BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Dalam perusahaan Kalbe Consumer Health, terdapat beberapa tingkatan dalam divisi IT dan *Digital Transformation*. Divisi IT dan *Digital Transformation* terdiri dari *Head of Digital Transformation*, *System Architect*, *Business Analyst*, *Application Developer*, *Oracle Functional*, *Oracle Technical*, *Network and Infra*, *IT Support Staff*, *Application Developer Intern*, *and Technical Oracle Intern.*

Kedudukan penulis selama masa magang adalah sebagai *Technical Oracle Support Intern* dengan mentor dari *Oracle Technical* bernama T. Ryan Adi Pangestu. Dalam masa program magang, *mentor* akan memberikan tugas baru ketika terdapat masalah dalam report yang user bicarakan kepada *mentor*.

3.2 Tugas yang Dilakukan

Tugas yang kerjakan selama melakukan program magang adalah sebagai berikut:

1. Membuat *procedure XSFL BATCH NUM MT F* dalam *oracle*
2. Mengupdate *report XSFL MT RFP F* dalam *oracle*
3. Membuat *procedure XSFL MT AUTO RFP* dalam *oracle*
4. Membuat *template API CREATE RFP* dalam *oracle*
5. Menganalisa *report XSFL CM List of Receipt and Payment Cash or Bank* dalam *oracle*
6. Menganalisa *report XSFL AP Nota Retur Faktur Pajak Report PDF* dalam *oracle*
7. Mencari cara membaca *JSON ARRAY* tanpa *JSON ARRAY T*
8. Mencari cara mengubah *expires time token API*
9. Mengupdate *report PO Budget Capex* dalam *Oracle*
10. Menganalisis *report XSFL AP Request For Payment* dalam *oracle apps*

11. Menganalisis *report XSFL AP Tax Bupot* dalam *oracle apps*
12. Mengupdate *view XXAR EINVOICE TRX ASSYS* dalam *Oracle*
13. Membuat *procedure INSERT JOURNAL* dalam *Oracle*
14. Mencari cara melakukan *retirement asset* dalam *Oracle*
15. Membuat *procedure XSFL VALIDATE CA NUMBER* dengan *Oracle*
16. Membuat *view XSFL CA SETTLEMENT* dengan *Oracle*
17. Menganalisis *report XSFL AP Request For Payment* dalam *oracle apps*
18. Membuat *program XSFL SYS - RECOMPILE OBJECT* dalam *Oracle*
19. Membuat *program API POST ASSYS JOURNAL* dalam *Oracle*
20. Membuat *program API POST VALIDATE RFP* dalam *Oracle*
21. Membuat *program API XSFL AR AUTO PVCUBE* dalam *Oracle*
22. Membuat *program XSFL PV RV AUTO* baru dalam *oracle apps*
23. Membuat *program XSFL AP SETTLEMENT PAY STG* dalam *Oracle*
24. Membuat *program API PAYMENT SETTLEMENT* dalam *Oracle*
25. Membuat *report XKLB GL GL ACTIVITY CSV* dalam *Oracle*

3.3 Uraian Pelaksanaan Magang

Berikut adalah *detail* terkait kegiatan-kegiatan yang dilakukan selama praktik kerja magang.

Tabel 3.1. Uraian pelaksanaan praktik kerja magang

Minggu ke	Pekerjaan yang dilakukan
1	• Membuat *procedure XSFL BATCH NUM MT F* dalam *oracle* • Mengupdate *report XSFL MT RFP F* dalam *oracle* • Membuat *procedure XSFL MT AUTO RFP* dalam *oracle*
2	• Membuat *procedure XSFL MT AUTO RFP* dalam *oracle* • Membuat *template API CREATE RFP* dalam *oracle* • Menganalisa *report XSFL CM List of Receipt and Payment Cash or Bank* dalam *oracle*
3	• Mengupdate *procedure XSFL MT AUTO RFP* dalam *oracle* • Melakukan tes *API CREATE RFP* dengan *postman* • Menganalisa *report XSFL AP Nota Retur Faktur Pajak Report PDF* dalam *oracle*
4	• Mengupdate *database XSFL AP MASTER TAX* dalam *Oracle* • Mencari cara membaca *JSON ARRAY* tanpa *JSON ARRAY T* • Mencari cara mengubah *expires time token API* • Membuat *client* baru untuk *postman* dan dokumentasi cara pembuatan *client* baru • Mengupdate *report PO Budget Capex* dalam *Oracle*
5	• Menganalisis *report XSFL AP Request For Payment* dalam *oracle apps* • Menganalisis *report XSFL AP Tax Bupot* dalam *oracle apps* • Mengupdate *view XSFL ORDS RFP AP V* dalam *Oracle* • Melakukan tes *API CREATE RFP* dengan *postman* • Mengupdate *procedure XSFL MT AUTO RFP* dalam *oracle*
	Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
6	• Mengupdate *view XXAR EINVOICE TRX ASSYS* dalam *Oracle* • Membuat *procedure INSERT JOURNAL* dalam *Oracle* • Mencari cara melakukan *retirement asset* dalam *Oracle* • Membuat *procedure XSFL VALIDATE CA NUMBER* dengan *Oracle* • Membuat *view XSFL CA SETTLEMENT* dengan *Oracle*
7	• Mengupdate *procedure budget capex* dalam *Oracle* • Melakukan *testing program* dengan basis *website* • Menganalisis *report XSFL AP Request For Payment* dalam *oracle apps* • Membuat *program XSFL SYS RECOMPILE OBJECT* dalam *Oracle* • Membuat *program API POST ASSYS JOURNAL* dalam *Oracle*
8	• Melakukan tes *program API POST ASSYS JOURNAL* dalam *Postman* • Membuat *program API POST VALIDATE RFP* dalam *Oracle* • Membuat *program API XSFL AR AUTO PVCUBE* dalam *Oracle* • Membuat *program XSFL PV RV AUTO* baru dalam *oracle apps* • Melakukan tes *API XSFL AR AUTO PVCUBE* dengan *postman*

Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
9	• Membuat *program XSFL AP SETTLEMENT PAY STG* dalam *Oracle* • Membuat *program API PAYMENT SETTLEMENT* dalam *Oracle* • Mengupdate *report XKLB GL GL ACTIVITY 2*
10	• Mengupdate *report XKLB GL GL ACTIVITY 2* • Membuat *report XKLB GL GL ACTIVITY CSV* dalam *Oracle*
11	• Mengupdate *program API XSFL MT AUTO RFP* dalam *Oracle* • Membuat *report XKLB GL GL ACTIVITY CSV* dalam *Oracle* • Menganalisis *report XXAP INVLIST* dalam *Oracle* • Menganalisis *report XSFL CM List of Receipt and Payment Cash or Bank*
12	• Membuat *report XXAP INVLSTS CSV* dalam *Oracle* • Menganalisis *program API XSFL MT AUTO RFP* dalam *Oracle* • Mengupdate *program API XSFL MT AUTO RFP* dalam *Oracle* • Menganalisis *report XSFL CM List of Receipt and Payment Cash or Bank*
13	• Menganalisa *report XSFL OPM RMPM VARIANCE* dalam *Oracle* • Menganalisis *program API XSFL MT AUTO RFP* dalam *Oracle* • Mengupdate *report XXAP INVLSTS CSV* dalam *Oracle* • Mengupdate *report XKLB GL GL ACTIVITY CSV* dalam *Oracle*

Lanjut pada halaman berikutnya

Tabel 3.1 Uraian pelaksanaan praktik kerja magang (lanjutan)

Minggu ke	Pekerjaan yang dilakukan
14	• Mengupdate *report XKLB GL GL ACTIVITY CSV* dalam *Oracle* • Menganalisis *report auto PV cube* dalam *Oracle* • Melakukan tes dalam *database* 04
15	• Melakukan tes dalam *database* 04 • Membuat *database link* dari *database* 04 ke *finmidd* dan sebaliknya • Menganalisa *report XSFL ASK MASTER VENDOR V* dalam *Oracle* • Membuat *report XSFL AP VENDOR LISTING* dalam *Oracle*

3.3.1 Mempelajari Mengenai Program

Ketika membuat *program ORBIT*, perlu dilakukan riset mengenai alat serta metode untuk membuat *program*. Proses pembelajaran mengenai alat serta metode dalam pembuatan *ORBIT* sendiri dibantu oleh *mentor*. Serta pembelajaran mengenai alur pembuatan *ORBIT* menggunakan *flow* yang telah diberikan oleh *mentor*.

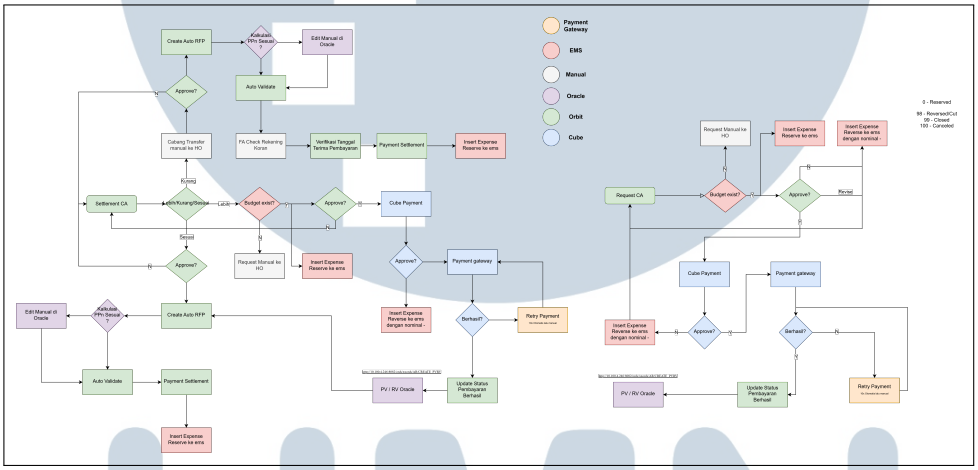
Database yang digunakan *Kalbe Consumer Helath* dalam menghubungkan data dengan *ORBIT* adalah *Oracle EBS R12. Oracle EBS R12* merupakan kumpulan aplikasi bisnis yang terintegrasi untuk mengotomatisasi proses bisnis utama di berbagai bidang seperti keuangan, sumber daya manusia, manufaktur, manajemen rantai pasokan, dan manajemen hubungan pelanggan. EBS dirancang untuk membantu perusahaan mengelola operasi bisnis mereka secara lebih efisien dan terintegrasi dalam satu platform.

Salah satu alat yang digunakan dalam membuat *API* dalam *ORBIT* adalah *Postman*. *Postman* adalah *platform API* untuk membangun dan menggunakan *API*. *Postman* menyederhanakan setiap langkah siklus hidup *API* dan menyederhanakan kolaborasi sehingga dapat membuat *API* yang lebih baik dan lebih cepat[4].

ORBIT dirancang untuk memperlancar operasi bisnis, meningkatkan produktivitas karyawan,dan memperbaiki pengalaman pengguna secara

keseluruhan. Aplikasi ini akan berfungsi sebagai platform terpusat untuk mengelola Aktivitas Kunjungan BR(*Bussines Representative*) ,Pemantauan *Event*, Proses Persetujuan, Sistem Manajemen Pembelajaran(LMS), Pelacakan Geolokasi, Gamifikasi (poin), *PO Transaction*(PO), dan manajemen *Operational* (CA)serta *Compensation*(CA).

CA merupakan *Cash Advance* atau proses dimana user dapat melakukan permintaan dana kepada perusahaan untuk keperluan perusahaan seperti *event* atau pembelian barang. Jika budget tidak sesuai dengan CA maka *ORBIT* akan melakukan *Auto RFP* yang merupakan Request For Payment yang dimana merupakan permintaan dana tambahan kepada perusahaan untuk menyelesaikan kebutuhan dalam perusahaan.



Gambar 3.1. Flowchart General ORBIT

Alur cara kerja dari ORBIT adalah

- Memilih proses pengeluaran dengan CA(*Cash Advance*) atau Non CA
- Jika CA memiliki budget kurang atau sesuai maka akan melakukan *Auto RFP* dan mengupdate data secara manual ke dalam *Oracle*, jika CA memiliki *budget* lebih maka akan melakkukan pembayaran dalam *Cube* lalu melakukan proses PV/RV dan dilanjutkan ke dalam *Auto RFP* dan mengupdate *data* secara *manual* ke dalam *Oracle*.
- RFP yang telah dibuat akan di validasi dengan *Auto Validate*.
- RFP yang telah di validasi akan diproses dengan *Payment Settlement* untuk menunjukan bahwa pengeluaran telah diproses.

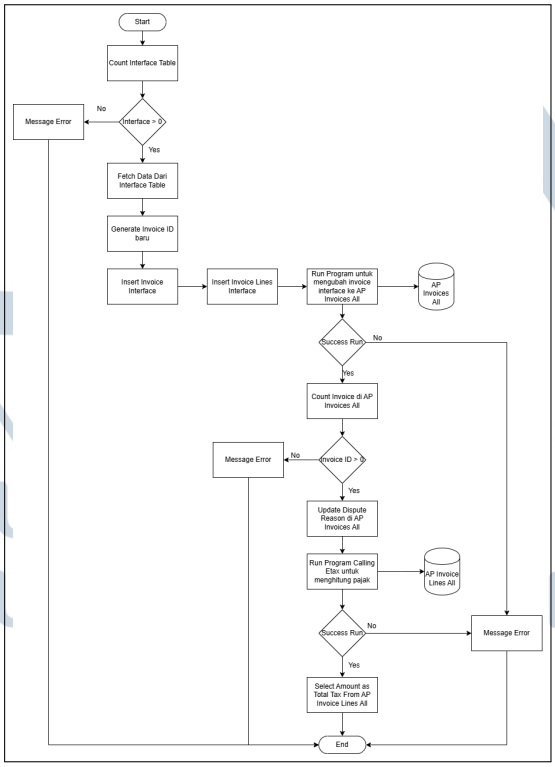
3.3.2 Proses Pembuatan Program

Proses dimulai dengan membuat *procedure program* yang dibutuhkan. Lalu dilanjutkan dengan proses membuat *API* dengan *ORDS oracle* dengan membentuk *template* dan *handle*, dan akhirnya melakukan tes *API* dalam *Postman* untuk melihat hasil *API* di dalam *Postman* dan *data* dalam *Oracle*. *Program* yang akan ditunjukan berupa:

- CREATE RFP
- GET AP RFP
- POST VALIDATE RFP
- CREATE PVRV
- POST ASSYS JOURNAL
- PAYMENT SETTLEMENT

A. Procedure Program

Berikut *procedure* dari *program CREATE RFP*:

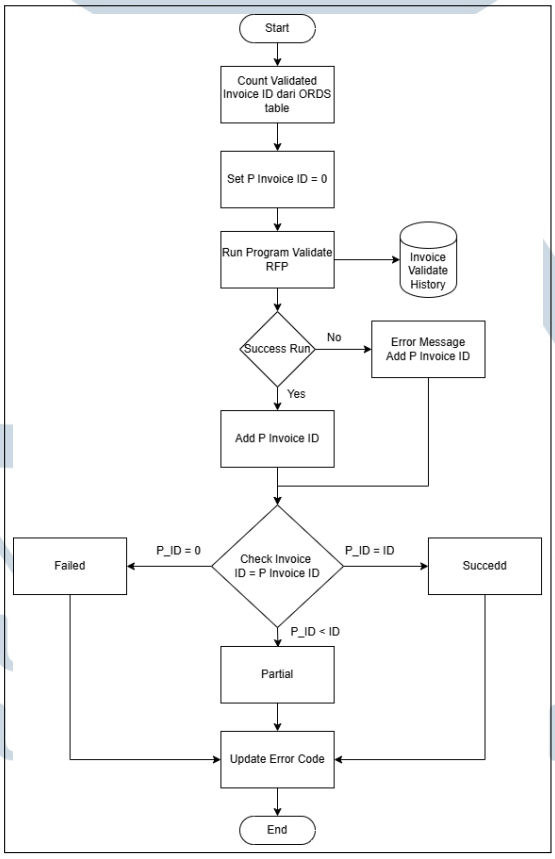


Gambar 3.2. Flowchart CREATE RFP

Program dimulai dengan menghitung jumlah *interface* dalam *table*. Jika nilai *interface* lebih dari 0, maka *program* akan mengambil *data* dari *table interface* dan jika tidak maka akan menghasilkan *message error*. Setelah *data* berhasil diambil, maka *program* akan membuat *invoice id* baru untuk dalam *table invoice interface*. *Data invoice id* baru dan *data* dari *table interface* akan dimasukan kedalam *table invoice interface* dan *invoice lines interface*.

Program akan memanggil *program* untuk mengubah *invoice interface* menjadi *data* di dalam *table AP Invoices All*. Jika berhasil maka *data* akan masuk ke dalam *AP Invoices All* dan jika tidak maka *program* akan mengeluarkan *message error*. Jika *data* berhasil masuk ke dalam *AP Invoices All* maka *program* akan mengupdate *Dispute Reason*. Setelah itu *program* akan memanggil *program* untuk menghitung pajak dari *AP Invoices All*. Jika berhasil maka *data* akan masuk ke dalam *AP Invoice Lines All* dan jika tidak maka *program* akan menghasilkan *message error*. Dan pada akhirnya *program* akan mencari jumlah nilai pajak sebagai *output* dari *program*.

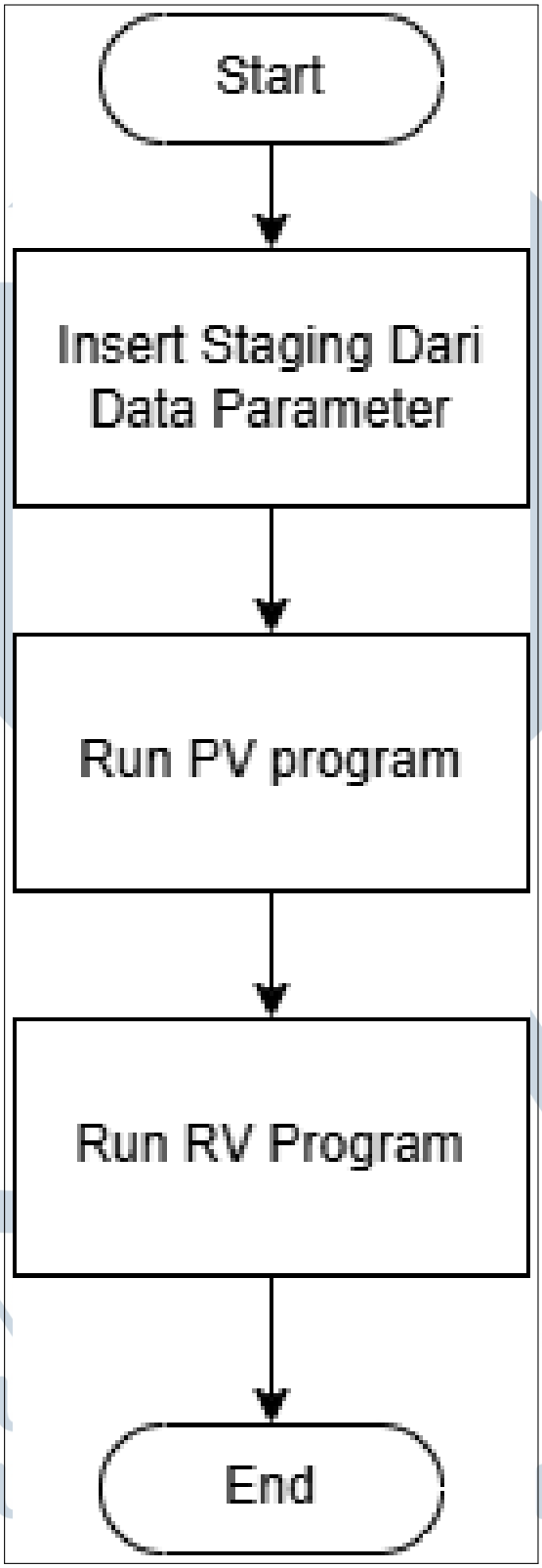
Berikut *procedure* dari *program POST VALIDATE RFP*:



Gambar 3.3. Flowchart POST VALIDATE RFP

Program dimulai dengan menghitung *validated invoice id* dari *table ORDS*. Setelah itu menentukan nilai *P Invoice ID* sebagai 0 untuk *looping*. Setelah itu melakukan *looping* memanggil *program Validate RFP* sesuai dengan *Invoice ID* dari *parameter program*. Jika *program* berhasil, maka *data* akan masuk ke dalam *table Invoice Validate History* dan jika gagal maka akan menghasilkan *message error* serta nilai *P Invoice ID* akan bertambah sebanyak 1 untuk setiap *Invoice ID* yang diproses baik jika *program* berhasil maupun gagal. Jika nilai *invoice ID* sama dengan *P Invoice ID* maka *program* akan menghasilkan *output succedd*, jika nilai tidak sama dan bukan 0 maka *program* akan menghasilkan *output partial*, dan jika nilai 0 maka *program* akan menghasilkan *output failed*. Nilai *output* akan dimasukan sebagai *error code output program*.

Berikut *procedure* dari *program CREATE PVRV*:



Gambar 3.4. Flowchart CREATE PVRV

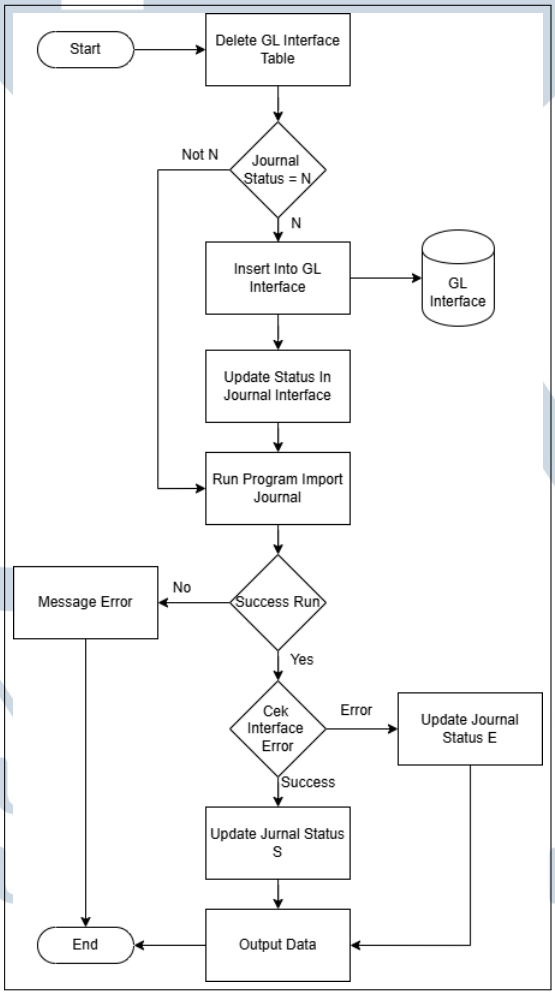
Program menjalankan staging dimana akan melakukan insert ke table

staging. Setelah menginput parameter ke table staging, maka program akan memulai untuk membuat PV dan RV. Program dijalankan 2 kali untuk membuat PV dan RV.

Dalam menjalankan program PVRV, pertama adalah menentukan akan membuat PV atau RV. Program akan menentukan parameter bedasarkan PV atau RV. Lalu program akan melihat apakh bank merupakan bagian dari bank cabang atau bukan.

Setelah itu program akan membuat nomor PVRV sesuai dengan parameter. Lalu menjalankan program utama untuk membuat nomor PV atau nomor RV kedalam database oracle. Jika berhasil maka program akan mengupdate table staging untuk menuliskan nomor PV atau RV dan jika gagal maka akan mengirimkan output 400.

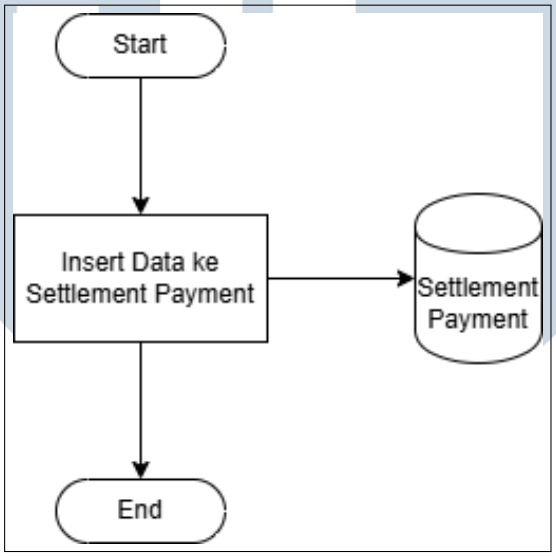
Berikut *procedure* dari *program POST ASSYS JOURNAL*:



Gambar 3.5. Flowchart POST ASSYS JOURNAL

Program dimulai dengan melakukan *loop* dengan *cursor*. Lalu menghapus *journal* dari *source* dan *group ID* yang sama shingga dapat diproses kembali. Setelah itu akan menjalankan *looping insert* ke dalam *table GL Interface. Data status staging* akan di *update* untuk diproses oleh *program*. Lalu *program* akan memanggil *program import interface* untuk mengirim *data* ke dalam *database*. Dan jika berhasil maka akan melakukan cek terhadap *data* yang tertinggal dan mengupdate *output database interface* sesuai dengan hasil menjalankan *program*.

Berikut *procedure* dari *program PAYMENT SETTLEMENT*:

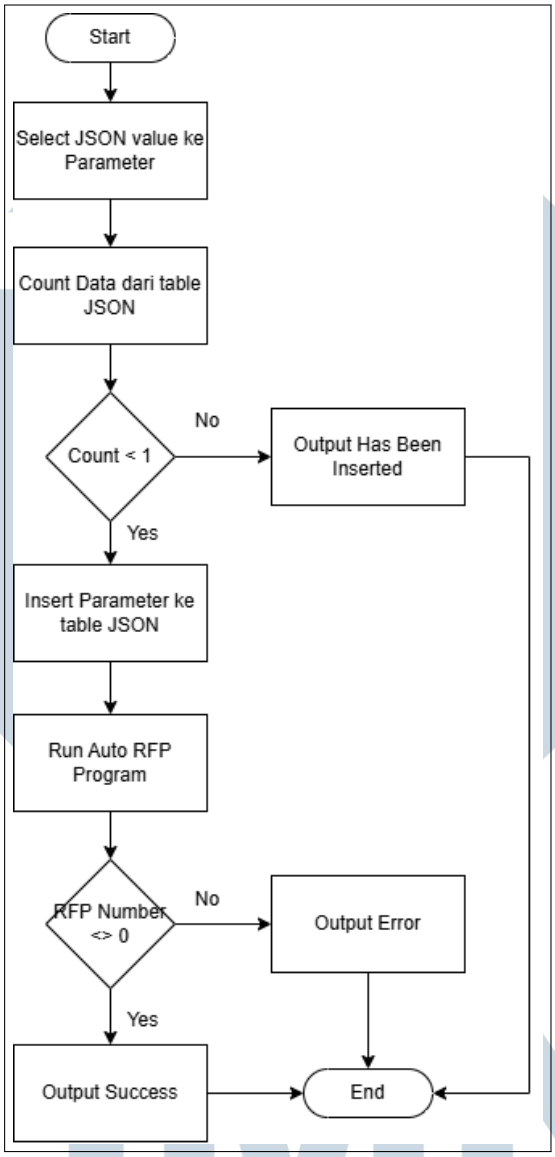


Gambar 3.6. Flowchart PAYMENT SETTLEMENT

Program akan melakukan *insert* kedalam *table settlement payment staging*.

B. Program API dalam Oracle

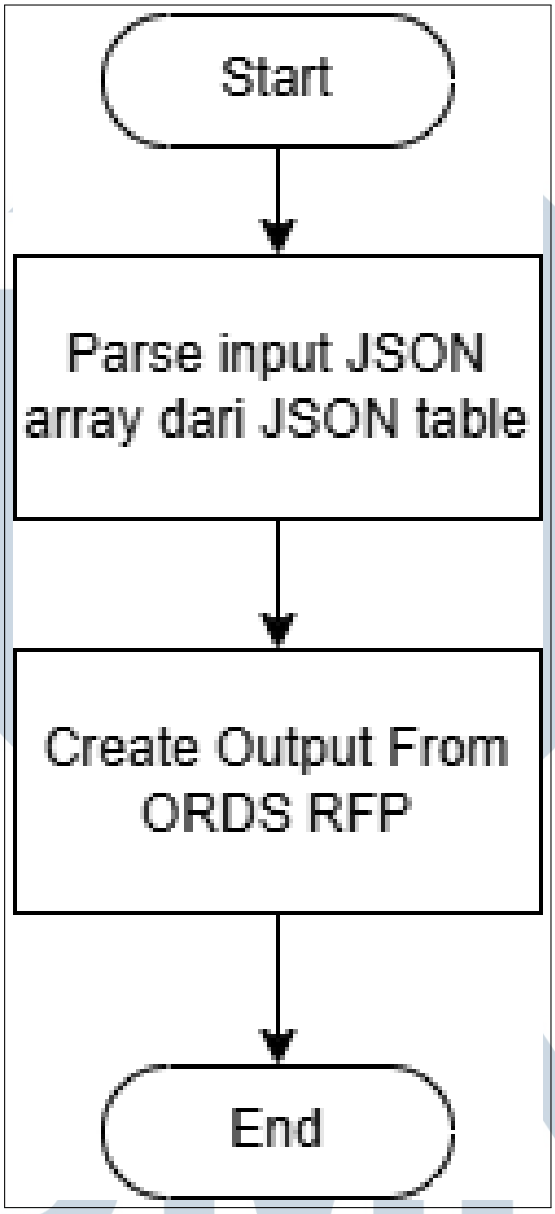
Berikut *program CREATE RFP*:



Gambar 3.7. Flowchart CREATE RFP API

Program dimulai dengan mengambil *data* dari *REST API* untuk diolah. *Data* yang telah diambil akan dicari dalam *table JSON*, jika ada maka akan menghasilkan *output* telah dibuat dan jika tidak ada dalam *table* maka *data* dari *REST API* akan dimasukan kedalam *table JSON*. Setelah itu menjalankan *program Auto RFP* dan jika hasil *RFP Number* bukan 0 maka *program* akan mengirimkan *output success* pada *Postman* dan jika nilai *RFP Number* adalah 0 maka akan menghasilkan *output failed* pada *Postman*.

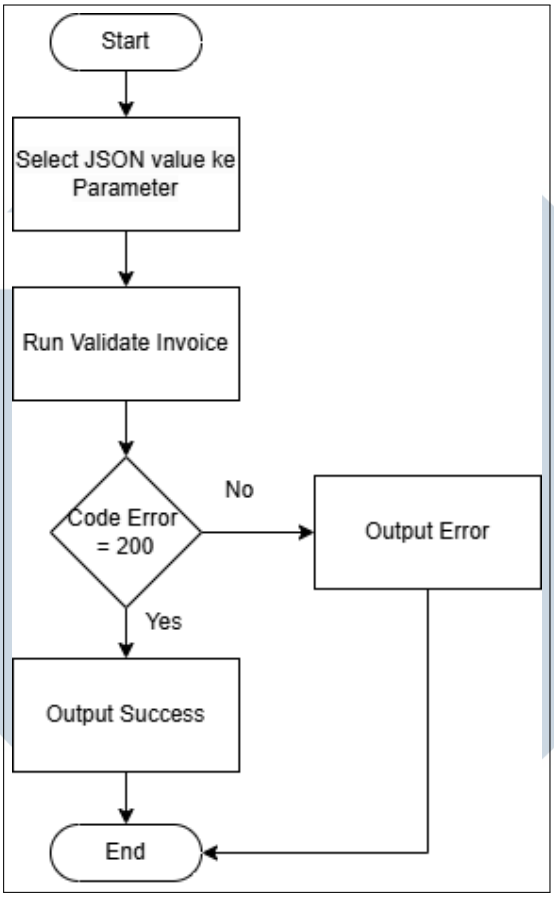
Berikut *program GET AP RFP*:



Gambar 3.8. Flowchart GET AP RFP API

Program dimulai dengan mengambil *data* dari *REST API* untuk diolah. *Data* yang diambil akan diolah dengan menggunakan *invoice number*. Setiap *invoice number* yang ada di dalam *table ords rfp* akan dikeluarkan menjadi *output*. Jika *program* berjalan dengan baik, maka *program* akan menghasilkan *data RFP* pada *Postman* dan jika dalam proses *loop* terjadi *error*, maka *program* akan menghasilkan *message error* pada *Postman*.

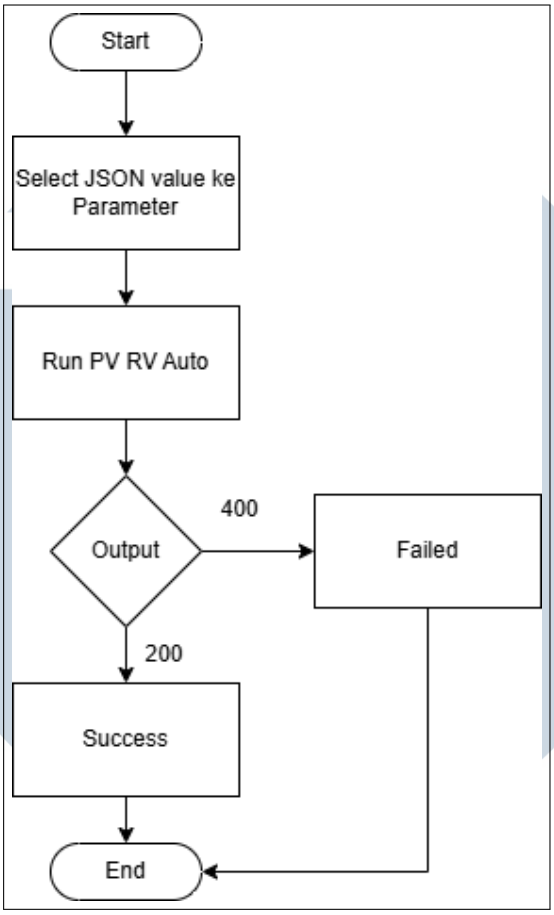
Berikut *program POST VALIDATE RFP*:



Gambar 3.9. Flowchart POST VALIDATE RFP API

Program dimulai dengan mengambil *data* dari *REST API* untuk diolah. *Parameter* yang telah diambil akan digunakan untuk memanggil *program* utama untuk validasi *invoice*. Jika *program* memberikan *output* 200, maka *program* akan memberikan *output success* dalam *postman* dan jika *program* memberikan *output* lain maka *output not success* dalam *postman*.

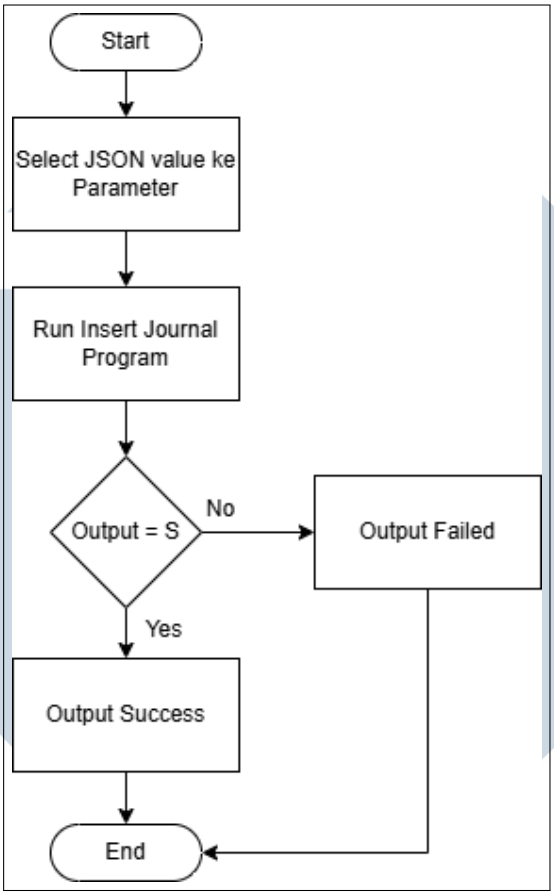
Berikut *program CREATE PVRV*:



Gambar 3.10. Flowchart CREATE PVRV API

Program dimulai dengan dengan mengambil *data* dari *REST API* untuk diolah. Lalu *program* akan menjalankan *program PV RV Auto*. Jika *output* 1 dan *output* 2 menghasilkan *code* 200 maka *program* akan menghasilkan *output success* pada *Postman*, jika ada *output* yang menghasilkan *code* selain 200 maka *program* maka menghasilkan *output erro*r pada *Postman*.

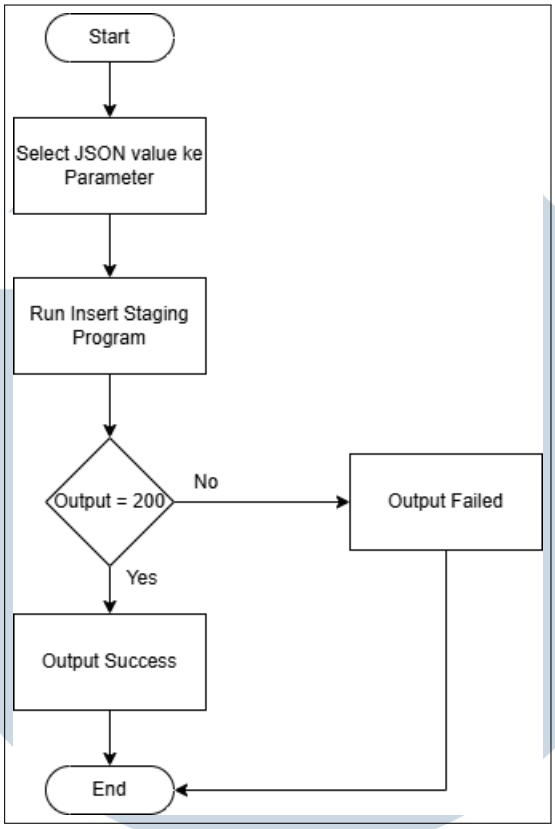
Berikut *program POST ASSYS JOURNAL*:



Gambar 3.11. Flowchart CREATE RFP

Program dimulai dengan dengan mengambil *data* dari *REST API* untuk diolah. *program* memanggil *program insert journal*. Jika *output* adalah S maka *program* akan menghasilkan *output* 200 pada *Postman* dan jika *output* lain maka *program* akan menghasilkan *output* 400 pada *Postman*.

Berikut *program PAYMENT SETTLEMENT*:

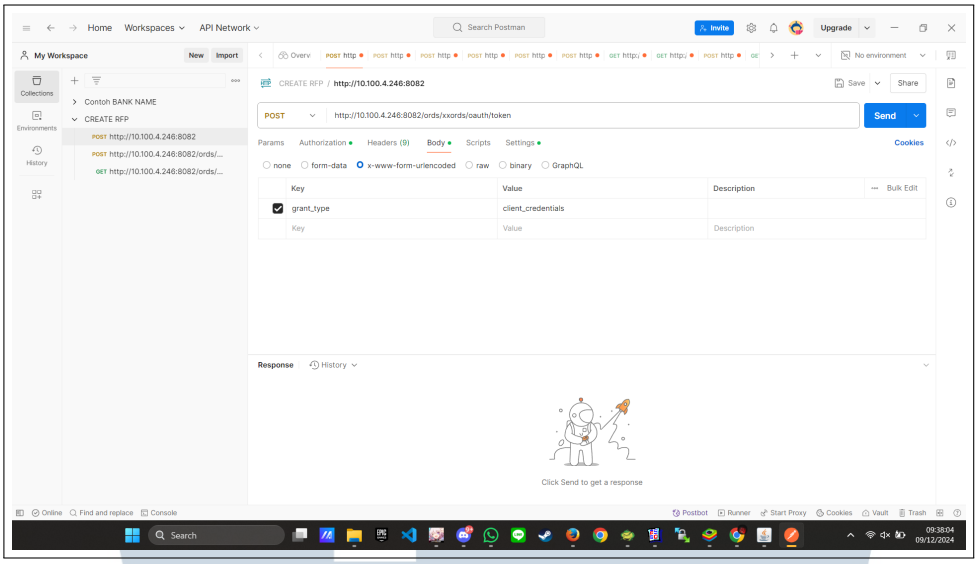


Gambar 3.12. Flowchart CREATE RFP

Program dimulai dengan dengan mengambil *data* dari *REST API* untuk diolah. *program* memanggil *program insert staging*. Jika *output* adalah 200 maka *program* akan menghasilkan *output success* pada *Postman* dan jika *output* lain maka *program* akan menghasilkan *output failed* pada *Postman*.

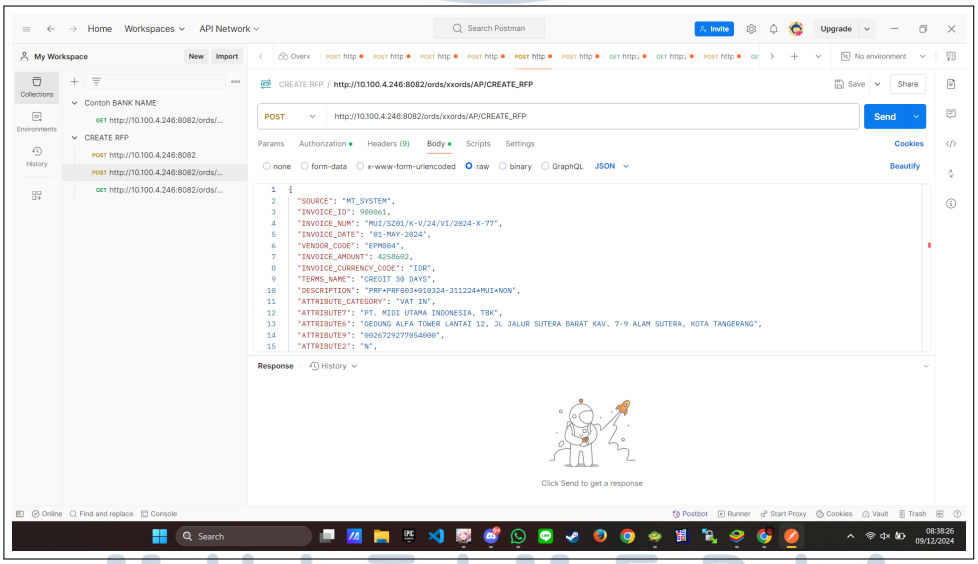
3.3.3 Proses Menjalankan Program

Dalam menjalankan *program API* dalam *ORBIT* diperlukan bantuan *Postman* untuk menghubungkan antara *database Oracle* dengan *Website ORBIT. Postman* digunakan untuk mengirim dan menerima *data* yang akan digunakan oleh *Oracle* dan *ORBIT. Oracle* akan menyimpan *data* yang dikirim oleh *ORBIT* dan memproses *data* yang dikirim menjadi laporan pembayaran. Laporan pembayaran yang telah dibuat oleh *Oracle* akan diambil *ORBIT* untuk dikirim kepada *user* sehingga proses pembayaran yang dilakukan oleh *user* dapat berjalan dengan lebih cepat. Berikut merupakan proses dalam menjalankan *Postman* dalam *program ORBIT*:



Gambar 3.13. Authorization Postman

Dalam menjalankan *Postman* perlu untuk mendapatkan *authorization* agar dapat mengakses *Postman* yang ingin digunakan. Dengan menjalankan *Postman* ini, maka akan muncul *token* sebagai *token authorization* terhadap *Postman* di *database* yang digunakan.



Gambar 3.14. CREATE RFP Postman

Dengan menggunakan *token* yang sudah didapatkan maka selanjutnya di dalam *Postman* yang dituju perlu untuk mengisi *body data* yang akan dikirim ke dalam *database*. *Data* ini disesuaikan dengan *setting* yang telah dibuat di dalam *database* dan akan menghasilkan respon dari *data* yang diberikan. Respon ini dapat berupa 200 yang berarti berhasil ataupun 400 yang berarti gagal.

3.4 Kendala dan Solusi yang Ditemukan

Kendala yang ditemukan selama melakukan proses magang adalah dengan pemahaman mengenai *ORBIT* dimana alur cara kerja *program ORBIT* cukup sulit untuk dimengerti untuk pertama kalinya. Serta terdapat beberapa metode dalam implementasi *API* dalam *Oracle* yang baru diketahui sehingga belum terlalu memahami mengenai cara pembuatan *API* dengan *Oracle*.

Solusi yang yang didapatkan adalah dengan membaca alur *flowchart* dan juga mempelajari mengenai cara implementasi *API* dalam *Oracle*. Serta juga dengan bertanya kepada mentor mengenai *project*.