

BAB 3

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Saat menjalani kegiatan magang di PT Cranium Royal Aditama, tugas yang dilakukan adalah sebagai *Fullstack Developer* yang memfokuskan pada *frontend* dan *backend*. Selama magang, Bapak Sugito selaku *VP Engineer* bertindak sebagai *Supervisor* langsung yang membimbing dan juga sebagai *Team Leader*. Untuk komunikasi antar supervisi dan anggota menggunakan aplikasi *Whatsapp* dan *Discord*. Platform yang digunakan untuk *sharing project* menggunakan *Github* sebagai *repository*.

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama kegiatan magang di PT Cranium Royal Aditama adalah membangun sistem *Enterprise Resource Planning* menggunakan *Java Springboot* sebagai *back-end* dan *Next.js* sebagai *framework* untuk *front-end*. Pengerjaan kali ini dimulai dengan pembuatan tampilan awal sebagai templat yang akan digunakan seterusnya berdasarkan desain pada *Figma*, lalu jika tampilan awal tersebut sudah selesai maka akan dilanjutkan pembuatan *CRUD* modul-modul berdasarkan tampilan awal yang sudah dibuat.

Peserta magang diberikan tugas untuk membuat templat halaman *Create* dan *Update*. Komponen global yang banyak digunakan seperti komponen *Modal*, *Breadcrumb* dan *Button* dibuat dan disesuaikan dengan desain *Figma*. Setelah template halaman untuk *CRUD* telah dibuat, pengerjaan selanjutnya adalah membuat halaman *CRUD* untuk tiap-tiap modul. Berikut modul dan submodul yang dikerjakan dalam pembuatan halaman *CRUD*

1. Modul Master
 - (a) Submodul *Item Supplier*
 - (b) Submodul *Salesman*
2. Modul Pembelian yaitu submodul *Purchasing Order Item Receipt*
3. Modul Produksi yaitu submodul *Production Planned Order*

3.3 Uraian Pelaksanaan Magang

Berikut adalah uraian pelaksanaan magang yang dilakukan di PT Cranium Royal Aditama.

3.3.1 Pelaksanaan Kerja Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

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

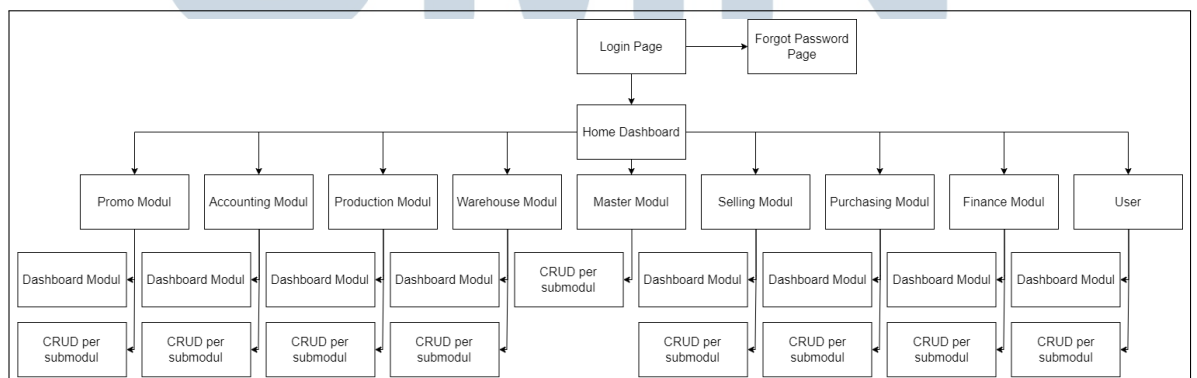
Minggu ke-	Aktivitas yang dikerjakan
1	Memepelajari dasar dari <i>React</i> dan <i>Next.js</i> dan mempelajari template yang diberikan
2	Mencoba menambahkan fitur CRUD untuk menghubungkan ke <i>backend</i>
3	Membuat halaman <i>create</i> dan <i>update</i> sesuai dengan <i>Figma</i>
4	Melanjutkan pembuatan halaman <i>create</i> dan <i>update</i> agar sesuai dengan <i>Figma</i> dan membuat komponen Modal
5	Melanjutkan pembuatan komponen modal agar dapat menjadi <i>Global</i> atau digunakan kembali dan merapihkan <i>styling</i> , mencoba membuat <i>Unit Test</i>
6	Mulai membuat halaman submodul master <i>Item Supplier</i> dimulai dengan perbaikan pada backend <i>Item Supplier</i>
7	Melanjutkan pembuatan halaman <i>Item Supplier</i> yaitu halaman <i>create</i> dan <i>update</i> , lalu penambahan fitur <i>searching</i>
8	Melanjutkan pembuatan halaman <i>dashboard view</i> dan <i>list paging</i> dan mulai membuat <i>unit test</i> untuk halaman <i>Item Supplier</i>
9	Melanjutkan ke pembuatan submodul master lainnya yaitu <i>Salesman</i> dan perbaikan <i>unit test</i> pada <i>Item Supplier</i>
10	Melanjutkan pembuatan halaman submodul master <i>Salesman</i>
11	Mulai membuat halaman modul <i>Purchasing</i> submodul <i>Order Item Receipt</i> dimulai dengan perbaikan pada backend <i>Order Item Receipt</i>
Dilanjutkan di halaman selanjutnya	

Tabel 3.1 – dilanjutkan dari tabel sebelumnya

Minggu ke-	Aktivitas yang dikerjakan
12	Melanjutkan pembuatan halaman <i>CRUD</i> dan implementasi fitur <i>searching</i> pada halaman <i>Order Item Receipt</i>
13	Melanjutkan pembuatan halaman <i>CRUD Order Item Receipt</i> dan revisi pada komponen global <i>breadcrumb</i>
14	Melakukan perbaikan pada <i>backend Master Warehouse</i>
15	Mulai membuat halaman modul <i>Production</i> submodul <i>Planned Order</i> dimulai dengan perbaikan pada <i>backend Planned Order</i>
16	Melanjutkan pembuatan halaman <i>CRUD Planned Order</i>
17	Melanjutkan pembuatan halaman <i>CRUD Planned Order</i>
18	Melakukan penambahan <i>breadcrumb</i> pada submodul yang belum diterapkan
19	Mulai membuat halaman modul <i>Warehouse</i> submodul <i>Item Receipt</i> dimulai dengan perbaikan pada <i>backend Item Receipt</i>

3.3.2 Sitemap

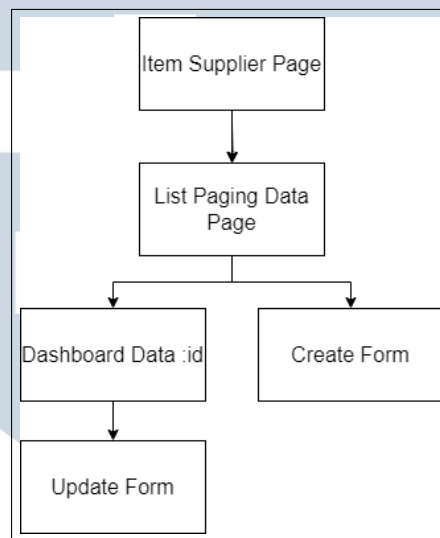
Sitemap adalah alat pengembangan web yang dapat membantu pengunjung melakukan navigasi situs web dengan lebih mudah. Sistem ERP yang sedang dikembangkan memiliki *sitemap* yang mempunyai halaman *login* merupakan halaman awal yang diakses pertama kali, lalu halaman berikutnya merupakan turunan yang hanya dapat diakses dari halaman tertentu.



Gambar 3.1. Sitemap sistem ERP

A. Sitemap Master Item Supplier

Submodul *Master Item Supplier* memiliki *sitemap* yang diawali dengan halaman *list paged data*. Halaman selanjutnya yang diakses yaitu *dashboard view* data berdasarkan id dan *create form*. Pada halaman *dashboard view* data, halaman yang dapat diakses yaitu *update form*. Berikut merupakan *sitemap* submodul *Master Item Supplier* yang dapat dilihat pada Gambar 3.2.

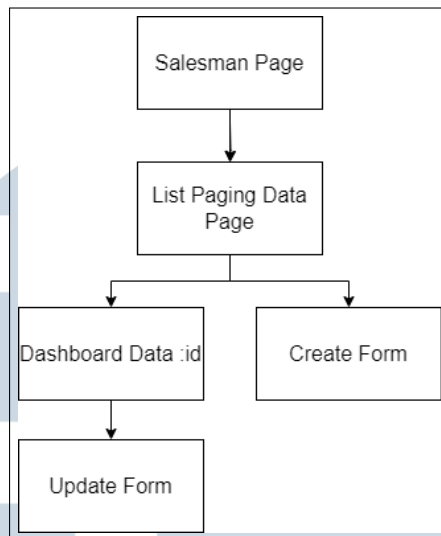


Gambar 3.2. Sitemap halaman *Item Supplier*

B. Sitemap Master Salesman

Submodul *Master Salesman* memiliki *sitemap* yang diawali dengan halaman *list paged data*. Halaman selanjutnya yang diakses yaitu *dashboard view* data berdasarkan id dan *create form*. Pada halaman *dashboard view* data, halaman yang dapat diakses yaitu *update form*. Berikut merupakan *sitemap* submodul *Master Salesman* yang dapat dilihat pada Gambar 3.3.

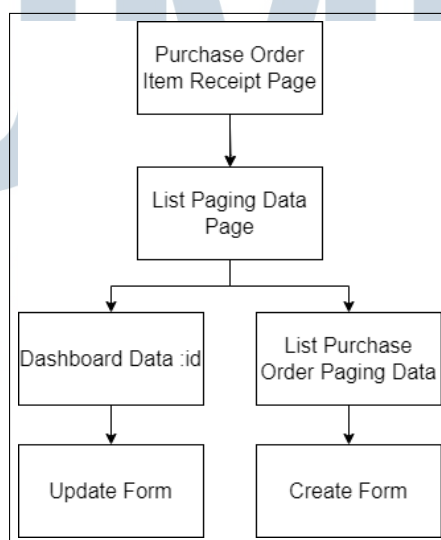
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.3. Sitemap halaman *Salesman*

C. Sitemap Purchasing Order Item Receipt

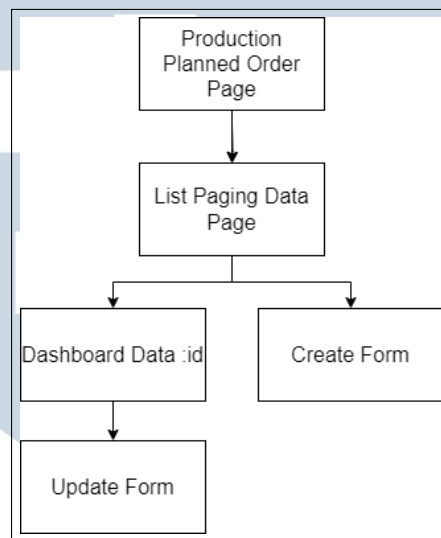
Submodul Purchasing Order Item Receipt memiliki *sitemap* yang diawali dengan halaman *list paged data*. Halaman selanjutnya yang diakses yaitu *dashboard view* data berdasarkan id dan *list Purchase Order data*. Pada halaman *dashboard view* data, halaman yang dapat diakses yaitu *update form*. Pada *list Purchase Order data*, halaman yang dapat diakses selanjutnya yaitu *create form*. Berikut merupakan *sitemap* submodul Purchasing Order Item Receipt yang dapat dilihat pada Gambar 3.4.



Gambar 3.4. Sitemap halaman *Purchasing Order Item Receipt*

D. Sitemap Production Planned Order

Submodul Production Planned Order memiliki *sitemap* yang diawali dengan halaman *list paged data*. Halaman selanjutnya yang diakses yaitu *dashboard view* data berdasarkan id dan *create form*. Pada halaman *dashboard view* data, halaman yang dapat diakses yaitu *update form*. Berikut merupakan *sitemap* submodul Production Planned Order yang dapat dilihat pada Gambar 3.5.



Gambar 3.5. Sitemap halaman *Production Planned Order*



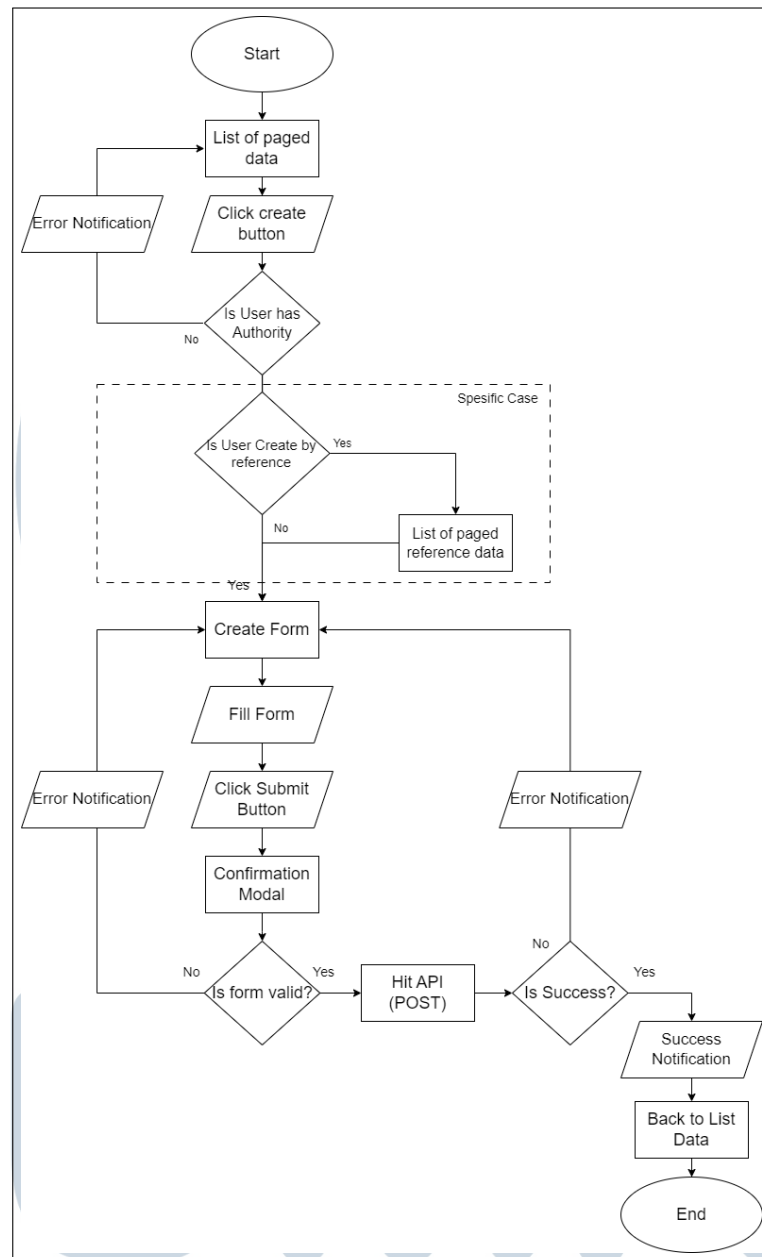
3.3.3 Flowchart

Flowchart digunakan untuk mengetahui alur kerja dari suatu sistem. Pada sistem ERP yang dibangun, *flowchart* digunakan untuk menjelaskan alur secara garis besar CRUD (*Create, Read, Update, Delete*) modul.

A. Create

Pada alur *create*, dimulai dari halaman *list paging* dari tiap-tiap submodul lalu *user* akan menekan tombol *create*. Sebelum masuk ke dalam halaman *create form*, *user* akan dicek apakah memiliki otoritas, jika tidak punya maka notifikasi *error* akan keluar. Jika sudah memasuki *create form*, *user* akan melakukan pengisian data pada *form* tersebut. Setelah data sudah diisi, *user* akan menekan tombol *create* yang memunculkan *modal* konfirmasi. Jika data tidak *valid* seperti ada *textfield* yang perlu diisi namun *user* tidak mengisinya, maka data tidak bisa di *submit*. Apabila data sudah *valid*, setelah menekan tombol konfirmasi pada *modal*, *frontend* akan melakukan *hit API*. Data yang berhasil masuk yang berarti *hit API* berhasil, sedangkan jika gagal maka akan mengeluarkan notifikasi *error*. Setelah data berhasil dibuat, halaman akan kembali ke *list paging* data submodul tersebut.





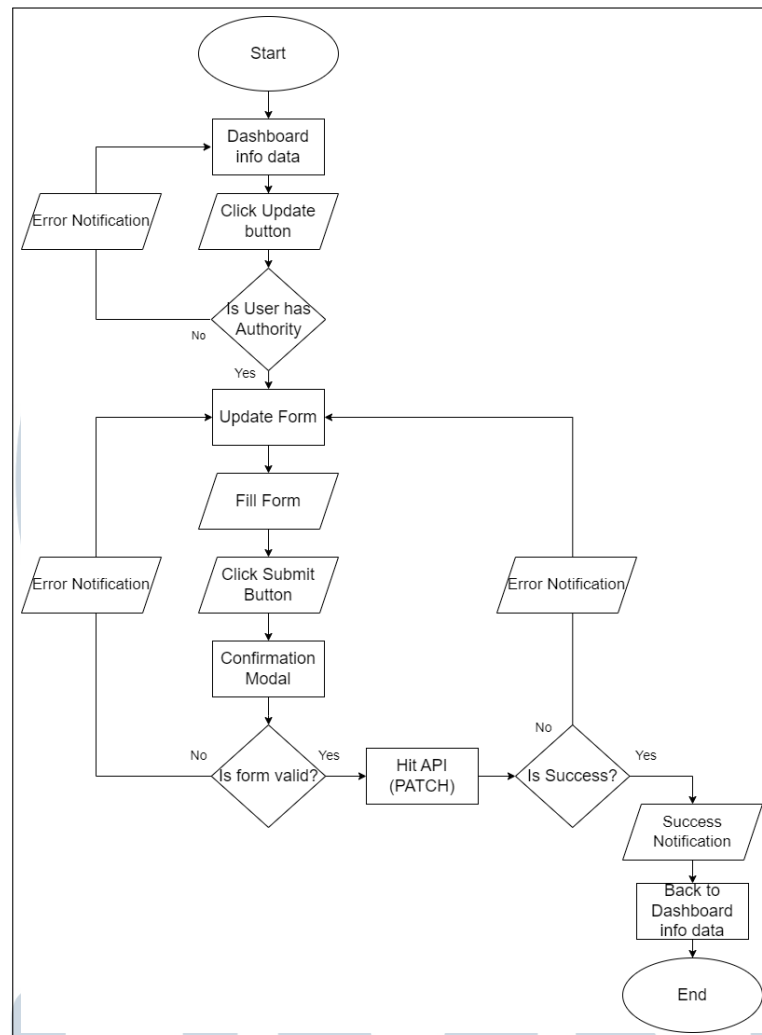
Gambar 3.6. Alur *create*

Gambar 3.6 menunjukkan bagaimana alur *create* dapat berkerja dari awal hingga akhir. Untuk sebagian kasus, pembuatan *create form* dapat berdasarkan dokumen referensi atau membuat tanpa dokumen referensi. Jika *user* memilih berdasarkan dokumen referensi, halaman akan memunculkan *list paging* dari dokumen referensi. Selanjutnya *user* dapat memilih data yang akan dijadikan referensi dokumen saat membuat data baru

B. Update

Alur *update* cukup mirip dengan alur *create*, dimulai dari halaman *dashboard info* yang berdasarkan *id* dari data submodul lalu *user* akan menekan tombol *update*. Sebelum masuk ke dalam halaman *update form*, *user* akan dicek apakah memiliki otoritas, jika tidak punya maka notifikasi *error* akan keluar. Jika sudah memasuki *update form*, data akan muncul pada *field* masing-masing dan *user* hanya perlu mengubah data. Setelah data sudah diubah, *user* akan menekan tombol *update* yang memunculkan *modal* konfirmasi. Jika data tidak valid seperti *textfield* yang perlu diisi namun *user* tidak mengisinya, maka data tidak bisa di *submit*. Apabila data sudah valid, setelah menekan tombol konfirmasi pada *modal*, *frontend* akan melakukan *hit API*. Data yang berhasil masuk yang berarti *hit API* berhasil, sedangkan jika gagal maka akan mengeluarkan notifikasi *error*. Setelah data berhasil dibuat, halaman akan kembali ke *dashboard* dari submodul tersebut berdasarkan *id* data.



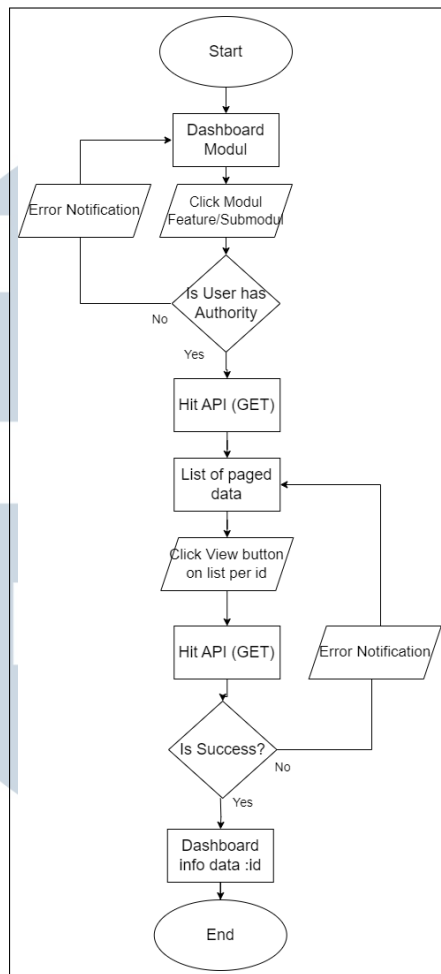


Gambar 3.7. Alur *update*

Berikut merupakan alur *update* yang dapat dilihat pada Gambar 3.7 yang menunjukkan bagaimana proses alur *update* dari awal hingga akhir.

C. Read

Pada alur *read*, dimulai dari halaman *dashboard* dari tiap-tiap modul lalu *user* akan menekan singkatan dari modul pada *sidebar* selain *dashboard*. Sebelum masuk ke dalam halaman *list paging*, *user* akan dicek apakah memiliki otoritas, jika tidak punya maka notifikasi *error* akan keluar. Jika sudah memasuki *list paging*, *user* akan menekan *icon opening file* untuk melihat data berdasarkan *id* yang ada pada *list paging*. *User* akan dicek otoritasnya kembali. Setelah klik berhasil, maka akan muncul halaman *dashboard* berdasarkan *id* data.

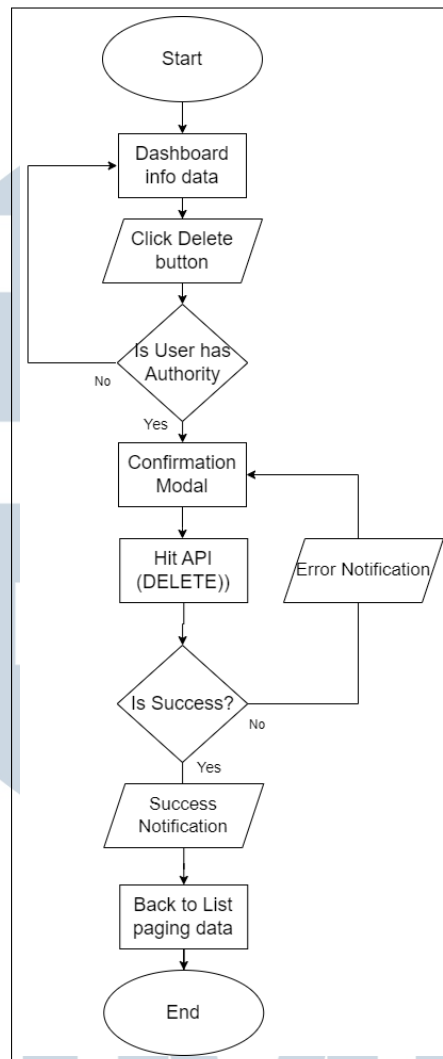


Gambar 3.8. Alur *read*

Berikut merupakan alur *read* yang dapat dilihat pada Gambar 3.8 yang menunjukkan bagaimana proses alur *read* dari awal hingga akhir.

D. Delete

Pada alur *delete*, dimulai dari halaman *dashboard info* yang berdasarkan *id* dari data submodul lalu *user* akan menekan tombol *delete*. Selanjutnya akan muncul *modal* sebagai konfirmasi penghapusan data. Jika data berhasil dihapus, maka halaman akan berpindah ke *list paging* dari submodul tersebut. Untuk otoritas *delete*, tombol *delete* tidak akan ditampilkan jika tidak memiliki otoritas.



Gambar 3.9. Alur *delete*

Berikut merupakan alur *delete* yang dapat dilihat pada Gambar 3.9 yang menunjukkan bagaimana proses alur *delete* dari awal hingga akhir.

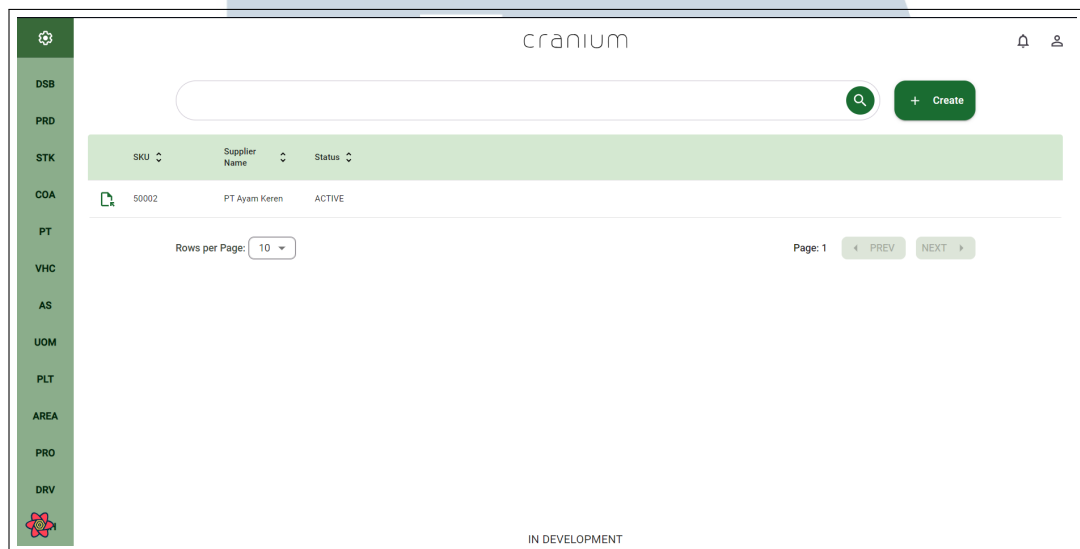
3.3.4 Implementasi Halaman

A. Halaman Master Item Supplier

Master Item Supplier Memiliki halaman *list paged data*, *dashboard view* berdasarkan id, *create form*, dan *update form*.

A.1 Halaman List Paged Data

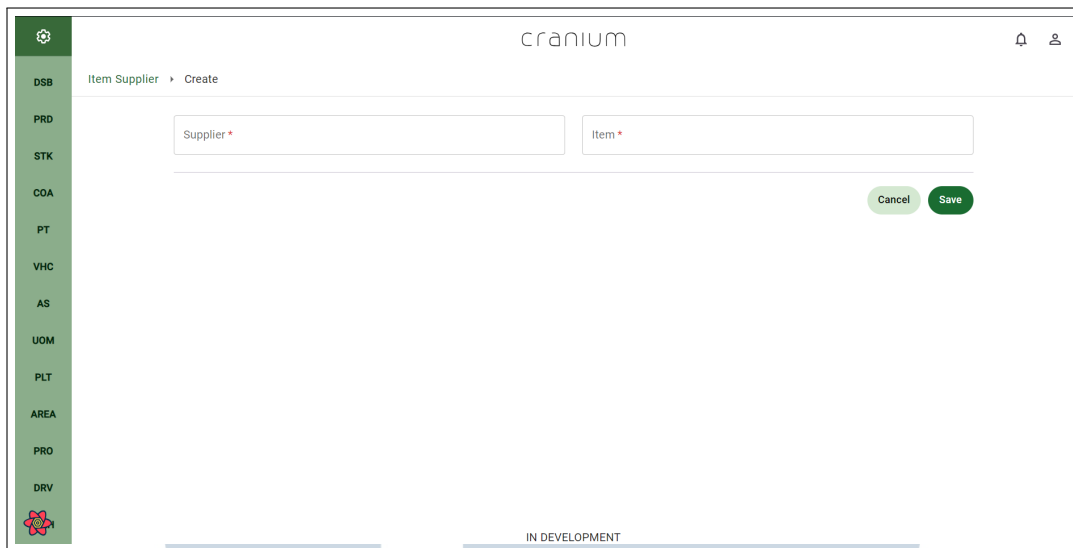
Halaman *list paged* data *Item Supplier* memiliki komponen *searching*, tombol *create* untuk menuju halaman *create form*, dan tabel berisi data yang memiliki tombol setiap baris untuk menuju halaman *dashboard view* berdasarkan id dari data. Berikut merupakan hasil implementasi dari halaman *list paged data Item Supplier* pada Gambar 3.10.



Gambar 3.10. Tampilan List Paged Data *Item Supplier*

A.2 Halaman Create

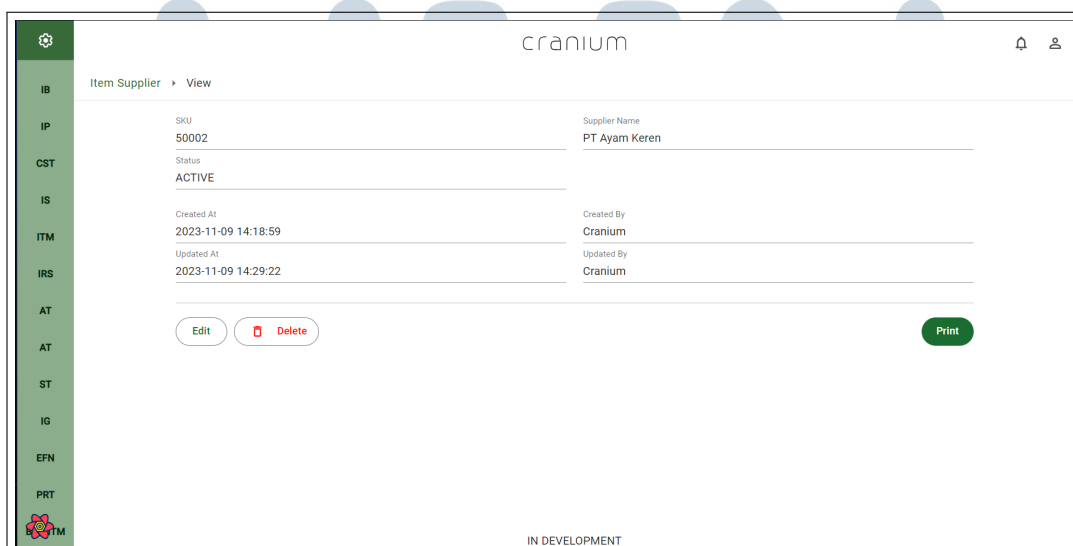
Halaman *create* data *Item Supplier* menampilkan sebuah *form* yang akan diisi oleh *user* untuk membuat data baru. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan dibuat dan akan berpindah ke halaman *list paged* data, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *list paged* data. Berikut merupakan hasil implementasi dari halaman *create* data *Item Supplier* pada Gambar 3.11.



Gambar 3.11. Tampilan Create Form *Item Supplier*

A.3 Halaman Dashboard View Data

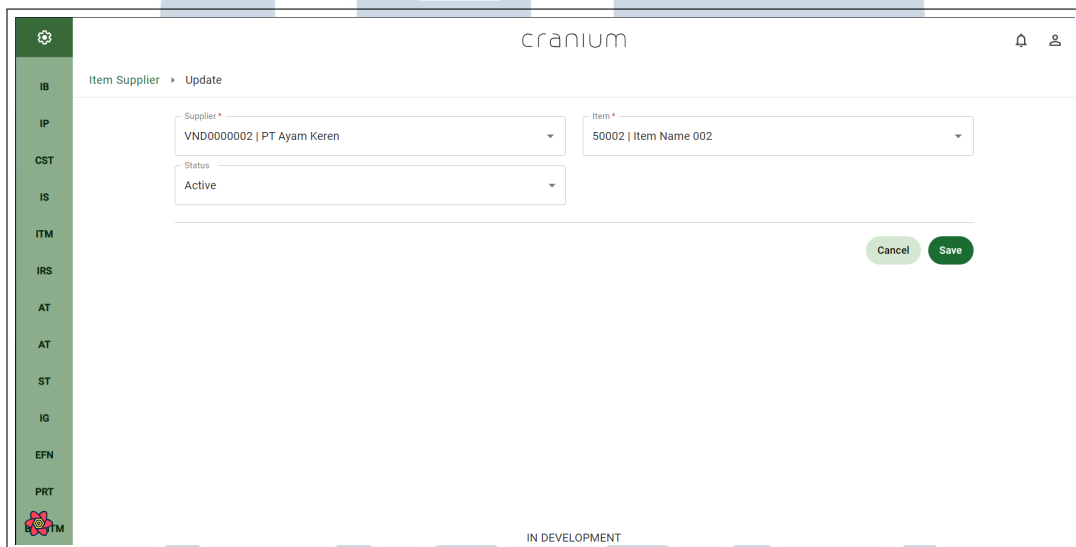
Halaman *dashboard view* data *Item Supplier* menampilkan informasi lebih lengkap dari data setiap *Item Supplier* berdasarkan id. Komponen tombol *edit* untuk menuju halaman *update* dari data dan tombol *delete* untuk menghapus data tersebut. Berikut merupakan hasil implementasi dari halaman *dashboard view* data *Item Supplier* pada Gambar 3.12.



Gambar 3.12. Tampilan Dashbaord View Data *Item Supplier*

A.4 Halaman Update

Halaman *update* data *Item Supplier* menampilkan sebuah *form* yang tiap *field* terisi dengan data dari *Item Supplier* berdasarkan id. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan diperbarui dan akan berpindah ke halaman *dashboard view* data, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *dashboard view* data berdasarkan id. Berikut merupakan hasil implementasi dari halaman *update* data *Item Supplier* pada Gambar 3.13.



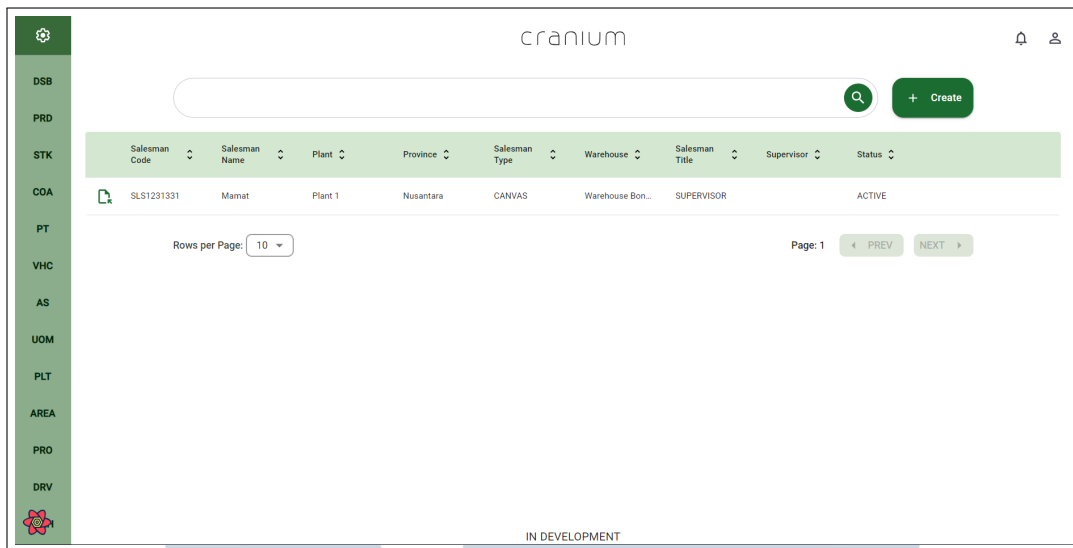
Gambar 3.13. Tampilan Update Form *Item Supplier*

B. Halaman Master Salesman

Master Salesman Memiliki halaman *list paged* data, *dashboard view* berdasarkan id, *create form*, dan *update form*.

B.1 Halaman List Paged Data

Halaman *list paged* data *Salesman* memiliki komponen *searching*, tombol *create* untuk menuju halaman *create form*, dan tabel berisi data yang memiliki tombol setiap baris untuk menuju halaman *dashboard view* berdasarkan id dari data. Berikut merupakan hasil implementasi dari halaman *list paged* data *Salesman* pada Gambar 3.14.



Gambar 3.14. Tampilan List Paged Data *Salesman*

B.2 Halaman Create

Halaman *create* data *Salesman* menampilkan sebuah *form* yang akan diisi oleh *user* untuk membuat data baru. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan dibuat dan akan berpindah ke halaman *list paged* data, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *list paged* data. Berikut merupakan hasil implementasi dari halaman *create* data *Salesman* pada Gambar 3.15.

Gambar 3.15. Tampilan Create Form *Salesman*

B.3 Halaman Dashboard View Data

Halaman *dashboard view data Salesman* menampilkan informasi lebih lengkap dari data setiap *Salesman* berdasarkan id. Komponen tombol *edit* untuk menuju halaman *update* dari data dan tombol *delete* untuk menghapus data tersebut. Berikut merupakan hasil implementasi dari halaman *dashboard view data Salesman* pada Gambar 3.16.

Field	Value	Field	Value
Salesman Code	SLS1231331	Chart of Account	1
Salesman Name	Mamat	Sales Chart of Account	1
KTP Number	00038497123	Sales Return Chart of Account	1
Salesman Address	Ji babat 3	Price Tag	1
Phone Number	0812344123112	Warehouse	Warehouse Bonang 1
Plant	Plant 1	Salesman Title	SUPERVISOR
Province	Nusantara	Supervisor	
Salesman Type	CANVAS	Status	ACTIVE
Created At	2023-11-18 17:27:34	Created By	Cranium
Updated At	2023-11-18 17:27:34	Updated By	Cranium

Gambar 3.16. Tampilan Dashboard View Data *Salesman*

B.4 Halaman Update

Halaman *update data Salesman* menampilkan sebuah *form* yang tiap *field* terisi dengan data dari *Salesman* berdasarkan id. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan diperbarui dan akan berpindah ke halaman *dashboard view data*, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *view data* berdasarkan id. Berikut merupakan hasil implementasi dari halaman *update data Salesman* pada Gambar 3.17.

Gambar 3.17. Tampilan Update Form *Salesman*

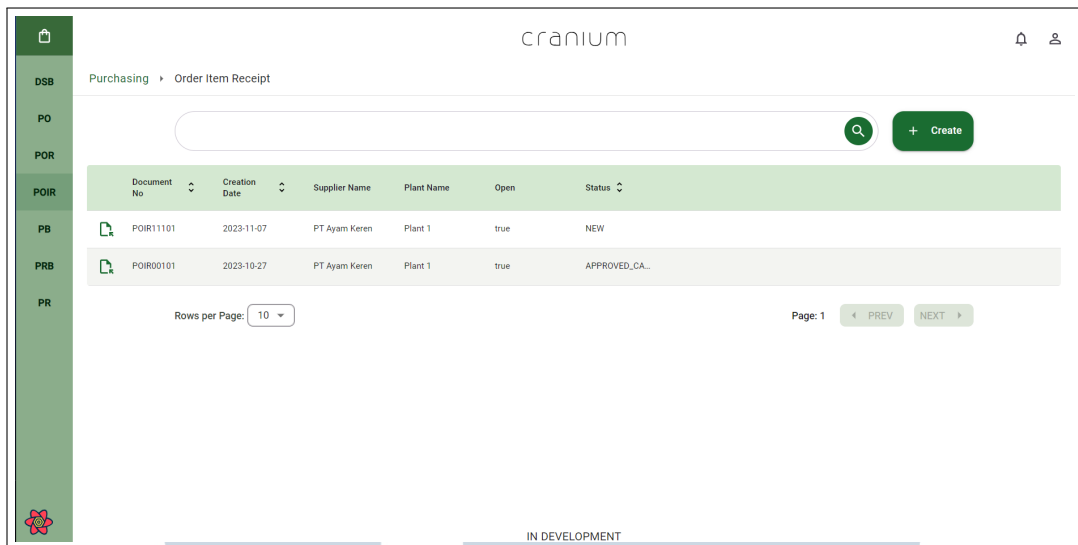
C. Halaman Purchasing Order Item Receipt

Purchasing Order Item Receipt Memiliki halaman *list paged* data, *list purchase order reference*, *dashboard view* berdasarkan id, *create form*, dan *update form*.

C.1 Halaman List Paged Data

Halaman *list paged* data *Order Item Receipt* memiliki komponen *searching*, tombol *create* untuk menuju halaman *create form*, dan tabel berisi data yang memiliki tombol setiap baris untuk menuju halaman *dashboard view* berdasarkan id dari data. Berikut merupakan hasil implementasi dari halaman *list paged* data *Order Item Receipt* pada Gambar 3.18.

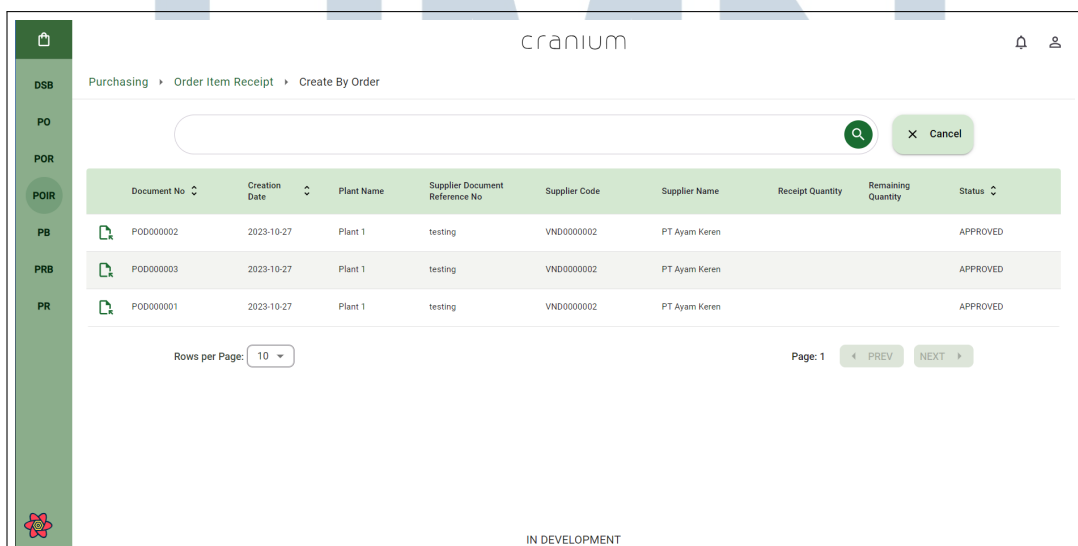
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.18. Tampilan List Paged Data *Order Item Receipt*

C.2 Halaman List Create by Order Paged Data

Halaman *list create by purchase order paged* data memiliki komponen *searching*, tombol *cancel* untuk menuju kembali ke halaman *list paged*, dan tabel berisi data yang memiliki tombol setiap baris untuk menuju halaman *create* berdasarkan id dari data *Purchase Order*. Berikut merupakan hasil implementasi dari halaman *list create by purchase order paged* data *Order Item Receipt* pada Gambar 3.19.



Gambar 3.19. Tampilan List *Purchase Order* Paged Data

C.3 Halaman Create

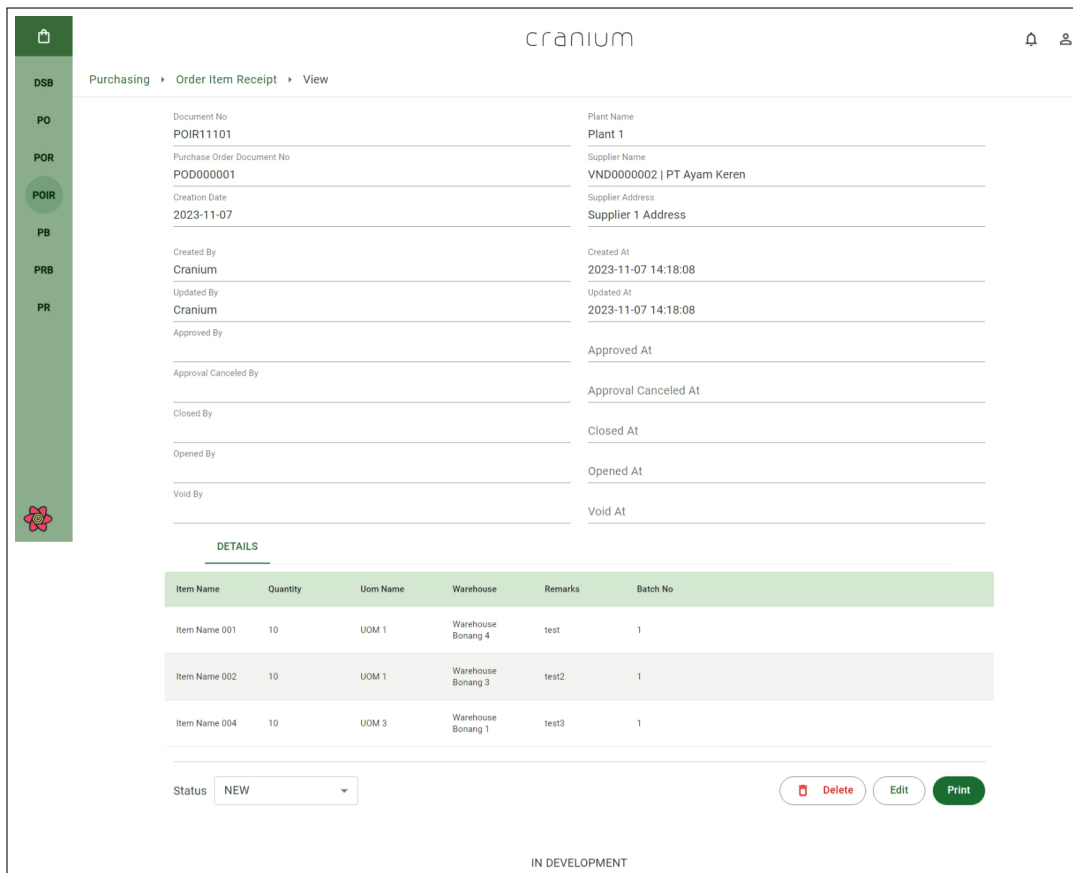
Halaman *create* data *Order Item Receipt* menampilkan sebuah *form* yang beberapa *field* terisi dengan data dari Purchase Order berdasarkan id. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan dibuat dan akan berpindah ke halaman *list paged* data, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *list paged* data. Berikut merupakan hasil implementasi dari halaman *create* data *Order Item Receipt* pada Gambar 3.20.

Skus	Item Name	Quantity	Uom Name	Warehouse	Remarks
50001	Item Name 001	10	UOM 1		test
50002	Item Name 002	10	UOM 1		test2
50004	Item Name 004	10	UOM 3		test3

Gambar 3.20. Tampilan Create Form *Order Item Receipt*

C.4 Halaman Dashboard View Data

Halaman *dashboard view* data *Order Item Receipt* menampilkan informasi lebih lengkap dari data setiap *Order Item Receipt* berdasarkan id. Komponen tombol *edit* untuk menuju halaman *update* dari data dan tombol *delete* untuk menghapus data tersebut. Lalu komponen *field* status untuk mengubah status dari dokumen tersebut. Berikut merupakan hasil implementasi dari halaman *dashboard view* data *Order Item Receipt* pada Gambar 3.21.



Gambar 3.21. Tampilan Dashboard View Data *Order Item Receipt*

C.5 Halaman Update

Halaman *update* data *Order Item Receipt* menampilkan sebuah *form* yang tiap *field* terisi dengan data dari Salesman berdasarkan id. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan diperbarui dan akan berpindah ke halaman *dashboard view* data, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *dashboard view* data berdasarkan id. Berikut merupakan hasil implementasi dari halaman *update* data *Order Item Receipt* pada Gambar 3.22.

Purchasing > Order Item Receipt > Update Order Item Receipt

Purchase Order Document No*
POD000001

Plant Name*
Plant 1

Document No*
POIR11101

Supplier Name*
VND0000002 | PT Ayam Keren

Creation Date*
07/11/2023

Supplier Address*
Supplier 1 Address

Details

<input type="checkbox"/>	Skus	Item Name	Quantity	Uom Name	Warehouse	Remarks
<input type="checkbox"/>	50001	Item Name 001	10	UOM 1	Warehous...	test
<input type="checkbox"/>	50002	Item Name 002	10	UOM 1	Warehous...	test2
<input type="checkbox"/>	50004	Item Name 004	10	UOM 3	Warehous...	test3

Delete selected item

Cancel Save

IN DEVELOPMENT

Gambar 3.22. Tampilan Update Form *Order Item Receipt*

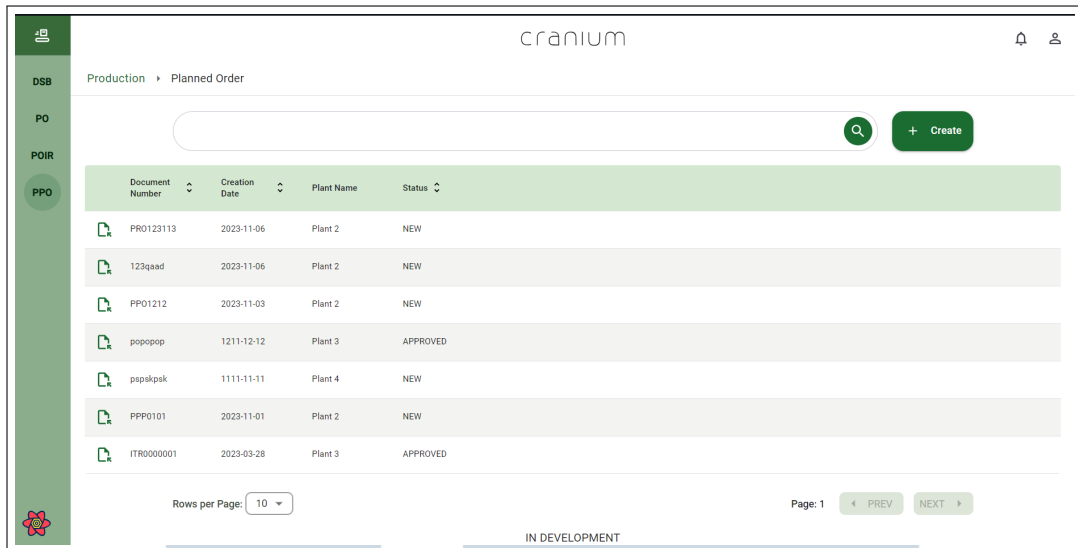
D. Halaman Production Planned Order

Production Planned Order memiliki halaman *list paged* data, *dashboard view* berdasarkan id, *create form*, dan *update form*.

D.1 Halaman List Paged Data

Halaman *list paged* data *Planned Order* memiliki komponen *searching*, tombol *create* untuk menuju halaman *create form*, dan tabel berisi data yang memiliki tombol setiap baris untuk menuju halaman *dashboard view* berdasarkan id dari data. Berikut merupakan hasil implementasi dari halaman *list paged* data *Planned Order* pada Gambar 3.23.

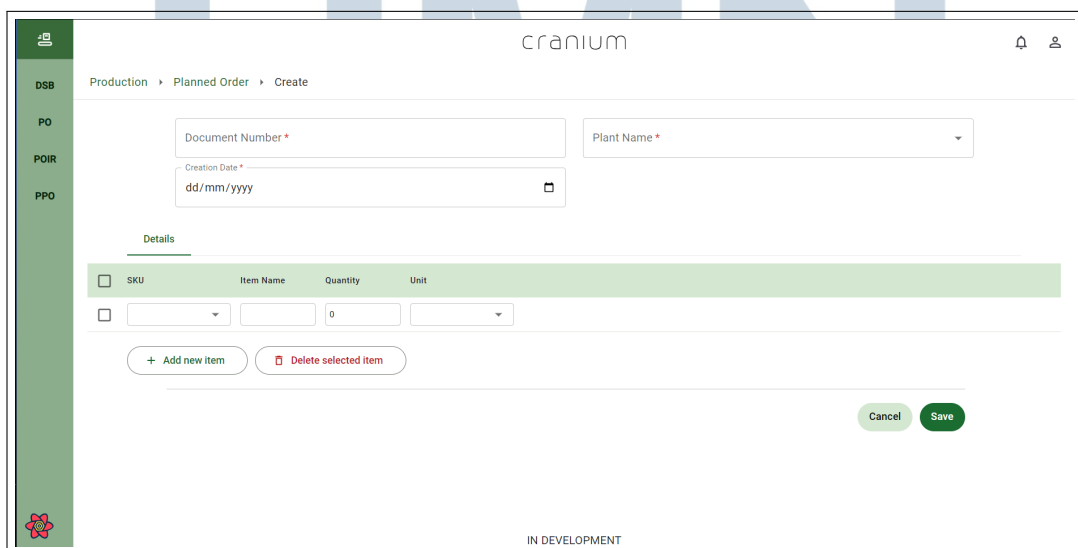
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.23. Tampilan List Paged *Data Planned Order*

D.2 Halaman Create

Halaman *create* data *Planned Order* menampilkan sebuah *form* yang akan diisi oleh *user* untuk membuat data baru. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan dibuat dan akan berpindah ke halaman *list paged* data, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *list paged* data. Berikut merupakan hasil implementasi dari halaman *create* data *Planned Order* pada Gambar 3.24.



Gambar 3.24. Tampilan Create Form *Planned Order*

D.3 Halaman Dashboard View Data

Halaman *dashboard view* data *Planned Order* menampilkan informasi lebih lengkap dari data setiap *Planned Order* berdasarkan id. Komponen tombol *edit* untuk menuju halaman *update* dari data dan tombol *delete* untuk menghapus data tersebut. Lalu komponen *field* status untuk mengubah status dari dokumen tersebut. Berikut merupakan hasil implementasi dari halaman *dashboard view* data *Planned Order* pada Gambar 3.25.

SKU	Item Name	Quantity	Unit
50001	Item Name 001	100	UOM 1
50002	Item Name 002	50	UOM 2

Gambar 3.25. Tampilan Dashboard View Data *Planned Order*

D.4 Halaman Update

Halaman *update* data *Planned Order* menampilkan sebuah *form* yang tiap *field* terisi dengan data dari Salesman berdasarkan id. Jika tombol *save* ditekan maka komponen *modal* akan muncul sebagai konfirmasi kepada *user* apakah data akan diperbarui dan akan berpindah ke halaman *dashboard view* data, sedangkan jika tombol *cancel* ditekan maka halaman akan berpindah ke halaman *dashboard view* data berdasarkan id. Berikut merupakan hasil implementasi dari halaman *update* data *Planned Order* pada Gambar 3.26.

Gambar 3.26. Tampilan Update Form *Planned Order*

3.3.5 Implementasi Unit Testing

Pembuatan *unit testing* dilakukan dengan tujuan bahwa program dapat berjalan sesuai yang diharapkan. Pengujian unit biasanya dilakukan secara otomatis dan fokus pada bagian-bagian terkecil dari kode, seperti fungsi atau metode, untuk memverifikasi kebenarannya. *Framework* yang digunakan pada *unit testing* adalah *Jest*. Pada sistem ERP ini, *unit testing* dibuat dengan menyesuaikan alur kerja berdasarkan *CRUD* yang sudah dibuat. Setiap submodul yang sudah selesai dilakukan implementasi, *unit test* akan dibuat untuk memastikan apakah setiap submodul sudah berjalan sesuai dengan alur kerja sistem.

Pada *list paging* dan *dashboard view* berdasarkan id, *unit testing* dibuat untuk memastikan data dapat tampil pada halaman. Bagian *create* dan *update* memastikan bahwa data dapat dikirim dengan *API* yang sesuai dan berhasil memunculkan notifikasi bahwa data berhasil dibuat. Pada bagian *delete*, *unit test* diawali dengan menampilkan *dashboard view* yang memiliki komponen tombol *delete*. Setelah tombol *delete* ditekan, *API* akan dipanggil dan akan menampilkan notifikasi jika data berhasil dihapus.

3.4 Kendala dan Solusi yang Ditemukan

Kendala yang ditemukan pada saat pelaksanaan magang yaitu:

- Beberapa komponen yang membutuhkan waktu yang cukup lama pengerjaan seperti *field autocomplete* dikarenakan kompleksitas yang cukup tinggi.

- Mempelajari *framework Next.js* seperti *file-system based router* yang dimiliki oleh *Next.js* untuk mendukung pengembangan sistem ERP ini karena belum pernah digunakan sebelumnya.
- Banyaknya perubahan pada desain *figma* dikarenakan *project* yang dikerjakan masih dalam tahap awal pengembangan.

Solusi yang ditemukan pada saat pelaksanaan magang yaitu:

- Mencari referensi terkait komponen yang sulit diimplementasikan.
- Mencari tutorial dan membaca dokumentasi dari *Next.js*
- Rutin melaksanakan rapat dengan supervisi, desainer sehingga proses pekerjaan dapat terkontrol dengan baik.

