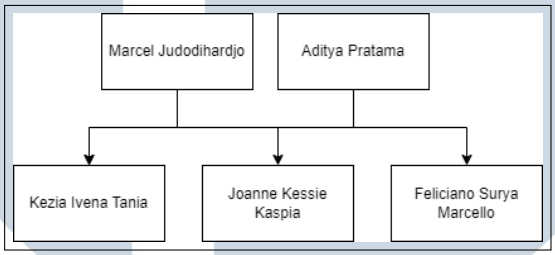
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Selama berjalannya praktik kerja magang di perusahaan Global Innovation Technology, posisi yang diberikan berada pada divisi Software Engineer di bawah pimpinan Bapak Aditya Pratama dan Bapak Marcel Judodihardjo. Proyek ini dikerjakan dalam sebuah *team* yang terdiri dari tiga orang seperti yang terlihat pada Gambar 3.1.



Gambar 3.1. Struktur organisasi tim

Tugas yang dikerjakan adalah membangun fitur-fitur *website sales tracking* yang dapat digunakan untuk memantau pekerjaan *sales* di luar kantor. Fitur-fitur ini akan dilengkapi dengan aspek lainnya, seperti fitur pencarian atau fitur filter, untuk membantu pengguna dalam menggunakan *website*.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang di PT Global Innovation Technology tugas yang harus dikerjakan antara lain membuat *database* produk, halaman dan *database product management*, halaman dan *database route management*, dan halaman *dashboard analytic*. *Website* ini dibangun menggunakan bahasa pemrograman JavaScript dan *framework* Angular, sedangkan *database* dibangun menggunakan bahasa pemrograman Java dan *framework* Spring Boot. Fitur-fitur yang akan dibangun akan diarahkan secara bertahap sehingga fitur sebelumnya harus diselesaikan terlebih dahulu untuk membangun fitur lainnya.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu

Minggu Ke -	Pekerjaan yang dilakukan
1	*M*embuat *database* dan *service* bagian produk.
2	*M*embuat CRUD produk dan tampilan halaman *product management*.
3	*M*enyiapkan service dan tampilan *dashboard analytic*.
4	*M*enghubungkan tampilan *dashboard analytic* dengan *database* dan menerapkan filter pada data tersebut.
5	*M*emperbaiki pengambilan data dan tampilan untuk *dashboard analytic* yang masih belum sesuai.
6	*M*emperbaiki fitur filter pada halaman *dahsboard analytic* yang kurang sesuai karena perubahan.
7	*M*embuat fitur *search* atau pencarian pada halaman *product management*.
8	*M*enerapkan filter *pie chart* pada halaman *dashboard analytic* bagian *top product*.
9	*M*embuat *database* dan *service* untuk *Route management*.
10	*M*elakukan penggabungan *database* dengan milik Joanne dan Feliciano.
11	*M*elakukan penggabungan file Angular dengan milik Joanne.
12	*M*enyiapkan halaman yang akan menampilkan *route management*.
13	*M*enghubungkan *database* dengan tampilan halaman *route management*.
14	*M*emperbaiki tabel pada halaman *route management* agar memudahkan pengguna.
15	*M*emperbaiki kesahalan saat penggabungaan file Angular pada halaman *order management*.
16	*M*emperbaiki kesahalan saat pengambilan data pada halaman *order management* dan *order history*.
17	*M*enambah filter pada halaman *order management* dan menghubungkan halaman ini dengan *customer management*.
18	*M*elaporkan hasil pekerjaan *website sales tracking* yang telah selesai.
	Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan yang dilakukan tiap minggu (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
19	*M*emperbaiki halaman-halaman *website* sesuai arahan *supervisor* dari minggu sebelumnya.
20	*M*emperbaiki data-data pada *database* yang belum lengkap sehingga dapat digunakan lebih maksimal.

Pada minggu pertama dan kedua mengerjakan *database* dan halaman produk. Halaman produk yang dibuat pada minggu kedua dapat menampilkan data produk, menambahkan data produk, mengubah data produk, dan menghapus data produk. Data produk diambil dari *database* yang telah dibuat terlebih dahulu di minggu pertama.

Pada minggu ketiga hingga minggu keenam membuat halaman *dashboard analytic*. *Dashboard analytic* akan menampilkan performa *sales* dan pembelian produk sesuai jangka waktu tertentu. Pada halamn ini pengguna dapat melihat peringkat *sales* maupun produk dari *database*.

Pada minggu ketujuh, membuat fitur pencarian dan filter untuk halaman produk. Untuk mempermudah pengguna mencari produk tertentu, dibuatlah fitur ini. Pengguna dapat mencari nama produk ataupun menggunakan *tree* level dan nama *parent* produk yang diinginkan.

Pada minggu kedelapan, menambahkan fitur pada *pie chart* di halaman *dashboard analytic*. Filter ini berada pada data produk yang dipisahkan menjadi tiga *pie chart* (sesuai level produk). Sehingga data produk *pie chart* untuk level yang lebih rendah akan menyesuaikan dengan level produk yang lebih tinggi yang dipilih.

Pada minggu kesembilan membuat *database* untuk *route management*. *Database* ini akan digunakan untuk menampilkan rute *sales* setiap bulannya untuk mengunjungi *customer tertentu*. Halaman yang akan menampilkan data ini akan dibuat di minggu selanjutnya.

Pada minggu kesepuluh dilakukan penggabungan *database* dengan tim sehingga tidak terjadi kesalahpahaman. Apabila terdapat fungsi yang sama pada *service database*, maka akan didiskusikan untuk menghapus salah satu *service*. Selain itu apabila ditemukan *service* yang tidak digunakan sama sekali, maka akan didiskusikan apabila *service* tersebut dapat dihapus.

Pada minggu kesebelas dilakukan penggabungan file Angular untuk mendapatkan seluruh halaman *website* yang dikerjakan tim. Hal ini dilakukan untuk memastikan seluruh halaman dapat diakses dan data yang ditampilkan sudah sesuai.

Pada minggu kedua belas hingga minggu keempat belas membuat halaman *route management*. Memastikan tampilan data sudah sesuai dengan *database* dan operasi yang dilakukan pada data melalui tampilan halaman sudah berhasil disimpan ke *database*. Tampilan halaman juga dipastikan memudahkan pengguna dalam membaca data.

Pada minggu kelima belas dan minggu keenam belas memperbaiki halaman yang tidak dapat diakses karena kesalahan dalam penggabungan file Angular. Selain itu juga terdapat beberapa kesalahan pengambilan data pada halaman yang telah digabungkan sehingga diperbaiki.

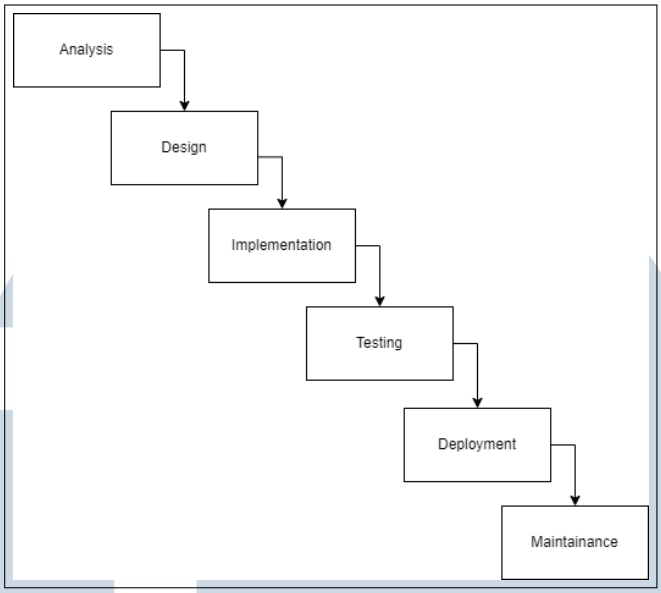
Pada minggu ketujuh belas menambahkan filter pada halaman *order management*. Hal ini untuk memudahkan pengguna untuk mencari riwayat pembelian tertentu yang dilakukan oleh *sales* maupun *customer*.

Pada minggu kedelapan belas, melaporkan pekerjaan *website sales tracking* yang telah selesai. Memperlihatkan alur kerja *website* kepada *supervisor* untuk meminta saran dan masukan.

Pada minggu kesembilan belas, terdapat beberap halaman yang perlu perbaikan sesuai yang diarahkan oleh *supervisor*. Tampilan yang kurang sesuai diperlukan beberapa perubahan atau tambahan agar tampilan halaman lebih baik dan mempermudah pengguna dalam menggunakan *website*.

3.4 Perancangan Sistem

Dalam perancangan sistem ini, digunakan model pendekatan SDLC (*System Development Life Cycle*) dengan model *waterfall*. Hal ini dikarenakan *website* ini lebih cocok dibangun dengan proses yang terstruktur dan sekuensial. Selain itu, setiap langkah yang telah selesai harus dikaji ulang untuk memastikan bahwa tahapan tersebut telah dikerjakan dengan benar dan sesuai dengan harapan. Model *waterfall* SDLC merupakan sebuah metodologi untuk merancang dan membangun sistem perangkat lunak, yaitu proses perancangan bertahap mengalir semakin ke bawah (seperti air terjun). SDLC model waterfall yang digunakan dapat dilihat pada Gambar 3.2.



Gambar 3.2. SDLC model waterfall

Perancangan *website sales tracking* ini hanya mencapai tahap *testing* atau pengujian, belum memasuki tahap *deployment* dan *maintainance*.

3.4.1 *Analysis* (Analisis)

Pada tahap ini dilakukan identifikasi masalah, usulan pemecahan masalah dan analisis kebutuhan sistem yang difokuskan untuk pembuatan perangkat lunak. Analisis telah disediakan oleh perusahaan sebagai berikut:

1. Diperlukannya suatu sistem untuk mengatur jadwal *sales* untuk mengunjungi *customer* tertentu setiap bulannya.
2. Diperlukannya suatu sistem untuk mengetahui transaksi produk yang dilakukan *sales* ke *customer*.
3. Perusahaan dapat melihat dan menilai performa dari *sales* dan *customer* serta produk yang paling diminati.

Berdasarkan analisis di atas, *website sales tracking* dibuat untuk memenuhi kebutuhan perusahaan. Fitur-fitur pada *website* akan disesuaikan sehingga memudahkan perusahaan dalam mengatur kerja *sales* ataupun mengatur perputaran barang di perusahaan.

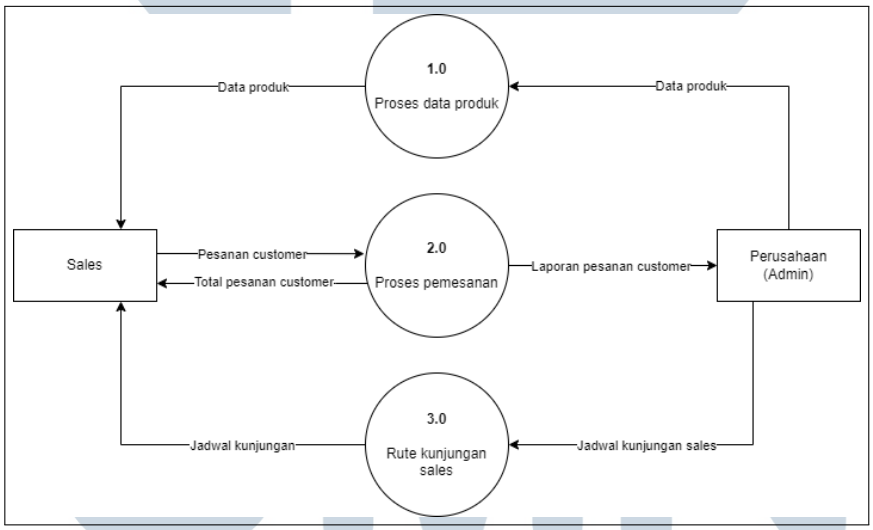
3.4.2 *Design* (Perancangan)

Melakukan perancangan agar dapat menyediakan rancangan yang diharapkan. Pada tahap ini dibuat *Data Flow Diagram* (DFD), *sitemap*, *flowchart*, dan *database schema*.

A. *Data Flow Diagram* (DFD) *Website Sales Tracking*

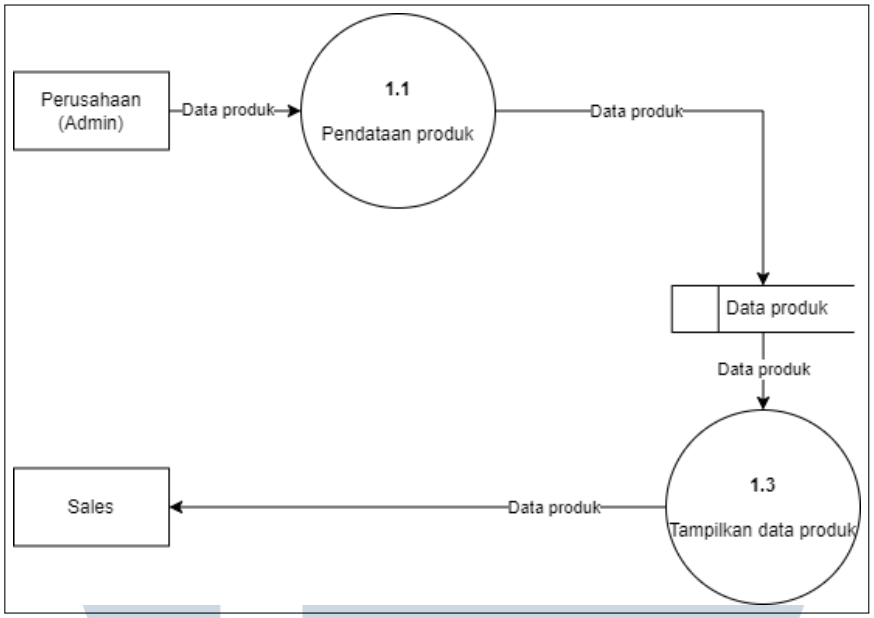
DFD atau *data flow diagram* merupakan diagram yang berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut-atribut. Salah satu keuntungan DFD adalah memudahkan pengguna yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan.

1. DFD Level 0 menggambarkan seluruh alur data pada *website sales tracking*.



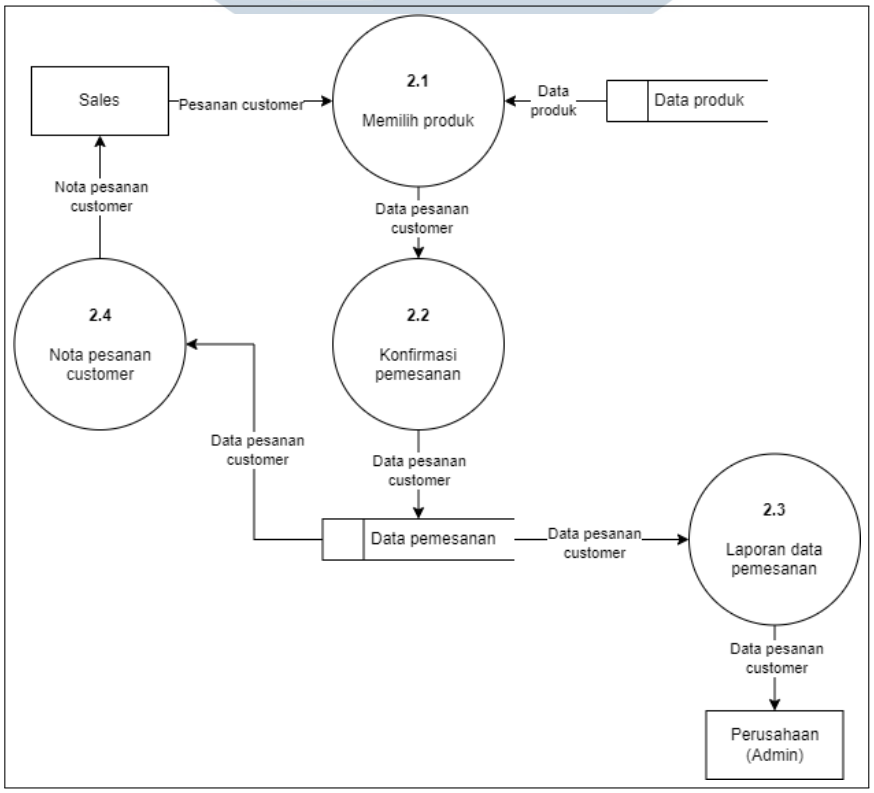
Gambar 3.3. DFD level 0

2. DFD Level 1 (proses data produk) merupakan pengembangan dari proses 1.0.



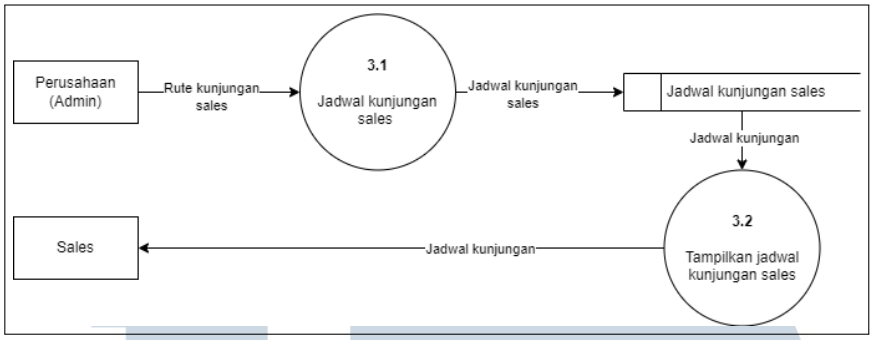
Gambar 3.4. DFD level 1 (proses data produk)

3. DFD Level 1 (proses pemesanan) merupakan pengembangan dari proses 2.0.



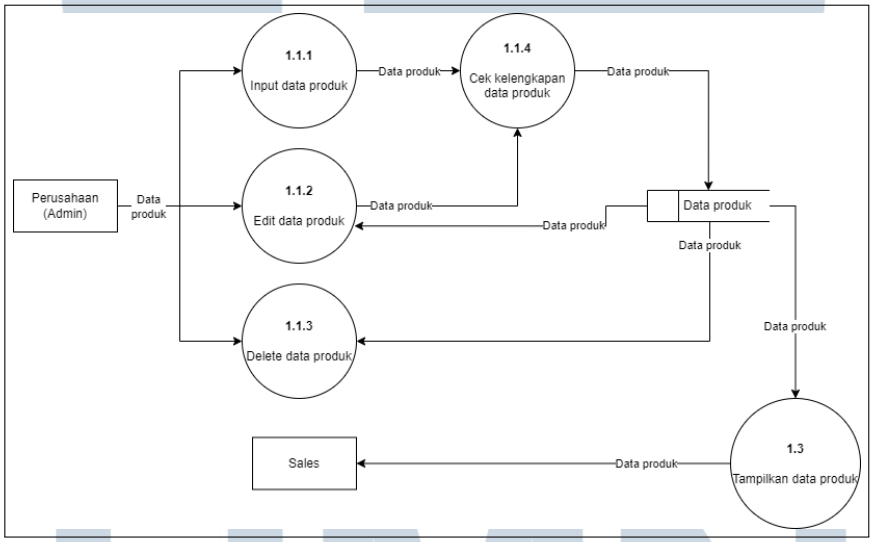
Gambar 3.5. DFD level 1 (proses pemesanan)

4. DFD Level 1 (proses pemesanan) merupakan pengembangan dari proses 3.0.



Gambar 3.6. DFD level 1 (rute kunjungan sales)

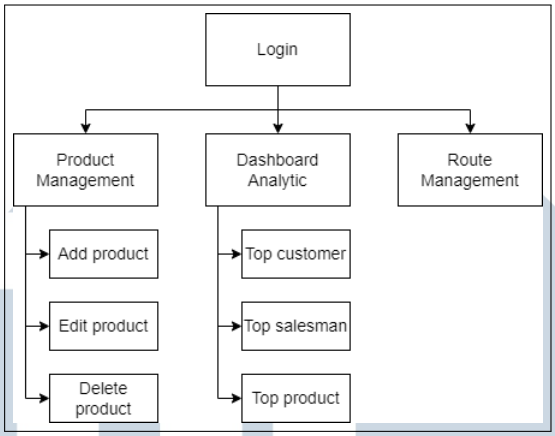
5. DFD Level 2 (pendataan produk) merupakan pengembangan dari proses 1.1.



Gambar 3.7. DFD level 2 (pendataan produk)

B. *Sitemap Website Sales Tracking*

Berdasarkan pembagian tugas yang telah diberikan secara bertahap, terdapat beberapa fitur yang dibangun pada *website sales tracking* ini. Aset dan *database* yang digunakan diambil dari project sebelumnya sehingga pembuatan *website* tidak memerlukan waktu lama. Beberapa fitur yang harus dibuat adalah *product management*, *dashboard analytic*, dan *route management*.



Gambar 3.8. *Sitemap website sales tracking* yang dibuat

Gambar di atas adalah *sitemap website sales tracking* yang dibuat. Halaman *product management* dapat digunakan pengguna untuk melihat data produk, menambah data produk, mengubah data produk, ataupun menghapus data produk. Halaman *dashboard analytic* membantu pengguna untuk melihat performa *customer* dan *sales*, selain itu juga akan menampilkan data produk yang banyak dibeli dalam jangka waktu tertentu. Dan halaman *route management* dapat digunakan untuk mengatur rute *sales* untuk mengunjungi *customer* tertentu setiap bulannya.

C. *Flowchart Website Sales Tracking*

1. *Flowchart* halaman *product management*

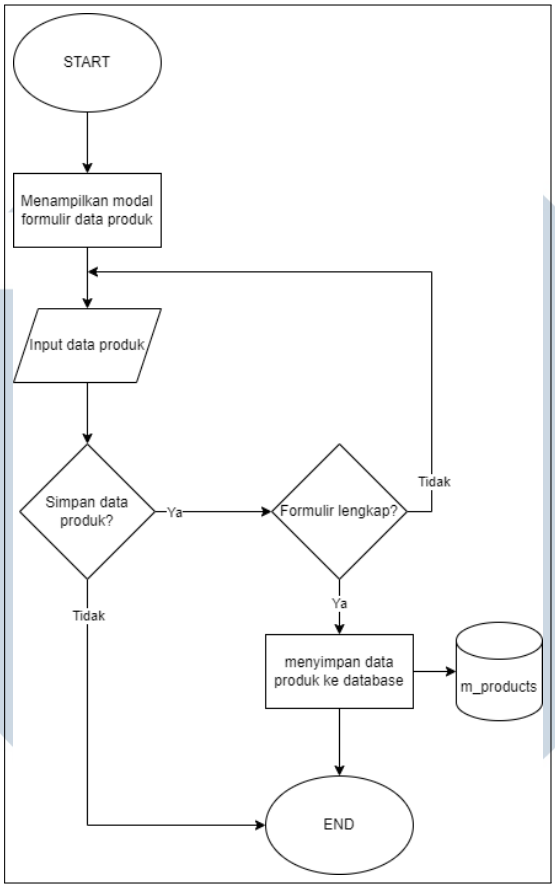
   Halaman *product management* akan menampilkan daftar seluruh data produk yang dimiliki, dan pilihan untuk menambah, menghapus, ataupun mengubah data produk. *Flowchart* halaman ini dapat dilihat pada Gambar 3.9.

Gambar 3.9. *Flowchart* halaman *product management*

Daftar (*list*) produk yang ditampilkan akan disesuaikan dengan *tree* level yang dipilih oleh pengguna. Tampilan awal akan menampilkan daftar produk level 1. Apabila pengguna mengubah *tree* level ke level 2 atau 3, maka daftar produk yang muncul akan disesuaikan dengan nama *product parent* yang dipilih. Terdapat juga beberapa pilihan yang dapat dipilih oleh pengguna, yaitu menambah produk baru, menghapus data produk yang ada, atau mengubah data produk tertentu.

2. *Flowchart* halaman penambahan produk

Halaman penambahan produk akan menampilkan formulir yang harus diisi oleh pengguna dalam bentuk modal. Formulir ini memiliki beberapa kriteria data yang harus dipenuhi untuk disimpan ke *database*. Bagian yang harus diisi adalah *tree* level, nama *parent* produk, dan nama produk. Flowchart untuk halaman ini terdapat pada Gambar 3.10.

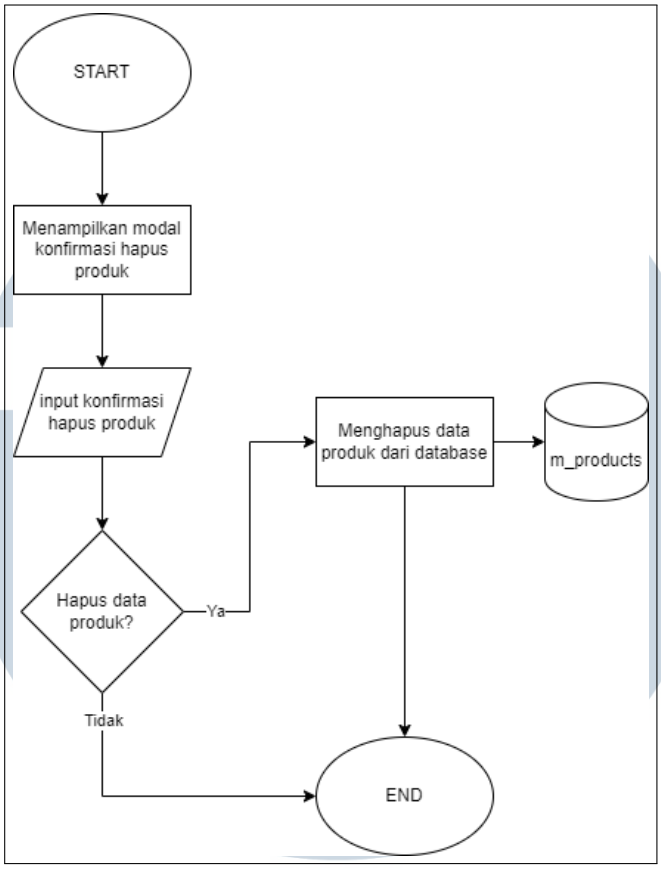


Gambar 3.10. *Flowchart* halaman penambahan produk

Ketika pengguna ingin menyimpan data, seluruh kriteria data yang disebutkan di atas harus dilengkapi, jika terdapat kriteria data yang kosong, maka data gagal disimpan dan harus mengisi kembali formulir. Untuk menambahkan data pada *tree* level 1, maka nama *parent* produk akan dikosongkan. Jika data sudah berhasil disimpan ke dalam *database*, atau pengguna membatalkan penambahan produk, maka akan kembali ke halaman *product management*.

3. *Flowchart* halaman konfirmasi penghapusan produk

   Halaman ini akan menampilkan tombol konfirmasi untuk menghapus suatu data produk dalam bentuk modal. Tujuan halaman ini adalah untuk mengantisipasi pengguna dalam menghapus produk secara tidak sengaja. *Flowchart* halaman konfirmasi penghapusan produk dapat dilihat pada Gambar 3.12.

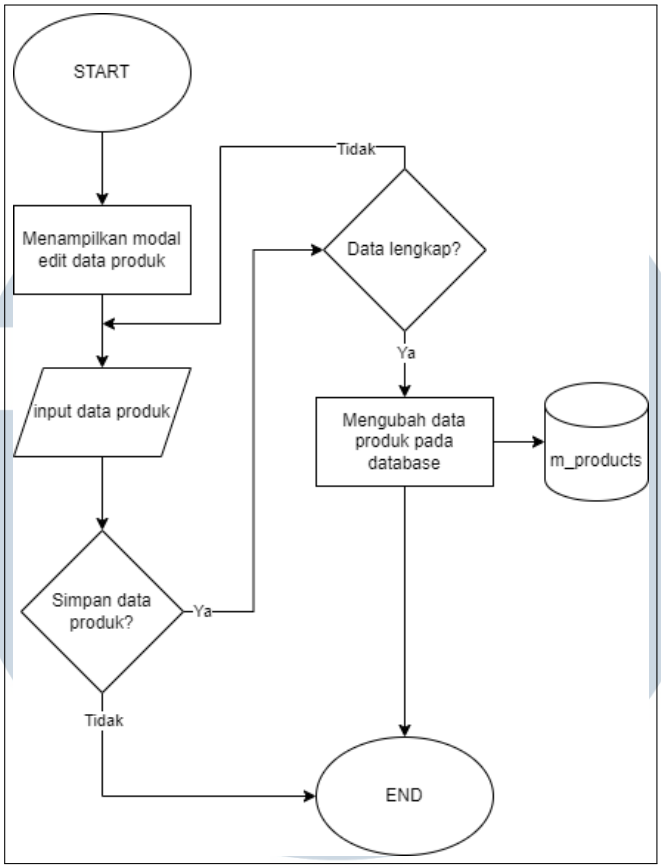


Gambar 3.11. *Flowchart* halaman konfirmasi penghapusan produk

Halaman ini berfokus pada tombol untuk menghapus produk atau membatalkan penghapusan produk. Apabila pengguna menyetuhui untuk menghapus produk, maka produk tersebut akan ditandai sebagai *non-active* pada *database* dan tidak akan lagi ditampilkan pada daftar produk di halaman *product management*. Ketika produk berhasil dihapus, atau pengguna membatalkan penghapusan produk, maka akan kembali ke halaman *product management*.

4. *Flowchart* halaman mengubah produk

Halaman mengubah produk akan menampilkan formulir yang telah terisi data produk yang tersimpan di *database* dalam bentuk modal. Pada halaman ini pengguna dapat mengubah data produk tersebut. Pengubahan data juga harus memenuhi kriteria data yang sama seperti penambahan produk, yakni kolom *tree* level, nama *parent* produk, dan nama produk tidak boleh kosong. *Flowchart* untuk halaman ini dapat dilihat pada Gambar 3.12.

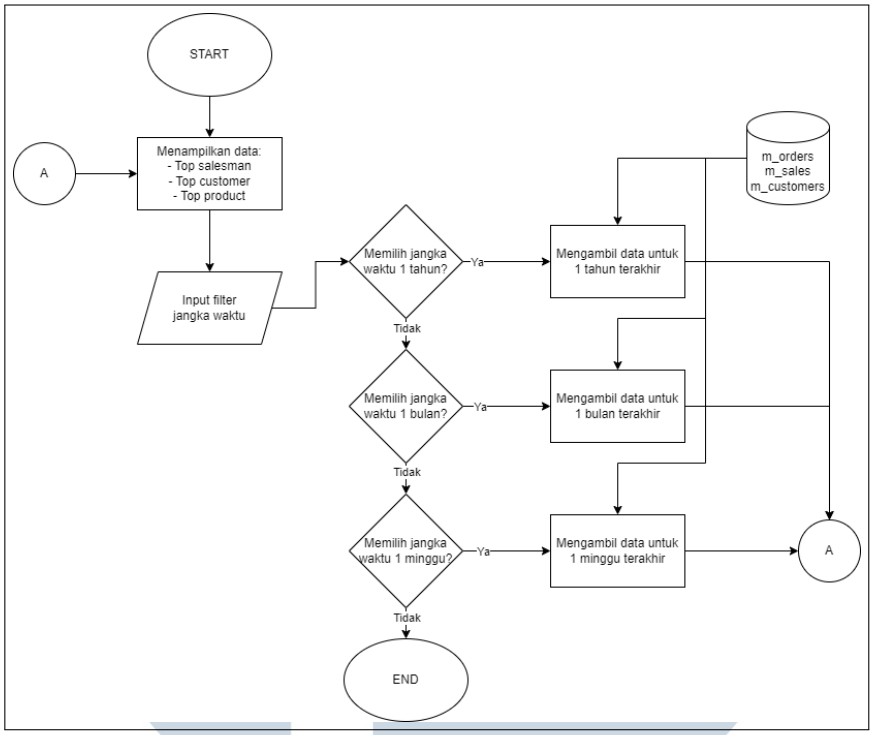


Gambar 3.12. *Flowchart* halaman mengubah produk

Apabila pengguna akan menyimpan perubahan data produk, seluruh kriteria data harus terisi. Apabila terdapat kriteria data yang kosong, kecuali pada *tree* level 1, maka data produk pada *database* akan diubah. Jika perubahan selesai dilakukan, atau pengguna membatalkan perubahan, maka akan kembali ke halaman *product management*.

5. *Flowchart* halaman *dashboard analytic*

Halaman *dashboard analytic* dibuat untuk membantu pengguna dalam melacak, menganalisis, dan membuat laporan mengenai KPI (*Key Performance Indicator)*. Halaman ini akan menampilkan daftar *sales*, *customer*, dan produk yang berada pada *database* transaksi. *Flowchart* untuk halaman ini dapat dilihat pada Gambar 3.13.

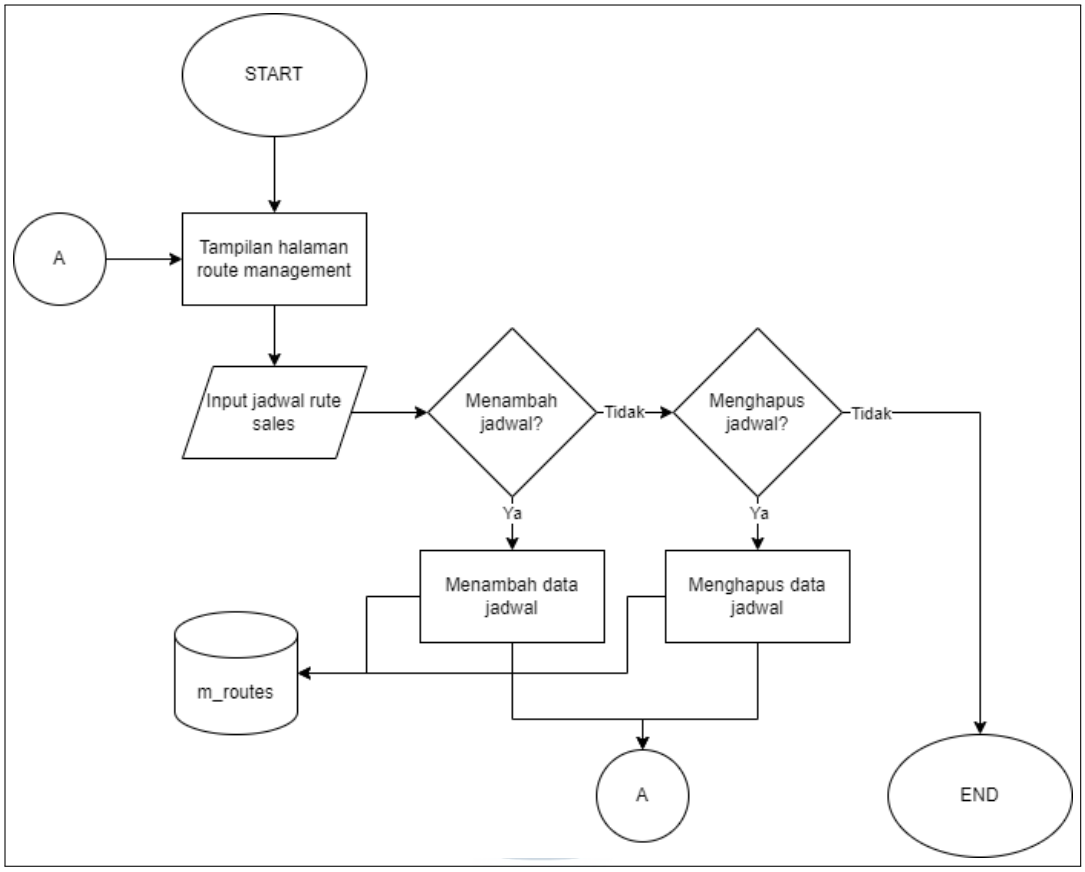


Gambar 3.13. *Flowchart* halaman *dashboard analytic*

Terdapat fitur untuk memilih jangka waktu pengambilan data dari *database*. Pilihan jangka waktu mencakup satu minggu terakhir, satu bulan terakhir, atau satu tahun terakhir. Data yang akan ditampilkan berupa grafik sehingga memudahkan pengguna dalam memahami informasi dari *database*. Pada pertama kali data ditampilkan, halaman akan menampilkan data untuk satu bulan terakhir, ketika pengguna memilih jangka waktu lain, maka data yang diinginkan pengguna akan diambil lagi dari *database* dan halaman akan menampilkan data yang diminta.

6. *Flowchart* halaman *route management*

Halaman *route management* berfungsi untuk mengatur jadwal *sales* melakukan kunjungan ke *customer* tertentu selama satu bulan. Tampilan halaman akan berupa tabel yang berisi *checkbox* untuk minggu ke-1 hingga minggu ke-4. *Flowchart* halaman *route management* dapat dilihat pada Gambar 3.14.

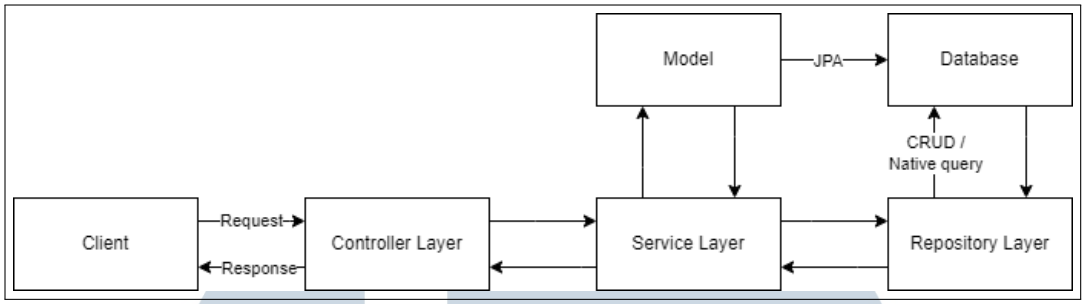


Gambar 3.14. *Flowchart* halaman *route management*

Apabila tombol *checkbox* diaktifkan oleh pengguna, maka data *sales*, *customer* dan waktu kunjungan akan disimpan ke dalam *database*. Sebaliknya, apabila tombol *checkbox* dihilangkan, maka data *sales*, *customer*, dan waktu kunjungan akan dihapus. Berdasarkan data ini, jadwal *sales* dapat diubah ke dalam bentuk tanggal setiap bulannya.

D. *Database Schema Website Sales Tracking*

Database yang digunakan dalam website ini dibangun menggunakan *framework* Spring Boot. Alur kerja Spring Boot secara sederhana dapat dilihat pada Gambar 3.15.



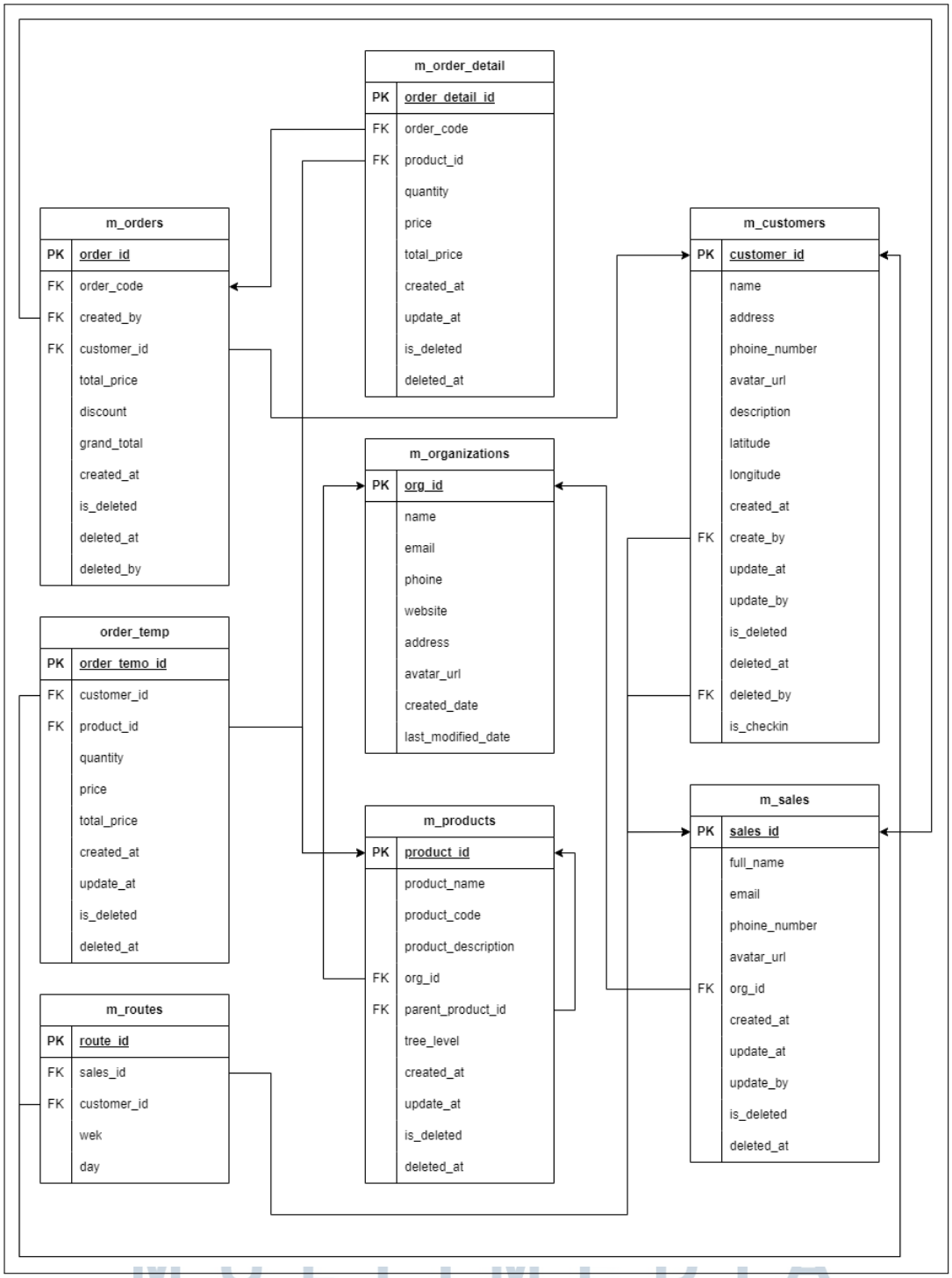
Gambar 3.15. Alur kerja Spring Boot

Alur kerja tersebut dibagi menjadi empat *layer* atau lapisan utama, yaitu *controller layer*, *service layer*, *repository layer*, dan *model*[3]. *Controller layer* merupakan bagian pertama dari sistem Spring Boot yang akan menerima permintaan *client*. Bagian ini akan menentukan metode yang diminta oleh *client*, *get*, *post*, atau *put*. Metode *get* mendefinisikan bahwa *client* meminta data dari *database*, metode *post* mendifinikan *client* ingin menambahkan data ke *database*, dan metode *put* mendefinisikan bahwa *client* ingin mengubah data yang ada di *database*¸

Lapisan selanjutnya adalah *service layer*. Lapisan ini yang akan memberikan pilihan kepada *controller layer* mengenai layanan (*services*) yang dibutuhkan. Lapisan ini juga akan menangani segala jenis komputasi pada data, oleh karena itu lapisan ini memerlukan pengambilan data yang dilakukan oleh *repository layer*. *Repository layer* juga seringkali disebut DAO *Layer*, DAO adalah singkatan untuk *Data Access Object*. Tujuan utama lapisan ini adalah mengakses data secara efisien dari *database* melalui *query* dan memberikan layanan pada *service layer*.

Lapisan terakhir adalah model, lapisan ini mewakili objek dunia nyata. JPA (Java Persistence API) memberikan referensi atau detail mengenai ORM (Object Relation Mapping), artinya pengguna dapat menghubungkan Java *class* dengan tabel *database*.

Terdapat delapan model yang dibangun pada *database website sales tracking*, yaitu m_organizations, m_sales, m_customers, m_products, m_orders, m_order_detail, oder_temp, dan m_routes. *Database schema* yang digunakan secara lengkap dapat dilihat pada Gambar 3.16.



Gambar 3.16. *Database schema website sales tracking*

1. Tabel m_organizations

   Tabel ini akan menyimpan informasi mengenai suatu perusahaan atau organisasi dan merupakan tabel utama yang membantu membedakan suatu organisasi dengan organisasi lainnya. Dengan tabel ini, sistem dapat

memisahkan produk, *sales*, dan *customer* masing-masing organisasi.

2. Tabel m_sales

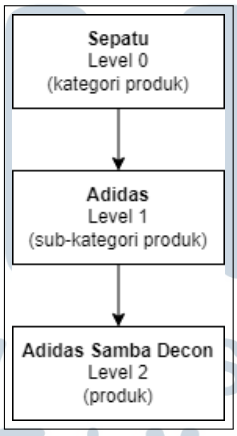
   Tabel m_sales akan menyimpan data *sales* yang dimiliki perusahaan. Terdapat kolom org_id yang terhubung dengan tabel m_organizations sehingga *sales* yang dimiliki perusahaan hanya akan terlihat oleh pengguna dari perusahaan atau organisasi tersebut. Terdapat juga kolom is_deleted yang berfungsi untuk menunjukkan apakah data *sales* tersebut masih ada atau sudah dihapus.

3. Tabel m_customers

   Tabel ini menyimpan data *customer* pada suatu perusahaan. Kolom created_by dan deleted_by terhubung dengan tabel m_sales. Hal ini membantu pengguna untuk mengetahui jumlah *customer* dan penanggung jawab masing-masing *customer*.

4. Tabel m_products

   Tabel ini menyimpan data produk yang dimiliki dan dijual oleh masing-masing perusahaan. Tabel dibangun dengan model hierarki sehingga data yang disimpan akan disusun ke dalam struktur seperti pohon. Dalam hal ini, model hierarki yang akan digunakan dibatasi hingga level 3. Contoh model hierarki yang digunakan dapat dilihat pada Gambar 3.17.



Gambar 3.17. Contoh model hierarki

Kolom parent_product_id akan menyimpan product_id yang menjadi *parent* dari suatu data produk. Apabila produk berada pada level 1, maka kolom tersebut akan kosong (*null*). Kolom org_id terhubung dengan tabel m_organizations yang akan membedakan produk milik suatu perusahaan dengan perusahaan lain.

5. Tabel m_orders

   Tabel ini menyimpan data transaksi yang dilakukan oleh *customer* ditandai dengan kolom customer_id yang terhubung dengan tabel m_customers. Kolom created_by terhubung dengan tabel m_sales yang menunjukkan penanggung jawab atas transaksi yang dilakukan oleh *customer*.

6. Tabel m_order_detail

   Tabel ini terhubung dengan tabel m_orders pada kolom order_code. Data yang disimpan pada tabel ini merupakan detail dari barang-barang yang dibeli oleh *customer* seperti jenis barang, harga, dan jumlah. Maka dari itu tabel ini terhubung dengan tabel m_products pada kolom product_id.

7. Tabel order_temp

   Tabel ini menyimpan data pembelian barang yang belum selesai dilakukan oleh *customer*. Data yang disimpan pada tabel ini serupa dengan tabel m_orders dan m_order_detail, namun tabel ini menyimpan data pembelian yang masih belum dilakukan dan tidak masuk ke dalam *history order*.

8. Tabel m_routes

   Pada tabel ini disimpan jadwal rute perjalanan *sales* untuk setiap bulannya. Dengan adanya tabel ini, *sales* dapat mengetahui *customer* yang harus dikunjungi pada tanggal tertentu. Data yang disimpan berupa *week* (minggu) dan day (hari), namun tampilan yang dilihat *sales* akan disesuaikan tanggalnya untuk setiap bulan.

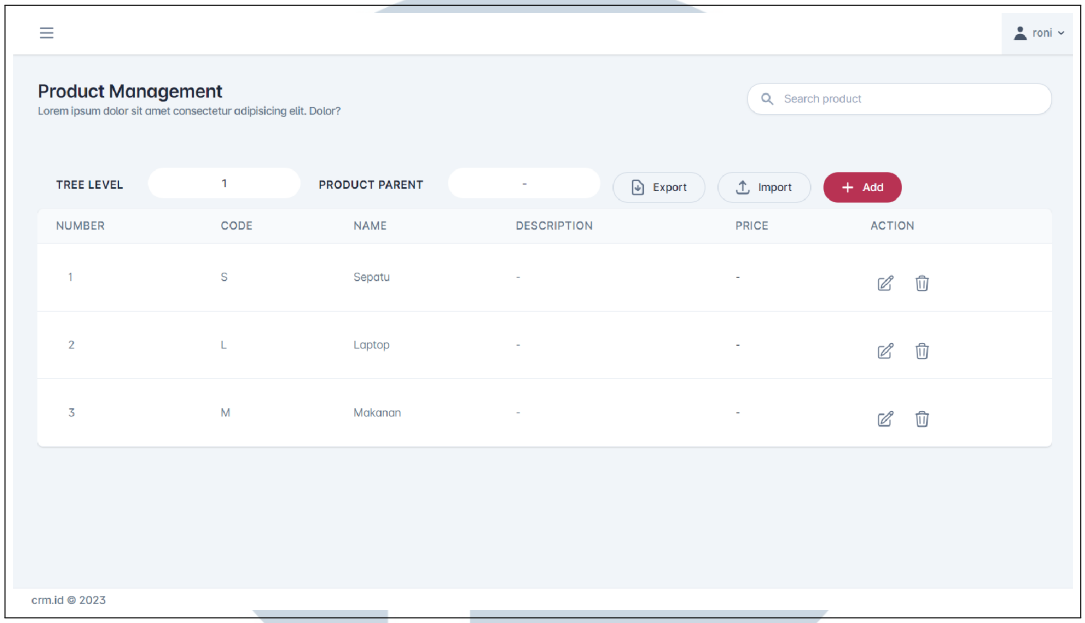
3.4.3 *Implementation* (Implementasi)

Melakukan penerapan hasil rancangan ke dalam bentuk yang dapat dibaca dan dimengerti oleh komputer.

1. Halaman *product management*

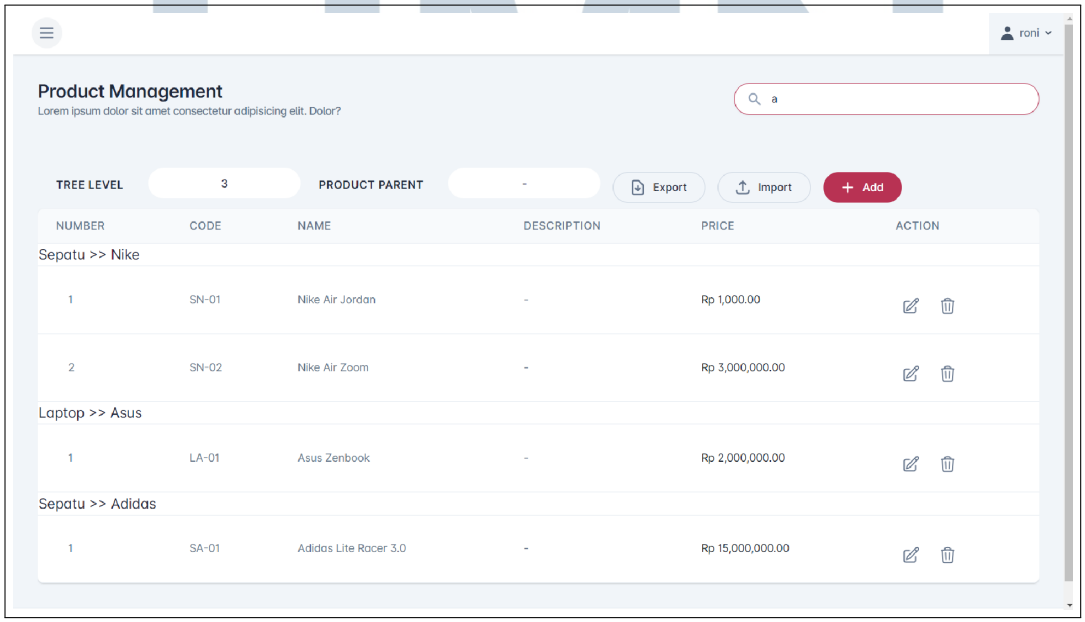
   Tampilan awal halaman ini akan menampilkan daftar produk pada *tree* level 1. Apabila pengguna menekan salah satu produk pada daftar ini, maka akan menampilkan daftar produk pada *tree* level 2 dengan *parent* produk yang dipilih. Hal ini berlaku hingga *tree* level 3. Selain itu terdapat pilihan *tree* level dan nama *parent* produk yang dapat dipilih langsung oleh pengguna

untuk mencari produk tertentu. Implementasi halaman *product management* dapat dilihat pada Gambar 3.18.



Gambar 3.18. Hasil implementasi halaman *product management*

Selain itu juga terdapat filter untuk mencari nama produk tertentu. Pencarian ini akan menampilkan daftar produk pada *tree* level 3 dan akan dikelompokkan berdasarkan nama *parent* produknya. Hasil filter menggunakan nama produk dapat dilihat pada Gambar 3.19.

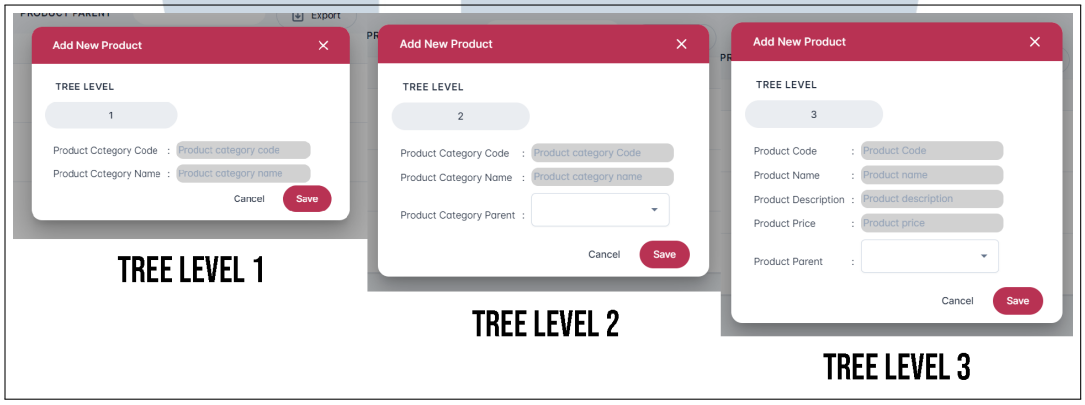


Gambar 3.19. Hasil implementasi halaman *product management* menggunakan filter pencarian

Terdapat tombol *add* pada bagian atas table untuk menambah produk baru kedalam *database*. Pada masing-masing produk juga terdapat tombol *edit* dan *delete* yang berguna untuk mengubah data produk atau menghapus produk tersebut.

2. Halaman *product management* untuk penambahan produk

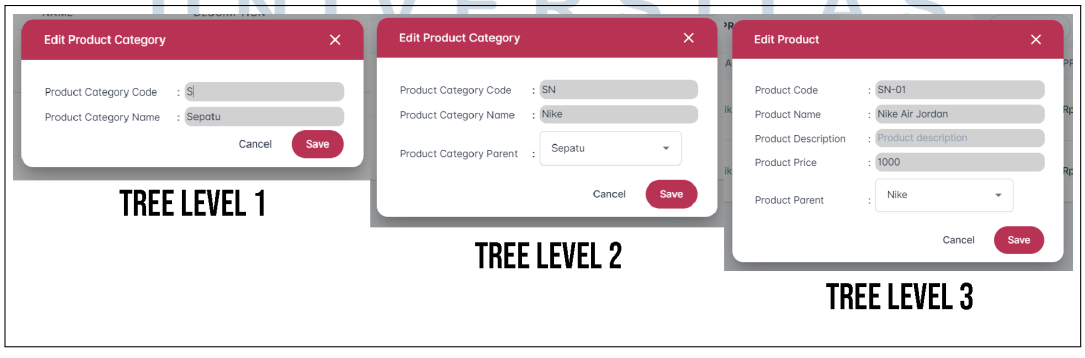
   Ketika menekan tombol *add*, modal untuk penambahan produk akan muncul. Modal ini berisi formulir yang harus diisi oleh pengguna untuk menambahkan produk baru. Data yang harus diisi pada formulir akan berbeda untuk produk level 1, 2, ataupun 3. Hasil implementasi halaman penambahan produk dapat dilihat pada Gambar 3.20.



Gambar 3.20. Hasil implementasi halaman penambahan produk

3. Halaman *product management* untuk pengubahan data produk

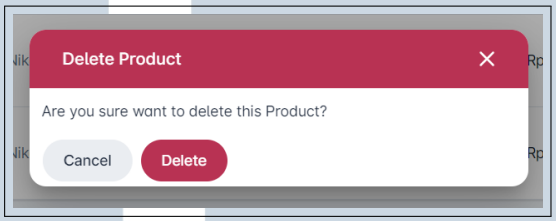
   Ketika tombol *edit* ditekan oleh pengguna pada salah satu produk, maka akan menampilkan modal untuk mengubah data produk tersebut pada *database*. Data yang dapat diubah juga disesuaikan dengan *tree* level produk tersebut, hal ini serupa dengan formulir pada halaman penambahan produk. Hasil implementasi halaman ini dapat dilihat pada Gambar 3.21.



Gambar 3.21. Hasil implementasi halaman pengubahan data produk

4. Halaman *product management* untuk menghapus produk

   Tombol *delete* pada daftar produk akan menampilkan halaman konfirmasi untuk menghapus data produk. Halaman ini akan memberikan pilihan pada pengguna untuk menghapus data produk tersebut atau membatalkan penghapusan data. Hasil implementasi halaman penghapusan produk dapat dilihat pada Gambar 3.22.



Gambar 3.22. Hasil implementasi halaman penghapusan data produk

   Apabila pengguna menekan tombol *delete*, maka produk tersebut akan dinonaktifkan pada *database*. Tombol *cancel* akan membatalkan penghapusan data produk.

5. Halaman *dashboard analytic*

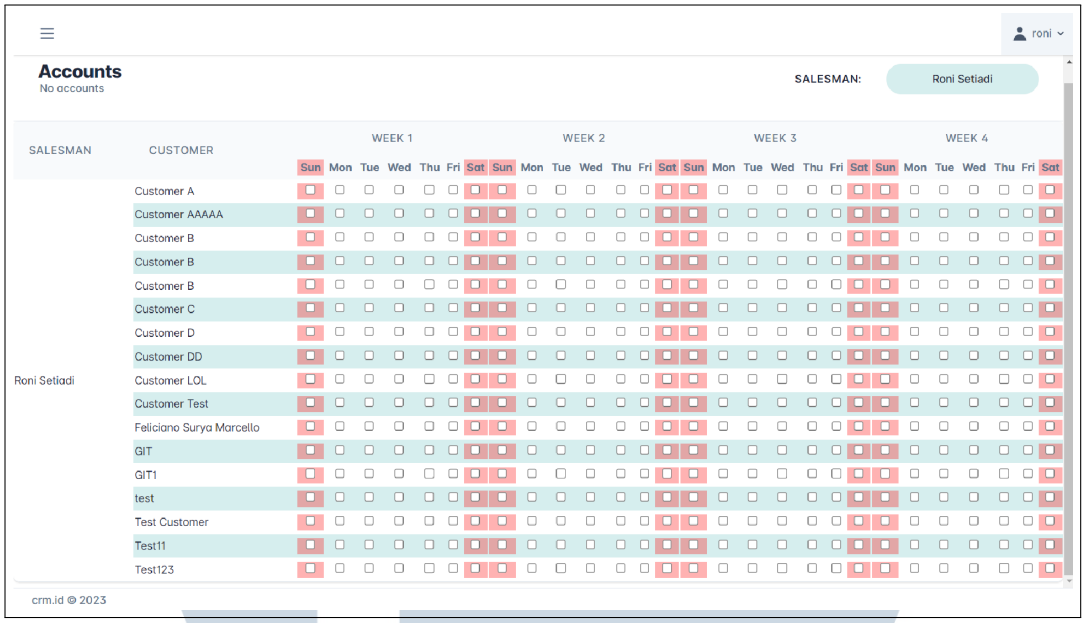
   Pada halaman ini akan ditampilkan beberapa aspek untuk melihat performa *sales* dan *customer* dalam bentuk grafik. Selain itu juga terdapat data produk yang dibeli. Pada pertama kali ditampilkan, halaman ini akan menampilkan data riwayat transaksi yang terjadi selama 1 bulan terakhir. Implementasi halaman *dashboard analytic* untuk performa *sales* dan *customer* dapat dilihat pada Gambar 3.23, sedangkan bagian grafik produk dapat dilihat pada Gambar 3.24.

Gambar 3.23. Hasil implementasi halaman *dashboard analytic* bagian performa *sales* dan *customer*

Gambar 3.24. Hasil implementasi halaman *dashboard analytic* bagian produk

6. Halaman *route managament*

   Halaman ini akan menampilkan daftar *customer* berdasarkan *sales* yang dipilih. Data *customer* ditampilkan dalam bentuk tabel dan disertai dengan *checkbox* untuk minggu pertama hingga minggu keempat. Tampilan halaman ini dapat dilihat pada Gambar 3.25.



Gambar 3.25. Hasil implementasi halaman *route management*

Apabila salah satu *checkbox* diaktifkan untuk salah satu *customer*, maka data tersebut akan disimpan ke dalam *database* sebagai jadwal *sales*. Sebaliknya, apabila *chechbox* yang aktif di-nonaktifkan, maka jadwal *sales* tersebut dihapus dari *database*.

3.4.4 *Testing* (Pengujian)

Program harus diuji coba yang difokuskan pada aktivitas pemastian bahwa semua perintah yang ada telah dicoba dan fungsi eksternal untuk memastikan bahwa dengan masukan tertentu suatu fungsi akan menghasilkan keluaran sesuai dengan yang dikehendaki. Pada sistem ini pengujian dilakukan secara mandiri untuk memastikan output yang diharapkan sudah sesuai. Hasil pengujian dapat dilihat pada Tabel 3.2.

Tabel 3.2. Pengujian pada *website sales tracking*

No	Halaman	Hasil yang diharapkan	Hasil pengujian
1	*Product management*	Menampilkan daftar produk	OK
Lanjut pada halaman berikutnya			

Tabel 3.2 Pengujian pada *website sales tracking* (lanjutan)

No	Halaman	Hasil yang diharapkan	Hasil pengujian
2	*Product management*	Menambahkan data produk baru	OK
3	*Product management*	Mengubah (*edit)* data produk	OK
4	*Product management*	Filter pencarian data produk	OK
5	*Dashboard analytic*	Menampilkan analisis data *sales*, *customer*, dan produk	OK
6	*Dashboard analytic*	Filter jangka waktu data yang ditampilkan	OK
7	*Dashboard analytic*	Filter *sales* dan *customer* tertentu	OK
8	*Route management*	Menampilkan rute perjalanan *sales* untuk satu bulan	OK
9	*Route management*	Menambahkan rute perjalanan *sales*	OK
10	*Route management*	Menghapus rute perjalanan *sales*	OK
11	*Route management*	Filter untuk menampilkan *sales* tertentu	OK

3.5 Kendala dan Solusi yang Ditemukan

Selama melakukan kerja magang di PT. Global Innovation Technology terdapat beberapa kendala yang ditemukan, yaitu:

1. Penggunaan *framework* untuk pembangunan *database* belum dipahami. Spring Boot merupakan salah satu *framework* dari Spring yang memberikan kemudahan untuk memilih *library* java yang akan digunakan. Meskipun terlihat sederhana, penggunaan Spring Boot harus dipelajari dengan baik agar dapat digunakan dengan maksimal.

2. Tantangan dalam memahami tugas. Terkadang tugas yang diberikan mungkin

sulit dipahami atau memerlukan keahlian khusus yang belum dimiliki. Contohnya dalam pembuatan grafik, dan membuat filter berdasarkan grafik lainnya.

3. Kesalahan komunikasi. Kesalahpahaman atau kurangnya komunikasi yang efektif dengan *supervisor* atau anggota tim dapat menyulitkan dalam menyelesaikan tugas.

Upaya yang dilakukan dalam mengatasi kendala-kendala yang ditemukan selama kerja magang adalah:

1. Mempelajari *framework* Spring Boot dengan membuat contoh *database*. Penggunaan metode *get*, *put*, dan *post* dipelajari dengan contoh *database* melalui aplikasi Postman untuk mengetahui data yang dikirim dan diterima sudah sesuai.

2. Jika terdapat kesulitan dalam memahami tugas, maka harus dipelajari dan meminta bantuan untuk meningkatkan kemampuan dalam mengerjakan tugas tersebut.

3. Sebelum mengerjakan tugas, memastikan instruksi sudah jelas dan tidak salah pemahaman. Meminta saran dan masukan untuk meningkatkan komunikasi.