BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kedudukan dalam perusahaan selama melakukan kerja magang adalah sebagai *back end programmer* pada divisi *product development*. Pada saat pertama kali bergabung dengan Jurnal, tugas pertama yang diberikan adalah mempelajari *ruby on rails* sebagai bahasa pemrograman utama dan *bitbucket* sebagai media *versioning* hasil kerja, di bawah pengawasan Bapak Daniel Witono selaku *head engineer* dan *supervisor* dari divisi *product development*.

Setelah mempelajari bahasa pemrograman *ruby on rails,* kedudukan yang diberikan adalah sebagai *back-end programmer* pada *website company profie* dan mendapatkan tugas untuk membuat *admin page* pada *website company profile* www.jurnal.id. Setelah *admin page* pada *website company profile* berhasil dibuat, tugas berikutnya yang diberikan adalah sebagai *back end programmer* pada aplikasi Jurnal dan menambahkan *chart* pada *admin page* aplikasi Jurnal.

Setelah mendapatkan materi tentang bisnis proses yang ada pada aplikasi Jurnal, tugas berikutnya yang diberikan adalah mengembangkan aplikasi Jurnal dengan membuat submodul *purchase order* dan *deposit*.

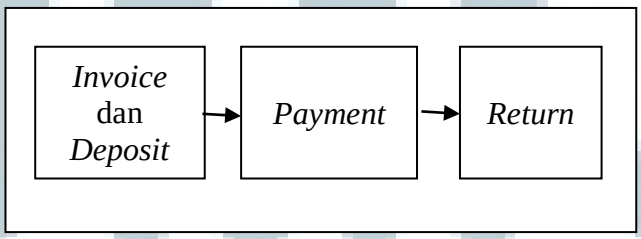
3.2 Tugas yang Dilakukan

Tugas pertama yang diberikan adalah membuat *admin page* pada *website company profile* dan aplikasi Jurnal. Dalam *admin page* terdapat informasi umum mengenai perusahaan dan pengguna, sedangkan informasi yang bersifat *private* tidak ditampilkan.

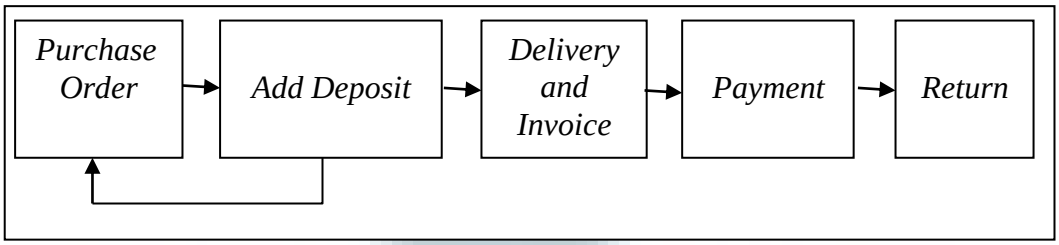
Setelah *admin page* berhasil dibuat, tugas berikutnya yang diberikan adalah membuat *chart* pada *admin page*. *Chart* ini berguna untuk menunjukkan perkembangan Jurnal dari sisi statistik penggunaan aplikasi Jurnal.

Setelah berhasil membuat *chart* pada *admin page* dan mendapatkan pengenalan bisnis oleh Bapak Anthony Kosasih, tugas berikutnya adalah menambahkan fitur *purchase order* dan *deposit* ke dalam transaksi pembelian produk. Pada fitur *purchase order*, stok barang tidak akan bertambah karena belum terjadi pengiriman barang, serta dapat melakukan pembayaran uang muka (*deposit)* lebih dari satu kali jika belum terdapat tagihan. *Deposit* yang dimaksud adalah satu jenis transaksi tersendiri dan hanya dapat dibuat berdasarkan *PO*.

Pada Gambar 3.1 dan Gambar 3.2 dapat dilihat perbedaan sebelum dan sesudah menambahkan submodul *PO* dan *deposit* ke dalam modul transaksi pembelian. Pada Gambar 3.1, setiap pembelian dianggap barang sudah diterima dan mendapatkan tagihan, sehingga stok barang pada gudang akan bertambah dan akan masuk ke dalam akun *payable*. Sedangkan pada Gambar 3.2, saat terjadi *PO* dan *add deposit*, stok pada gudang tidak bertambah jika tidak ada *invoice* dengan nomor *PO* yang bersangkutan, serta semua transaksi yang terjadi pada *PO* dan *add deposit* akan dicatat dalam akun *prepayment*, bukan akun *payable*.

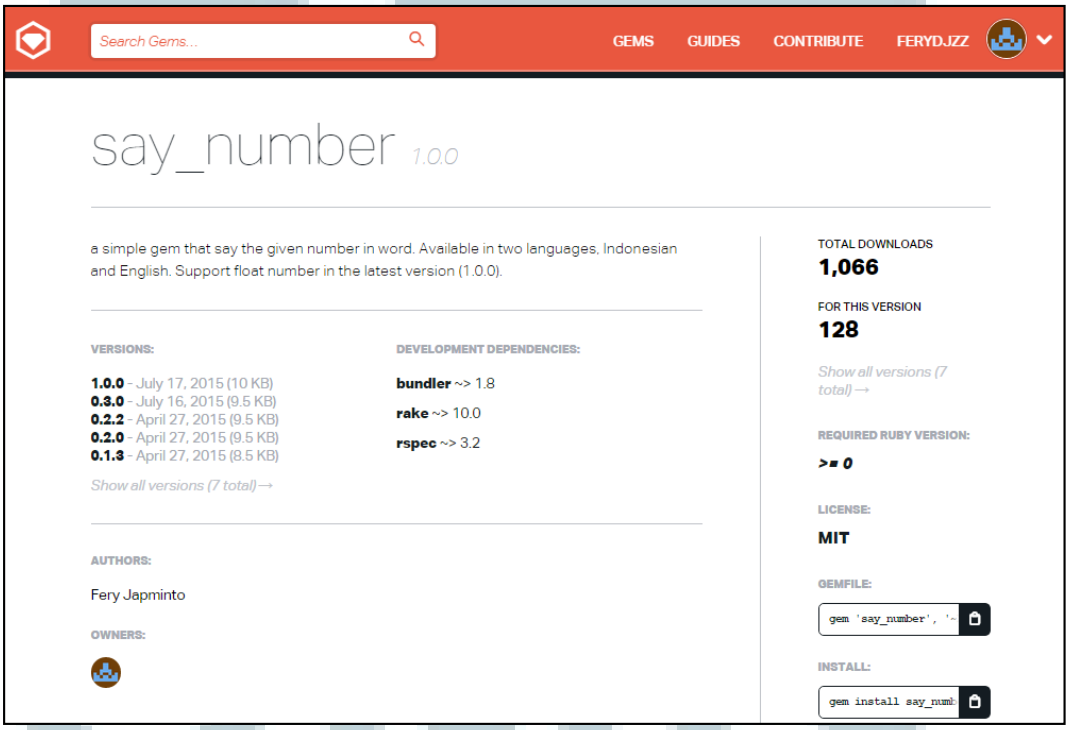


Gambar 3.1 Proses Transaksi Pembelian tanpa *PO* dan *Deposit*



Gambar 3.2 Proses Transaksi Pembelian dengan *PO* dan *Deposit*

Tugas berikutnya adalah membuat *gem say_number* yang berguna untuk meng-*generate* terbilang dari sejumlah angka yang dimasukkan. *Gem* ini digunakan dalam pembuatan *invoice* (*print invoice*). *Gem* atau *gemfile* merupakan istilah dalam *ruby*, yang berarti *library open source* yang dapat di-*upload* ke situs www.rubygems.org, dan dapat digunakan oleh siapa saja.



Gambar 3.3 Gem Say_Number pada Situs Rubygems.org

3.3 Uraian Pelaksanaan Kerja Magang

Pekerjaan yang dilakukan selama kerja magang di PT Jurnal Consulting Indonesia dapat dijelaskan melalui penjelasan dalam *admin pages* dan *chart*, *flowchart*, *Data Flow Diagram* (*DFD*), *Entity Relationship Diagram* (*ERD*), struktur tabel, dan hasil implementasi.

3.3.1 Admin Pages dan Chart

Admin pages dibuat pada *website company profile* dan aplikasi Jurnal dengan tujuan mempermudah tim Jurnal. Selain itu, terdapat tiga jenis *chart* yang dibuat pada *admin pages* aplikasi Jurnal, yaitu *chart login, chart unique login* dan *chart register*. *Chart* tersebut memiliki *filter* yang dapat menampilkan data berdasarkan *daily*, *weekly*, dan *monthly*. Dengan adanya *chart* ini, tim Jurnal dapat memonitor perkembangan aplikasi Jurnal.

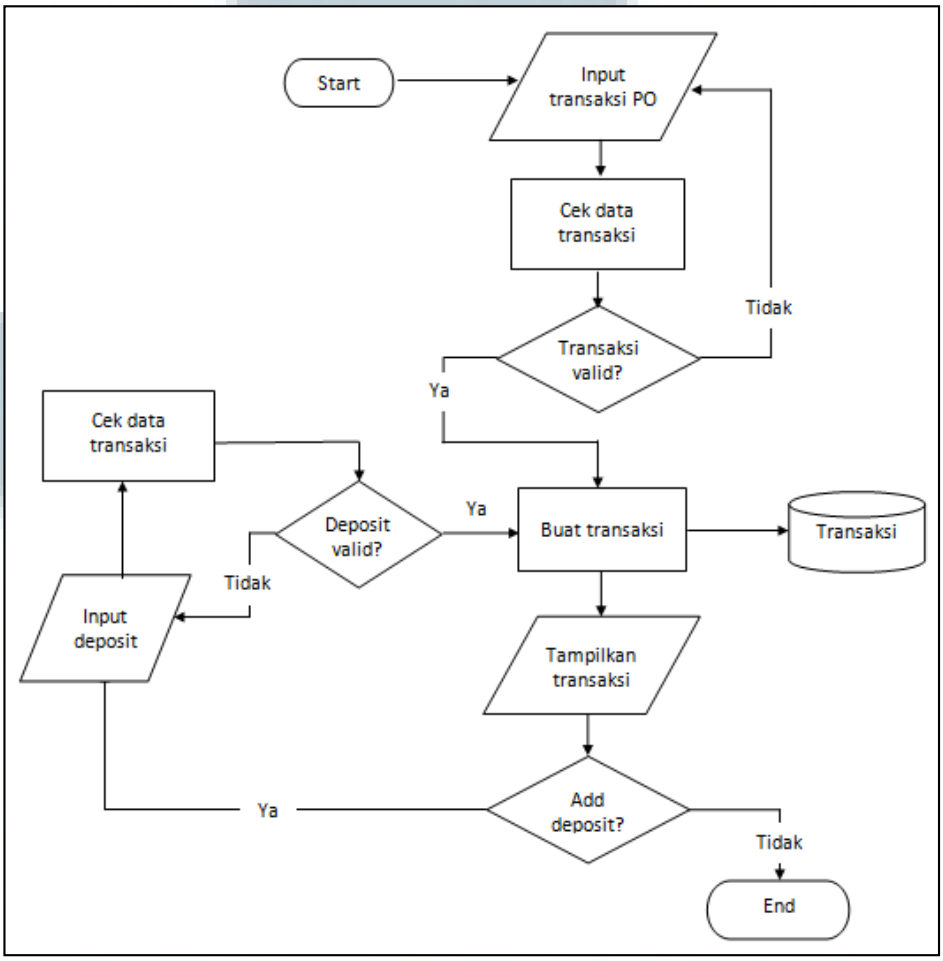
Informasi pada *chart login* adalah jumlah total *login* yang terjadi pada aplikasi Jurnal, sedangkan pada *chart unique login* hanya menghitung jumlah pengguna yang *login* dengan mengabaikan jumlah *login* yang dilakukan per pengguna. Pada *chart register*, data diambil dari jumlah register yang terjadi pada aplikasi Jurnal.

3.3.2 Flowchart dan Data Flow Diagram

Pada *Flowchart* akan digambarkan bagaimana alur *user* untuk menggunakan submodul *purchase order* dan *deposit* sedangkan pada *Data Flow Diagram* (*DFD*) akan dijelaskan mengenai alur data pada aplikasi Jurnal. Berikut adalah penjabarannya.

1. *Flowchart*

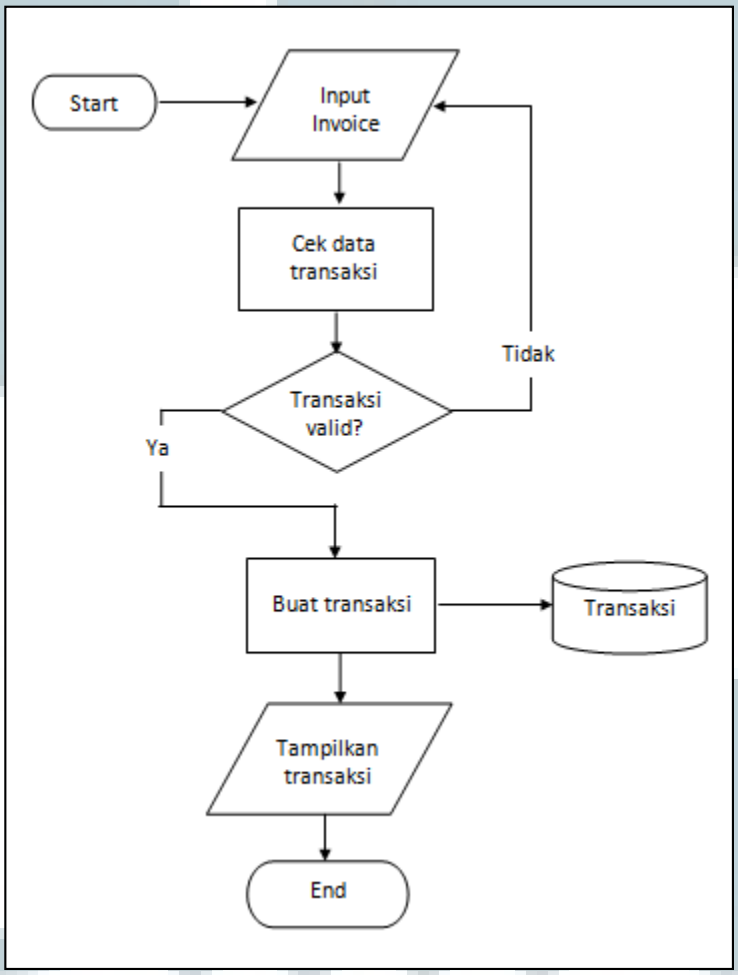
Flowchart digunakan untuk menjelaskan proses dalam submodul *purchase order* dan *deposit* yang dapat dilihat pada Gambar 3.4, serta penerapannya dalam modul transaksi pembelian yang dapat dilihat pada Gambar 3.5 dan Gambar 3.6.



Gambar 3.4 *Flowchart* Submodul *PO* dan *Deposit PO*

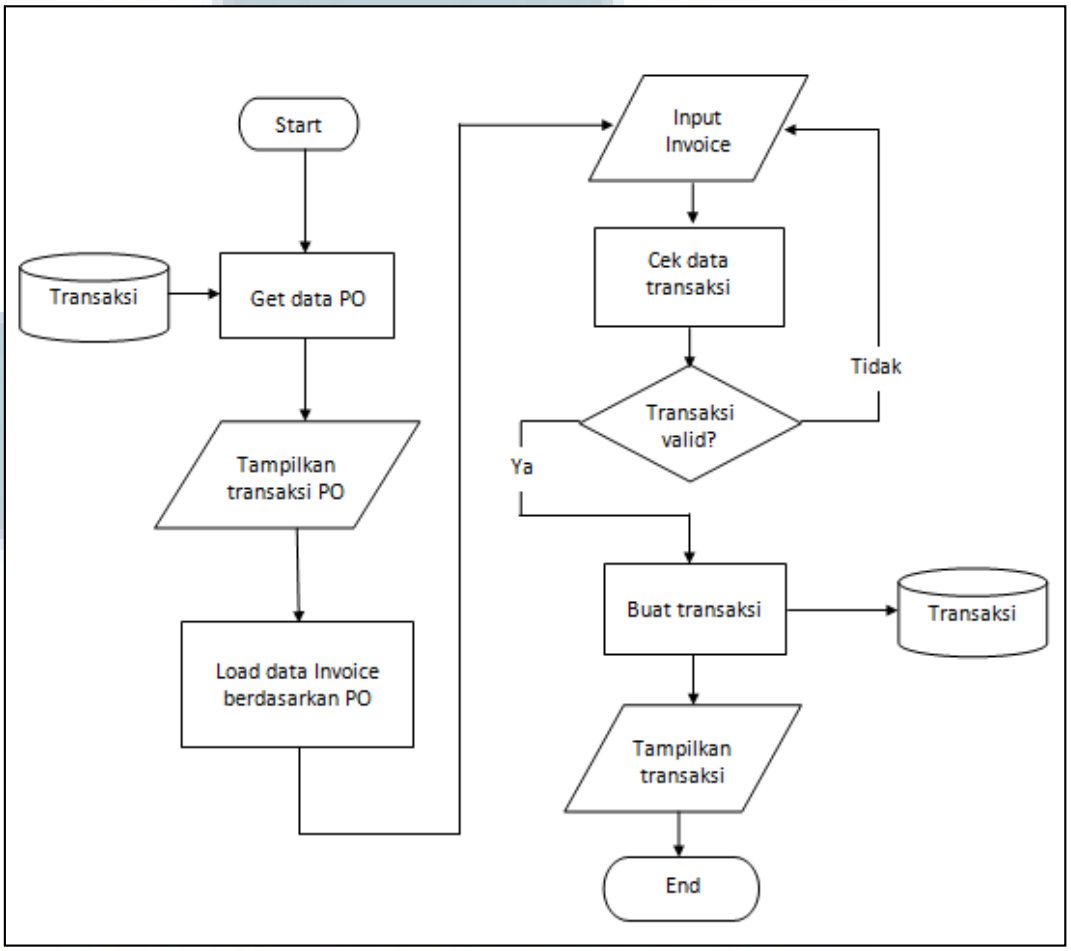
Pada Gambar 3.4 dijelaskan proses yang terjadi pada saat membuat *PO* dan dapat membuat *deposit* berdasarkan *PO* yang dibuat. *Deposit PO* bersifat *optional* dan dapat dilakukan penambahan deposit lebih dari satu kali selama belum ada tagihan atas *PO* tersebut. *Deposit* yang dimaksud adalah satu jenis transaksi tersendiri dan hanya dapat dibuat berdasarkan transaksi *PO*.

Pada Gambar 3.5 dan Gambar 3.6, dapat dilihat perbedaan sebelum dan sesudah menambahkan submodul *PO* pada traksaksi *invoice*. Perbedaan paling mendasar adalah dalam *PO* stok barang tidak bertambah hingga ada *invoice* yang dibuat berdasarkan nomor *PO* tersebut. Jika sudah ada *deposit* pada *PO*, maka pada saat membuat *invoice* berdasarkan *PO* tersebut hanya akan ditagih sesuai dengan jumlah yang belum terbayar saja. Selain itu, fitur *PO* memungkinkan pengguna untuk melakukan *invoicing* lebih dari satu kali dan tagihan akan disesuaikan dengan jumlah *deposit* yang sudah pernah dibuat sebelumnya.



Gambar 3.5 *Flowchart Invoice* tanpa *PO*

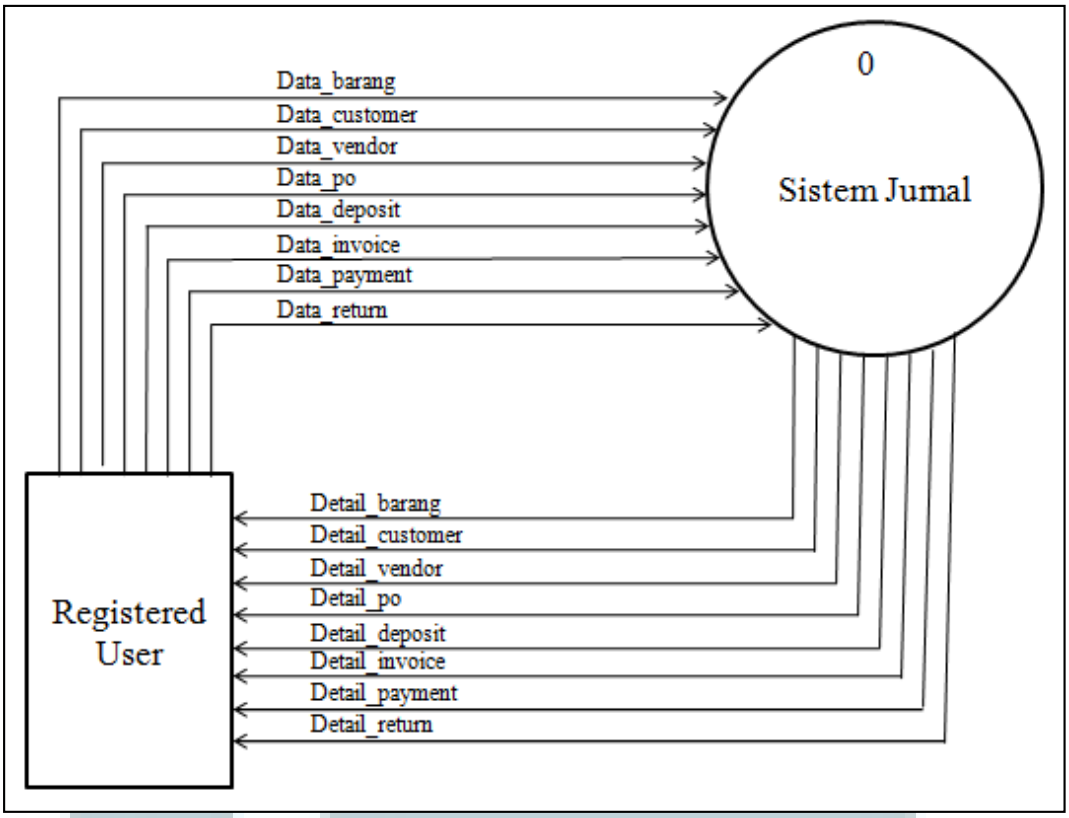
Pada Gambar 3.5 dijelaskan bahwa *invoice* yang dibuat merupakan data langsung dari pengguna dan bukan berdasarkan *PO* yang telah dibuat. Data yang dimasukkan tidak tergantung pada *PO*, berbeda dengan Gambar 3.6 yang hanya dapat memasukkan data berdasarkan *PO* yang telah dibuat sebelumnya.



Gambar 3.6 *Flowchart Invoice* Berdasarkan *PO*

2. *Context Diagram*

Context diagram menggambarkan secara umum sistem yang dikembangkan. Ilustrasi pada *context diagram* dapat dilihat pada Gambar 3.7.



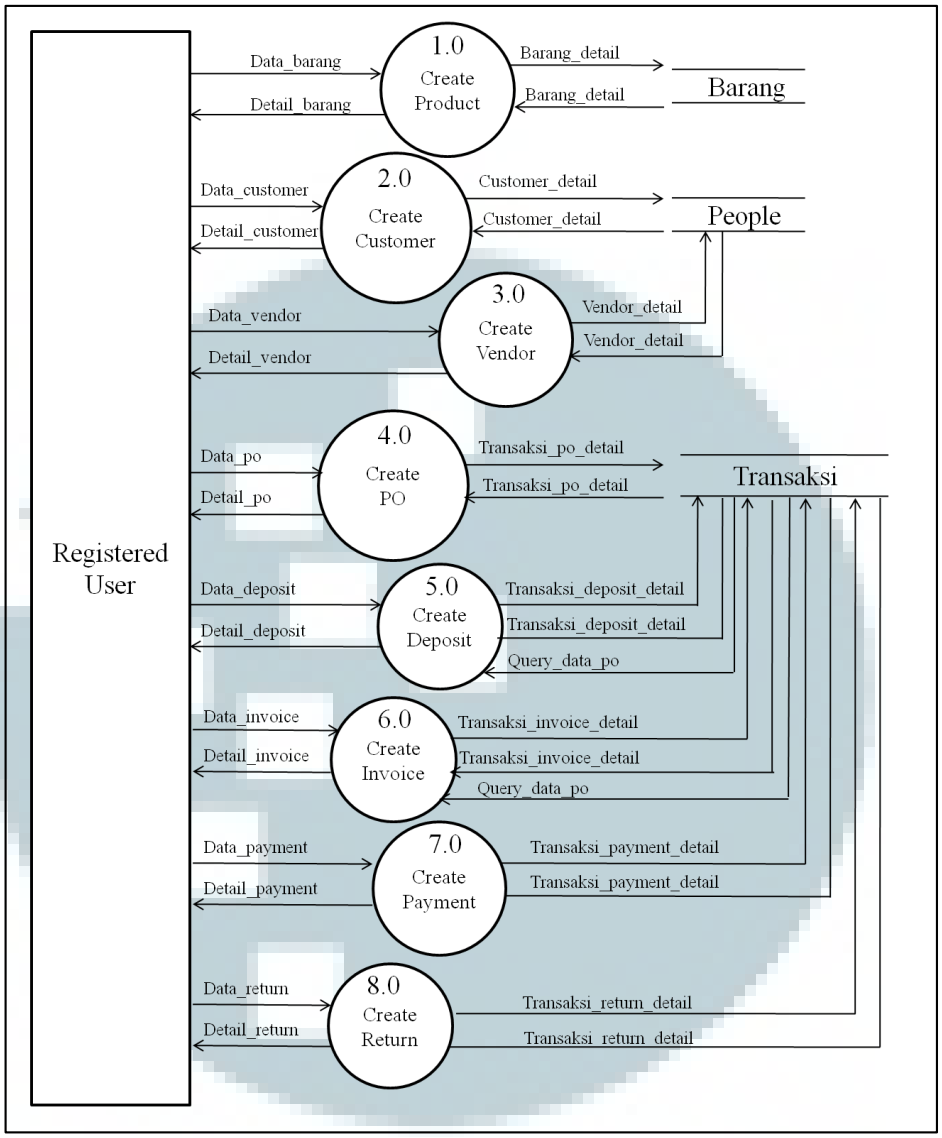
Gambar 3.7 *Context Diagram* Jurnal

Pada Gambar 3.7 dijelaskan bahwa entitas *registered user* dapat memasukkan data barang, *customer*, *vendor*, *PO*, *deposit*, *invoice*, *payment* dan *return*, serta dapat melihat detail barang, *customer*, *vendor*, *PO*, *deposit*, *invoice*, *payment* dan *return.*

3. DFD Level 1

Dalam DFD level 1 dapat dilihat proses yang ada dalam aplikasi Jurnal, seperti yang ditunjukkan pada Gambar 3.8. Data *customer* dan *vendor* disimpan pada tabel yang sama, yaitu tabel *people*. Data *PO, deposit, invoice, payment* dan *return* disimpan pada tabel transaksi dengan tipe transaksi yang berbeda.

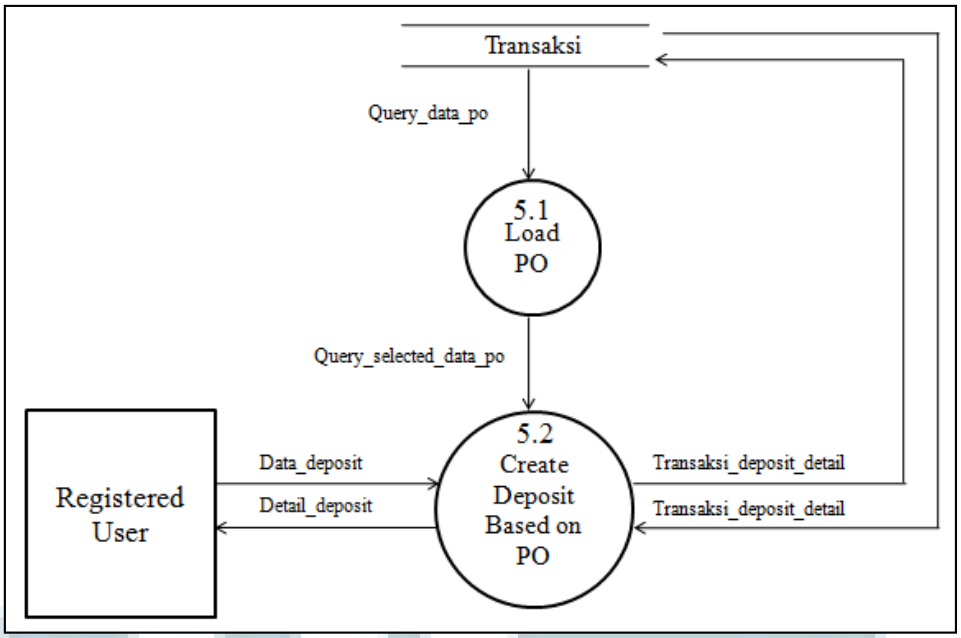
.



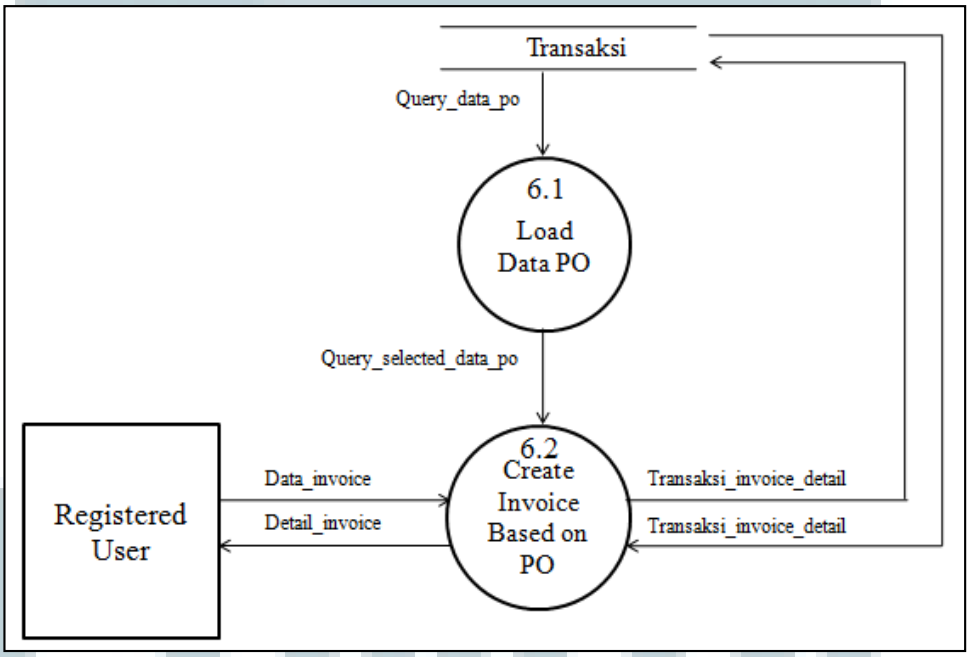
Gambar 3.8 DFD Level 1 Submodul *PO* dan *Deposit*

4. DFD Level 2

Dalam DFD level 2 dijelaskan lebih lengkap mengenai hubungan *PO*, *deposit* dan *invoice*. *Deposit* hanya dapat dibuat berdasarkan transaksi *PO*. *Invoice* dapat dibuat berdasarkan transaksi *PO* maupun tanpa transaksi *PO* seperti pada Gambar 3.9 dan Gambar 3.10.



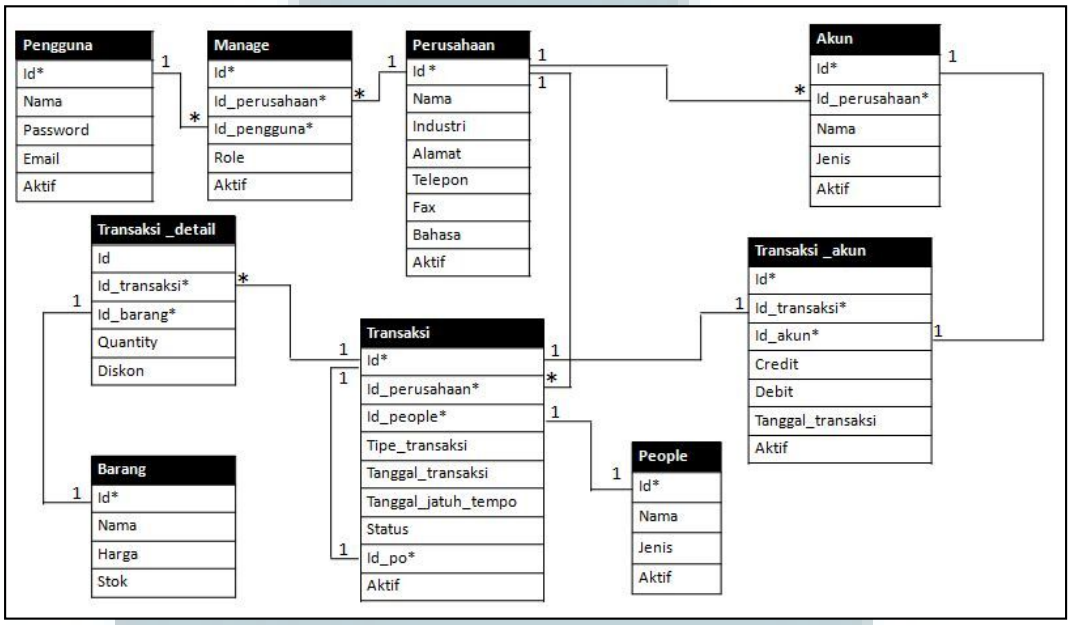
Gambar 3.9 DFD Level 2 *Create Deposit Based on PO*



Gambar 3.10 DFD Level 2 *Create Invoice Based on PO*

3.3.3 Entity Relationship Diagram

Entity Relatiobship Diagram (ERD) menjelaskan tabel-tabel dalam *database* aplikasi Jurnal Consulting Indonesia yang digunakan dalam pengembangan yang dilakukan beserta hubungannya.



Gambar 3.11 ERD Aplikasi Jurnal

3.3.4 Struktur Tabel

Tabel yang digunakan dalam pengembangan submodul *PO* dan *deposit* ini hanya beberapa tabel saja seperti yang digambarkan dalam ERD. Rincian tabel tersebut adalah sebagai berikut.

1. Nama tabel : Perusahaan
   Fungsi : Menyimpan informasi perusahaan
   Primary key : id_perusahaan
   Foreign key : -

Tabel 3.1 Tabel Perusahaan

Nama Kolom	Tipe data	Panjang	Keterangan
Id_perusahaan	Int	11	ID perusahaan, *auto increment*
Nama	Varchar	255	Nama perusahaan pengguna
Industri	Varchar	255	Jenis perusahaan pengguna
Alamat	Text		Alamat perusahaan pengguna
Telepon	Varchar	255	Telepon perusahaan pengguna
Fax	Varchar	255	Fax perusahaan pengguna
Bahasa	Int	11	Bahasa yang ingin digunakan dalam aplikasi
Aktif	Boolean	1	Status perusahaan pengguna dalam aplikasi (aktif/non aktif)

2. Nama tabel : Pengguna

   Fungsi : Menyimpan informasi pengguna

   Primary key : id_pengguna

   Foreign key : id_perusahaan

Tabel 3.2 Tabel Pengguna

Nama Kolom	Tipe data	Panjang	Keterangan
Id_pengguna	Int	11	ID pengguna, *auto increment*
Nama	Varchar	255	Nama pengguna
Password	Varchar	255	Password pengguna
Email	Varchar	255	Email pengguna
Aktif	Boolean	1	Status pengguna dalam aplikasi (aktif/non aktif)

3. Nama tabel : Manage

   Fungsi : Menyimpan hubungan antara pengguna dan perusahaan

   Primary key : id_manage

   Foreign key : id_perusahaan, id_pengguna

Tabel 3.3 Tabel Manage

Nama Kolom	Tipe data	Panjang	Keterangan
Id_manage	Int	11	ID manage, *auto increment*
Id_perusahaan	Int	11	Id perusahaan yang terkait
Id_pengguna	Int	11	Id pengguna yang terkait
Role	Int	11	Jenis aturan
Aktif	Boolean	1	Status pengguna dalam aplikasi (aktif/non aktif)

4. Nama tabel : People

   Fungsi : Menyimpan informasi *customer/vendor*

   Primary key : id_people

   Foreign key : -

Tabel 3.4 Tabel People

Nama Kolom	Tipe data	Panjang	Keterangan
Id_people	Int	11	ID *client*, *auto increment*
Nama	Varchar	255	Nama *client*
Jenis	Int	11	Jenis *client*
Aktif	Boolean	1	Status *client* dalam aplikasi (aktif/non aktif)

5. Nama tabel : Akun

   Fungsi : Menyimpan daftar akun dari perusahaan

   Primary key : id_akun

   Foreign key : -

Tabel 3.5 Tabel Akun

Nama Kolom	Tipe data	Panjang	Keterangan
Id_pengguna	Int	11	ID *account*, *auto increment*
Id_perusahaan	Int	11	Id perusahaan yang terkait
Nama	Varchar	255	Nama *account*
Jenis	Int	11	Jenis *account*
Aktif	Boolean	1	Status *account* dalam aplikasi (aktif/non aktif)

6. Nama tabel : Barang

   Fungsi : Menyimpan informasi *product*

   Primary key : id_barang

   Foreign key : -

Tabel 3.6 Tabel Barang

Nama Kolom	Tipe data	Panjang	Keterangan
Id_barang	Int	11	ID *product*, *auto increment*
Nama	Varcghar	255	Nama *product*
Harga	Float		Harga *product*
Stok	Int	11	Stok per *product*
Aktif	Boolean	1	Status barang dalam aplikasi (aktif/non aktif)

7. Nama tabel : Transaksi

   Fungsi : Menyimpan informasi transaksi yang terjadi

   Primary key : id_transaksi

   Foreign key : id_perusahaan, id_people, id_po

Tabel 3.7 Tabel Transaksi

Nama Kolom	Tipe data	Panjang	Keterangan
Id_transaksi	Int	11	ID transaksi, *auto increment*
Id_perusahaan	Int	11	ID perusahaan yang terkait
Id_people	Int	11	ID *product* yang terkait
Tipe_transaksi	Int	11	Jenis transaksi
Tanggal_transaksi	Date		Tanggal transaksi dilakukan
Tanggal_jatuh_tempo	Date		Tanggal jatuh tempo transaksi tersebut
Status	Int	11	Status transaksi *(open, partial, paid, overdue)*
Id_po	Int	11	ID *purchase order* yang terkait
Aktif	Boolean	1	Status transaksi dalam aplikasi (aktif/non aktif)

8. Nama tabel : Transaksi_detail

   Fungsi : Menyimpan informasi detail dari transaksi yang berhubungan dengan *product*

   Primary key : id_transaksi_detail

   Foreign key : id_transaksi, id_barang

Tabel 3.8 Tabel Transaksi Detail

Nama Kolom	Tipe data	Panjang	Keterangan
Id_transaksi_detail	Int	11	ID transaksi_detail, *auto increment*
Id_transaksi	Int	11	ID transaksi yang terkait
Id_barang	Int	11	ID *product* yang terkait
Quantity	Int	11	Jumlah *product* per unit
Diskon	Float		Diskon harga

9. Nama tabel : Transaksi_akun

   Fungsi : Menyimpan informasi transaksi yang terjadi dari tiap akun berhubungan dengan tranaksi dan akun

   Primary key : id_transaksi_akun

   Foreign key : id_transaksi, id_akun

Tabel 3.9 Tabel Transaksi Akun

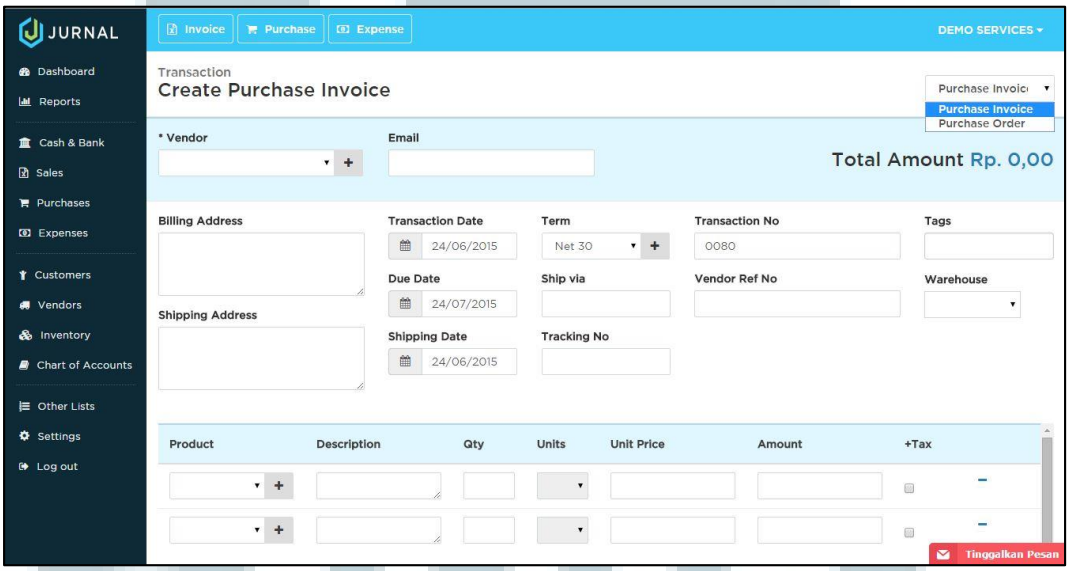
Nama Kolom	Tipe data	Panjang	Keterangan
Id_transaksi_akun	Int	11	ID transaksi_akun, *auto increment*
Id_transaksi	Int	11	ID transaksi yang terkait
Id_akun	Int	11	ID *account* yang terkait
Credit	Float		*Credit* dalam transaksi
Debit	Float		*Debit* dalam transaksi
Tanggal_transaksi	Date		Tanggal transaksi
Aktif	Boolean	1	Status transaksi_akun dalam aplikasi (aktif/non aktif)

3.3.5 Hasil Implementasi

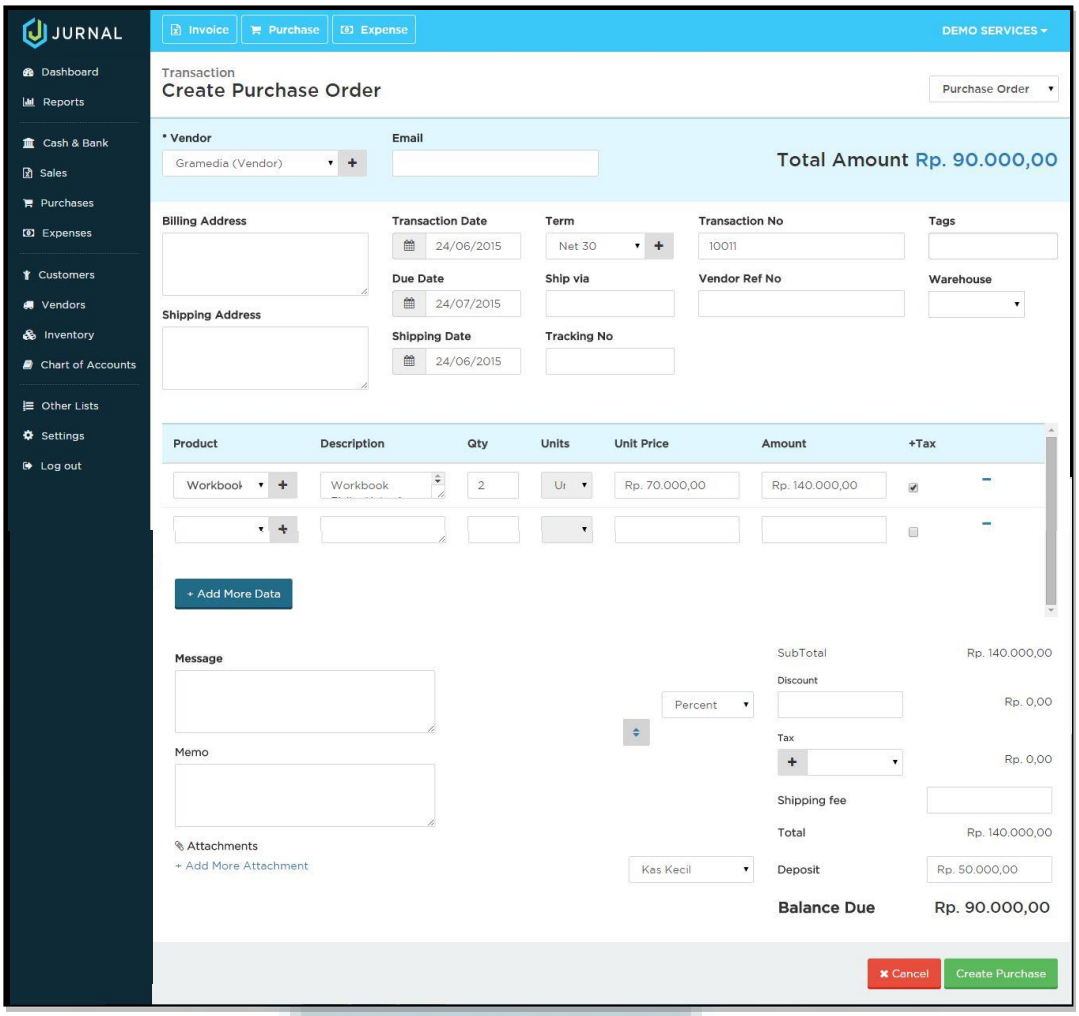
Berdasarkan desain-desain antarmuka yang telah ada sebelumnya, maka dilakukan implementasi berdasarkan rancangan awal yang sudah dijelaskan. Hasil implementasi berdasarkan rancangan yang sudah dibuat akan dijelaskan dengan tampilan antarmuka, berikut adalah rinciannya.

1. Transaksi *Purchase Order*

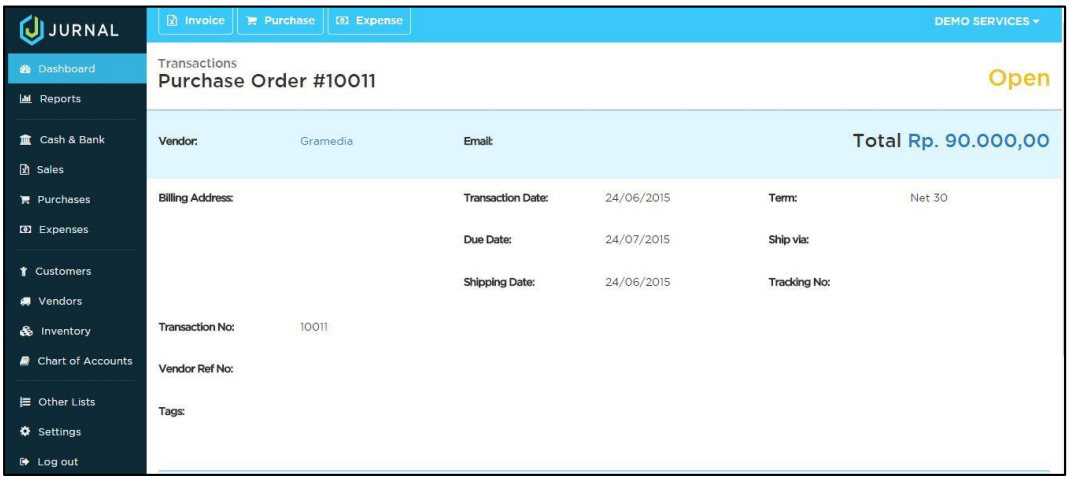
Untuk membuat transaksi *PO*, klik *purchase* pada bagian menu kiri dan pilih *create new purchase* atau klik *shortcut purchase* pada bagian atas *page*. Setelah itu pilih *purchase order* pada pilihan tipe transaksi di kanan atas seperti pada Gambar 3.12. Masukkan data yang diperlukan seperti yang terlihat pada Gambar 3.13 dan tekan tombol *create*. Jika semua data yang dimasukkan *valid*, maka transaksi akan dibuat dan akan dialihkan ke halaman detail transaksi seperti pada Gambar 3.14.



Gambar 3.12 Halaman *Create Purchases*



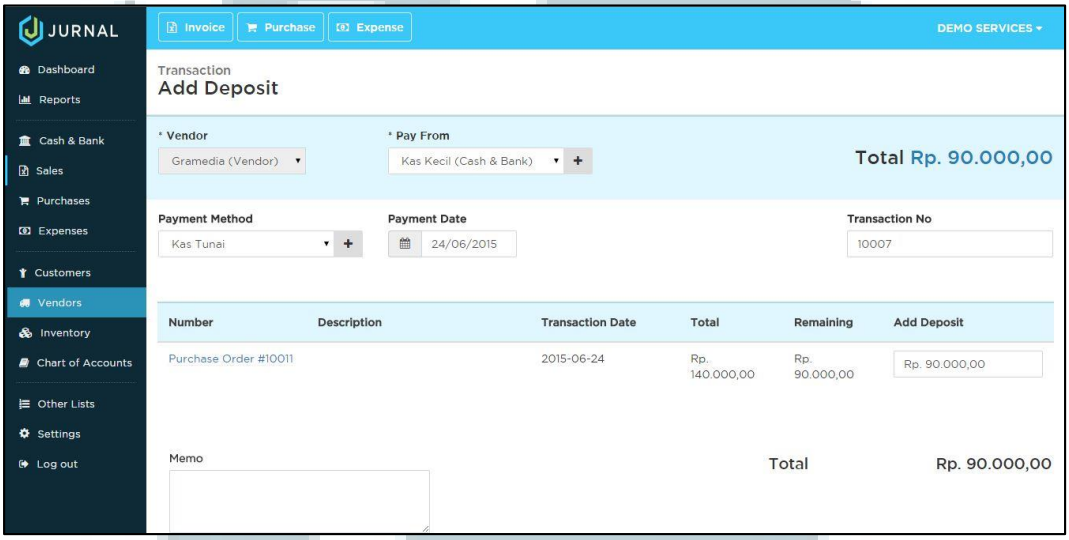
Gambar 3.13 Halaman *Create Purchases Order*



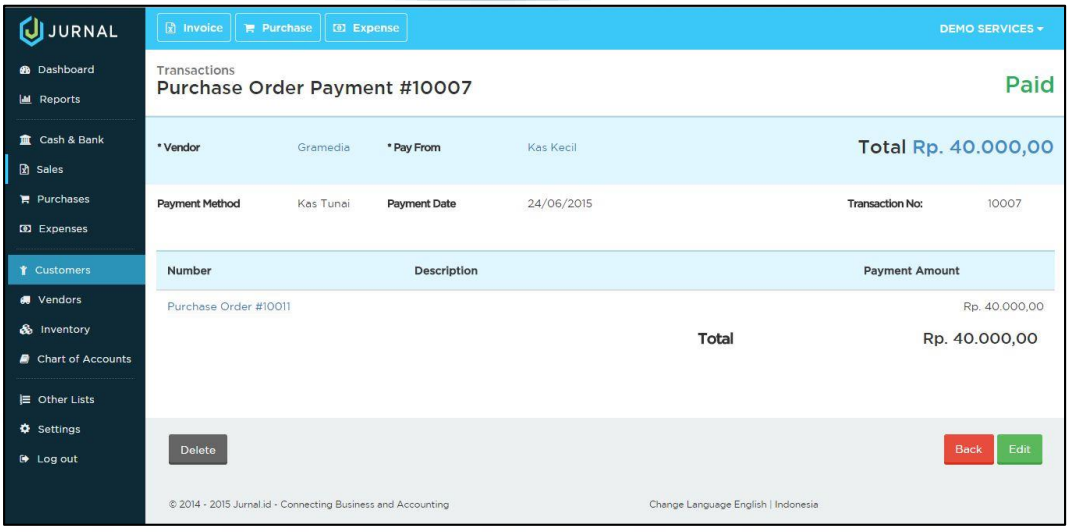
Gambar 3.14 Halaman *Feedback Create PO*

2. Transaksi *Add Deposit*

Penambahan *deposit* hanya dapat dilakukan pada transaksi *PO* saja, seperti pada Gambar 3.15. Dalam satu penambahan *deposit*, bisa terdiri dari satu atau beberapa *PO*, dan *deposit* tidak dapat melebihi total yang harus dibayar dalam *PO*. Pada saat membuat *invoice* berdasarkan *PO*, maka jumlah tagihan akan dikurangi *deposit* yang ada pada *PO* tersebut.



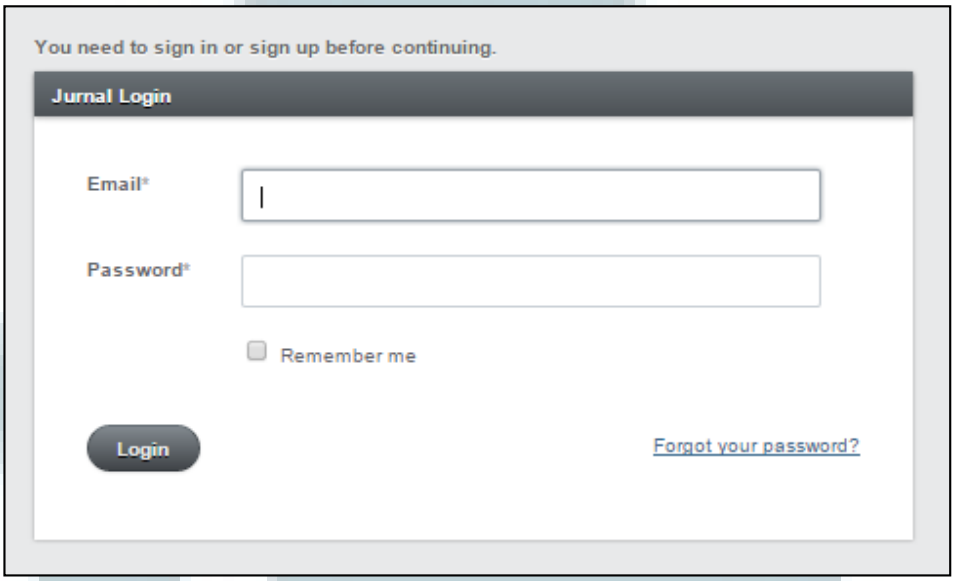
Gambar 3.15 Halaman *Add Deposit* pada *PO*



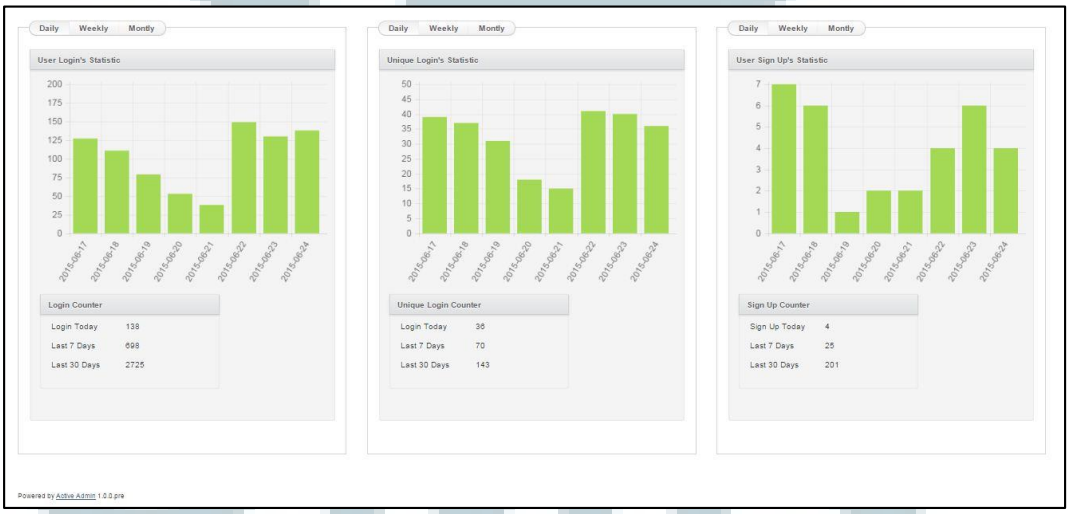
Gambar 3.16 Halaman *Feedback Add Deposit* pada *PO*

3. *Admin Pages* dan *Chart*

Login admin hanya diberikan kepada tim Jurnal dan dibatasi hak aksesnya sesuai dengan jabatan. *Sales* hanya dapat melihat data perusahaan dan data *user* sedangkan CEO dan direktur dapat melihat semua informasi pada *admin page.*



Gambar 3.17 Halaman *Login Admin Page*



Gambar 3.18 *Chart* pada *Admin Page*

Chart pada *admin page* berguna untuk memperlihatkan perkembangan Jurnal dalam periode harian, mingguan dan bulanan. Tampilan dalam bentuk

c*hart* mempermudah membaca data statistik dari *user login*, *unique user login* dan *register* seperti pada Gambar 3.18.

3.3.6 Kendala yang Ditemukan

Selama berlangsungnya proses kerja magang, terdapat beberapa kendala yang berhubungan dengan pengembangan aplikasi Jurnal. Hal-hal tersebut adalah sebagai berikut.

1. Tidak mengerti proses akuntasi dan harus mempelajari bisnis proses dari sisi akuntasi, seperti *chart of account, credit debit account,* kategori *account* dan lain sebagainya.
2. Belum pernah mempelajari dan menggunakan bahasa pemrograman *ruby on rails* sehingga membutuhkan waktu untuk mempelajari dan menyesuaikan diri dengan *environment rails.*
3. Belum pernah menggunakan *git repository* saat bekerja dalam tim, sehingga harus banyak membaca dan mempelajari tentang *git repository*.

3.3.7 Solusi atas Kendala yang Ditemukan

Berikut merupakan solusi yang ditempuh untuk mengatasi kendala yang ditemukan tersebut.

1. Mempelajari proses bisnis dan akuntansi dari sesi pengenalan yang dibimbing oleh Bapak Anthony Kosasih.
2. Mendapatkan sesi pengenalan terhadap *ruby on rails* secara singkat pada saat pertama kali bergabung dengan tim Jurnal yang dibimbing oleh Bapak Daniel

Witono dan pengetahuan dari UMN mengenai pemrograman *object oriented* sangat membantu.

3. Tidak ragu untuk bertanya kepada karyawan senior serta mencari informasi yang dibutuhkan di internet.