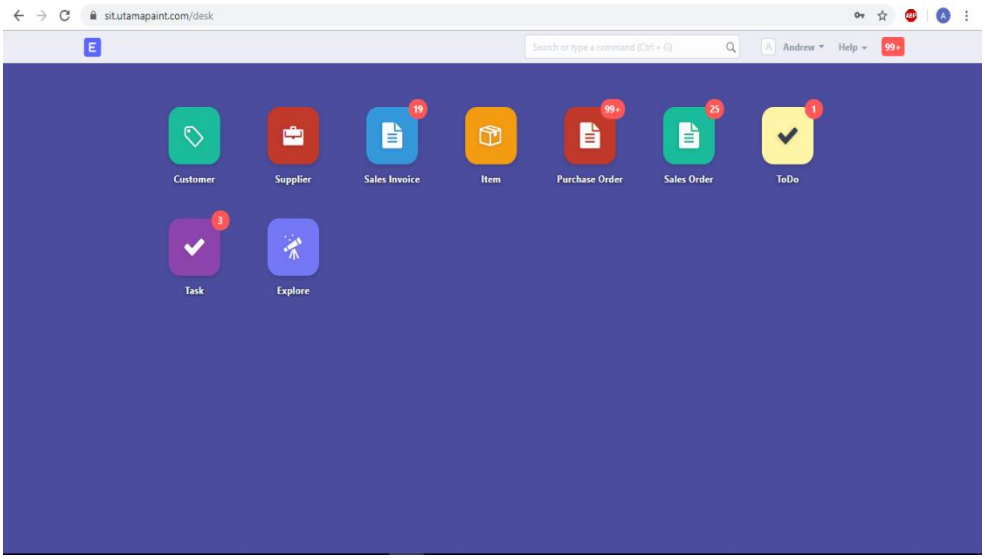
2. Bahasa Pemrograman Python dan Javascript

Bahasa pemrograman python dan javascript sangat sering dipakai pada pembuatan *software* ERPNext tersebut. Maka dari itu, mahasiswa diminta untuk mengerti *syntax* yang berada pada python dan javascript karna kurangnya pemahaman akan bahasa pemrograman tersebut. Python digunakan pada *server-side* serta javascript digunakan pada *client-side*. Mahasiswa melakukan pembelajaran bahasa pemrgoraman ini dengan melakukan *online course* pada *website*. Setelah melakukan pembelajaran python dan javascript secara dasar, mahasiswa langsung melakukan

pembelajaran mengenai *function* yang ada pada *framework* pendukung ERPNext yaitu *frappe*.

3.3.2 Pengembangan aplikasi ERP untuk Toko Cat Utama

Toko cat utama merupakan sebuah perusahaan yang menjual produk seperti *decorative paint, protective and marine paint,* dan mesin *tinting* pada kontraktor, pemilik rumah, maupun toko cat lainnya. Toko Cat Utama ini mempunyai masalah krusial dikarenakan penjualan yang semakin meningkat dan membutuhkan suatu sistem yang dapat menampung semua proses bisnis pada Toko Cat Utama. Maka dari itu, dibutuhkan suatu sistem ERP yang dapat mengintegrasikan proses bisnis yang terjadi pada Toko Cat Utama.



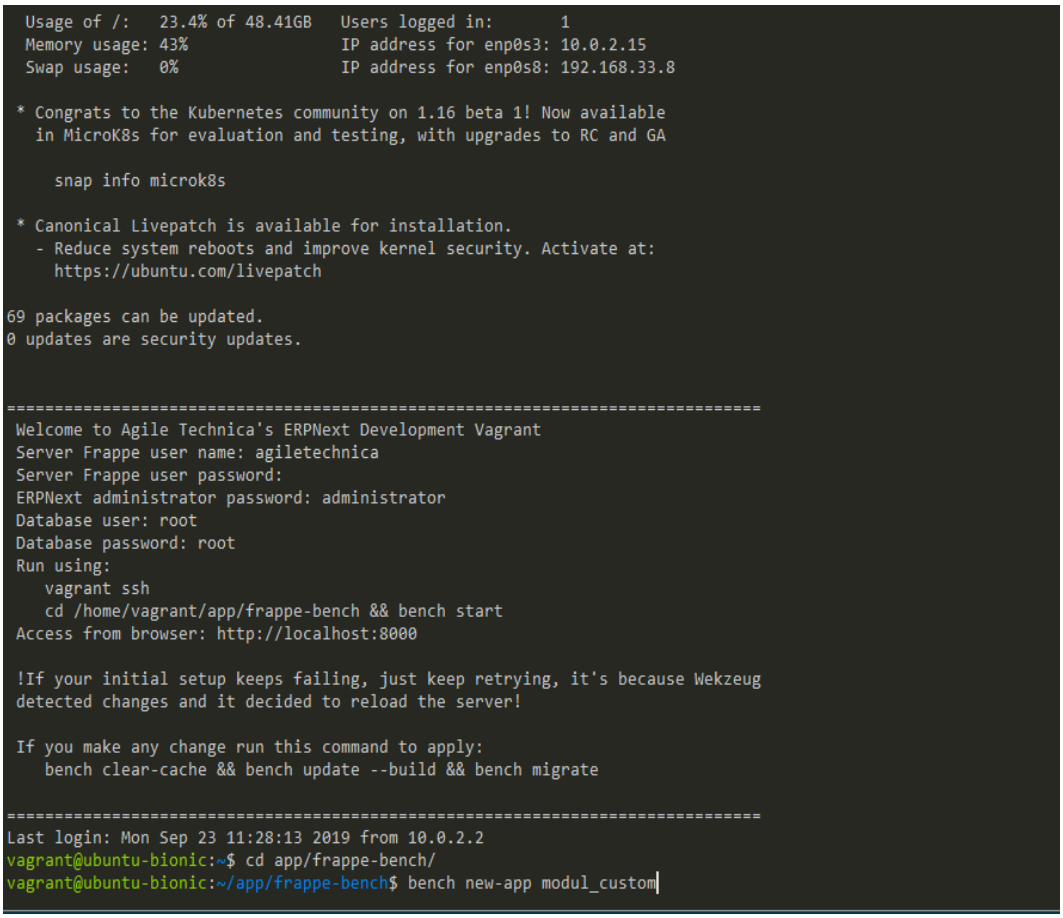
Gambar 3. 2 Tampilan Halaman Awal ERPNext pada Toko Cat Utama

Pada gambar 3.2, terdapat halaman awal ERPNext untuk Toko Cat Utama. Pekerjaan yang dilakukan oleh mahasiswa adalah melakukan *custom* pada

ERPNext tersebut. Pengembangan yang dilakukan mahasiswa untuk memenuhi kebutuhan klien yaitu:

1. Membuat *new-app*

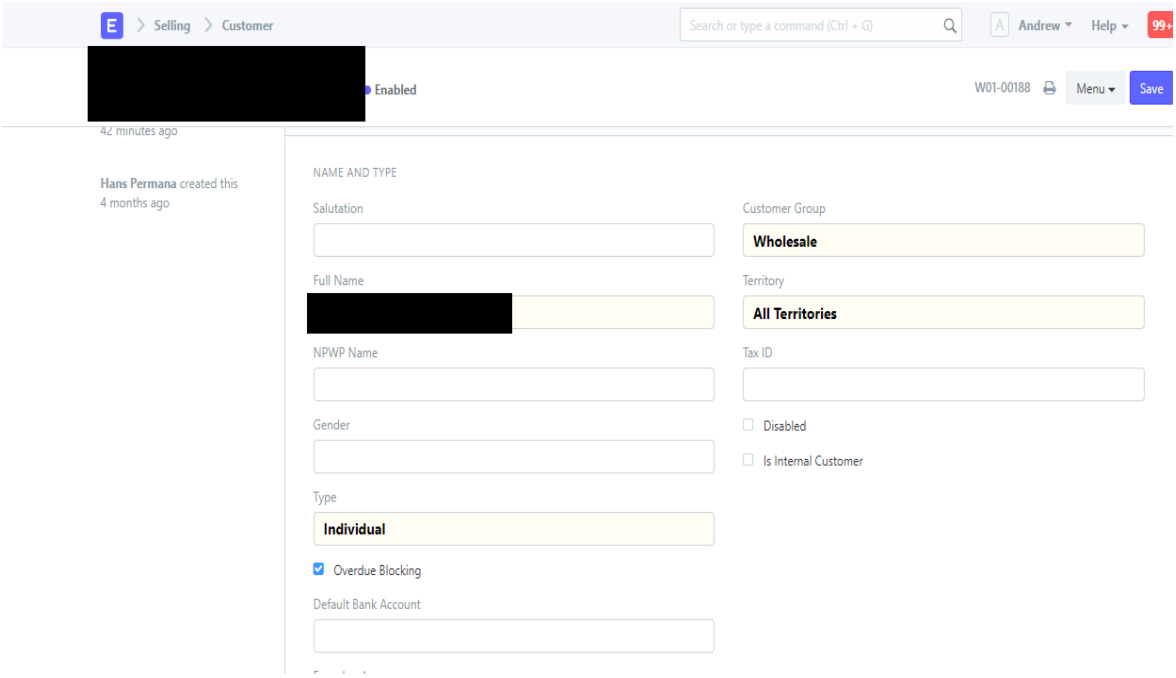
   Dengan menggunakan *framework frappe*. Mahasiswa dapat menggunakan fungsi dari *frappe* yaitu membuat *app* baru tanpa mengganti *core* yang ada pada ERPNext. *New-app* dilakukan ketika ingin membuat suatu modul *custom*.



Gambar 3. 3 Tampilan *Console Frappe*

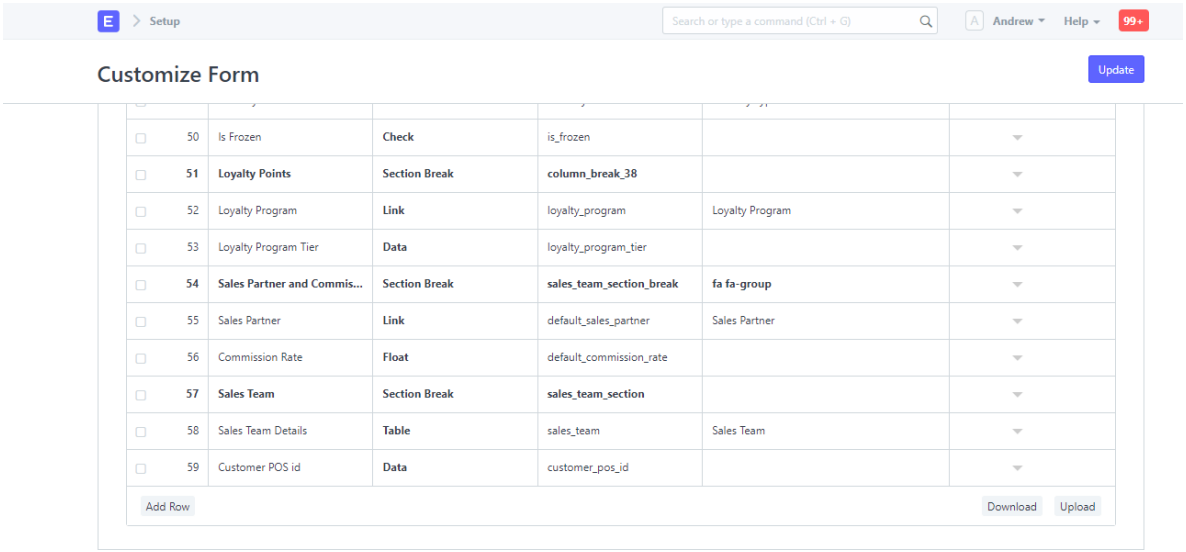
Pada gambar 3.3, "*bench new-app modul_custom*" berfungsi untuk membuat modul baru yang memiliki nama yaitu "modul_custom". Pada *app* tersebut mahasiswa dapat membuat suatu *doctype*. *DocType* atau Tipe Dokumen adalah alat untuk memasukkan formulir di ERPNext. Contohnya seperti *Sales Order*, *Work Order* ditambahkan sebagai *Doctype* di backend. Pada projek ini, *new-app* yang dibuat adalah "erpnext_overdue_invoice_sales_blocker". Setelah melakukan *bench new-app* tersebut. Maka *framework frappe* akan secara otomatis membuat *directory* serta *file* yang dibutuhkan.

2. Membuat *Custom Field* pada *Doctype Customer*



Gambar 3. 4 *Doctype Customer*

Pada gambar 3.4, mahasiswa harus membuat sebuah *custom field* untuk menampilkan *checkbox* dengan label "*Overdue Blocking"*. *Checkbox* tersebut berfungsi untuk menampilkan *customer* tersebut mempunyai *Sales Invoice* yang belum dibayar sesuai *payment terms* yang berlaku. Jika ada yang belum dibayar, maka *customer* tersebut akan ditandai dengan *checkbox overdue blocking*.



Gambar 3. 5 Menu *Customize Form* pada *Customer*

Berdasarkan gambar 3.5, mahasiswa dapat menambahkan *field* yang dibutuhkan pada *doctype customer* dengan pengaturan yang sudash disediakan pada ERPNext. Dengan klik *Add Row* pada pengaturan di atas.

hooks.py 3.61 KB

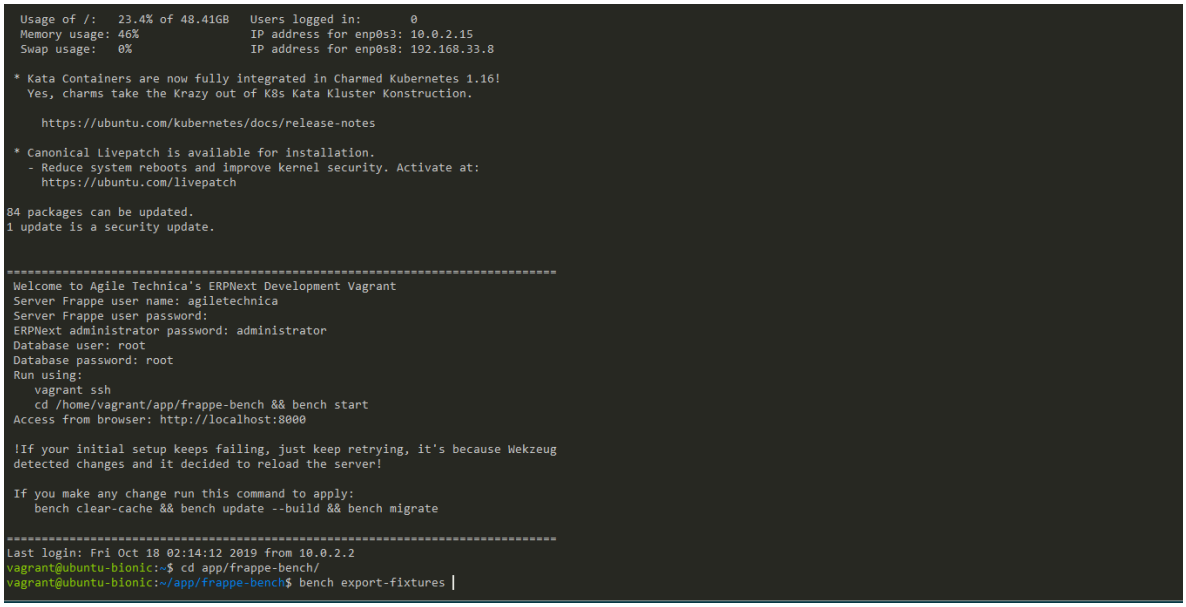
```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals
from . import __version__ as app_version

app_name = "erpnext_overdue_invoice_sales_blocker"
app_title = "Erpnext Overdue Invoice Sales Blocker"
app_publisher = "Agile Technica"
app_description = "This all will block invoice creation if the customer has overdue invoice"
app_icon = "octicon octicon-file-directory"
app_color = "grey"
app_email = "info@agiletechnica.com"
app_license = "MIT"
fixtures = ["Custom Field","Custom Script"]
doctype_js = {
        "Sales Order": "custom_scripts/Sales Order.js"
    }

scheduler_events = {
        "hourly": [
                "erpnext_overdue_invoice_sales_blocker.tasks.set_blocking_for_overdue"
        ]
}
```

Gambar 3. 6 *Script* hooks.py

Berdasarkan gambar 3.6, di tambahkan beberapa *code* pada *hooks.py* yaitu *file* python untuk melakukan *export* pada modul yang sudah dibuat tadi.

Gambar 3. 7 Menambahkan *Field*

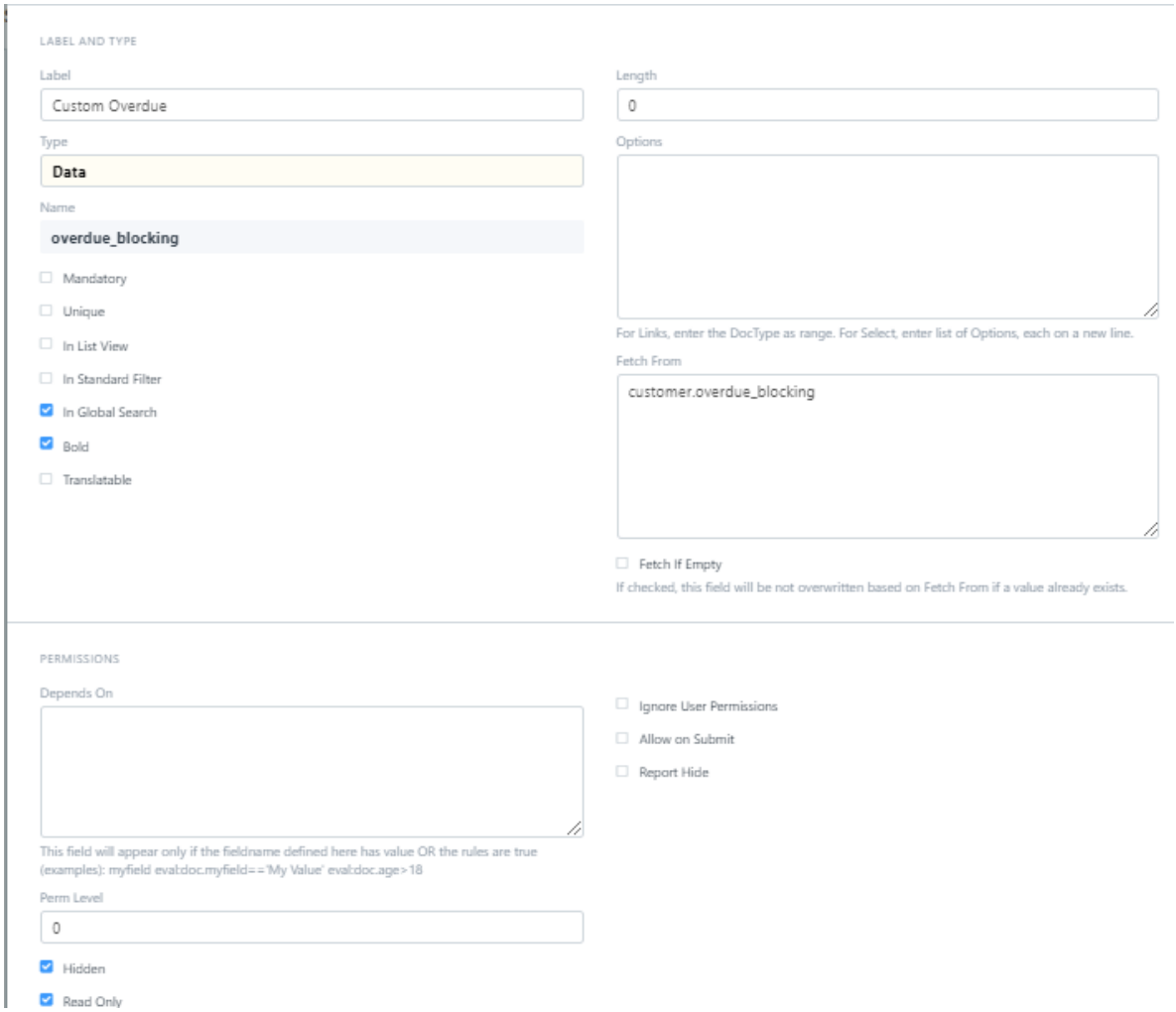
Berdasarkan gambar 3.7, mahasiswa membuat *field* yang dibutuhkan dengan mengisi label dan *type*. Setelah melakukan pembuatan *field*, mahasiswa harus memindahkan *custom field* yang ditambahkan sebelumnya pada *new-app* agar mengurangi kesalahan melakukan perubahan langsung pada *core* ERPNext.



Berdasarkan gambar 3.8, *bench export-fixtures* adalah suatu *command* dari *framework frappe* untuk melakukan *export* dapat berupa *field*, *custom script* dan lainnya.

3. Membuat *custom field* pada *Sales Order*

   Custom field pada *sales order* dibuat untuk membuat koneksi antara *sales order* dengan *customer*. Langkah-langkah yang dilakukan hampir sama dengan langkah pembuatan *custom field* pada *customer*. Perbedaannya pada field yang di butuhkan pada *sales order* berbeda dengan *customer*.



Gambar 3. 9 Tampilan Menambah *Field*

Pada gambar 3.9, mahasiswa membuat *field* pada *sales order* dengan beberapa perubahan seperti *type* "Data" , lalu memasukkan *fetch from* dengan *"customer.overdue_blocking"*. Maksud dari *fetch from* tersebut yaitu mengambil *value dari customer* yang berasal dari *doctype customer dan overdue_blocking* merupakan salah satu *field* yang terdapat pada *doctype customer*. *Field* dibuat *hidden* karena tujuannya yaitu menyambungkan *doctype customer* dengan *doctype sales order.* Setelah itu melakukan *bench export-fixtures* pada *command console.*

4. Membuat *custom script* untuk *sales order*

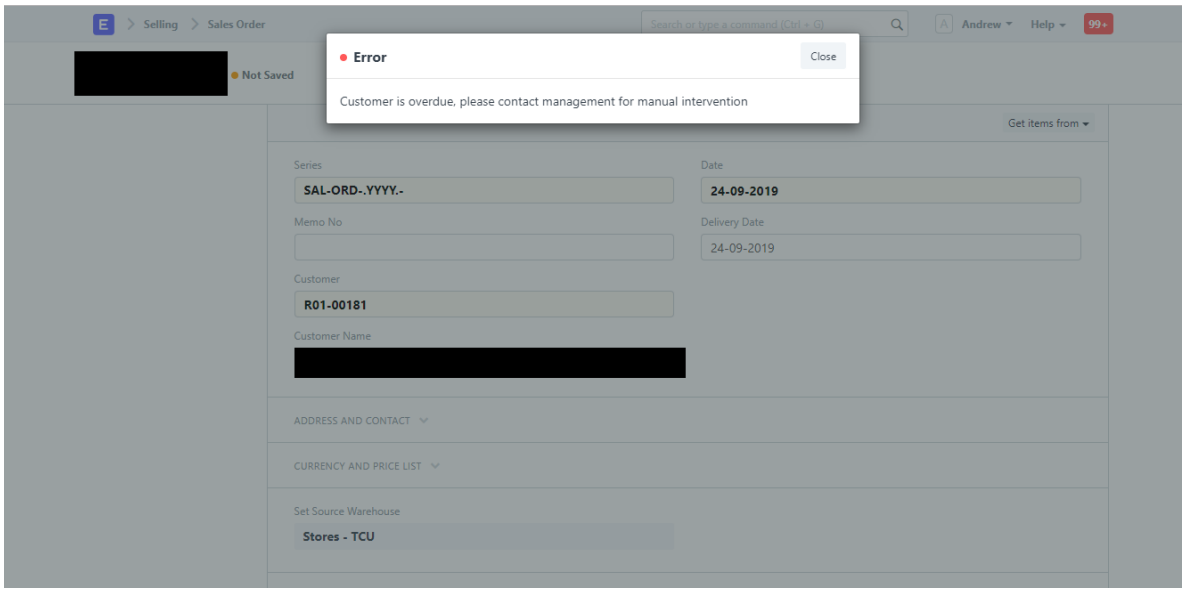
```
frappe.ui.form.on('Sales Order', {
        refresh: function(frm) {
                frm.enable_save();
                if (frm.doc.overdue_blocking == '1') {
                        frappe.msgprint({
                        title: __('Error'),
                        indicator: 'red',
                        message: __('Customer is overdue, please contact management for manual intervention')
                        })
            frm.disable_save();
                }
                else if (frm.doc.overdue_blocking == '0'){
                        frm.enable_save();
                }
        }

});
```

Gambar 3. 10 *Custom Script* pada *Sales Order*.js

Pada gambar 3.10, *custom script* yang dibuat berguna untuk mencegah para *customer* yang memiliki *sales invoice* melewati masa pembayarannya tidak dapat membuat *sales order* lagi.

Berdasarkan *code* yang dibuat jika *checkbox* pada *overdue blocking* adalah bernilai 1. Maka muncul sebuah *notification* yang berisi pesan pada *code* diatas serta tidak dapat menyimpan dokumen *sales order tersebut.*

Pada gambar 3.11, hasil dari *custom script* yang dibuatlah adalah seperti ini. *Customer* yang memiliki tunggakan pada *sales invoice* mendapatkan *notification* serta tombol untuk *save* tidak berfungsi.



Gambar 3. 11 Hasil dari *Custom Script*

3.3.3 Pengembangan aplikasi ERP untuk Coway International

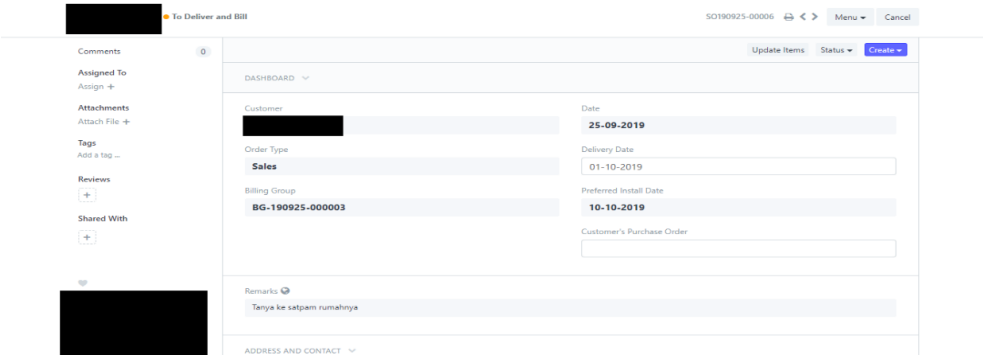
Coway International merupakan perusahaan yang menjual berbagai produk seperti *air purifier, water purifier,* dan lain-lain. Namun perusahaan tersebut memiliki masalah karena banyak penjualan. Perusahaan ingin membuat *mobile app* untuk penjualan serta ingin membuat sistem *ERP* untuk *backend*

pada *mobile app* tersebut. Mahasiswa mengerjakan bagian *backend* untuk projek Coway International.

```
4  from __future__ import unicode_literals
5
6  import frappe
7
8
9  def create_project(sales_order, method):
10     """
11     :param sales_order:
12     :param method:
13     :return: crete new project based on sales order
14     """
15     doc = frappe.new_doc('Project')
16     doc.project_name = construct_project_name(sales_order.name, sales_order.customer)
17     doc.status = 'Open'
18     doc.coway_installation_id = construct_project_name(sales_order.name, sales_order.customer)
19     doc.sales_order = sales_order.name
20     doc.customer = sales_order.customer
21     doc.insert()
22     # A reload is required because this document is opened. When not reloaded, an error
23     # "Error: Document has been modified after you have opened it" will appear
24     sales_order.reload()
25
```

Gambar 3. 12 *Code* membuat *Project*

Pada gambar 3.12, terlihat *code* yaitu "*frappe.new_doc('Project')*". *Code* tersebut berfungsi untuk membuat suatu dokumen baru pada *doctype project*. Dari *function* diatas dihasilkan *project* baru berdasarkan *sales order* yang sudah dibuat pada gambar 3.13 di bawah ini.

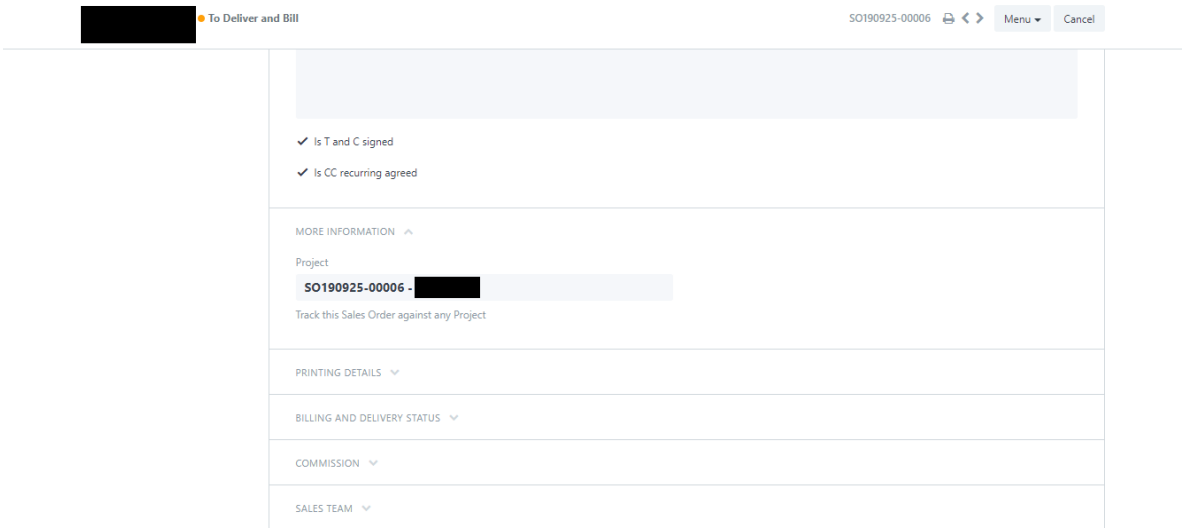


Gambar 3. 13 Tampilan Membuat *Sales Order*

```
def assign_project_to_sales_order(sales_order, method):
    """
    :param sales_order:
    :param method:
    :return: assign project to sales order after project created based on sales order
    """
    doc_projects = frappe.get_list('Project', filters={
        'coway_installation_id': construct_project_name(sales_order.name, sales_order.customer)})
    for doc_project in doc_projects:
        sales_order.project = doc_project.name
        sales_order.save()
```

Gambar 3. 14 *Code* untuk menyimpan *Project* pada *Sales Order*

Pada gambar 3.14, *code* diatas merupakan pencarian dokumen *project* berdasarkan nama dari *sales order* yang sudah dibuat sebelumnya. Setelah itu digunakan proses *looping* agar dapat mencari dokumen *project* yang sesuai dengan *sales order* masing-masing.



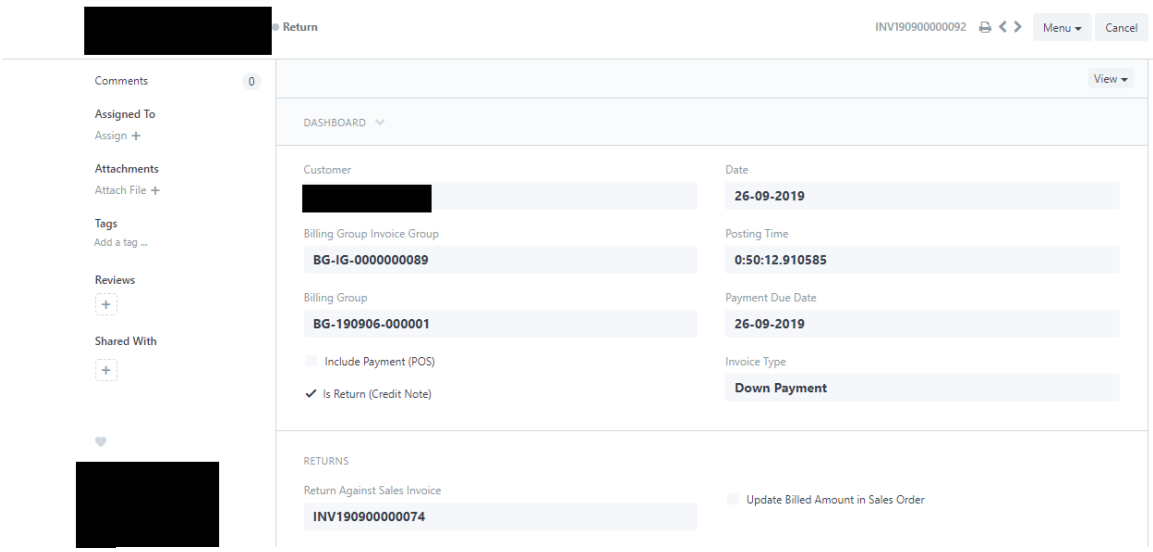
Gambar 3. 15 *Project* terlihat pada *Sales Order*

Pada gambar 3.15, terlihat hasil dari *project* yang sudah dibuat dimasukkan pada *sales order* yang sebelumnya di buat.

```
def auto_cancel_project_task(sales_invoice, method):
    """
    :param sales_invoice:
    :param method:
    :return: cancel project when sales return is 1
    """
    sales_invoice.reload()
    sales_order_number = ""
    for sales_invoice_item in sales_invoice.items:
        sales_order_number = sales_invoice_item.sales_order

    if sales_invoice.is_return == 1:
        doc_projects = frappe.get_list('Project', filters={
            'coway_installation_id': construct_project_name(sales_order_number, sales_invoice.customer)})

        for doc_project in doc_projects:
            project_doc = frappe.get_doc('Project', doc_project.name)
            project_doc.status = 'Cancelled'
            project_doc.save()

        doc_tasks = frappe.get_list('Task', filters={
            'coway_installation_id': "Installation: " + construct_project_name(sales_invoice.return_against,
                                                                               sales_invoice.customer)})

        for doc_task in doc_tasks:
            task_doc = frappe.get_doc('Task', doc_task.name)
            task_doc.status = 'Cancelled'
            task_doc.save()
    else:
        print("")
```

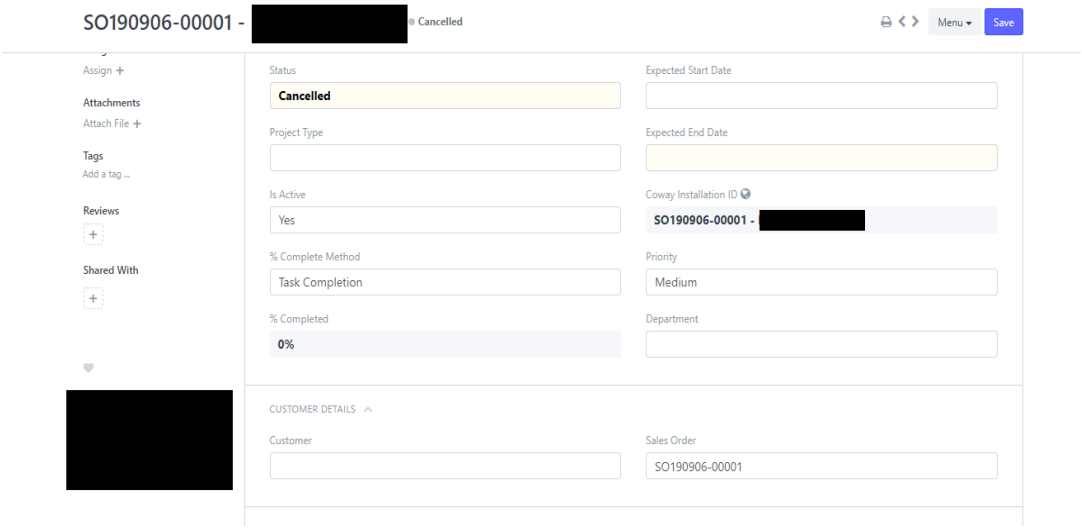
Gambar 3. 16 *Code Sales Invoice Return*

Pada gambar 3.16, *function* diatas aktif ketika *customer* ingin mengembalikan barang atau dapat juga dengan retur *sales invoice.* Jika *sales invoice* mempunyai nilai retur adalah 1, maka dokumen *project dan task* yang disambungkan dari *sales invoice* tersebut berubah statusnya dari *"Open"* menjadi "*Cancelled"*.



Gambar 3. 17 Dokumen *Sales Invoice Return*

Pada gambar 3.17, *sales invoice* memiliki nilai *"Is Return (Credit Note)"* adalah 1 serta status dokumen *sal*es *invoice* menjadi *"Cancelled"*.



Gambar 3. 18 Hasil dari *Sales Invoice Return*

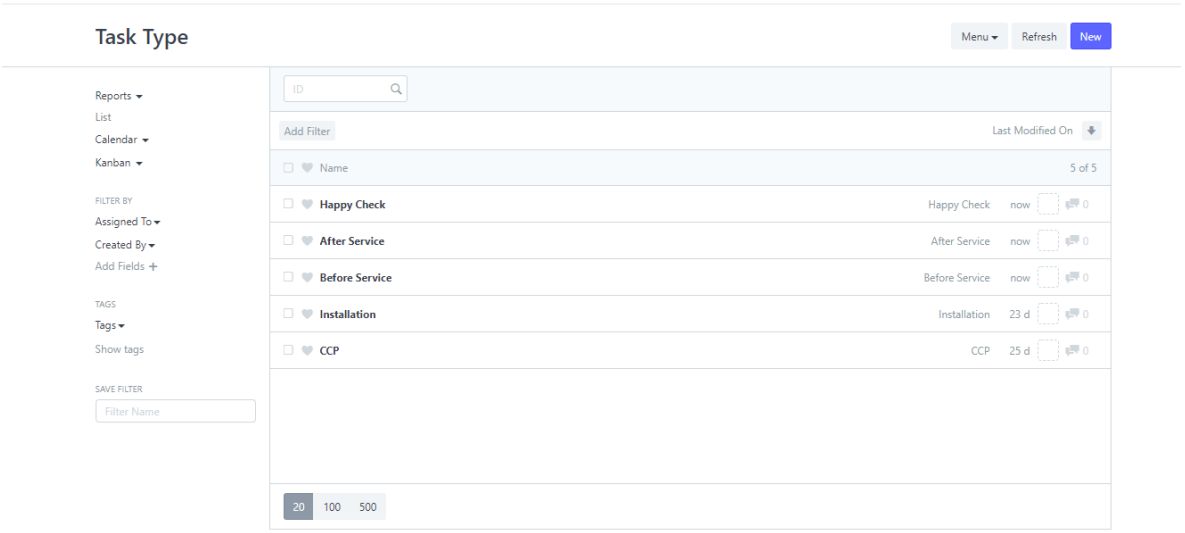
Pada gambar 3.18, Dokumen *sales invoice* terintegrasi dengan *sales order* yang sudah dibuat. Setelah itu, dokumen *project* berubah statusnya menjadi *"Cancelled"* karena *sales invoice* memiliki *is return* bernilai 1.

22 lines (17 sloc) | 514 Bytes

```
from __future__ import unicode_literals
import frappe


def create_task_type(task_name):
    if not frappe.db.exists("Task Type", task_name):
        return {
            frappe.get_doc({
                "doctype": "Task Type",
                "name": task_name
            }).insert()
        }

    return False


def after_install():
    create_task_type('CCP')
    create_task_type('Installation')
    create_task_type('Before Service')
    create_task_type('After Service')
    create_task_type('Happy Check')
```

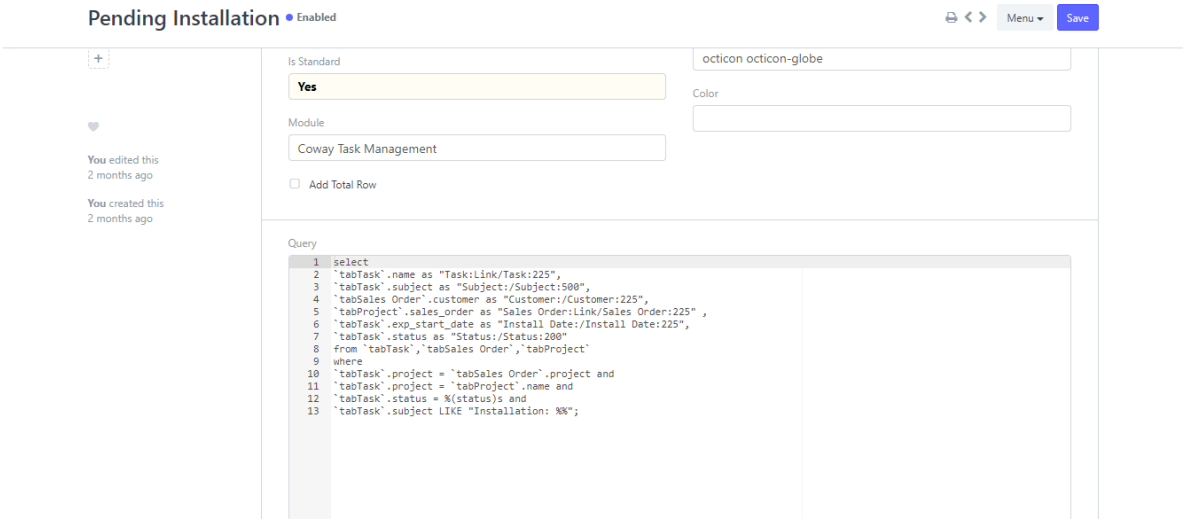
Gambar 3. 19 Code untuk menambahkan *Task Type*

Pada gambar 3.19 diatas, *function* yang terdapat pada *code* diatas berfungsi untuk menambahkan *record task type* setelah *install app* tersebut. *Function* akan berfungsi jika *task type* sebelumnya belum ada pada ERPNext. Jika sudah ada *task type* sebelumnya, maka *task type* tidak akan dibuat lagi.



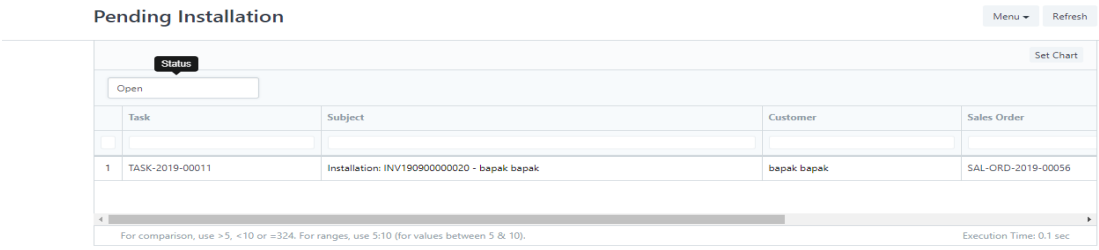
Gambar 3. 20 Hasil dari pembuatan *Task Type*

Pada gambar 3.20, *task type* terbuat seperti yang di harapkan dari *code* gambar 3.19 tersebut.



Gambar 3. 21 *Query Report*

Pada gambar 3.21, *Query* dibuat untuk menampilkan beberapa atribut yang dibutuhkan *customer service*.



Gambar 3. 22 Hasil dari *Query Report*

Pada gambar 3.22, terlihat hasil dari *Query Report* menunjukkan hanya beberapa *field* yang ingin di lihat dari berbagai *doctype* untuk mempermudah *customer service* melakukan pencarian.

```
from __future__ import unicode_literals

import re
from datetime import *

import frappe


def construct_project_name(invoice_doctype_name, customer_doctype_name):
    """
    :param invoice_doctype_name: sales invoice name
    :param customer_doctype_name:  sales invoice customer
    :return:
    """
    return invoice_doctype_name + " - " + customer_doctype_name


def get_customer_type(coway_credit_check, method):
    customer_doc = frappe.get_doc('Customer', coway_credit_check.customer_id)
    coway_credit_check.customer_type = customer_doc.customer_type

```

Gambar 3. 23 *Code* untuk mendapatkan *Customer Type*

Pada gambar 3.23, *function get_customer_type* diatas berfungsi untuk mendapatkan tipe dari *doctype customer* lalu menyimpannya pada *doctype coway credit check.*

```
def create_ccp_if_down_payment(sales_invoice, method):
    """
    Create CCP if new customer and don't check overdue for existing customer
    :param sales_invoice:
    :param method:
    :return:
    """
    sales_payment_settings_master = frappe.get_doc("Sales and Payment API Settings")
    for sales_invoice_item in sales_invoice.items:
        sales_order_doc = frappe.get_doc('Sales Order', sales_invoice_item.sales_order)

    ccp_exist = frappe.db.exists({
        'doctype': 'Coway Credit Check',
        'customer_id': sales_invoice.customer
    })

    if ccp_exist and sales_payment_settings_master.enable_multiple_ccp_customer_toggle == 0:
        print('CCP already exist')
    else:
        match = re.search('^direct debit', sales_order_doc.installments_payment_method, flags=re.IGNORECASE)
        if sales_invoice.invoice_type == 'Down Payment' and match and sales_order_doc.down_payment_month <= 18:
            ccp_doc = frappe.new_doc('Coway Credit Check')
            ccp_doc.customer_id = sales_invoice.customer
            ccp_doc.sales_invoice = sales_invoice.name
            for item in sales_invoice.items:
                ccp_doc.sales_order = item.sales_order
                ccp_doc.save()
        elif sales_order_doc.down_payment_month == 36 or sales_order_doc.down_payment_type == 'Full':
            print('No need ccp')

```

Gambar 3. 24 *Function create_ccp_if_down_payment*

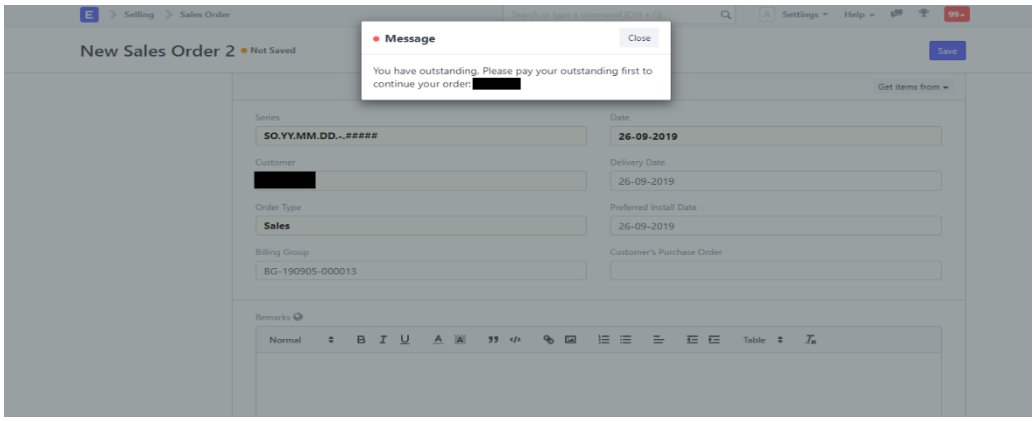
Pada gambar 3.24, *function create_ccp_if_down_payment* ini berfungsi untuk mengecek apakah *sales order* memiliki bulan dp lebih rendah dari 18, *invoice type* yaitu *"Down Payment"*, dan cek *customer* menggunakan

"*Direct Debit"* dari jenis bank apapun atau tidak. Jika kondisi terpenuhi maka dibuatnya lah dokumen *CCP*.

```
def block_overdue(sales_order, method):
    """
    Check existing customer overdue for next order
    :param sales_order:
    :param method:
    :return:
    """
    so_exist = frappe.db.exists({
        'doctype': 'Sales Order',
        'customer': sales_order.customer
    })

    if so_exist and sales_order.down_payment_type == "Partial":
        sales_invoice_docs = frappe.get_list('Sales Invoice',
                                             filters={'customer': sales_order.customer, 'status': 'overdue'})
        for sales_invoice_doc in sales_invoice_docs:
            doc_invoice = frappe.get_doc('Sales Invoice', sales_invoice_doc.name)
            customer_doc = frappe.get_doc('Customer', sales_order.customer)
            if doc_invoice.customer == customer_doc.name:
                if customer_doc.customer_type in ['Individual', 'Individual - WNI', 'Individual - WNA']:
                    ccp_docs = frappe.get_list('Coway Credit Check', filters={'customer_id': sales_order.customer})
                    for ccp in ccp_docs:
                        ccp_doc = frappe.get_doc('Coway Credit Check', ccp.name)
                        if doc_invoice.status == 'Overdue' or ccp_doc.credit_check_status == 'Pending' or ccp_doc.credit_check_status == 'R
                            frappe.throw('You have outstanding, Please pay your outstanding first to continue your '
                                         'order. ' + doc_invoice.name)
                elif customer_doc.customer_type == 'Company':
                    today_date = date.today()
                    total_overdue_month = abs(today_date.month - doc_invoice.due_date.month)
                    total_overdue_day = (today_date.day - doc_invoice.due_date.day)
                    ccp_docs = frappe.get_list('Coway Credit Check', filters={'customer_id': sales_order.customer})
                    for ccp in ccp_docs:
                        ccp_doc = frappe.get_doc('Coway Credit Check', ccp.name)
                        if total_overdue_month >= 2 and total_overdue_day <= 0 and doc_invoice.status != 'Cancelled' or ccp_doc.credit_chec
                            frappe.throw('You have outstanding, Please pay your outstanding first to continue your '
                                         'order. ' + doc_invoice.name)
```

Gambar 3. 25 *Function block_overdue*

Pada gambar 3.25, *function block_overdue* secara singkat berfungsi untuk memeriksa apakah *customer* mempunyai tunggakan pada pesanan sebelumnya atau tidak. Pada *function* tersebut, *customer* yang mempunyai *type individual* dan memiliki status *overdue* tidak dapat memesan barang serta mendapat *notification*. Untuk *type company* jika waktu *overdue* melebihi jangka waktu dua bulan maka tidak dapat memesan barang.



Gambar 3. 26 Hasil dari *block_overdue*

Pada gambar 3.26, terlihat notifikasi yang berisi "*You Have outstanding, Please pay your understanding first to continue your order*: (*customer_name)*". Dikarenakan *customer* tersebut memiliki tunggakan yang dijelaskan pada gambar 3.25 yang membuat *customer* tersebut tidak dapat memesan barang.

```
def create_installation_task():
    """
    Create installation task with background jobs if ccp exists(also approve) and payment exist
    or ccp not exists and payment exist
    :return:
    """
    created_tasks = []

    sales_invoices = frappe.db.sql(
        "SELECT name from `tabSales Invoice` as si "
        "WHERE si.creation > (CURDATE() - INTERVAL 90 DAY) "
        "AND si.name not in (SELECT sales_invoice from `tabTask` WHERE creation > (CURDATE() - INTERVAL 90 DAY)) "
        "ORDER BY si.creation ASC",
        as_dict=True)

    for sales_invoice in sales_invoices:
        sales_invoice_doc = frappe.get_doc('Sales Invoice', sales_invoice.name)
        for item in sales_invoice_doc.items:

            # sales_order_docs = frappe.get_all('Sales Order')
            # for sales_order in sales_order_docs:
            sales_order_doc = frappe.get_doc('Sales Order', item.sales_order)

            project_name = sales_order_doc.project

            installation_task_subject = "Installation: " + project_name + " " + sales_order_doc.name
            coway_installation_id = installation_task_subject

            if coway_installation_id in created_tasks:
                continue

            task_exists = frappe.db.exists({
                'doctype': 'Task',
                'subject': installation_task_subject
            })
```

Gambar 3. 27 *Function* untuk membuat *installation task*

Pada gambar 3.27, *function create_installation_task* berawal dari *query database* berisi *sales invoice* yang tanggal pembuatannya lebih besar dari (*current date – interval 90day)*. Setelah itu *sales invoice* yang sudah di *query* tersebut di *looping* lagi untuk mendapatkan dokumen *sales order* yang saling berhubungan. *Function* ini mempunyai fungsi yaitu membuat *installation task* jika dokumen *CCP* mempunyai status "*Approve"* serta adanya bukti pembayaran.

```
customer_doc = frappe.get_doc("Customer", sales_invoice_doc.customer)

ccp_docs = frappe.get_all('Coway Credit Check', fields=["name", "credit_check_status"],
                          filters={'customer_id': sales_invoice_doc.customer,
                                   'sales_invoice': sales_invoice_doc.name})
proceed_create_installation_task = False

# COW-326 for company we ignore downpayment, we create task as soon as the CCP is approved
if customer_doc.customer_type == "Company":
    if sales_invoice_doc.invoice_type == "Outright":  # no need for CCP if it's outright
        proceed_create_installation_task = True
    else:
        for ccp in ccp_docs:
            if ccp.credit_check_status == 'Approved':
                proceed_create_installation_task = True

else:
    # we check advances if this is an individual customer
    for advance in sales_invoice_doc.advances:
        payment_entry_doc = frappe.get_doc('Payment Entry', advance.reference_name)

        # check if reference existed
        for references in payment_entry_doc.references:

            if not ccp_docs and references.reference_doctype == 'Sales Order' and \
                    payment_entry_doc.docstatus == 1:
                proceed_create_installation_task = True

            else:
                for ccp in ccp_docs:
                    if ccp.credit_check_status == 'Approved' and \
                            references.reference_doctype == 'Sales Order' and payment_entry_doc.docstatus == 1:
                        proceed_create_installation_task = True

if proceed_create_installation_task:
    insert_installation_task_doc(installation_task_subject, coway_installation_id,
                                 project_name, sales_order_doc.preferred_install_date,
                                 sales_order_doc.preferred_install_date, sales_order_doc,
                                 sales_invoice_doc)
    created_tasks.append(coway_installation_id)
```

Gambar 3. 28 Lanjutan dari *function* membuat *installation task*

Berdasarkan gambar 3.28, jika *customer* melakukan pembayaran sekaligus atau *"Outright"* maka *installation task* akan terbuat langsung khususnya untuk *customer type company*. Tetapi jika *customer type company* membayar selain "*Outright"*, maka dokumen *CCP* harus di cek terlebih dahulu apakah statusnya "*Approve*" atau tidak. Sedangkan, untuk *customer type individual* di cek apakah dokumen *CCP* mempunyai status "Approve" dan sudah dibayar sebesar "Down Payment" jika menggunakan "Down Payment".

```
192 scheduler_events = {
193     "hourly": [
194         "coway_sales_enhancements.tasks.auto_cancel_sales_invoice",
195         "coway_sales_enhancements.tasks.assign_blocked_overdue"
196     ],
197     "cron": {
198         "0/10 * * * *": [
199             "coway_sales_enhancements.tasks.create_installation_task"
200         ]
201     }
202 }
```

Gambar 3. 29 *Scheduler Event* untuk membuat *Installation Task*

Pada gambar 3.29, fungsi dari *scheduler event* ini yaitu untuk melakukan *function* secara otomatis oleh waktu yang dipasang. "0/10 * * * *" merupakan waktu yang dipasang untuk *function* tersebut adalah sepuluh menit.

```
4  from __future__ import unicode_literals
5
6  import frappe
7
8
9  def create_warehouse(warehouse_name, warehouse_type, account, parent_warehouse, employee_id):
10     new_warehouse = frappe.get_doc({
11         "doctype": "Warehouse",
12         "warehouse_name": warehouse_name,
13         "warehouse_type": warehouse_type,
14         "account": account,
15         "parent_warehouse": parent_warehouse,
16         "employee_id": employee_id
17     }).insert(True)
18
19     return new_warehouse
20
```

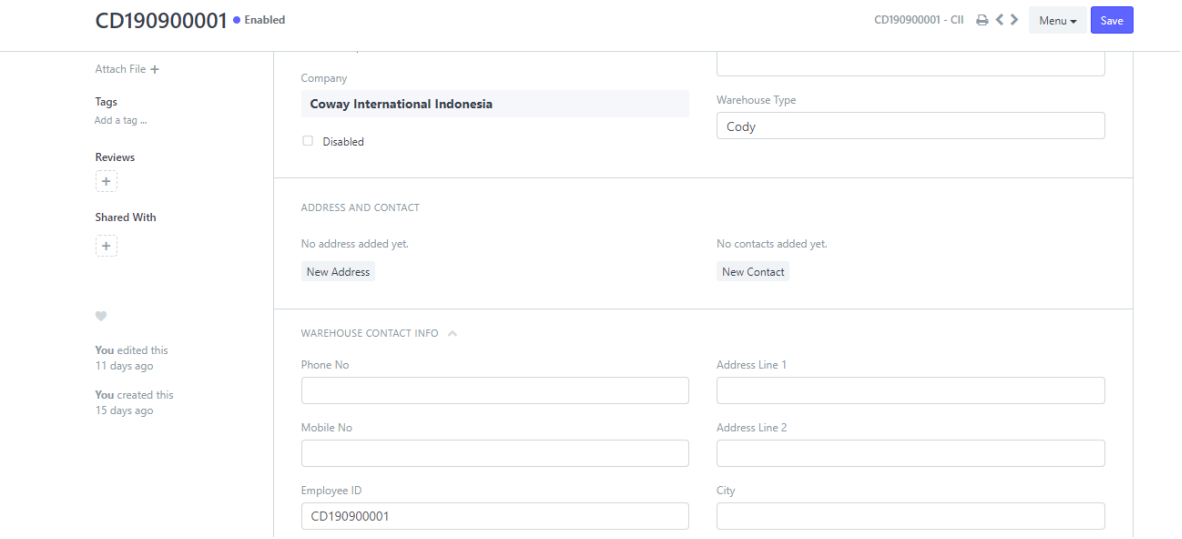
Gambar 3. 30 *Function* untuk membuat *warehouse*

Pada gambar 3.30, pada kasus kali ini klien meminta untuk satu karyawan atau *employee* mempunyai satu "*warehouse virtual*". *Function create_warehouse* berfungsi untuk membuat *warehouse* pada ERPNext.

```
def assign_cody_ce_to_warehouse():
    stock_api_settings = frappe.get_doc('Stock API Settings')

    if stock_api_settings.enable_account_and_parent_warehouse_settings == 1:
        employee_docs = frappe.get_all('Employee', filters=[{'designation': ("in", ["Cody", "Coway Technician"])}, ['warehouse', '=', '']])
        for employee in employee_docs:
            employee_doc = frappe.get_doc('Employee', employee.name)
            try:
                if employee_doc.designation == 'Cody':
                    employee_warehouse = create_warehouse \
                        (employee_doc.name, employee_doc.designation, stock_api_settings.account,
                         "Cody - CII", employee_doc.name)
                    employee_doc.warehouse = employee_warehouse.name
                    employee_doc.save()
                else:
                    employee_warehouse = create_warehouse \
                        (employee_doc.name, employee_doc.designation, stock_api_settings.account,
                         "CT - CII", employee_doc.name)
                    employee_doc.warehouse = employee_warehouse.name
                    employee_doc.save()
            except:
                pass
```

Gambar 3. 31 *Function assign_cody_ce_to_warehouse*

Pada gambar 3.31, *function assign_cody_ce_to_warehouse* berfungsi untuk mencari *employee* yang mempunyai *designation* yaitu “Cody” atau “Coway Technician” untuk dibuatkan *warehouse virtual*.



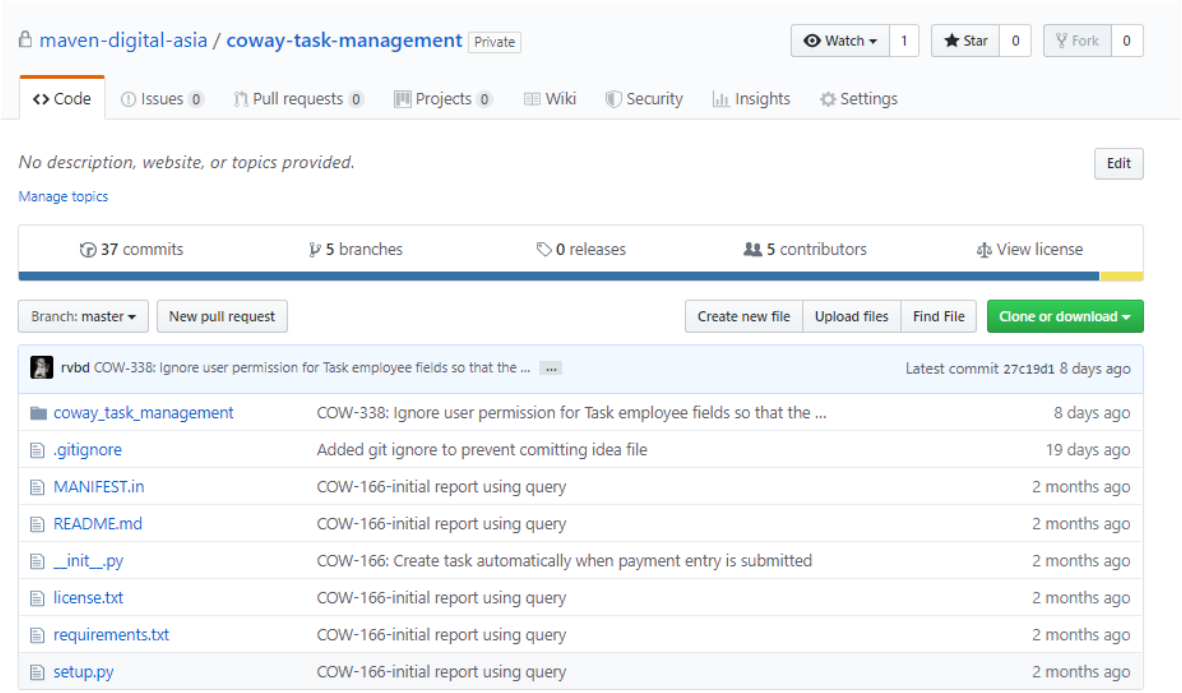
Gambar 3. 32 Hasil dari pembuatan *Warehouse*

Pada gambar 3.32, *warehouse* yang berhasil dibuat akan mempunyai hasil seperti gambar berikut. *Warehouse* ini mempunyai koneksi dengan *employee* yang mempunyai *designation* yaitu “Cody”.

```
scheduler_events = {
    "cron": {
        "0/10 * * * *": [
            "coway_stock_enhancements.tasks.assign_cody_ce_to_warehouse"
        ]
    }
}
```

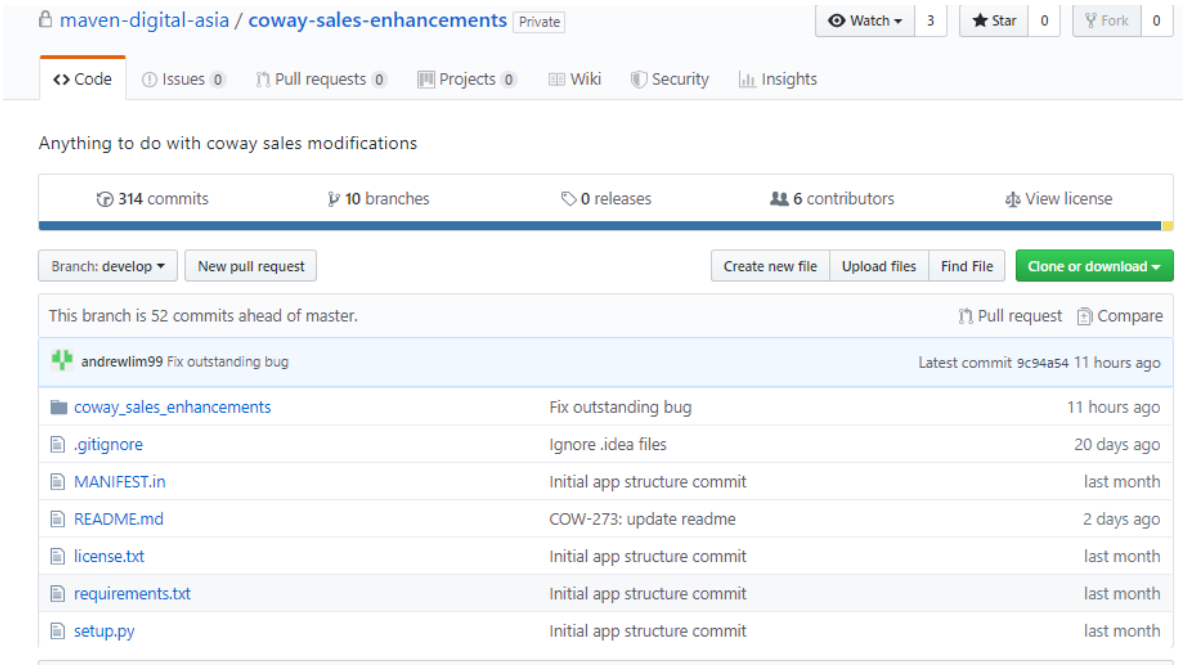
Gambar 3. 33 *Scheduler Event* pada *Warehouse*

Pada gambar 3.33, *function assign_cody_ce_to_warehouse* dijalankan dengan menggunakan *scheduler event* dalam jangka waktu sepuluh menit.



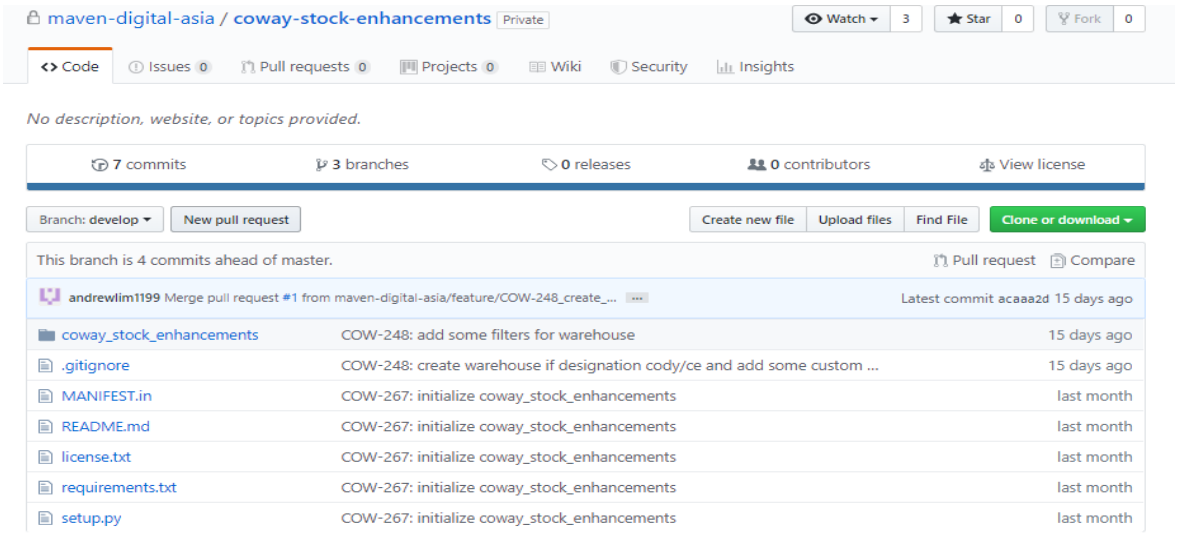
Gambar 3. 34 Github untuk penyimpanan *code* coway_task_management

Pada gambar 3.34, *code* yang sudah dibuat disimpan pada Github. Github coway_task_management tersebut berisi *code* yang berhubungan dengan pembuatan *installation task* pada projek ini.



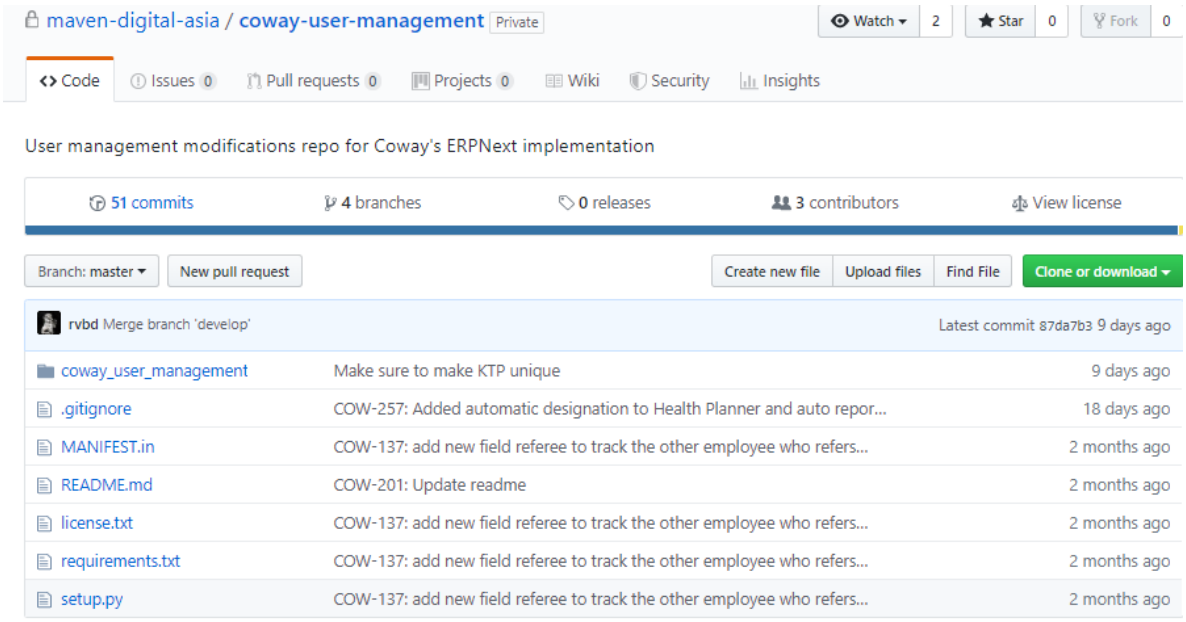
Gambar 3. 35 Github penyimpanan *code* untuk coway_sales_enhancements

Pada gambar 3.35 ini, Github coway_sales_enhancements tersebut berisi *code* yang berisi dokumen *CCP*, *custom API* untuk *sales order*, dan *code* yang berkaitan dengan penjualan pada projek ini.



Gambar 3. 36 Github penyimpanan *code* untuk coway_stock_enhancements

Pada gambar 3.36, Github coway_stock_enhancements tersebut berisi *code* yang berkaitan dengan aktivitas gudang seperti *warehouse*.



Gambar 3. 37 Github penyimpanan *code* untuk coway_user_management

Pada gambar 3.37, Github coway_user_management tersebut berisi *code* yang berkaitan dengan *user* atau *employee*.

3.4 Kendala yang dihadapi

Kendala yang dihadapi mahasiswa selama melakukan program kerja magang pada PT Sumber Inovasi Informatika yakni:

1. Kurang pengetahuan akan bahasa pemrograman python dan javascript.
2. Koneksi internet yang kurang stabil.
3. Spesifikasi *desktop* yang tinggi untuk ERPNext.

3.5 Solusi atas Kendala yang Dihadapi

Didapatkan solusi atas kendala yang dihadapi mahasiswa pada program kerja magang pada PT Sumber Inovasi Informatika yakni:

1. Mahasiswa melakukan *online course* seperti pada https://www.codecademy.com tentang python dan javascript.
2. Menggunakan *hotstpot* pribadi untuk mendapatkan koneksi yang lebih stabil.
3. Perusahaan membelikan *desktop* yang sesuai spesifikasi untuk aplikasi yang dikembangkan.