BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pada masa magang batch kedua di PT Cranium Royal Aditama, penulis ditempatkan sebagai *Full Stack Developer* di bawah divisi *IT and Software Solutions*, tepatnya dalam tim *Product Developer* yang menangani proyek pengembangan sistem *Enterprise Resource Planning* (ERP). Dalam periode ini, penulis bekerja bersama beberapa peserta magang lain dengan bimbingan seorang *supervisor* yang berperan dalam mengarahkan dan mengawasi jalannya pengembangan. Fokus utama yang dikerjakan selama magang adalah mengembangkan dan menyempurnakan modul *Selling* pada sistem *ERP* internal yang digunakan perusahaan.

Penulis berada di bawah bimbingan langsung Bapak Sugito selaku *Tech Lead*, serta Bapak Angga sebagai *Project Manager* yang mengoordinasikan keseluruhan proyek ERP. Keduanya berperan penting dalam memberikan arahan teknis, meninjau hasil pekerjaan, serta memastikan seluruh proses pengembangan berjalan sesuai dengan standar perusahaan. Selain itu, penulis juga mendapatkan pendampingan dari dua *mentor*, yaitu Kak Petra yang berfokus pada pengembangan sisi *backend* dan Kak Sabil yang bertanggung jawab pada sisi *frontend*. Dari bimbingan para mentor dan supervisor, penulis belajar bagaimana berkolaborasi lintas tim serta menerapkan teknologi yang digunakan dalam proyek secara langsung.

Secara umum, struktur dan tanggung jawab dalam tim magang dapat dijelaskan sebagai berikut:

1. Supervisor: Mengarahkan, mengevaluasi, serta menjamin standar kualitas pekerjaan pada setiap tahapan proyek.
2. Mentor: Memberikan panduan teknis dan memastikan mahasiswa memahami alur kerja pengembangan, termasuk penggunaan *tools* seperti GitHub dan Figma.
3. Mahasiswa Magang: Bertugas mengimplementasikan fitur, memperbaiki bug, serta mendokumentasikan hasil pekerjaan sesuai arahan dan tenggat waktu yang telah ditetapkan.

Dalam menjalankan tugasnya, tim pengembang di Cranium mengikuti alur kerja (*workflow*) yang terstruktur untuk memastikan koordinasi dan kualitas hasil tetap terjaga. Proses tersebut terdiri dari beberapa tahapan, yaitu:

1. Pembagian Tugas
   (a) Penugasan diberikan setelah pekerjaan sebelumnya selesai dan telah digabungkan ke cabang utama (*main branch*).
   (b) Supervisor menentukan prioritas tugas yang disampaikan melalui rapat atau komunikasi langsung.
   (c) Rincian kebutuhan fitur biasanya dijelaskan menggunakan diagram, rancangan tabel, maupun desain antarmuka dari Figma.
   (d) Setiap tugas dikerjakan di cabang pengembangan terpisah untuk menghindari konflik kode.
2. Pelaksanaan Tugas
   (a) Pengembang mulai mengerjakan tugas setelah menerima penjelasan teknis dan tenggat waktu yang telah ditentukan.
   (b) Apabila menemui kendala, pengembang dapat berkoordinasi dengan mentor atau supervisor untuk mendapatkan solusi.
3. Asistensi dan Revisi
   (a) Setelah pekerjaan selesai, mahasiswa melakukan asistensi guna mendapatkan masukan dan evaluasi.
   (b) Revisi dilakukan hingga hasil sesuai dengan standar yang ditentukan.
   (c) Komunikasi tetap dibuka baik secara langsung maupun melalui media daring.
4. Pengujian Fitur
   (a) Setiap perubahan diuji melalui pengujian otomatis (*unit test, contract test*) dan pengujian manual.
   (b) Pengujian otomatis dilakukan menggunakan kerangka *testing* bawaan dari Spring Boot dan Next.js.
   (c) Pengujian manual dilakukan bersama mentor dan supervisor sebelum hasil kerja diunggah ke *repository*.

5. Penggabungan Kode (*Pull Request*)

   (a) Setelah hasil dinyatakan valid, kode digabungkan ke *main branch* melalui *pull request*.

   (b) Pengujian tambahan dilakukan untuk memastikan tidak ada kesalahan setelah penggabungan.

6. Finalisasi

   (a) Tugas dianggap selesai setelah penggabungan berhasil tanpa konflik.

   (b) Cabang pengembangan (*branch*) kemudian dihapus dan pengembang membuat cabang baru untuk fitur berikutnya.

Dalam hal komunikasi dan koordinasi, perusahaan menerapkan sistem kerja yang efisien dan terorganisasi. Komunikasi harian dilakukan melalui *WhatsApp* untuk kebutuhan informasi cepat, sedangkan *Discord* digunakan sebagai media diskusi teknis terutama saat bekerja dari rumah. Ketika bekerja di kantor, koordinasi dilakukan melalui pertemuan langsung dan diskusi tim secara informal.

Selain itu, progres setiap anggota tim dipantau melalui tabel *roadmap* di Figma. Setiap peserta magang wajib memperbarui status pekerjaannya secara mandiri. Google Sheets juga digunakan untuk mencatat laporan harian serta perkembangan tugas masing-masing anggota. Kolaborasi teknis dalam pengembangan kode dilakukan melalui GitHub sebagai *repository* utama, lengkap dengan sistem *branching* dan *pull request*.

Dengan sistem kerja yang terstruktur dan komunikasi yang terbuka, penulis memperoleh gambaran nyata mengenai bagaimana proses pengembangan perangkat lunak berskala besar dijalankan di lingkungan profesional. Pengalaman ini menjadi dasar penting bagi penulis untuk memahami praktik kolaboratif dalam dunia industri teknologi.

3.2 Tugas yang Dilakukan

Selama pelaksanaan program magang di PT Cranium Royal Aditama, penulis berpartisipasi secara langsung dalam proses pengembangan sistem *Enterprise Resource Planning* (ERP) yang sedang dikembangkan oleh perusahaan. Fokus utama pekerjaan yang dilakukan adalah pengembangan serta penyesuaian fitur sesuai dengan kebutuhan internal perusahaan maupun permintaan dari

pengguna sistem. Setiap aktivitas dikerjakan berdasarkan arahan dari pembimbing dan disesuaikan dengan prioritas proyek yang tengah berjalan.

Pada proyek ERP ini, pengembangan sisi *backend* dilakukan menggunakan Java dengan *Framework Spring Boot*, sedangkan sisi *frontend* memanfaatkan *Next.js* dan TypeScript untuk membangun antarmuka yang interaktif dan dinamis. Sistem ERP ini mengadopsi pendekatan arsitektur *modular monolith*, yang menggabungkan kemudahan pengelolaan sistem monolitik dengan fleksibilitas pemisahan modul seperti pada arsitektur *microservices*. Pendekatan tersebut membuat setiap modul memiliki batas tanggung jawab yang jelas, sehingga proses pengembangan dan pengujian dapat dilakukan secara terpisah tanpa memengaruhi stabilitas sistem secara keseluruhan.

Selama magang, penulis mendapatkan berbagai jenis tugas yang cukup beragam, di antaranya memperbaiki kesalahan kode atau *bug fixing*, menambahkan fitur baru pada sisi *backend* dan menyempurnakan fungsi yang telah ada agar lebih efisien serta mudah digunakan. Setiap hasil pekerjaan kemudian ditinjau oleh pembimbing sebelum digabungkan ke dalam *repository* utama proyek.

Dalam proses pengerjaannya, digunakan berbagai perangkat pengembangan untuk menunjang efisiensi dan kolaborasi antaranggota tim. Adapun teknologi dan alat bantu yang digunakan dalam proyek ERP selama kegiatan magang ini adalah sebagai berikut:

1. IntelliJ IDEA: digunakan sebagai lingkungan pengembangan utama untuk membangun dan menguji kode di sisi *backend* berbasis Java dan Spring Boot.
2. Spring Boot: berfungsi sebagai kerangka kerja utama dalam pengembangan layanan *backend* yang mengatur logika bisnis dan komunikasi antar modul.
3. Visual Studio Code: digunakan dalam pengembangan antarmuka pengguna ERP berbasis *Next.js* sekaligus untuk menulis *unit test* pada sisi *frontend*.
4. Next.js dengan TypeScript: dimanfaatkan untuk membangun antarmuka aplikasi ERP berbasis web yang interaktif dan dinamis.
5. PostgreSQL: digunakan sebagai sistem manajemen basis data utama yang menyimpan seluruh data operasional sistem ERP.
6. DBeaver: digunakan untuk mengelola struktur dan isi basis data PostgreSQL secara visual.

7. Postman: dimanfaatkan untuk menguji API dan memastikan integrasi antara sisi *frontend* dan *backend* berjalan dengan baik.

8. GitHub: digunakan sebagai platform kolaborasi untuk menyimpan kode proyek dan mengelola *pull request*, *branching*, serta *code review*.

Melalui keterlibatan langsung dalam proyek ini, penulis mendapatkan pengalaman berharga dalam pengembangan perangkat lunak berskala besar, khususnya dalam penerapan teknologi *Spring Boot* dan *Next.js*. Selain memperkuat pemahaman teknis, kegiatan magang ini juga memberikan wawasan mengenai proses kolaboratif, manajemen versi, serta penerapan standar pengembangan yang digunakan di lingkungan industri profesional.

3.3 Uraian Pelaksanaan Magang

Bagian ini berisi rangkuman kegiatan yang penulis lakukan selama masa kerja praktik. Uraian disusun untuk memberikan gambaran mengenai alur tugas, peran yang dijalankan, serta perkembangan pekerjaan dari minggu ke minggu.

3.3.1 Pelaksanaan Kerja Magang

Selama masa magang, penulis mengerjakan beberapa tugas yang berkembang sesuai kebutuhan proyek. Pada tahap awal, pekerjaan berfokus pada pengembangan backend ERP Cranium, khususnya implementasi fitur dan revisi kode. Memasuki pertengahan periode, penulis juga terlibat sebagai Quality Assurance pada proyek BCA LMS untuk melakukan pengujian dan verifikasi defect. Rincian aktivitas tiap minggu ditampilkan pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Mengerjakan konversi tipe data UOM ID menjadi Long lalu melakukan Pull Request dan menunggu di merge
2	Melakukan Revisi Pull Request dan Mengerjakan generate document number untuk Selling Order
3	Melanjutkan mengerjakan generate document number untuk Selling Order dan melakukan Pull Request dan menunggu merge serta revisi Pull Request

Minggu Ke -	Pekerjaan yang dilakukan
4	Melakukan Revisi Pull Request
5	Melakukan Revisi Pull Request dan Perbaikan Fulfillment Selling Order Return
6	Memperbaiki selling order return fulfillment dan revisi Pull Request
7	Melakukan Revisi Pull Request dan Mengerjakan document number untuk selling delivery order
8	Revisi Pull Request dan Mengerjakan document number untuk selling order return
9	Mengerjakan document number untuk selling invoice dan juga invoice return serta Revisi Pull Request
10	Mengerjakan document number untuk selling deliver order return
11	Onboarding Quality Assurance pada Proyek BCA LMS dan membaca requirement aplikasi yang sedang dibuat
12	Melakukan verifikasi tiket UAT fase 1 pada fitur *sorting* dan *searching* untuk memastikan hasil pencarian dan pengurutan data sesuai dengan requirement
13	Melakukan pengujian UAT fase 1 pada tampilan *detail item* dan validasi kesesuaian data yang ditampilkan dengan data sebelumnya
14	Melakukan pengujian UAT fase 1 pada *success message*, *alert message*, dan popup konfirmasi untuk memastikan pesan sistem sesuai dengan konteks aksi pengguna
15	Melakukan pengujian UAT fase 1 pada fitur pencarian di tab *Assignment Profile* dan menu *Library*, termasuk validasi hasil dan wording yang ditampilkan
16	Melakukan pengujian UAT fase 1 pada aspek antarmuka pengguna, seperti fitur *drag kolom*, format tanggal, dan konsistensi tampilan halaman
17	Melakukan pengujian UAT fase 1 pada fitur *lazy load* dan navigasi halaman untuk memastikan performa dan alur pengguna berjalan sesuai requirement

Minggu Ke -	Pekerjaan yang dilakukan
18	Melakukan *retesting* hasil perbaikan UAT fase 1 serta validasi ulang tiket untuk memastikan bug yang diperbaiki tidak menimbulkan dampak pada fitur lain

3.3.2 Konsep Software Development Life Cycle ERP Cranium

Dalam pengembangan sistem ERP Cranium, perusahaan menerapkan konsep *Software Development Life Cycle* (SDLC) dengan pendekatan *Agile Development*. Pendekatan ini memungkinkan proses pengembangan dilakukan secara iteratif dan adaptif terhadap perubahan kebutuhan pengguna. Setiap siklus pengembangan mencakup tahapan analisis kebutuhan, perancangan, implementasi, pengujian, serta evaluasi hasil sebelum memasuki ke tahap berikutnya.

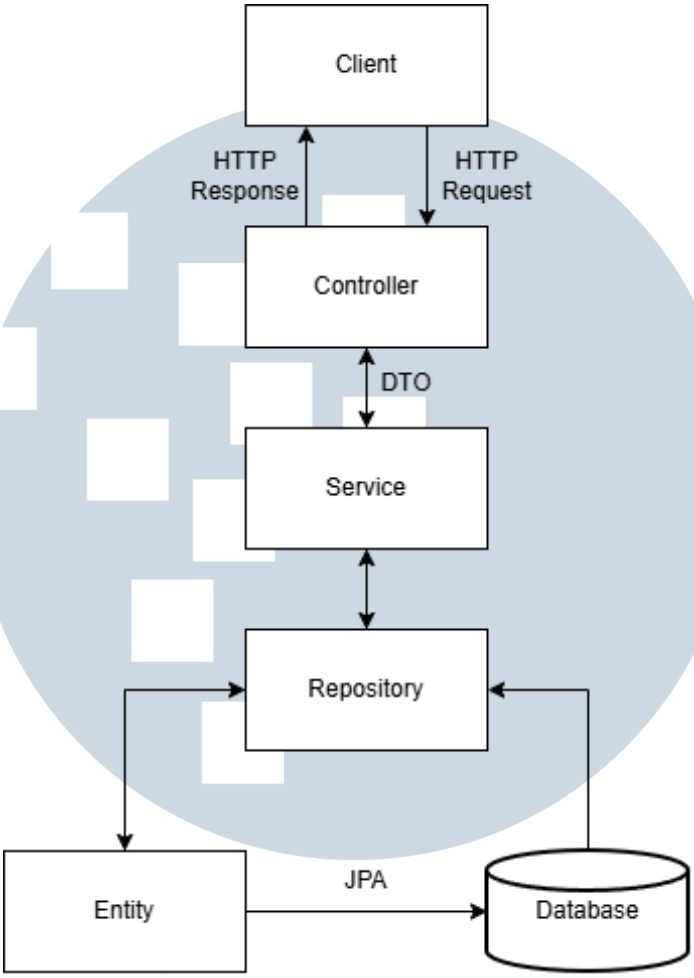
Pada periode magang batch kedua ini, penulis berfokus pada pengembangan bagian *backend* menggunakan *Java Spring Boot*, khususnya untuk submodul-submodul dalam modul *Selling*. Salah satu pekerjaan utama adalah implementasi fitur *generate document number* yang berfungsi untuk menghasilkan nomor dokumen otomatis berdasarkan format yang telah ditentukan oleh pengguna melalui modul *Master*.

Proses pengembangan dilakukan menggunakan *framework Spring Boot* dengan arsitektur berlapis yang terdiri dari:

1. Controller — menangani permintaan API dari *frontend*, melakukan validasi awal, dan meneruskan data ke lapisan *service*.
2. Service — memuat logika bisnis utama, termasuk proses pengambilan format dari modul *Master* dan pembuatan *document number* otomatis.
3. Repository — berfungsi untuk mengakses database menggunakan *Jakarta Persistence API (JPA)*.

Selain itu, pengujian dilakukan menggunakan dua pendekatan utama, yaitu:

- Unit Test, untuk memastikan logika bisnis berjalan sesuai harapan pada tingkat metode.
- Contract Test, untuk menjamin komunikasi antar modul dan format respons API tetap konsisten.



Gambar 3.1. Diagram Arsitektur Backend ERP Cranium

Gambar 3.1 menunjukkan alur kerja lapisan *backend* pada sistem ERP Cranium yang dikembangkan menggunakan *Spring Boot*. Lapisan *Controller* menerima permintaan dari pengguna, kemudian meneruskannya ke *Service* untuk diproses, sebelum data disimpan melalui *Repository* ke dalam basis data.

Konsep pengembangan aplikasi yang digunakan pada tahap ini tetap mengacu pada implementasi sebelumnya yang telah dijelaskan dalam laporan magang batch pertama, yaitu penggunaan *Data Transfer Object* (DTO), *Entity*, *Mapper*, serta struktur berlapis *Controller–Service–Repository*. Pendekatan tersebut membantu menjaga konsistensi, keterpisahan tanggung jawab antar lapisan, serta memudahkan proses pengujian dan pemeliharaan sistem.

3.3.3 Ruang Lingkup Pengembangan

Selama periode magang batch kedua, ruang lingkup pekerjaan yang dilakukan penulis berfokus pada pengembangan dan penyempurnaan modul *Selling* dalam sistem ERP Cranium. Pengembangan dilakukan secara bertahap berdasarkan prioritas proyek dan hasil evaluasi dari supervisor.

Secara umum, ruang lingkup kegiatan dapat dikategorikan ke dalam tiga bagian utama sebagai berikut:

1. Penanganan Bug

   Perbaikan *bug* dilakukan pada modul dan *Selling*. *Bug* yang diperbaiki berasal dari laporan supervisor maupun yang ditemukan secara langsung saat pengerjaan modul. Penanganannya mencakup sisi *backend*, dengan bentuk perbaikan berupa penyesuaian struktur data antara *frontend* dan *backend*.

2. Penambahan Fitur pada Submodul Eksisting

   Fitur tambahan dikembangkan pada submodul-submodul di dalam modul *Master* dan *Selling*. Salah satu pembaruan penting adalah penambahan fitur pengaturan format document number pada modul *Master* serta generate document number otomatis pada modul *Selling*. Fitur ini memungkinkan sistem untuk secara otomatis membuat nomor dokumen dengan pola yang sudah ditentukan oleh pengguna di modul *Master*.

3. Integrasi Antar Modul

   Integrasi antara modul *Master* dan *Selling* dilakukan untuk memastikan proses pengambilan format dan pembuatan nomor dokumen berjalan lancar. Ketika pengguna membuat data baru di modul *Selling*, sistem *backend* akan mengambil format dari modul *Master* dan menghasilkan nomor dokumen yang sesuai dengan pola yang sudah disimpan.

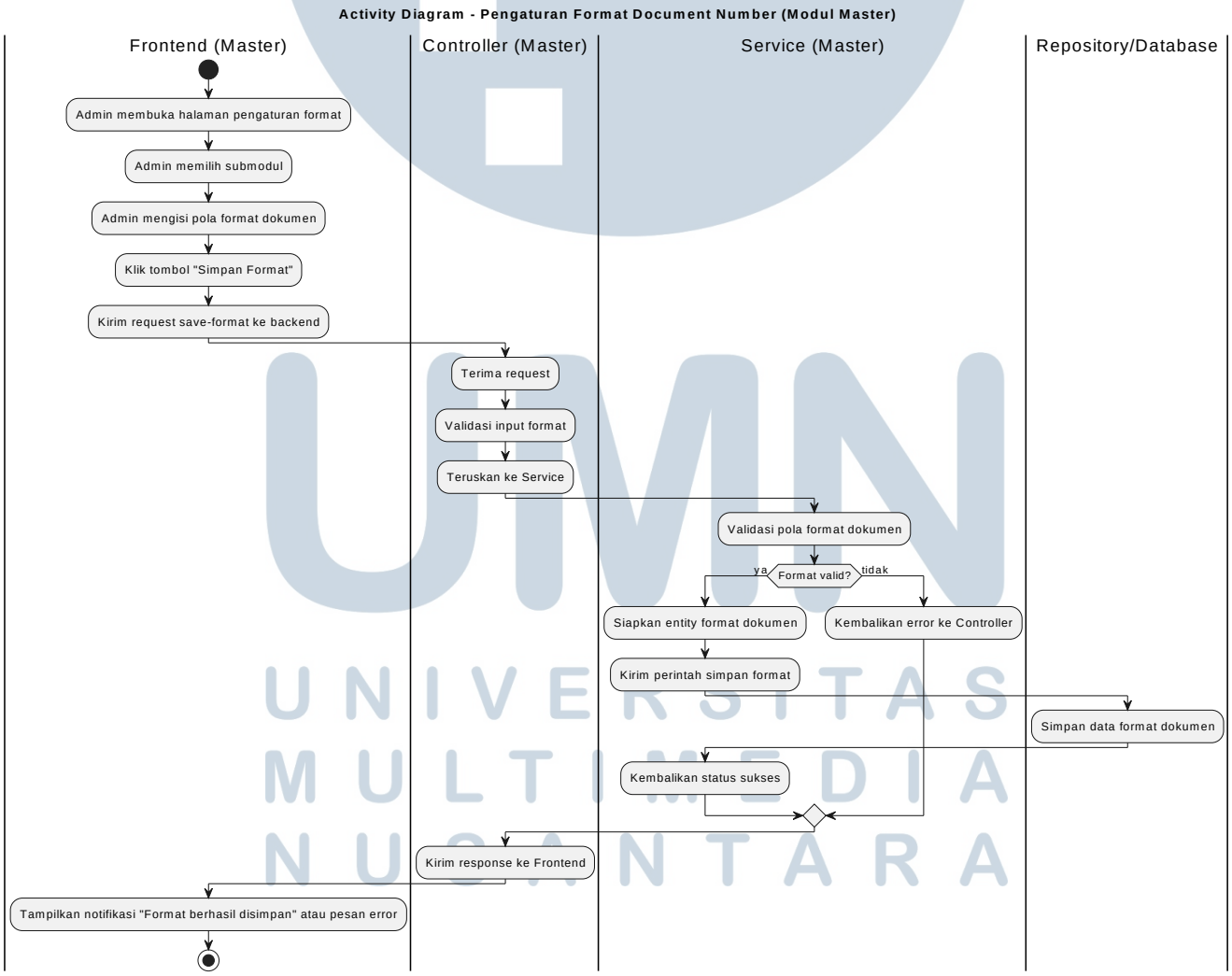
3.3.4 Diagram Sistem

Bagian ini menyajikan diagram untuk menggambarkan alur proses utama yang terkait dengan fitur *document number* pada sistem ERP Cranium. Diagram dibuat dalam bentuk *activity diagram* dengan pembagian *swimlane* untuk menunjukkan peran masing-masing komponen, yaitu pengguna, sisi *frontend*, *backend* Spring Boot, serta *database*.

Diagram pertama menjelaskan bagaimana pengguna melakukan pengaturan format nomor dokumen melalui modul *Master*, sementara diagram kedua menunjukkan proses otomatisasi pembuatan nomor dokumen ketika pengguna melakukan transaksi pada modul *Selling*. Kedua diagram ini digunakan untuk memperjelas alur kerja, interaksi antar layer, serta integrasi antara modul *Master* dan *Selling*.

A Activity Diagram Pengaturan Format Document Number (Modul Master)

Gambar 3.2 menampilkan alur proses saat pengguna mengatur format nomor dokumen pada *modul Master*. Tahapan ini dilakukan oleh admin untuk menentukan pola format yang akan digunakan oleh modul-modul lain di sistem ERP.

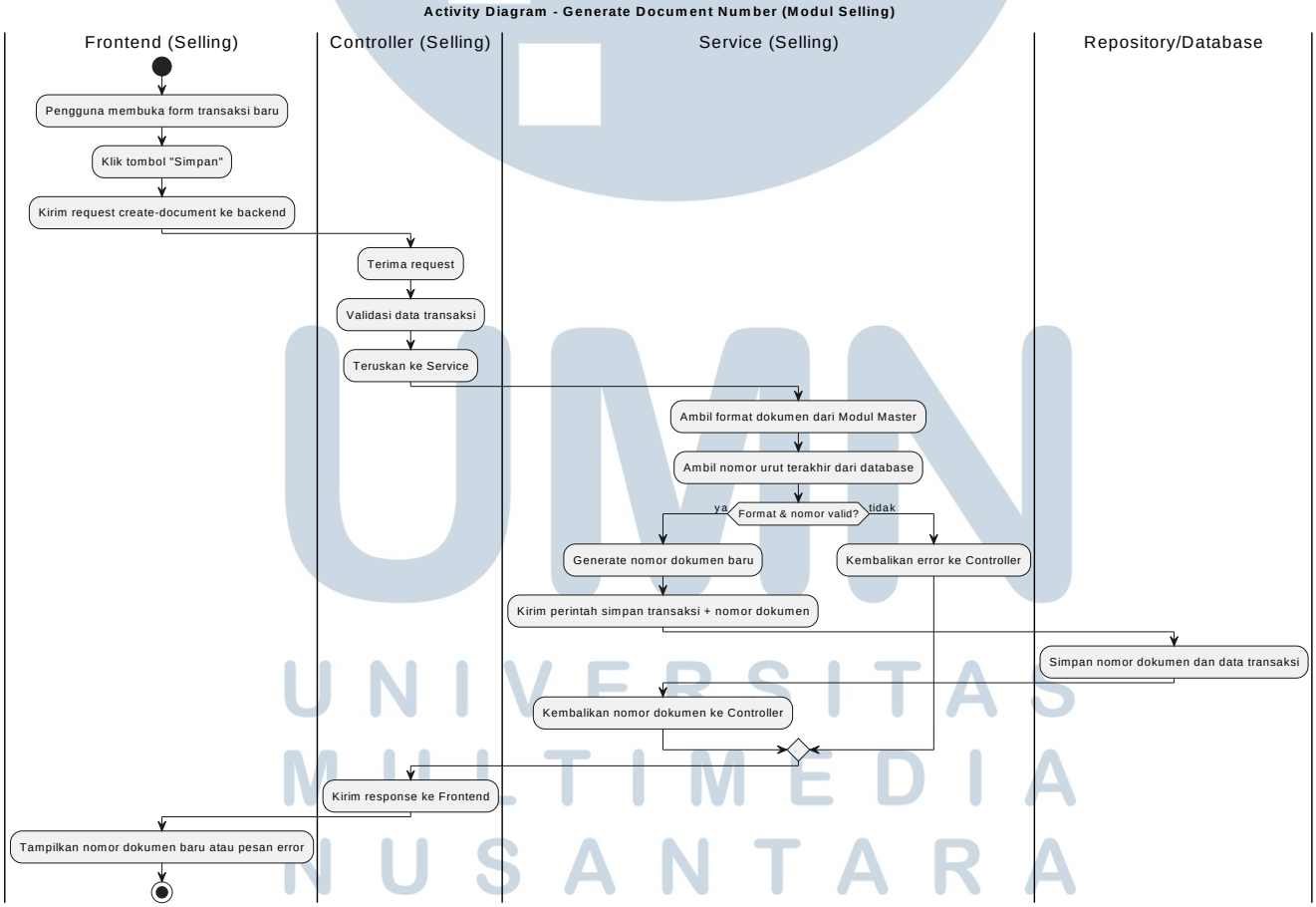


Gambar 3.2. Activity Diagram Pengaturan Format Document Number pada Modul Master

Prosesnya dimulai ketika pengguna membuka halaman pengaturan format dokumen, memilih submodul yang akan dikonfigurasi, kemudian mengisi pola format (misalnya `SO-{tahun}-{nomorUrut}`). Sistem akan melakukan validasi format, menyimpannya ke basis data, dan menampilkan notifikasi bahwa pengaturan berhasil disimpan.

B Activity Diagram Generate Document Number (Modul Selling)

Gambar 3.3 menunjukkan alur proses pembuatan nomor dokumen otomatis pada *modul Selling*. Setiap kali pengguna membuat transaksi baru, sistem secara otomatis mengambil pola format yang telah disimpan di modul *Master*, menghasilkan nomor dokumen berdasarkan urutan terakhir, dan menyimpannya bersamaan dengan data transaksi.



Gambar 3.3. Activity Diagram Generate Document Number pada Modul Selling

Pendekatan ini memastikan setiap dokumen yang dibuat memiliki nomor

yang unik dan konsisten dengan aturan yang telah ditetapkan, sekaligus meminimalkan kesalahan akibat input manual.

3.3.5 Pengembangan Modul Selling

Pengembangan pada modul *Selling* berfokus pada tiga aktivitas utama, yaitu implementasi fitur *generate document number*, perbaikan proses *fulfillment* pada *Selling Order Return*, serta penyesuaian struktur data agar konsisten dengan standar ERP Cranium. Setiap pekerjaan mengikuti alur SDLC internal: analisis kebutuhan, implementasi, pengujian, asistensi, dan penggabungan kode melalui *pull request*.

A Implementasi Generate Document Number

Pada modul *Selling*, beberapa transaksi sebelumnya belum memiliki mekanisme penomoran dokumen yang terpusat dan mengikuti konfigurasi dari modul *Master*. Kondisi tersebut menyebabkan format nomor dokumen tidak konsisten serta sulit dilakukan pengelolaan ulang secara terpusat. Untuk mengatasi permasalahan tersebut, diterapkan mekanisme pembuatan nomor dokumen otomatis yang terintegrasi langsung dengan modul *Master*. Sistem terlebih dahulu mengambil konfigurasi nomor dokumen berdasarkan *domain* dan *menu*, kemudian menyusunnya sesuai urutan format yang telah ditentukan sebelum data transaksi disimpan.

Untuk mengatasi permasalahan tersebut, diterapkan mekanisme pembuatan nomor dokumen otomatis yang terintegrasi langsung dengan modul *Master*. Sistem terlebih dahulu mengambil konfigurasi nomor dokumen berdasarkan *domain* dan *menu*, kemudian menyusunnya sesuai urutan format yang telah ditentukan sebelum data transaksi disimpan. Implementasi perbaikan ini mencakup utilitas pembentuk nomor dokumen, penggunaan *WebClient* untuk mengambil konfigurasi dari modul *Master*, serta integrasi mekanisme tersebut pada proses penyimpanan transaksi. Contoh implementasi ditunjukkan pada potongan kode berikut.

```
1 @Component
2 @NoArgsConstructor(access = AccessLevel.PRIVATE)
3 public class DocumentNumberUtil {
4
5     public static String buildDocumentNumber(DocumentNumberDto
    documentNumberDto) {
6         StringBuilder documentNumber = new StringBuilder();
```

```
7
8           for (DocumentNumberPreferenceDto format :
    documentNumberDto.getFormat()) {
9               switch (format.getField()) {
10                  case "YEAR":
11                      documentNumber.append(LocalDate.now().getYear
    ());
12                      break;
13                  case "MONTH":
14                      documentNumber.append(String.format("%02d",
    LocalDate.now().getMonthValue()));
15                      break;
16                  case "PREFIX":
17                      documentNumber.append(documentNumberDto.
    getMenuPrefix());
18                      break;
19                  case "SEPARATOR":
20                      documentNumber.append("/");
21                      break;
22                  case "SEQUENCE_5":
23                      documentNumber.append(
24                          String.format("%05d", documentNumberDto.
    getSequenceNo()));
25                      break;
26                  default:
27                      throw new IllegalArgumentException("Format
    document number tidak didukung");
28              }
29          }
30          return documentNumber.toString();
31      }
32  }
```

Kode 3.1: Util pembentuk nomor dokumen berdasarkan konfigurasi Master

Kode 3.1 berfungsi menyusun nomor dokumen berdasarkan urutan format yang diperoleh dari modul *Master*. Setiap elemen format diproses secara dinamis, mulai dari informasi tanggal, prefix menu, hingga nomor urut dengan panjang digit tertentu, sehingga perubahan format dapat dilakukan tanpa modifikasi kode pada submodul transaksi.

```
1 @Component
2 public class SellingDocumentNumberWebClient {
3
```

```
    @Autowired
    private WebClient internalWebClient;

    public DocumentNumberDto getDocumentNumberByDomainNameMenuName
    (
            Optional<HttpHeaderDto> httpHeaderDto,
            String domainName,
            String menuName) {

        return internalWebClient.get()
            .uri(uriBuilder -> uriBuilder
                .path("/document-number")
                .queryParam("domainName", domainName)
                .queryParam("menuName", menuName)
                .build())
            .retrieve()
            .bodyToMono(DocumentNumberDto.class)
            .block();
    }
}
```

Kode 3.2: WebClient pengambilan konfigurasi nomor dokumen dari modul Master

Untuk memperoleh konfigurasi nomor dokumen secara terpusat, sistem menggunakan mekanisme komunikasi antar modul melalui *WebClient*. Kode 3.2 menunjukkan proses pengambilan konfigurasi nomor dokumen dari modul *Master* berdasarkan parameter *domain* dan *menu* yang sesuai dengan submodul transaksi.

```
DocumentNumberDto documentNumberDto =
    sellingDocumentNumberWebClient.
    getDocumentNumberByDomainNameMenuName(
        httpHeaderService.getHttpHeader(),
        "SELLING",
        "SELLING_ORDER");

String documentNo =
    DocumentNumberUtil.buildDocumentNumber(documentNumberDto);

try {
    Orders orders = Orders.builder()
        .documentNo(documentNo)
        .status(SellingStatus.NEW.getValue())
        .systemDate(LocalDateTime.now())
        .build();
```

```
16
17      ordersRepository.save(orders);
18  } catch (Exception ex) {
19      DocumentNumberReserveCreateEvent event =
20          DocumentNumberReserveCreateEvent.builder()
21              .documentNumberId(documentNumberDto.getId())
22              .sequenceNo(documentNumberDto.getSequenceNo())
23              .build();
24
25      applicationEventPublisher.publishEvent(event);
26      throw ex;
27  }
```

Kode 3.3: Integrasi generate document number dan mekanisme pengamanan transaksi

Pada lapisan service, proses *Generate Document Number* diterapkan pada tahap awal alur pembuatan transaksi, sebagaimana ditunjukkan pada Kode 3.3. Proses diawali dengan pemanggilan modul *Master* melalui *SellingDocumentNumberWebClient* untuk memperoleh konfigurasi nomor dokumen berdasarkan parameter *domain* `"SELLING"` dan *menu* `"SELLING_ORDER"`. Konfigurasi yang diperoleh kemudian diproses menggunakan *DocumentNumberUtil.buildDocumentNumber()* untuk membentuk nomor dokumen baru sebelum entitas transaksi dibuat.

Nomor dokumen yang telah dihasilkan selanjutnya diset ke dalam objek entitas *Orders* melalui properti `documentNo`, kemudian transaksi disimpan ke basis data menggunakan *ordersRepository.save()*, sebagaimana terlihat pada bagian *try block* pada Kode 3.3. Pendekatan ini memastikan bahwa setiap transaksi memiliki identitas dokumen yang valid dan sesuai dengan standar penomoran sebelum proses penyimpanan dilakukan.

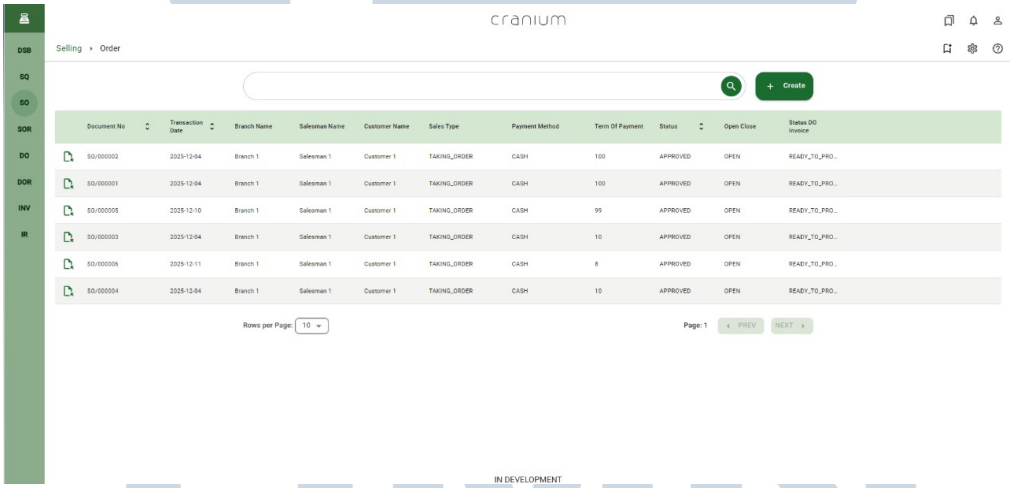
Sebagai mekanisme pengamanan, Kode 3.3 juga memperlihatkan penanganan kondisi kegagalan penyimpanan transaksi melalui *exception handling*. Apabila terjadi kegagalan pada proses penyimpanan, sistem akan memicu *DocumentNumberReserveCreateEvent* yang berisi informasi *documentNumberId* dan *sequenceNo*. Mekanisme ini berfungsi untuk menjaga konsistensi nomor dokumen dengan mencegah terjadinya lonjakan nomor urut atau hilangnya *sequence* akibat transaksi yang tidak berhasil disimpan.

Pola implementasi pada lapisan service ini diterapkan secara konsisten pada submodul lain, seperti *Delivery Order*, *Invoice*, dan *Selling Order Return*, dengan perbedaan utama terletak pada parameter *menu* yang digunakan saat pengambilan

konfigurasi nomor dokumen. Dengan alur pemanggilan konfigurasi, pembentukan nomor dokumen, serta mekanisme pengamanan yang seragam, sistem menjadi lebih terstruktur, mudah dipelihara, dan andal dalam menangani transaksi berskala besar.

Pengujian dilakukan dengan membuat transaksi *Selling Order* untuk memastikan bahwa nomor dokumen dihasilkan sesuai dengan konfigurasi pada modul *Master*. Pada skenario pengujian ini, sistem berhasil memanggil konfigurasi format, menyusun nomor dokumen sesuai pola yang ditentukan, serta menyimpan transaksi tanpa kendala. Nomor dokumen yang terbentuk telah mengikuti urutan, *prefix*, dan format tanggal yang ditetapkan, sehingga konsistensi penomoran antar transaksi pada modul *Selling* dapat terjaga.

Gambar 3.4 berikut menunjukkan hasil akhir setelah transaksi berhasil dibuat. Nomor dokumen ditampilkan pada antarmuka dan tersimpan di database sesuai proses yang telah diimplementasikan.



Gambar 3.4. Hasil pembuatan Selling Order dengan nomor dokumen yang berhasil digenerate

Melalui pengujian ini dapat disimpulkan bahwa proses pembentukan nomor dokumen telah berjalan dengan benar, terintegrasi penuh dengan modul *Master*, dan konsisten pada setiap transaksi *Selling Order*. Dengan mekanisme ini, seluruh submodul pada *Selling* kini menggunakan pola penomoran terstandarisasi, sehingga memudahkan pengelolaan maupun perubahan format di kemudian hari.

B Perbaikan Proses Fulfillment Selling Order Return

Pada proses *Selling Order Return* ditemukan beberapa ketidaksesuaian, seperti kuantitas retur yang memungkinkan melebihi jumlah pada transaksi awal,

struktur data yang belum sepenuhnya mendukung alur bisnis retur, serta kebutuhan penyesuaian data untuk mendukung komponen *infinite autocomplete*. Kondisi tersebut berpotensi menimbulkan inkonsistensi data dan kesalahan validasi pada proses retur.

Untuk mengatasi permasalahan tersebut, dilakukan sejumlah perbaikan pada alur *Selling Order Return*. Perbaikan mencakup penyesuaian struktur DTO agar selaras dengan kebutuhan bisnis, penerapan validasi kuantitas retur, serta penyempurnaan mekanisme pencarian dokumen untuk mendukung *infinite autocomplete*. Selain itu, penyesuaian juga dilakukan pada lapisan service, controller, dan repository untuk memastikan proses retur berjalan konsisten, valid, dan sesuai dengan alur bisnis yang ditetapkan.

Perubahan ini tidak hanya menambahkan lapisan validasi baru pada level input, tetapi juga memastikan perhitungan subtotal, diskon, dan pajak dilakukan secara terpusat dan akurat. Dengan begitu, setiap retur yang dibuat maupun diperbarui tetap berada dalam batas kuantitas yang seharusnya.

```
@Data
@Builder
@Jacksonized
@NewOrderReturnDetailFulfillmentQuantityIsWithinLimit
public class OrdersReturnDetailCreateDto {

    @NotNull(message = "{selling.itemid.notnull}")
    @Positive(message = "{selling.itemid.positive}")
    @Min(value = 0, message = "{selling.discountamount.min}")
    @Max(value = 100000000, message = "{selling.discountamount.max
    }")
    private BigDecimal discountAmount;

    private Long ordersDetailId;
}
```

Kode 3.4: DTO OrdersReturnDetailCreateDto dengan validasi kuantitas (Bagian 1)

DTO ini memperkenalkan anotasi `NewOrderReturnDetailFulfillment QuantityIsWithinLimit` yang berfungsi memastikan bahwa permintaan retur tidak melebihi batas kuantitas yang masih tersedia dari transaksi awal, sebagaimana diimplementasikan pada Kode 3.4. Validasi ini penting untuk mencegah data retur yang tidak wajar, terutama ketika detail pesanan sudah pernah dipenuhi sebagian pada transaksi sebelumnya.

```
public class NewOrderReturnDetailFulfillmentQuantityValidator
    implements ConstraintValidator<
    NewOrderReturnDetailFulfillmentQuantityIsWithinLimit,
    OrdersReturnDetailCreateDto> {

    @Autowired
    private OrdersDetailRepository ordersDetailRepository;

    @Autowired
    private ResourceBundleMessageSource sellingMessageSource;

    private final String
    SELLING_ORDER_RETURN_DETAIL_QUANTITY_EXCEED = "selling.
    orderreturndetail.quantity.exceed";

    @Override
    public boolean isValid(OrdersReturnDetailCreateDto
    ordersReturnDetailCreateDto, ConstraintValidatorContext context
    ) {
        if (ordersReturnDetailCreateDto.getOrdersDetailId() ==
    null) {
            return true;
        }

        Optional<OrdersDetail> ordersDetailOptional =
    ordersDetailRepository.findById(ordersReturnDetailCreateDto.
    getOrdersDetailId());
        if (ordersDetailOptional.isEmpty()) {
            return true;
        }

        OrdersDetail ordersDetail = ordersDetailOptional.get();
        Double remainingQuantity = ordersDetail.
    getOrdersReturnDetailRemainingQuantity();
        Double requestedQuantity = ordersReturnDetailCreateDto.
    getQuantity().doubleValue();

        if (remainingQuantity > 0 && !ordersDetail.
    getOrdersReturnDetailFulfilled()) {
            if (requestedQuantity > remainingQuantity) {
                context.disableDefaultConstraintViolation();
                context.buildConstraintViolationWithTemplate(
                        sellingMessageSource.getMessage(
```

```
31
    SELLING_ORDER_RETURN_DETAIL_QUANTITY_EXCEED,
32                                  new Object[]{requestedQuantity,
    remainingQuantity},
33                                  LocaleContextHolder.getLocale()
34                          )
35                  ).addConstraintViolation();
36                  return false;
37              }
38          } else {
39              if (requestedQuantity > ordersDetail.getQuantity().
    doubleValue()) {
40                  context.disableDefaultConstraintViolation();
41                  context.buildConstraintViolationWithTemplate(
42                          sellingMessageSource.getMessage(
43
    SELLING_ORDER_RETURN_DETAIL_QUANTITY_EXCEED,
44                                  new Object[]{requestedQuantity,
    ordersDetail.getQuantity()},
45                                  LocaleContextHolder.getLocale()
46                          )
47                  ).addConstraintViolation();
48                  return false;
49              }
50          }
51
52          return true;
53      }
54  }
```

Kode 3.5: Validator NewOrderReturnDetailFulfillmentQuantityValidator (Bagian 2)

Validator ini menjalankan pemeriksaan kuantitas secara langsung pada DTO sebelum masuk ke lapisan service, sebagaimana ditunjukkan pada Kode 3.5. Mekanismenya memastikan bahwa nilai retur tidak melebihi *remaining quantity*, dan apabila detail pesanan sudah pernah dipenuhi penuh, validator membandingkannya dengan kuantitas awal pesanan. Pesan error disesuaikan menggunakan *message bundle* agar konsisten dengan standar error aplikasi.

```
1  public class OrderReturnDetailFulfillmentQuantityValidator
       implements ConstraintValidator<
       OrderReturnDetailFulfillmentQuantityIsWithinLimit,
       OrdersReturnDetailUpdateDto> {
2      @Autowired
```

```
    private OrdersDetailRepository ordersDetailRepository;

    @Autowired
    private OrdersReturnDetailRepository
ordersReturnDetailRepository;

    @Autowired
    private ResourceBundleMessageSource sellingMessageSource;

    private final String
SELLING_ORDER_RETURN_DETAIL_QUANTITY_EXCEED = "selling.
orderreturndetail.quantity.exceed";

    @Override
    public boolean isValid(OrdersReturnDetailUpdateDto
ordersReturnDetailUpdateDto, ConstraintValidatorContext
constraintValidatorContext) {
        if (ordersReturnDetailUpdateDto.getOrdersDetailId() ==
null) {
            return true;
        }

        Optional<OrdersReturnDetail> ordersReturnDetailOptional =
ordersReturnDetailRepository.findById(
ordersReturnDetailUpdateDto.getId());
        if (ordersReturnDetailOptional.isEmpty()) {
            return true;
        }
        OrdersReturnDetail ordersReturnDetail =
ordersReturnDetailOptional.get();

        BigDecimal updateQuantity = BigDecimal.valueOf(0);
        if (ordersReturnDetailUpdateDto.getId() == null ||
ordersReturnDetailUpdateDto.getId() == 0) {
            updateQuantity = ordersReturnDetailUpdateDto.
getQuantity();
        } else {
            updateQuantity = ordersReturnDetailUpdateDto.
getQuantity().subtract(ordersReturnDetail.getQuantity());
        }

        Optional<OrdersDetail> ordersDetailOptional =
ordersDetailRepository.findById(ordersReturnDetailUpdateDto.
```

```
    getOrdersDetailId());
33          if (ordersDetailOptional.isEmpty()) {
34              return true;
35          }
36
37          OrdersDetail ordersDetail = ordersDetailOptional.get();
38          Double remainingQuantity = ordersDetail.
    getOrdersReturnDetailRemainingQuantity();
39
40          if (updateQuantity.doubleValue() > remainingQuantity) {
41              constraintValidatorContext.
    disableDefaultConstraintViolation();
42              constraintValidatorContext.
    buildConstraintViolationWithTemplate(
43                      sellingMessageSource.getMessage(
44
    SELLING_ORDER_RETURN_DETAIL_QUANTITY_EXCEED,
45                              new Object[]{updateQuantity,
    remainingQuantity},
46                              LocaleContextHolder.getLocale()
47                      )
48              ).addConstraintViolation();
49              return false;
50          }
51
52          return true;
53      }
54  }
```

Kode 3.6: Validator OrderReturnDetailFulfillmentQuantityValidator untuk Update (Bagian 3)

Validator ini digunakan saat proses pembaruan retur. Nilai yang dicek bukan hanya kuantitas baru, tetapi selisihnya dari jumlah sebelumnya, sebagaimana diimplementasikan pada Kode 3.6. Dengan cara ini, setiap perubahan tetap berada dalam batas kuantitas yang belum terpakai dari detail pesanan terkait.

```
1 @GetMapping(value = "/order/document-nos", headers = "X-Api-
    Version=1")
2 @IsSellingOrdersRead
3 @ResponseStatus(HttpStatus.OK)
4 public Page<OrdersDocumentNoDto> findAllOrdersDocumentNo(
5         Pageable pageable,
6         @RequestParam(required = false) List<Long> orderIds) {
```

```
7       return ordersService.findAllOrdersDocumentNo(pageable,
        orderIds);
8   }
```

Kode 3.7: Endpoint untuk mengambil document number dengan pagination (Bagian 4)

Endpoint ini menyediakan akses data dokumen dengan pagination serta opsi penyaringan berdasarkan ID, sebagaimana ditunjukkan pada Kode 3.7. Fungsionalitas ini sangat membantu frontend, khususnya pada komponen *infinite autocomplete* yang membutuhkan data bertahap dan stabil.

```
1   @Query(value = findOrderDocumentNoQuery,
2           countQuery = "SELECT COUNT(1) FROM \"selling\".\"orders\"
        WHERE deleted = false AND status = :status",
3           nativeQuery = true)
4   Page<OrdersDocumentNo> findOrderDocumentNoByStatus(
5           @Param("status") Short status, Pageable pageable);
```

Kode 3.8: Repository query untuk document number dengan status dan pagination (Bagian 5)

Query repository ini mengembalikan document number berdasarkan status tertentu, sebagaimana diimplementasikan pada Kode 3.8. Dengan pagination dari Spring Data, beban data dapat dikontrol sehingga lebih efisien untuk kebutuhan frontend.

```
1   private void buildNewOrderReturnDetails(List<
        OrdersReturnDetailCreateDto> ordersReturnDetailCreateDtoList,
        OrdersReturn ordersReturn, List<OrdersReturnDetail>
        ordersReturnDetails) throws ClientException, ServerException{
2       BigDecimal subtotal = BigDecimal.valueOf(0);
3       BigDecimal totalItemDiscountAmount = BigDecimal.valueOf(0);
4       BigDecimal totalPPN = BigDecimal.valueOf(0);
5       BigDecimal totalPPH = BigDecimal.valueOf(0);
6       for (OrdersReturnDetailCreateDto detail :
        ordersReturnDetailCreateDtoList) {
7           OrdersReturnDetail ordersReturnDetail =
        ordersReturnDetailBuild(detail);
8           ordersReturnDetail.setOrdersReturn(ordersReturn);
9
10          BigDecimal itemSubtotal = getItemSubtotal(
        ordersReturnDetail.getPrice(), ordersReturnDetail.getQuantity()
        );
11          subtotal = subtotal.add(itemSubtotal);
12
```

```
13          OrdersReturnTotalDetailDiscountAmountParameterDto paramDto
     = OrdersReturnTotalDetailDiscountAmountParameterDto
14                  .builder()
15                  .itemSubtotal(itemSubtotal)
16                  .disc1(ordersReturnDetail.getDiscount1())
17                  .disc2(ordersReturnDetail.getDiscount2())
18                  .disc3(ordersReturnDetail.getDiscount3())
19                  .disc4(ordersReturnDetail.getDiscount4())
20                  .disc5(ordersReturnDetail.getDiscount5())
21                  .disc1Amount(ordersReturnDetail.getDiscount1Amount
    ())
22                  .disc2Amount(ordersReturnDetail.getDiscount2Amount
    ())
23                  .disc3Amount(ordersReturnDetail.getDiscount3Amount
    ())
24                  .disc4Amount(ordersReturnDetail.getDiscount4Amount
    ())
25                  .disc5Amount(ordersReturnDetail.getDiscount5Amount
    ())
26                  .build();
27
28          BigDecimal itemDiscountAmount =
    getTotalDetailDiscountAmount(paramDto);
29          totalItemDiscountAmount = totalItemDiscountAmount.add(
    itemDiscountAmount);
30
31          BigDecimal totalAfterDiscount = itemSubtotal.subtract(
    itemDiscountAmount);
32          BigDecimal ppnValue = getPpnValueByVatId(
    ordersReturnDetail.getVatId());
33
34          totalPPN = totalPPN.add(getPpnPphAmount(totalAfterDiscount
    , ppnValue));
35          totalPPH = totalPPH.add(getPpnPphAmount(totalAfterDiscount
    , ordersReturnDetail.getPphPercentage()));
36
37          ordersReturnDetail.setDiscountAmount(itemDiscountAmount);
38          ordersReturnDetails.add(ordersReturnDetail);
39      }
40
41      ordersReturn.setOrdersReturnDetail(ordersReturnDetails);
42      ordersReturn.setSubTotalAmount(subtotal);
43      ordersReturn.setItemDiscountAmount(totalItemDiscountAmount);
```

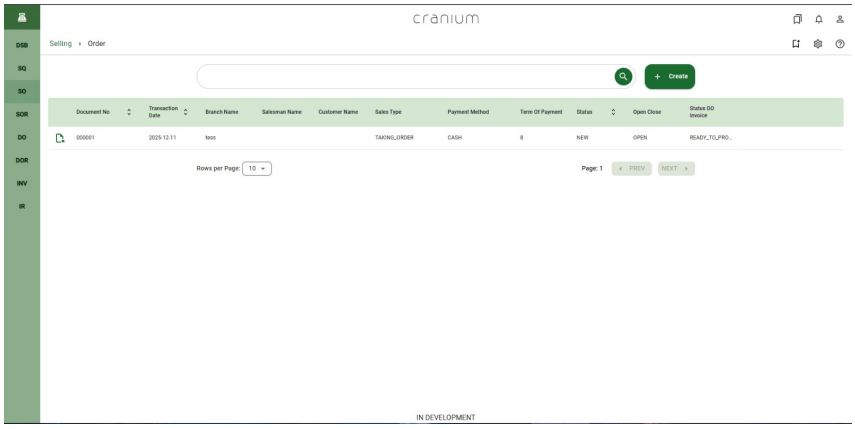
```
44      ordersReturn.setTotalPphAmount(totalPPH);
45      ordersReturn.setTotalPpnAmount(totalPPN);
46  }
```

Kode 3.9: Service perhitungan subtotal, diskon, dan pajak (Bagian 6)

Seluruh proses perhitungan subtotal, diskon, dan pajak dilakukan secara konsisten dalam metode ini, sebagaimana ditunjukkan pada Kode 3.9. Setiap detail retur dibangun satu per satu, lalu dihitung nilai diskon, PPN, dan PPH berdasarkan parameter yang ada. Pendekatan seperti ini membantu menjaga konsistensi nilai antar transaksi serta mencegah kemungkinan kesalahan hitung ketika jumlah detail semakin banyak.

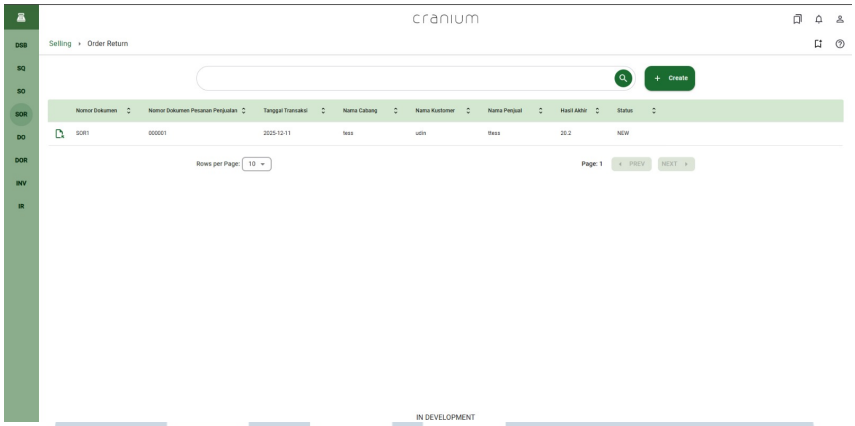
Pengujian fitur *Selling Order Return* dilakukan menggunakan tiga skenario utama dengan variasi kondisi input untuk memastikan proses retur berjalan sesuai kebutuhan bisnis. Pada setiap skenario, sistem berhasil memvalidasi data, memproses retur, dan menghasilkan keluaran yang sesuai. Dokumentasi hasil pengujian ditampilkan pada Gambar 3.5, 3.6, dan 3.7.

Skenario pertama menunjukkan bahwa sistem berhasil membuat transaksi *Selling Order* (SO) menggunakan nomor dokumen yang telah dihasilkan secara otomatis. Pada tahap ini, seluruh data berhasil tersimpan dan SO dapat dilanjutkan ke proses berikutnya seperti ditunjukkan pada Gambar 3.5.



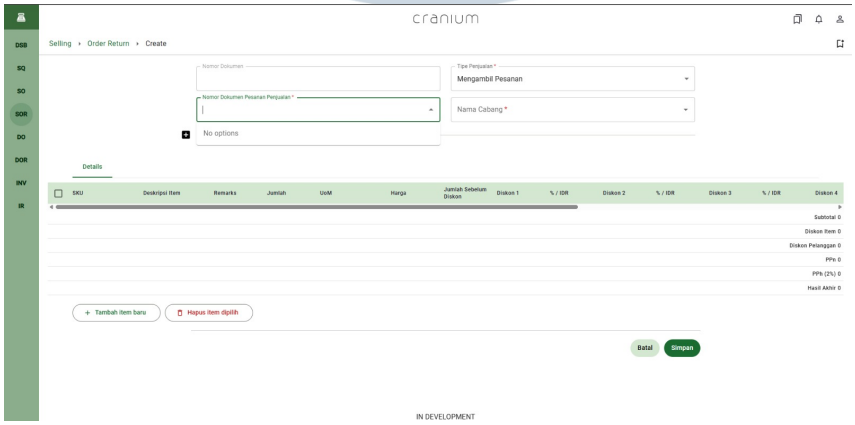
Gambar 3.5. Hasil pembuatan Selling Order (SO) berhasil

Pada skenario kedua, sistem berhasil membuat *Selling Order Return* (SOR) berdasarkan SO yang telah di-approve. Proses retur berjalan sesuai perhitungan baru yang telah diperbaiki, mencakup validasi kuantitas, subtotal, diskon, dan pajak. Hasilnya dapat dilihat pada Gambar 3.6.



Gambar 3.6. Hasil pembuatan Selling Order Return (SOR) berdasarkan SO yang valid

Skenario ketiga menguji pembatasan kuantitas retur. Ketika pengguna mencoba membuat SOR kedua untuk SO yang sama, daftar SO tidak muncul pada komponen *infinite autocomplete*. Kondisi ini terjadi karena SO tersebut sudah sepenuhnya digunakan pada retur sebelumnya sehingga tidak memiliki kuantitas tersisa. Perilaku ini membuktikan bahwa validasi berjalan dengan benar. Hasilnya ditampilkan pada Gambar 3.7.



Gambar 3.7. SO tidak muncul karena sudah digunakan pada retur sebelumnya (validasi berhasil)

Melalui ketiga skenario tersebut, dapat disimpulkan bahwa perbaikan pada proses *Selling Order Return* telah menghasilkan sistem yang lebih konsisten, aman, dan sesuai dengan aturan bisnis yang berlaku. Validasi kuantitas berjalan efektif, perhitungan nilai transaksi akurat, dan mekanisme pemilihan dokumen kini bekerja secara terkendali.

C Penyesuaian Struktur Data pada Modul Selling

Pada modul *Selling*, masih terdapat sejumlah atribut yang menggunakan tipe data `Integer`, sementara standar sistem ERP Cranium telah menetapkan penggunaan `Long` untuk seluruh identitas data dan referensi utama. Ketidakseragaman tipe data ini menyebabkan beberapa kendala, seperti kebutuhan konversi manual yang berulang, ketidakkonsistenan pada proses validasi, serta potensi kesalahan pada pemetaan entitas antar lapisan sistem.

Untuk mengatasi permasalahan tersebut, dilakukan penyesuaian tipe data dari `Integer` menjadi `Long` pada berbagai lapisan aplikasi, meliputi DTO, Entity, Controller, Service, Interface Custom, hingga Validator. Penyesuaian ini bertujuan untuk menyelaraskan struktur data dengan standar sistem serta menghilangkan kebutuhan konversi manual yang tidak diperlukan. Contoh perubahan implementasi ditunjukkan pada potongan Kode 3.10.

```
private Integer baseUomId;
private Integer targetUomId;

// Menjadi
private Long baseUomId;
private Long targetUomId;
```

Kode 3.10: Perubahan tipe data pada DTO

Perubahan ini memastikan bahwa seluruh ID pada modul *Selling* mengikuti standar sistem dan tidak lagi membutuhkan konversi manual di layer lain.

Controller sebelumnya masih menerima parameter ID dengan tipe dasar `long`. Tipe tersebut kemudian diseragamkan menjadi `Long` agar konsisten dengan DTO dan Entity, sebagaimana ditunjukkan pada Kode 3.11.

```
public UnitOfMeasurementDto findUnitOfMeasurementById(
    @PathVariable long id)
// Menjadi
public UnitOfMeasurementDto findUnitOfMeasurementById(
    @PathVariable Long id)

public UomConversionDto updateUomConversion(@RequestBody
    UomConversionUpdateDto dto, @PathVariable long id)
// Menjadi
public UomConversionDto updateUomConversion(@RequestBody
    UomConversionUpdateDto dto, @PathVariable Long id)
```

Kode 3.11: Penyesuaian tipe data pada Controller

Dengan perubahan ini, proses pengambilan dan pembaruan data menjadi lebih konsisten karena seluruh ID kini menggunakan tipe yang sama.

Beberapa atribut pada Entity juga masih menggunakan tipe `int`. Atribut tersebut diperbarui agar sejajar dengan standar ID lain di sistem, seperti ditunjukkan pada Kode 3.12.

```
private int baseFormUomId;

// Menjadi
private Long baseFormUomId;
```

Kode 3.12: Penyesuaian tipe data pada Entity

Selain itu, interface yang digunakan sebagai hasil proyeksi query kustom juga mengalami penyesuaian tipe data agar selaras dengan perubahan pada Entity, sebagaimana ditampilkan pada Kode 3.13.

```
Integer getBaseUomId();

// Menjadi
Long getBaseUomId();
```

Kode 3.13: Perubahan pada Custom Interface

Penyesuaian ini dapat mencegah error saat pemetaan entitas, terutama ketika data yang diterima frontend menggunakan tipe Long.

Pada lapisan service, penyesuaian tipe data juga dilakukan untuk menghilangkan kebutuhan konversi manual dari `Integer` ke `Long`. Perubahan tersebut ditunjukkan pada Kode 3.14.

```
.baseFormUomId(1)
// Menjadi
.baseFormUomId(1L)

// Sebelumnya
private UnitOfMeasurement getUom(Integer uomId) {
    Optional<UnitOfMeasurement> uomOptional =
        uomRepository.findByIdAndDeletedFalse(Long.valueOf(uomId))
    ;
}

// Menjadi
private UnitOfMeasurement getUom(Long uomId) {
    Optional<UnitOfMeasurement> uomOptional =
        uomRepository.findByIdAndDeletedFalse(uomId);
```

```
}
```

Kode 3.14: Penyesuaian logika Service agar tidak lagi melakukan konversi manual

Dengan ini, data yang diterima repository langsung kompatibel dengan lapisan lainnya tanpa konversi tambahan.

Penyesuaian serupa juga diterapkan pada Validator agar konsisten dengan tipe data terbaru, seperti ditunjukkan pada Kode 3.15.

```
public class MasterUomIsExistsValidator implements
    ConstraintValidator<MasterUomIsExist, Integer> {

public boolean isValid(Integer uomId, ConstraintValidatorContext
    context) {
     if (Objects.isNull(uomId)) return true;
     uomService.findUnitOfMeasurementById((long) uomId);
     return true;
}

// Menjadi
public class MasterUomIsExistsValidator implements
    ConstraintValidator<MasterUomIsExist, Long> {

public boolean isValid(Long uomId, ConstraintValidatorContext
    context) {
     if (Objects.isNull(uomId)) return true;
     uomService.findUnitOfMeasurementById(uomId);
     return true;
}
```

Kode 3.15: Penyesuaian Validator dari Integer ke Long

Dengan tipe yang sudah seragam, proses validasi dan pemanggilan repository menjadi lebih sederhana dan minim error.

Hasil dari penyesuaian ini menunjukkan bahwa struktur data pada seluruh submodul *Selling* menjadi lebih konsisten. Selain itu, proses validasi dan integrasi antar modul berjalan lebih stabil, serta potensi terjadinya error akibat perbedaan tipe data dapat diminimalkan.

3.3.6 Daftar Service Backend yang Dikembangkan

Selama pelaksanaan kerja praktik pada modul *Selling*, dilakukan pengembangan dan perbaikan sejumlah service backend untuk mendukung

implementasi *generate document number*, perbaikan proses *Selling Order Return*, serta penyesuaian struktur data agar sesuai dengan standar sistem ERP Cranium. Service-service tersebut diakses melalui endpoint REST yang digunakan oleh frontend maupun komunikasi antar modul. Ringkasan daftar service, endpoint, metode HTTP, serta bentuk request dan response ditampilkan pada Tabel 3.2.

Tabel 3.2. Daftar Service Backend pada Modul Selling

Nama Service	**Endpoint**	**Method**	**Request**	**Response**
Generate Document Number	`/document-number`	GET	domainName, menuName	DocumentNumber Dto
Create Selling Order	`/selling/orders`	POST	OrdersCreateDto	OrdersDto
Get Selling Order Document Number	`/selling/order-document-nos`	GET	Pageable, orderIds	Orders DocumentNoDto
Create Selling Order Return	`/selling/order-returns`	POST	OrdersReturn CreateDto	OrdersReturnDto
Update Selling Order Return	`/selling/order-returns/\{id\}`	PUT	OrdersReturn UpdateDto	OrdersReturnDto
Validasi Kuantitas Retur	Validator Internal	–	OrdersReturnDetail CreateDto	Valid / Error
Get Unit of Measurement by ID	`/master/uom/\{id\}`	GET	id (Long)	UnitOf MeasurementDto

Tabel 3.2 menunjukkan bahwa setiap service yang dikembangkan memiliki peran spesifik dalam mendukung proses bisnis pada modul *Selling*. Service *Generate Document Number* digunakan untuk memastikan konsistensi penomoran dokumen antar transaksi, sementara service *Selling Order Return* mendukung validasi kuantitas, perhitungan nilai transaksi, serta integrasi dengan komponen *infinite autocomplete*. Penyesuaian struktur data memastikan bahwa seluruh service menggunakan tipe data yang seragam dan sesuai dengan standar sistem, sehingga meminimalkan potensi kesalahan pada proses integrasi antar modul.

3.3.7 UAT BCA LMS

Selain berperan sebagai *developer* pada modul *Selling*, penulis juga menjalankan tugas sebagai Assurance Analyst pada proyek BCA LMS. Peran ini berfokus pada proses verifikasi dan validasi fungsi aplikasi untuk memastikan kesesuaiannya dengan kebutuhan bisnis yang telah ditetapkan oleh client. Pengujian dilakukan mengikuti alur SDLC, dimulai dari memahami requirement, menyusun skenario pengujian, melakukan eksekusi, dan mendokumentasikan hasilnya secara sistematis.

Ruang lingkup pekerjaan meliputi verifikasi tiket UAT, melakukan *retesting* setelah perbaikan developer, memastikan apakah temuan merupakan bug, misunderstanding, atau enhancement, serta mencatat seluruh proses dalam *Defect List* internal. Selama proses UAT berlangsung, client mengirimkan daftar temuan melalui sistem tiket, dan penulis bertanggung jawab untuk memvalidasi apakah isu dapat direproduksi dan sesuai dengan acceptance criteria yang berlaku.

Untuk memastikan setiap fungsi berjalan sesuai requirement, penulis menyusun skenario pengujian dalam bentuk tabel yang berisi deskripsi skenario, langkah uji, dan hasil yang diperoleh. Beberapa contoh skenario uji selama fase UAT ditunjukkan pada tabel berikut:

Tabel 3.3. Contoh Skenario Pengujian selama fase UAT

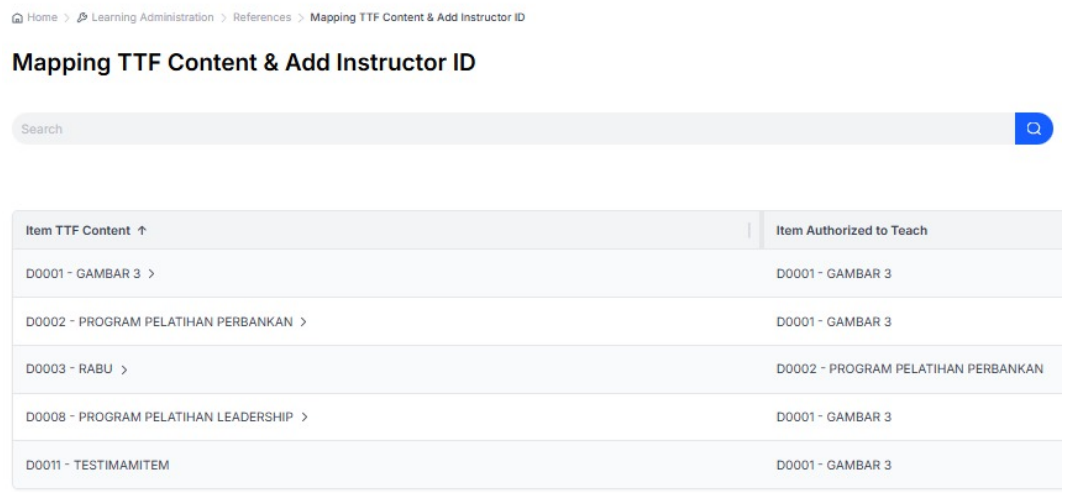
Skenario	**Langkah Pengujian**	**Hasil**
Sorting	1. Pada menu utama, pilih *References*. 2. Pilih sub menu *Mapping TTF*. 3. Klik pada field *Item Authorized to teach*. 4. Cek apakah sorting bekerja atau tidak.	Sesuai
Searching	1. Pilih menu *References*. 2. Masuk ke sub menu *Course Group*. 3. Input data yang ingin dicari lalu klik *Search* 4. Pastikan data yang keluar sesuai dengan apa yang dicari	Sesuai

Skenario	Langkah Pengujian	Hasil
Detail item	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Klik tombol *View* 4. Ketika sudah berada di *Detail items*, cek apakah data *Course Group* muncul.	Sesuai
Success Message	1. Pilih menu *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Klik tombol *View* 4. Ketika sudah di *Details* pindah tab menuju *Teaching Materials*. 5. Lalu upload file dan liat apakah success message yang keluar sesuai dengan req.	Sesuai
Searching Item - Tab AP	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Klik tombol *View* 4. Ketika sudah berada di *Detail items*, pindah ke tab *Assignment Profile*. 5. Lalu search dan lihat apakah data yang keluar sesuai dengan apa yang diinput	Sesuai
Searching Library	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Library*. 4. Masukkan judul yang ingin dicari 5. Klik tombol *Search* 6. Pastikan data yang keluar adalah data yang sesuai dengan judul yang dicari.	Sesuai
Wordingan Library	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Library*. 3. Masukkan 2 filter di selection screen 4. Klik tombol *Search* 5. Pastikan wordingan sesuai dengan req.	Sesuai

Skenario	Langkah Pengujian	Hasil
Wordingan AP - Library	1. Pilih *Manage User Learning*. 2. Masuk ke sub menu *Asssignment Profile*. 3. Klik tombol *View* 4. Ketika sudah berada di *Detail*, pindah ke tab *Library*. 5. Klik *Add Item*. 6. Pastikan Wordingan sesuai dengan req.	Sesuai
Popup edit message AP - Library	1. Pilih *Manage User Learning*. 2. Masuk ke sub menu *Asssignment Profile*. 3. Klik tombol *View* 4. Ketika sudah berada di *Detail*, pindah ke tab *Library*. 5. Klik *Add Item*. 6. Pilih data lalu klik *Save*. 7. Pastikan muncul pop up *Edit Confirmation* lalu klik *Save*.	Sesuai
Alert message success AP - Library	1. Pilih *Manage User Learning*. 2. Masuk ke sub menu *Asssignment Profile*. 3. Klik tombol *View* 4. Ketika sudah berada di *Detail*, pindah ke tab *Library*. 5. Klik *Add Item*. 6. Pilih data lalu klik *Save*. 7. Pastikan wordingan merujuk ke Assignment profile dan tidak null	Sesuai
Drag kolom	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Pilih pagination 100/page 4. Ketika data muncul pastikan kolom bisa dilebar kecilkan.	Sesuai

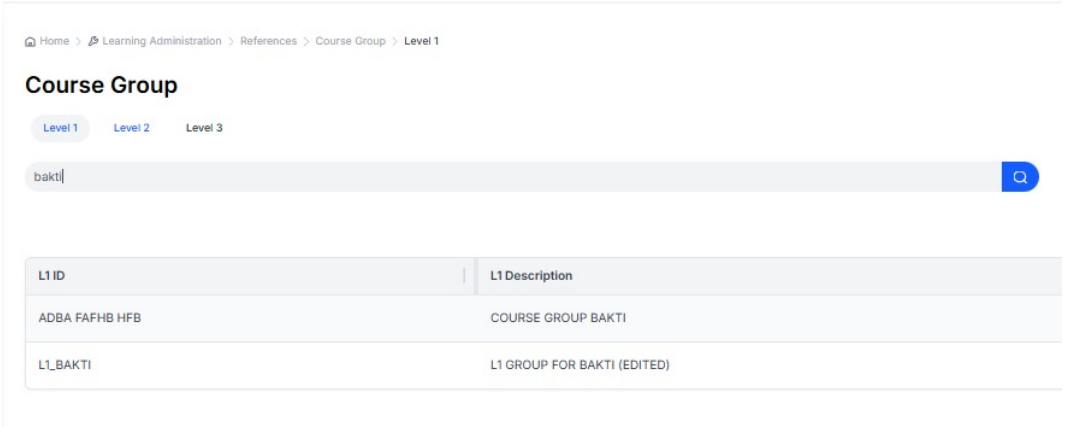
Skenario	Langkah Pengujian	Hasil
Wording discard confirmation	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Klik tombol *View* 4. Ketika sudah berada di *Detail items*, pindah ke tab *Categories*. 5. Klik *Add New* 6. Pilih data lalu klik *Cancel*. 7. Pastikan wording Discard Confirmation sesuai.	Sesuai
Format date	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Klik tombol *View* 4. Ketika sudah berada di *Detail items*, cek apakah format *Creation Date* sudah sesuai dengan req atau belum.	Sesuai
Pop up library access	1. Pilih menu *People*. 2. Masuk ke sub menu *User*. 3. Klik tombol *View*. 4. Pada halaman *User Profile*, buka tab *Library Access*. 5. Pilih data lalu klik *Preview*. 6. Pastikan lebar preview sesuai	Sesuai
Lazy load	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Klik tombol *Add New* 4. Pilih data yang ingin dicari di *Course Group L1*. 5. Pastikan lazy load berjalan ketika scroll ke bawah	Sesuai

Berikut merupakan dokumentasi hasil pengujian User Acceptance Test (UAT) pada aplikasi BCA LMS. Dokumentasi ini disajikan dalam bentuk gambar pendukung yang dilengkapi dengan narasi penjelasan untuk memastikan hasil pengujian dapat dipahami dengan jelas.



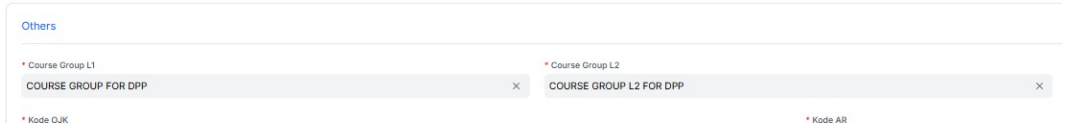
Gambar 3.8. Hasil pengujian fitur sorting

Gambar 3.8 menunjukkan hasil pengujian fitur *sorting*. Data berhasil diurutkan sesuai dengan kriteria yang dipilih pengguna, menandakan bahwa fungsi pengurutan berjalan dengan baik dan sesuai dengan requirement.



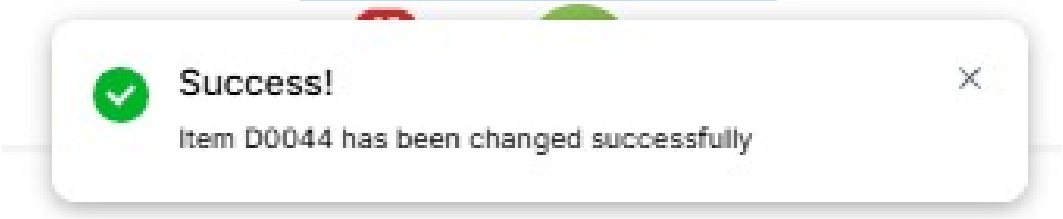
Gambar 3.9. Hasil pengujian fitur searching

Gambar 3.9 menunjukkan hasil pengujian fitur *searching*. Sistem mampu menampilkan data yang relevan berdasarkan kata kunci yang dimasukkan oleh pengguna, sehingga fungsi pencarian berjalan sesuai dengan requirement.



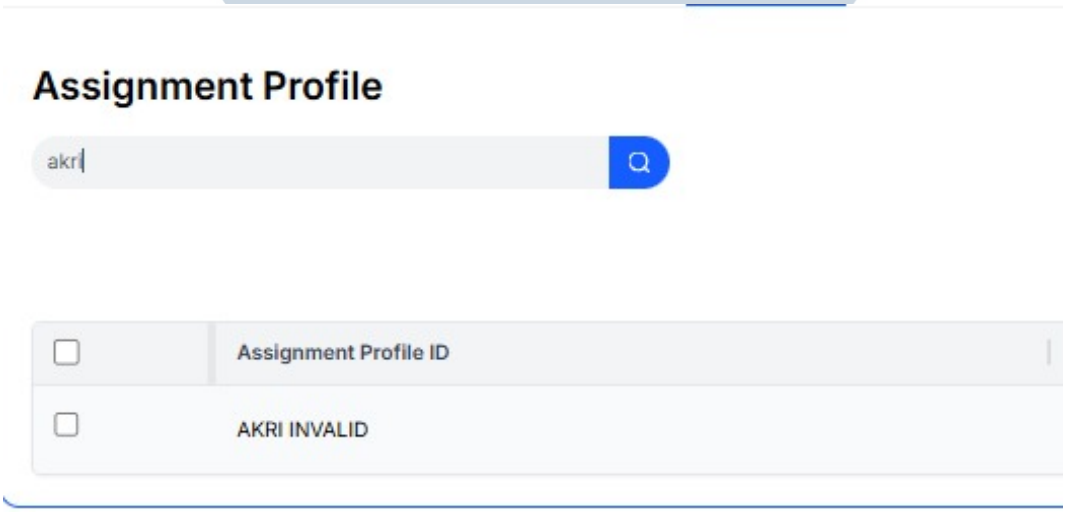
Gambar 3.10. Tampilan halaman detail item

Gambar 3.10 menunjukkan tampilan halaman *detail item*. Informasi yang ditampilkan telah sesuai dengan data item yang dipilih dan konsisten dengan data pada halaman sebelumnya.



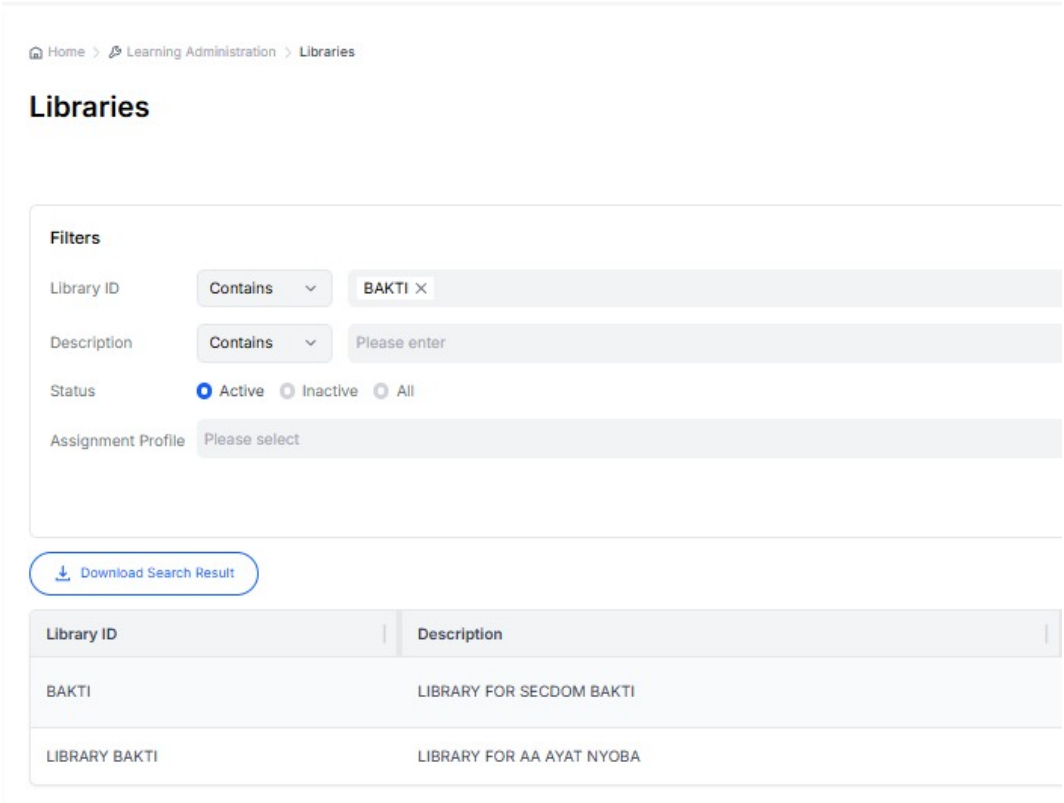
Gambar 3.11. Tampilan success message

Gambar 3.11 menunjukkan pesan keberhasilan yang muncul setelah proses berhasil dilakukan. Pesan yang ditampilkan sesuai dengan aksi pengguna dan memberikan umpan balik yang jelas.



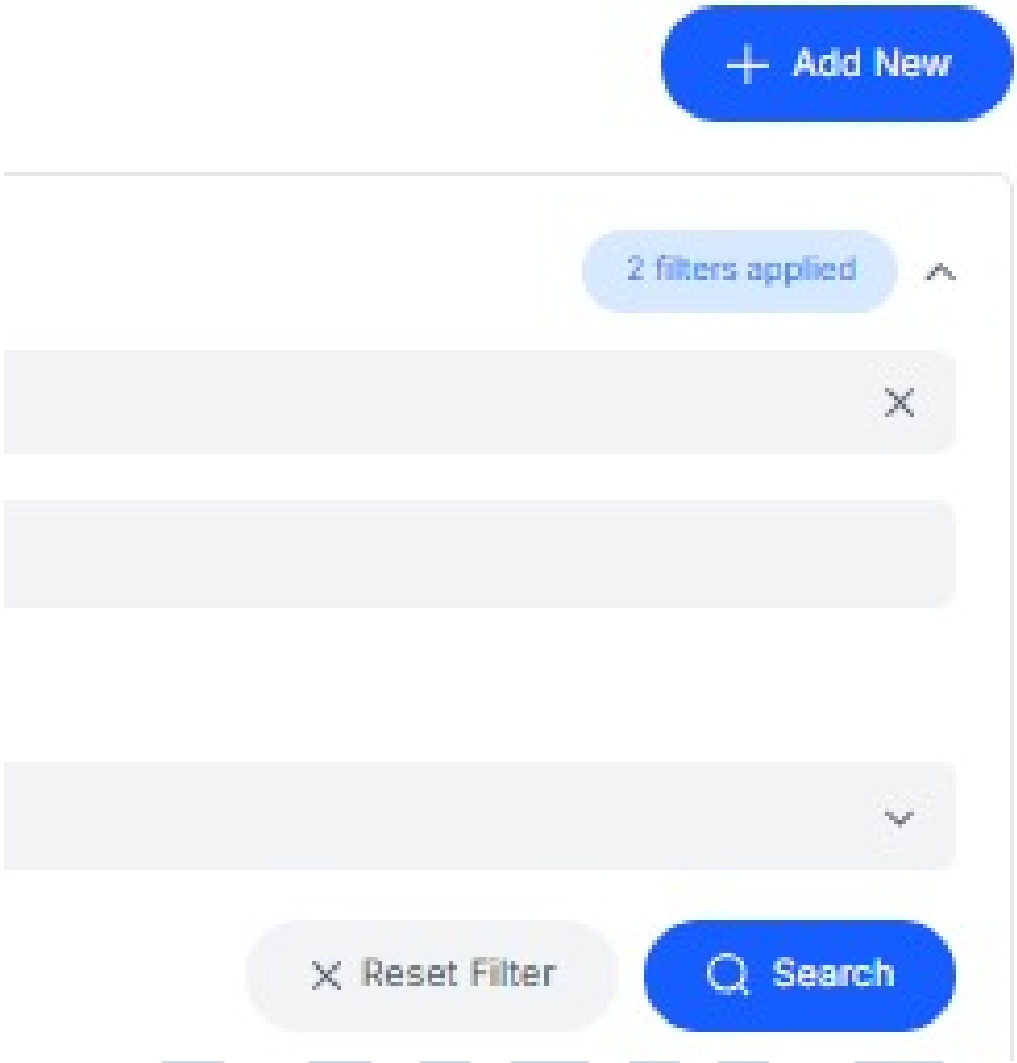
Gambar 3.12. Hasil pengujian searching pada tab Assignment Profile

Gambar 3.12 menunjukkan hasil pengujian fitur pencarian pada tab *Assignment Profile*. Data yang ditampilkan sesuai dengan input pencarian yang diberikan oleh pengguna.



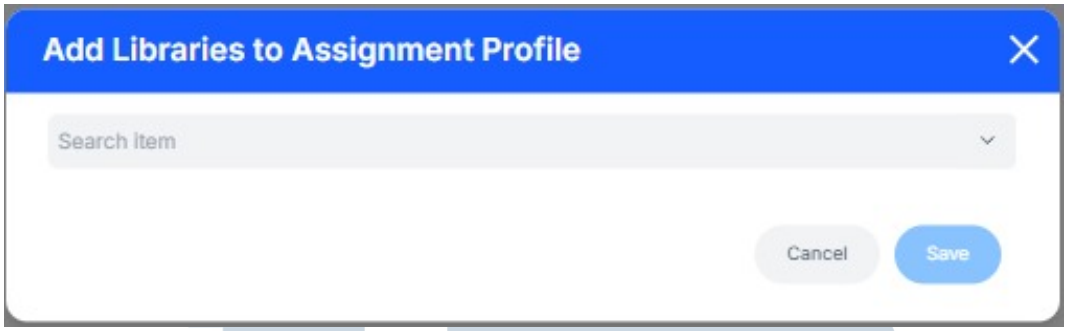
Gambar 3.13. Hasil pengujian searching pada menu Library

Gambar 3.13 menunjukkan bahwa sistem berhasil menampilkan data pada menu *Library* sesuai dengan kata kunci pencarian yang dimasukkan.



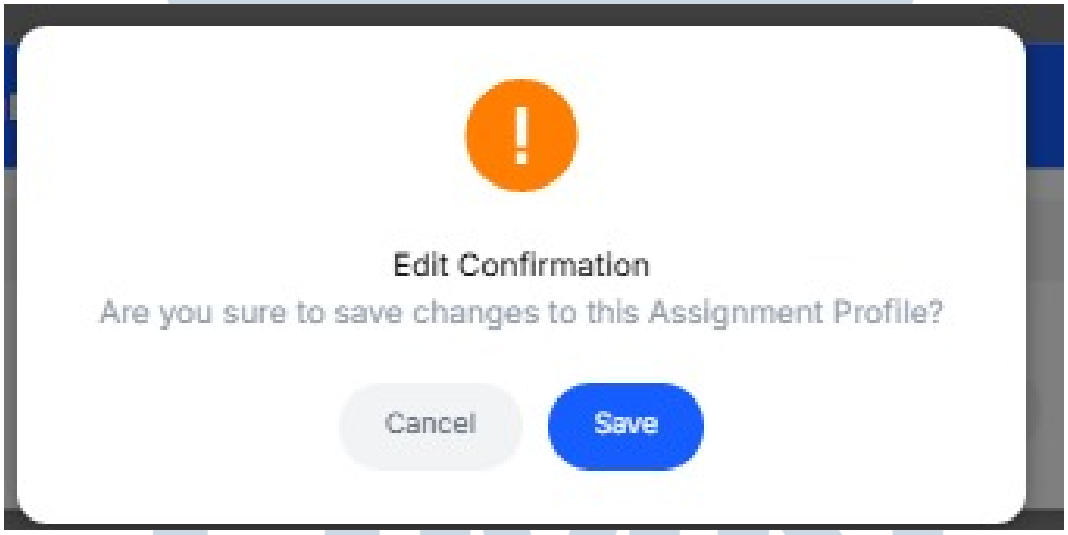
Gambar 3.14. Penyesuaian wording pada Library

Gambar 3.14 menunjukkan bahwa wording yang ditampilkan telah menyesuaikan kondisi filter yang digunakan dan sesuai dengan ketentuan requirement.



Gambar 3.15. Penyesuaian wording pada Assignment Profile - Library

Gambar 3.15 menunjukkan bahwa penyesuaian wording pada tab *Library* di *Assignment Profile* telah diterapkan dengan benar dan konsisten.

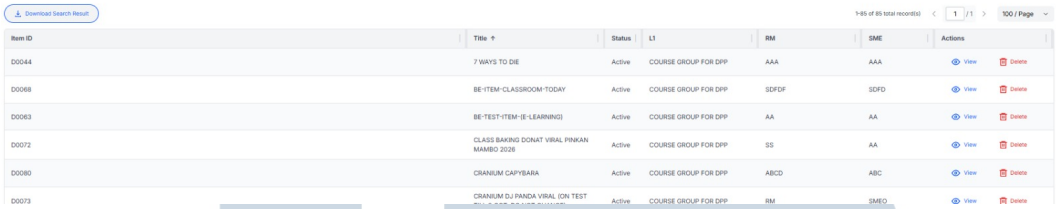


Gambar 3.16. Popup edit message pada Assignment Profile

Gambar 3.16 menunjukkan popup konfirmasi edit yang muncul saat pengguna melakukan perubahan data. Informasi yang ditampilkan telah sesuai dengan konteks aksi pengguna.

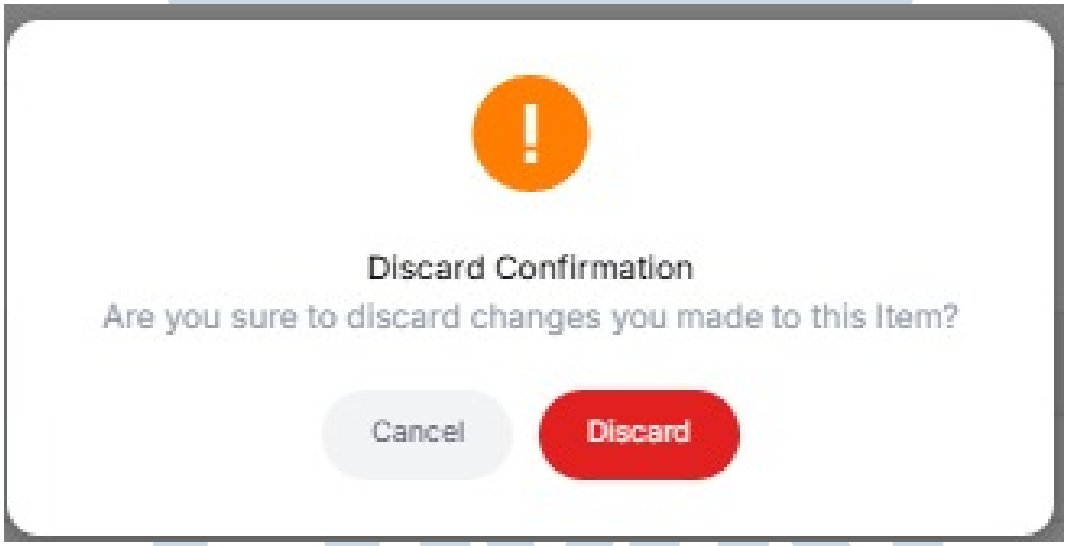
Gambar 3.17. Alert success pada Assignment Profile

Gambar 3.17 menunjukkan alert keberhasilan setelah proses penyimpanan data berhasil dilakukan. Pesan yang ditampilkan sesuai dengan requirement.



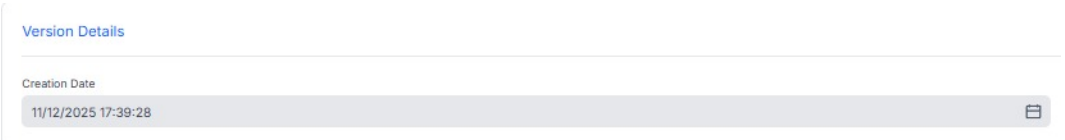
Gambar 3.18. Hasil pengujian fitur drag kolom

Gambar 3.18 menunjukkan bahwa fitur *drag kolom* berjalan dengan baik, sehingga pengguna dapat menyesuaikan tampilan tabel sesuai kebutuhan.



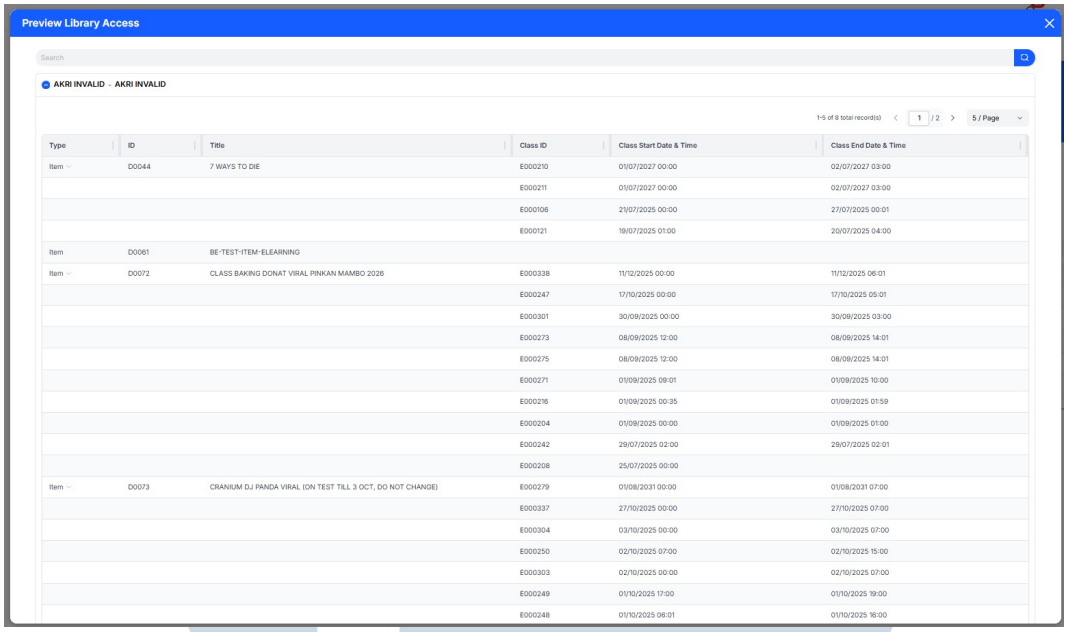
Gambar 3.19. Wording discard confirmation

Gambar 3.19 menunjukkan pesan konfirmasi pembatalan yang ditampilkan ketika pengguna membatalkan proses. Wording yang digunakan sudah jelas dan sesuai dengan requirement.



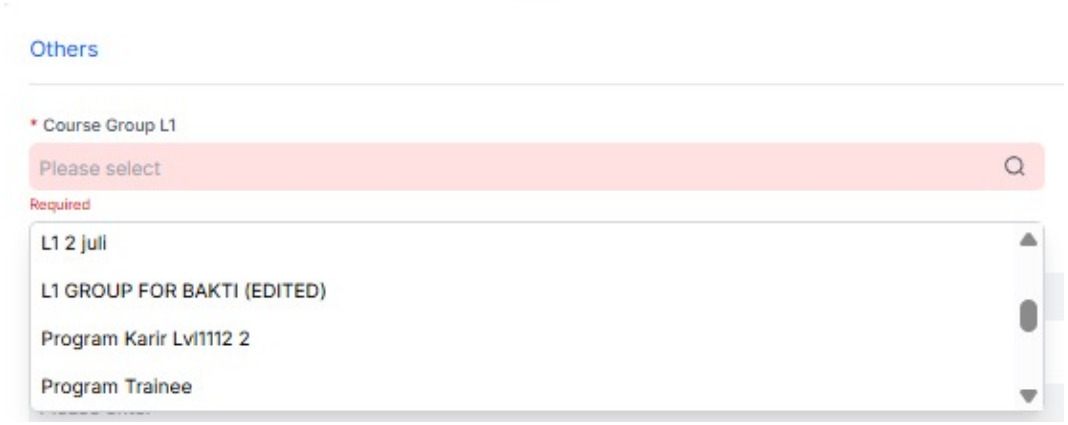
Gambar 3.20. Penyesuaian format tanggal

Gambar 3.20 menunjukkan bahwa format tanggal yang ditampilkan pada aplikasi telah sesuai dengan standar dan ketentuan yang ditetapkan.



Gambar 3.21. Popup library access

Gambar 3.21 menunjukkan popup *library access* dengan tampilan yang jelas dan mudah dibaca oleh pengguna.



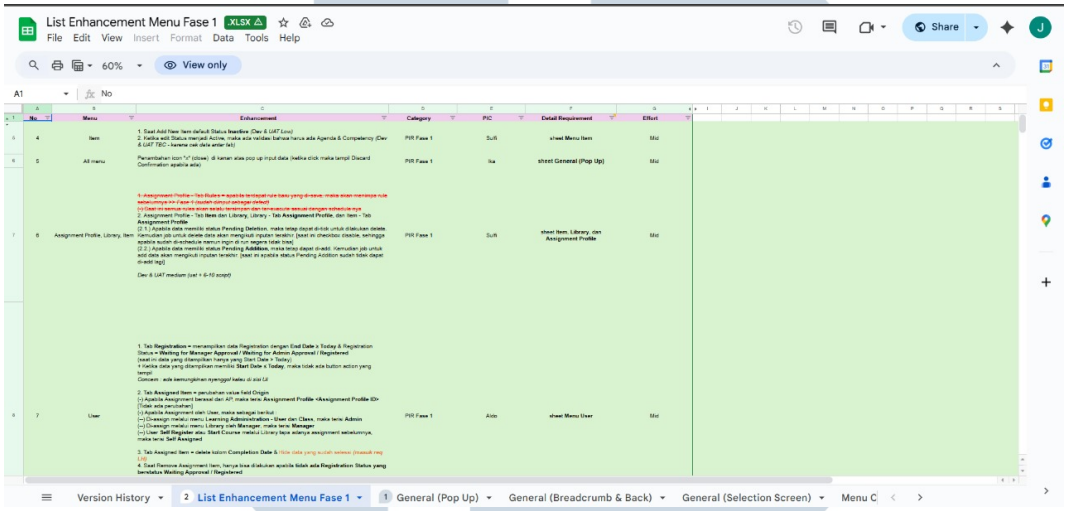
Gambar 3.22. Hasil pengujian fitur lazy load

Gambar 3.22 menunjukkan bahwa fitur *lazy load* berjalan dengan baik, di mana data dimuat secara bertahap saat pengguna melakukan scroll.

Selain melakukan pengujian terhadap bug, penulis juga menangani permintaan peningkatan fitur atau *Product Improvement Request* (PIR) yang diajukan oleh client setelah proses UAT utama selesai. Setiap permintaan enhancement dicatat dalam Daftar PIR, kemudian dianalisis untuk menentukan perubahan yang diperlukan serta dampaknya terhadap alur bisnis aplikasi. Setelah

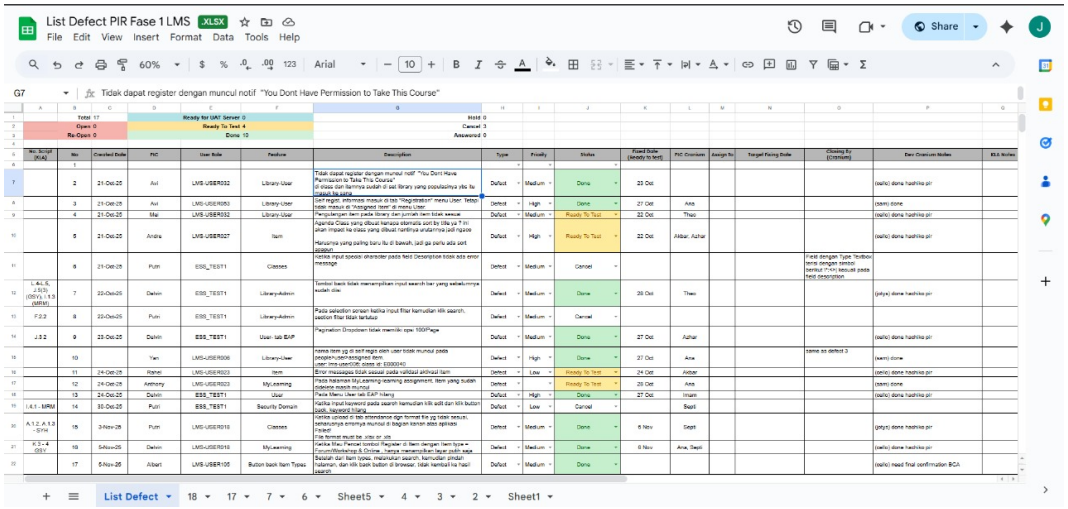
developer menyelesaikan implementasinya, penulis menyusun skenario uji dan melakukan verifikasi hasil.

Berikut merupakan contoh daftar PIR yang diterima selama tahap peningkatan fitur:



Gambar 3.23. Contoh Daftar PIR pada Proyek BCA LMS

Client juga menyampaikan setiap *bug enhancement* melalui tiket resmi. Contoh tiket *bug* enhancement dapat dilihat pada gambar berikut:



Gambar 3.24. Contoh Tiket Enhancement dari Client

Setelah developer menyelesaikan implementasi *enhancement*, penulis melakukan pengujian tambahan untuk memastikan setiap perubahan telah berjalan

sesuai kebutuhan dan tidak menimbulkan dampak pada fitur lain. Contoh skenario pengujian PIR ditunjukkan pada tabel berikut:

Tabel 3.4. Contoh Skenario Pengujian Enhancement (PIR)

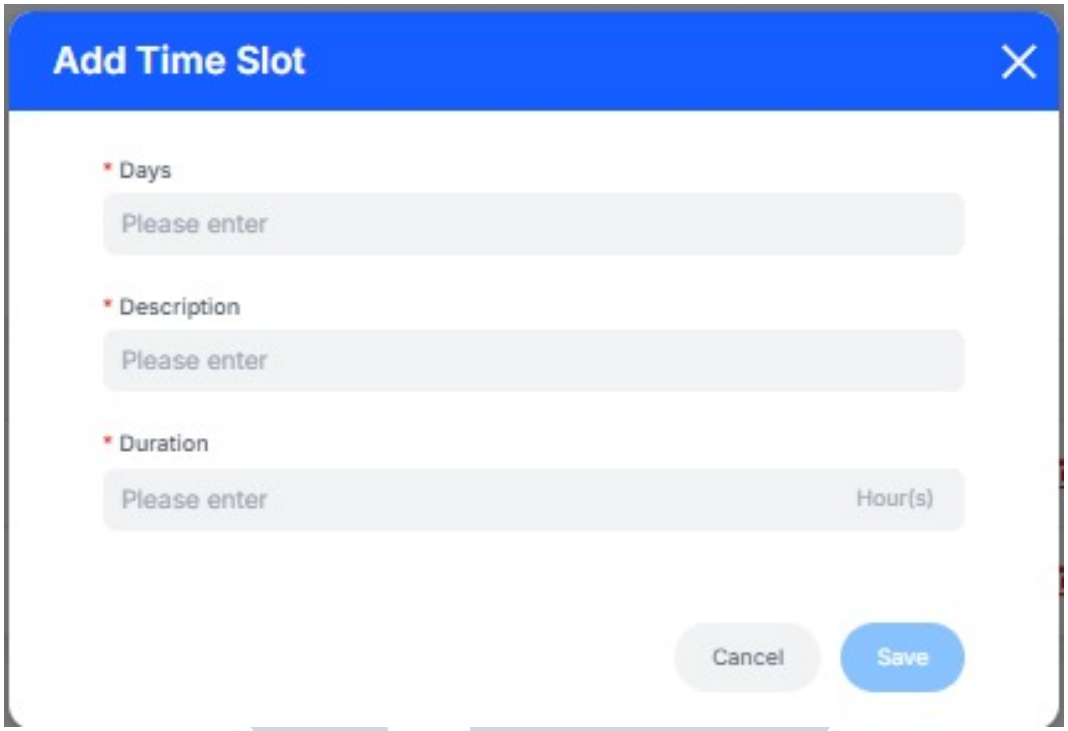
Enhancement	**Langkah Pengujian**	**Hasil**
Penambahan icon ”x” (close)	1. Pada menu utama, pilih *Learning Activities*. 2. Pilih salah satu sub menu pada *Learning Activities*. 3. Klik tombol *View*. 4. Pada halaman *Item Details*, masuk ke tab lain dan pastikan ketika mode edit, icon “x” (close) muncul sesuai kebutuhan.	Sesuai
Penambahan Tab EAP	1. Pilih menu *People*. 2. Masuk ke sub menu *User*. 3. Klik tombol *View*. 4. Pada halaman *User Profile*, buka tab *EAP* dan pastikan sesuai dengan requirement.	Sesuai
Penambahan Field Textbox menjadi multiple input	1. Pilih *Learning Activities*. 2. Masuk ke sub menu *Items*. 3. Pada selection screen, masukkan input search lalu tekan enter. 4. Ulangi hingga sistem menerima beberapa input (multiple). 5. Lakukan hal yang sama pada sub menu *Libraries*.	Sesuai

Enhancement	Langkah Pengujian	Hasil
Penyesuaian Behaviour halaman	1. Pilih *Learning Activities*. 2. Pilih salah satu sub menu terkait. 3. Pada selection screen, masukkan input pencarian lalu tekan enter. 4. Klik tombol *View*. 5. Pada halaman *Details*, klik tombol *Back* pada browser. 6. Pastikan sistem kembali ke halaman sebelumnya dengan data input tetap muncul. 7. Ulangi untuk seluruh menu terkait.	Sesuai
Penyesuaian Behaviour halaman	1. Pilih *Learning Activities*. 2. Pilih salah satu sub menu terkait. 3. Pada selection screen, masukkan input pencarian lalu tekan enter. 4. Klik tombol *View*. 5. Pada halaman *Details*, klik tombol *Breadcrumb*. 6. Pastikan sistem kembali ke halaman sebelumnya dengan data input tetap muncul. 7. Ulangi untuk seluruh menu terkait.	Sesuai
Penyesuaian Description character	1. Pilih *Learning Activities*. 2. Pilih sub menu *Items*. 3. Klik tombol *Add New*. 4. Pada halaman *Add New Item*, cek field Description apakah sudah 4000 karakter atau belum.	Sesuai

Enhancement	Langkah Pengujian	Hasil
Penyesuaian Description character	1. Pilih *Learning Activities*. 2. Pilih sub menu *Items*. 3. Klik tombol *View*. 4. Pada halaman *Details*, pindah ke tab *Agenda Template* 5. Klik button *Add New*. 6. Cek apakah field description memiliki limit 300 karakter.	Sesuai
Penyesuaian nama	1. Pilih *My Learning*. 2. Lalu cek apakah *Featured Learning* sudah berubah menjadi *Learning Recommendation*.	Sesuai
Penyesuaian table list	1. Pilih *Learning Activities*. 2. Pilih sub menu *Items*. 3. Lalu cek di table list apakah field description sudah tidak ada	Sesuai
Penyesuaian behaviour (*Items*)	1. Pilih *Learning Activities*. 2. Pilih sub menu *Items*. 3. Klik tombol *Search*. 4. Pilih berapa data yang ingin ditampilkan per page lalu klik 5. Pastikan setelah klik pagination akan auto scroll ke bawah tabel.	Sesuai
Penyesuaian behaviour (*Class*)	1. Pilih *Learning Activities*. 2. Pilih sub menu *Class*. 3. Klik tombol *Search*. 4. Pilih berapa data yang ingin ditampilkan per page lalu klik 5. Pastikan setelah klik pagination akan auto scroll ke bawah tabel.	Sesuai

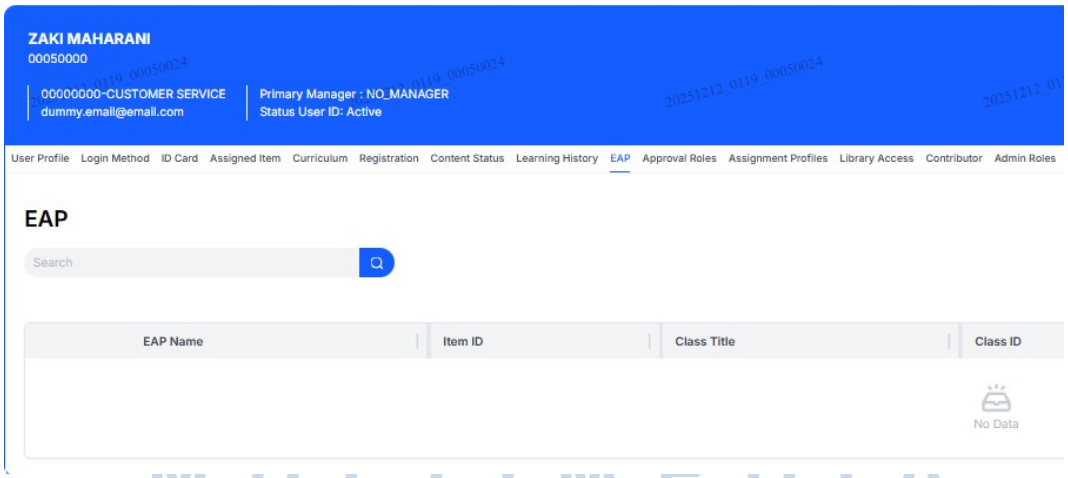
Enhancement	Langkah Pengujian	Hasil
Penyesuaian behaviour (*Libraries*)	1. Pilih *Learning Activities*. 2. Pilih sub menu *Libraries*. 3. Klik tombol *Search*. 4. Pilih berapa data yang ingin ditampilkan per page lalu klik 5. Pastikan setelah klik pagination akan auto scroll ke bawah tabel.	Sesuai
Penyesuaian nama (*Libraries*)	1. Pilih *Learning Activities*. 2. Pilih sub menu *Libraries*. 3. Klik tombol *View*. 4. Pada halaman *Details*, cek ke tab *Items* apakah di field Highlight as Featured Learning sudah berubah menjadi Learning Recommendation	Sesuai
Penyesuaian nama (*Items*)	1. Pilih *Learning Activities*. 2. Pilih sub menu *Items*. 3. Klik tombol *View*. 4. Pada halaman *Details*, cek ke tab *Libraries* apakah di field Highlight as Featured Learning sudah berubah menjadi Learning Recommendation	Sesuai
Penyesuaian field	1. Pilih *Learning Activities*. 2. Pilih sub menu *Items*. 3. Klik tombol *View*. 4. Pada halaman *Details*, cek ke tab *Libraries* apakah field *Information Message* sudah di hapus	Sesuai

Berikut merupakan dokumentasi hasil pengujian PIR setelah perubahan atau enhancement diimplementasikan oleh developer. Setiap gambar menunjukkan bahwa fitur yang disesuaikan telah berfungsi sesuai kebutuhan.



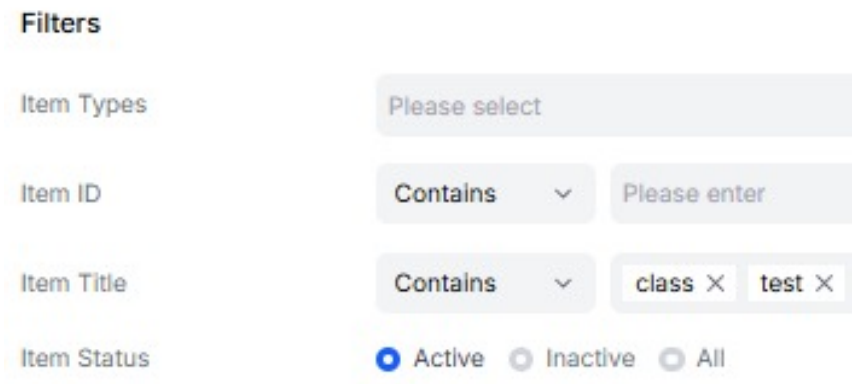
Gambar 3.25. Penambahan ikon close (x)

Gambar 3.25 menunjukkan penambahan ikon *close (x)* yang berfungsi untuk memudahkan pengguna menutup tampilan dengan lebih intuitif.



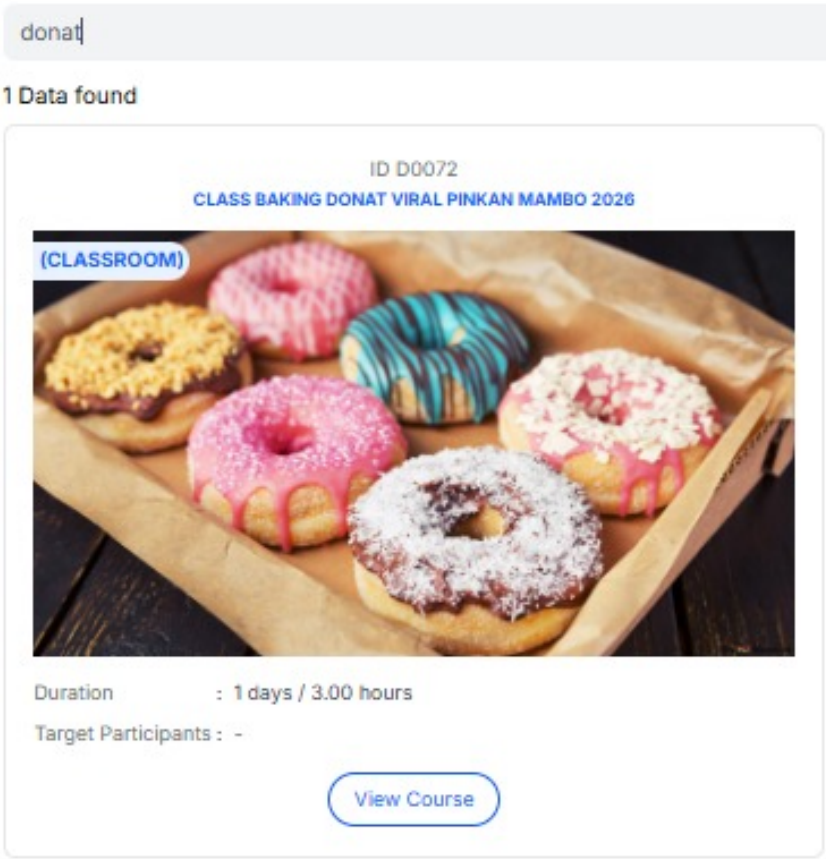
Gambar 3.26. Penambahan tab EAP

Gambar 3.26 menunjukkan penambahan tab *EAP* sebagai pemisahan konten agar struktur halaman lebih jelas.



Gambar 3.27. Penambahan fitur multiple input

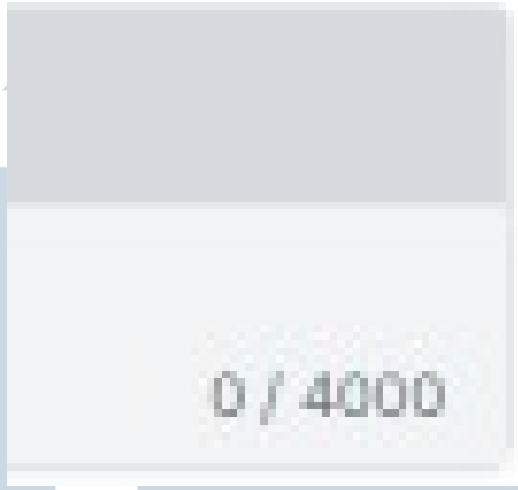
Gambar 3.27 menunjukkan penambahan fitur *multiple input* yang memungkinkan pengguna memasukkan lebih dari satu data dalam satu proses.



Gambar 3.28. Penyesuaian behaviour halaman

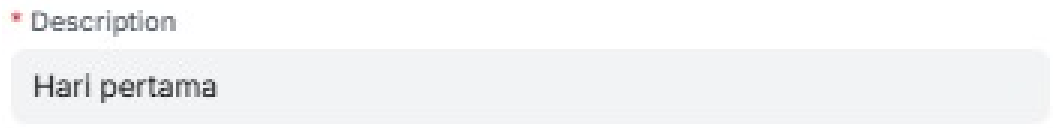
Gambar 3.28 menunjukkan penyesuaian perilaku halaman agar alur interaksi

pengguna lebih konsisten.



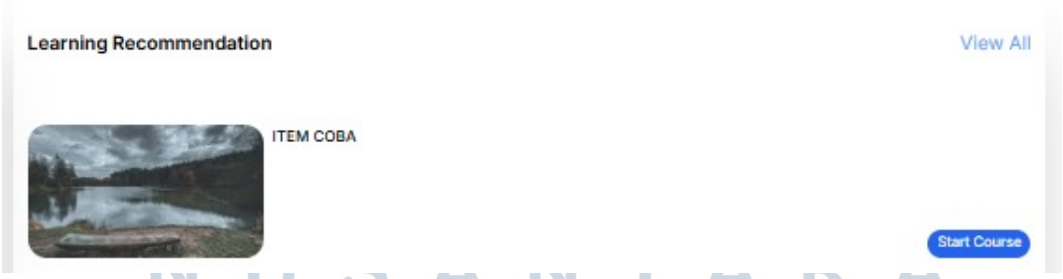
Gambar 3.29. Penyesuaian panjang karakter description

Gambar 3.29 menunjukkan penyesuaian batas maksimal karakter pada field *description* agar sesuai dengan kebutuhan input data.



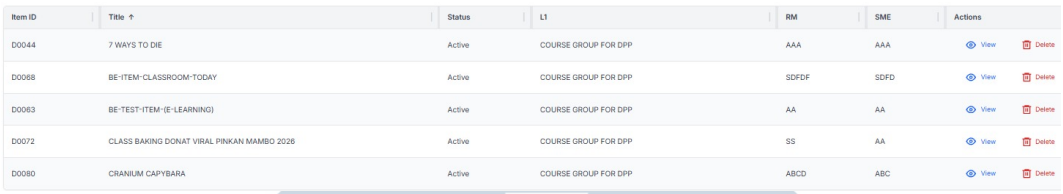
Gambar 3.30. Penyesuaian description alternatif

Gambar 3.30 menunjukkan penyesuaian panjang karakter *description* pada kondisi tertentu agar tetap konsisten dengan aturan sistem.



Gambar 3.31. Penyesuaian penamaan

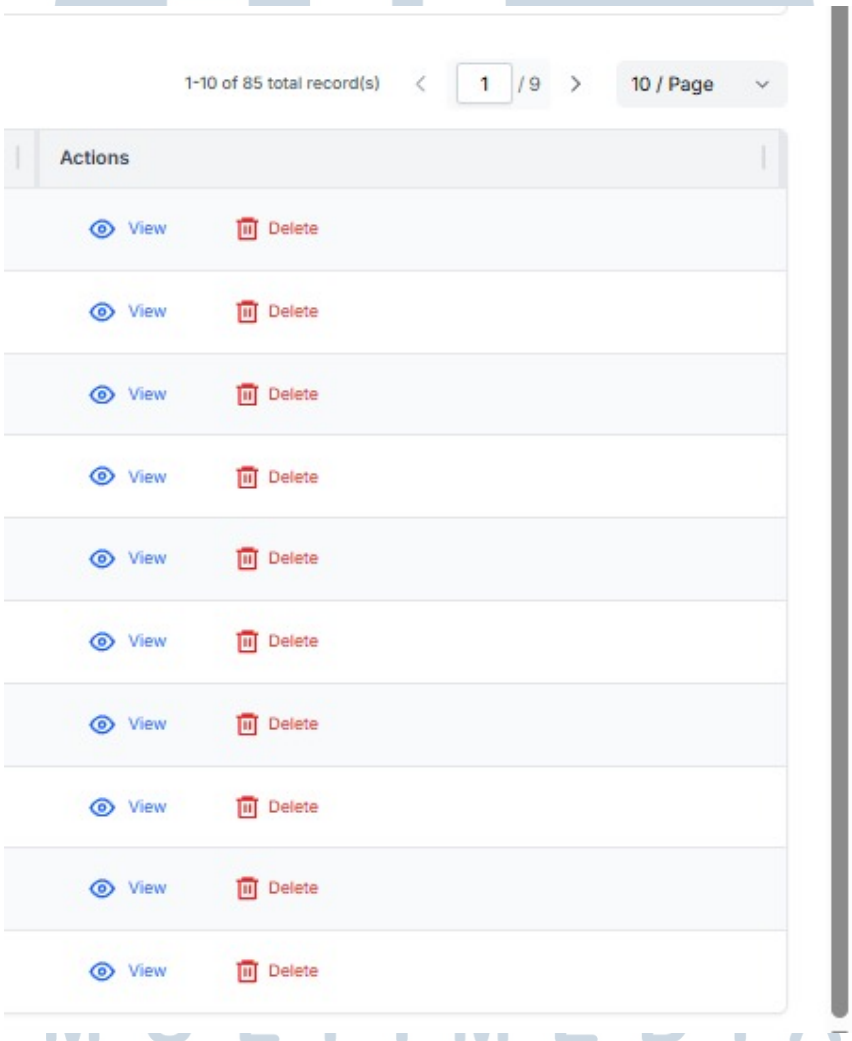
Gambar 3.31 menunjukkan penyesuaian penamaan agar lebih sesuai dengan requirement.



Item ID	Title ↑	Status	L1	RM	SME	Actions
D0044	7 WAYS TO DIE	Active	COURSE GROUP FOR DPP	AAA	AAA	View Delete
D0068	BE-ITEM-CLASSROOM-TODAY	Active	COURSE GROUP FOR DPP	SDFDF	SDFD	View Delete
D0063	BE-TEST-ITEM-(E-LEARNING)	Active	COURSE GROUP FOR DPP	AA	AA	View Delete
D0072	CLASS BAKING DONAT VIRAL PINKAN MAMBO 2026	Active	COURSE GROUP FOR DPP	SS	AA	View Delete
D0080	CRANIUM CAPYBARA	Active	COURSE GROUP FOR DPP	ABCD	ABC	View Delete

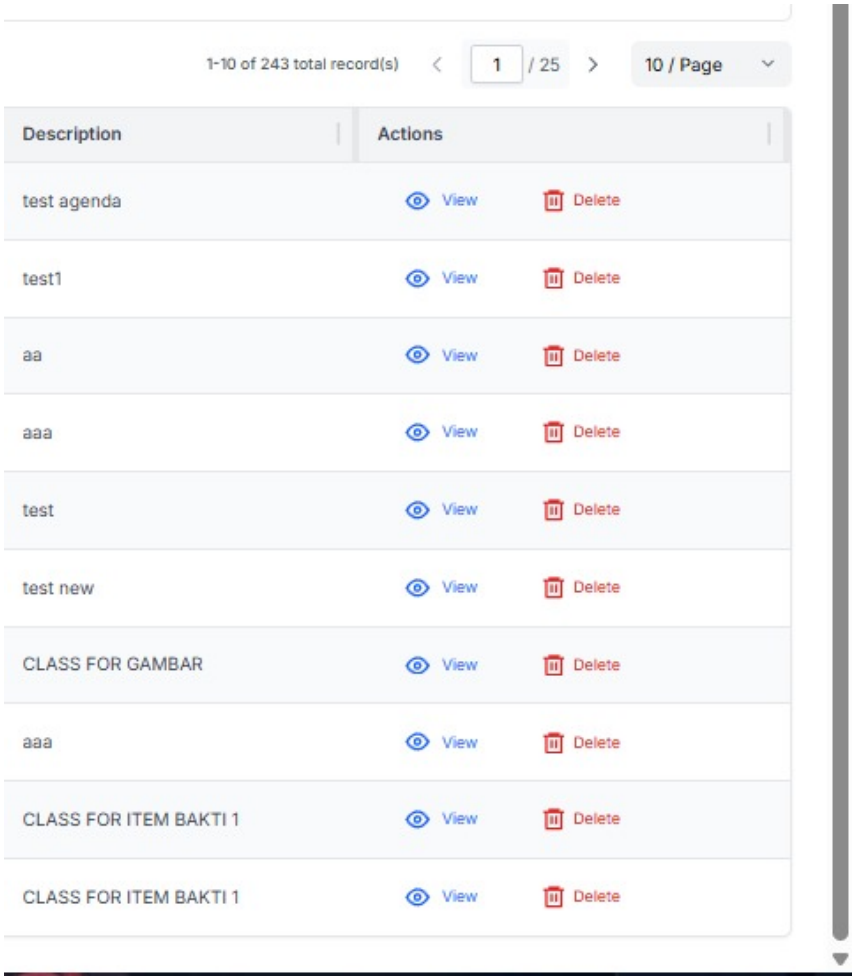
Gambar 3.32. Penyesuaian tampilan table list

Gambar 3.32 menunjukkan penyesuaian tampilan tabel agar data lebih mudah dibaca.



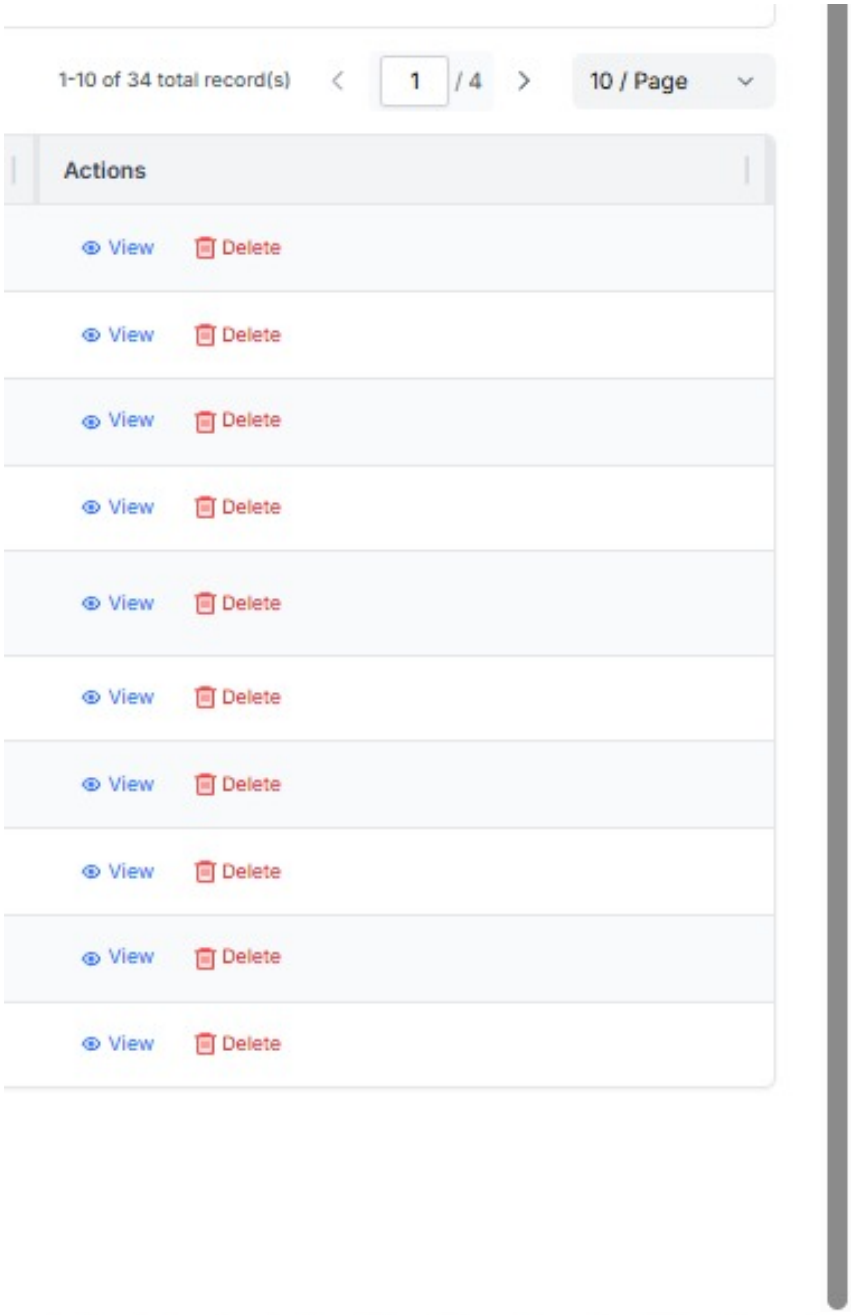
Gambar 3.33. Penyesuaian behaviour items

Gambar 3.33 menunjukkan perubahan perilaku pada data *items* agar proses pengelolaan data berjalan optimal.



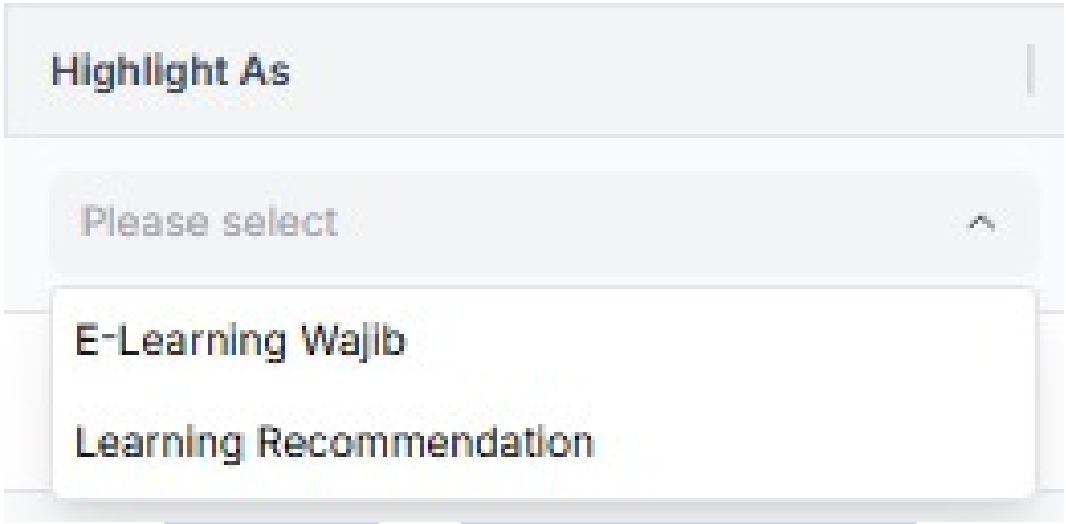
Gambar 3.34. Penyesuaian behaviour class

Gambar 3.34 menunjukkan penyesuaian perilaku pada data *class* agar selaras dengan kebutuhan sistem.



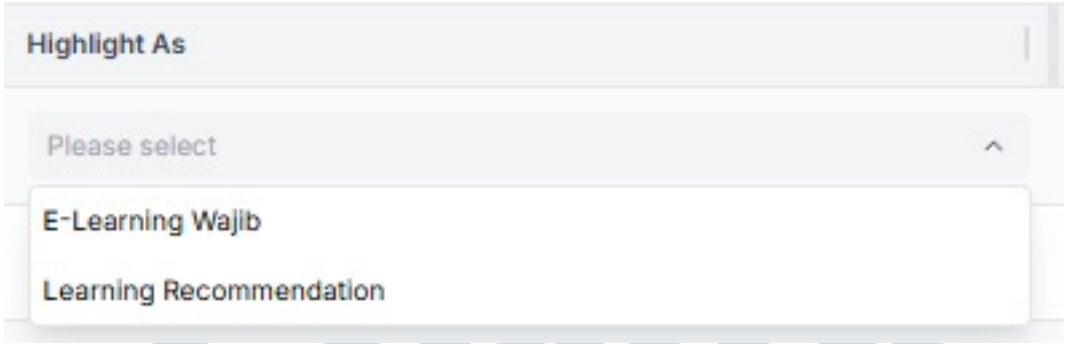
Gambar 3.35. Penyesuaian behaviour libraries

Gambar 3.35 menunjukkan penyesuaian perilaku pada modul *libraries* agar proses pengelolaan data lebih konsisten.



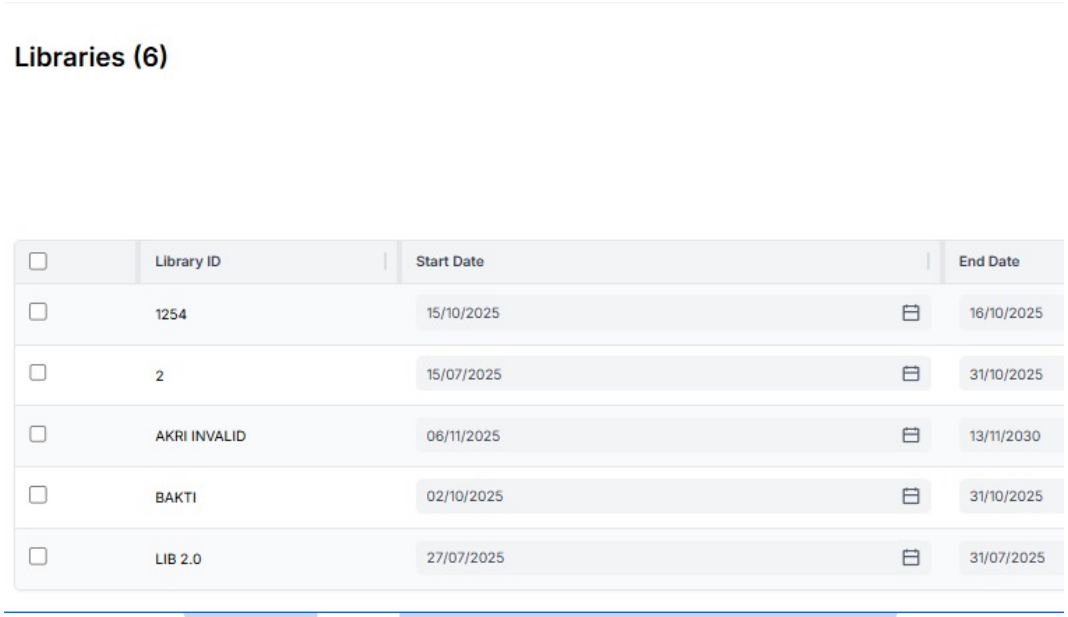
Gambar 3.36. Penyesuaian nama libraries

Gambar 3.36 menunjukkan penyesuaian penamaan pada data *libraries* agar sesuai dengan standar sistem.



Gambar 3.37. Penyesuaian nama items

Gambar 3.37 menunjukkan penyesuaian penamaan pada data *items* untuk meningkatkan kejelasan informasi.



Gambar 3.38. Penyesuaian field

Gambar 3.38 menunjukkan penyesuaian field input agar sesuai dengan kebutuhan data dan aturan validasi sistem.

Melalui pengujian PIR ini, seluruh perubahan dan enhancement yang dilakukan telah tervalidasi dengan baik serta tidak memengaruhi fungsi lain di dalam aplikasi.

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kerja praktik, ada beberapa kendala. Kendala tersebut di antaranya:

1. Beberapa pekerjaan memerlukan acuan *best practice* tertentu, sehingga membutuhkan waktu tambahan untuk mempelajari standar implementasi yang tepat.
2. Terdapat bagian requirement yang kurang jelas atau masih bersifat ambigu, sehingga berpotensi menimbulkan kesalahan dalam proses pengembangan apabila tidak dikonfirmasi lebih lanjut.

Untuk mengatasi kendala tersebut, terdapat solusinya, yaitu:

1. Melakukan diskusi secara rutin dengan anggota tim, mentor, maupun senior untuk memperoleh arahan mengenai pendekatan teknis yang sesuai.

2. Melakukan klarifikasi langsung kepada user atau pihak terkait guna memastikan interpretasi requirement sudah tepat dan mengurangi risiko kesalahan dalam implementasi.