BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama pelaksanaan kerja magang di PT Kompas Media Nusantara, penempatan dilakukan pada Departemen *Technology, Product, & Data* (TPD) dengan peran sebagai *Software Engineer*. Fokus utama pekerjaan adalah pengembangan dan penyempurnaan fitur *Payment* CC pada modul *Purchase Order Detail* di GMMS (*GoMED Material Management System*) berbasis ASP.NET MVC, dengan tujuan agar proses pencatatan dan konfirmasi pembayaran dapat terintegrasi secara optimal dengan alur kerja internal perusahaan.

Kegiatan magang berada di bawah bimbingan Bapak Rokhmat PN selaku *Lead Developer* sekaligus mentor teknis, serta dikoordinasikan bersama Bapak Rudyanto selaku *Product Manager*. Bimbingan yang diberikan mencakup penentuan ruang lingkup pekerjaan, penetapan prioritas pengembangan, serta evaluasi hasil implementasi agar selaras dengan kebutuhan bisnis dan standar teknis yang berlaku di lingkungan pengembangan sistem GMMS.

Dalam pelaksanaan pengembangan sistem, tim Departemen *Technology, Product, & Data* menerapkan pendekatan pengembangan perangkat lunak berbasis *Agile*. Pendekatan ini digunakan untuk mendukung kebutuhan bisnis yang dinamis serta memungkinkan penyesuaian fitur secara bertahap berdasarkan hasil evaluasi dan umpan balik yang diperoleh selama proses pengembangan. Pola kerja yang diterapkan bersifat iteratif dan inkremental, di mana pengembangan dilakukan melalui pembagian tugas dalam periode kerja singkat yang menyerupai *sprint*.

Secara konseptual, penerapan pendekatan *Agile* tersebut mengadopsi praktik kerja yang sejalan dengan kerangka *Scrum*, dengan penyesuaian terhadap struktur organisasi internal perusahaan. *Product Manager* berperan dalam menyampaikan kebutuhan bisnis, prioritas fitur, dan kriteria keberhasilan pengembangan, sedangkan *Lead Developer* bertindak sebagai pengarah teknis, pengambil keputusan teknis utama, serta penanggung jawab terhadap kualitas implementasi. Peran *Software Engineer* (Magang) berfokus pada pelaksanaan pengembangan fitur, perbaikan *bug*, serta pengujian fungsional sesuai dengan tugas yang telah ditetapkan pada setiap iterasi pengembangan.

Secara struktural, kedudukan dalam tim pengembangan diatur sebagai berikut:

1. *Lead Developer* (Mentor): memberikan arahan teknis, melakukan *code review*, serta mengevaluasi kualitas dan konsistensi implementasi.
2. *Product Manager*: menyampaikan kebutuhan bisnis, aturan proses, serta kriteria penerimaan fitur yang dikembangkan.
3. *Software Engineer* (Magang): melaksanakan pengembangan fitur, perbaikan *bug*, serta pengujian sistem sesuai dengan arahan dan prioritas yang ditetapkan.

Alur koordinasi kerja diterapkan secara sistematis untuk memastikan setiap tugas dapat diselesaikan secara efektif, terukur, dan terdokumentasi dengan baik. Tahapan koordinasi kerja meliputi:

1. *Briefing* dan Penetapan Tugas
    a) *Briefing* awal dilakukan untuk membahas ruang lingkup pekerjaan, dependensi antar fitur, serta target pengembangan fitur *Payment* CC.
    b) Spesifikasi yang dibahas mencakup integrasi dengan status dokumen (misalnya *New*, *Sent*, *Need Revision*, *Need Approval*, dan *Closed*), aturan aktivasi tombol aksi, serta konsistensi tampilan antarmuka dengan modul lain pada GMMS.
2. Pelaksanaan Pengembangan
    a) Implementasi dilakukan pada sisi *controller*, *model*, dan *view*, termasuk penyesuaian kueri terhadap tabel terkait seperti *PaymentTable*, *PaymentInvoiceTable*, dan *PaymentXDocument*.
    b) Kendala teknis yang muncul selama proses pengembangan dikomunikasikan secara langsung melalui kanal koordinasi tim untuk memperoleh solusi dan keputusan teknis dalam waktu singkat.
3. Evaluasi dan Revisi
    a) Setiap hasil pekerjaan diserahkan untuk dilakukan proses *review* yang mencakup aspek fungsionalitas, keandalan sistem, serta kesesuaian dengan aturan bisnis.
    b) Revisi dan perbaikan dilakukan berdasarkan umpan balik yang diterima, termasuk penajaman validasi data, penyesuaian tipe data, serta penyempurnaan antarmuka pengguna.

4. Finalisasi dan Implementasi

    a) Pengujian dilakukan pada tingkat unit dan integrasi, khususnya pada skenario status dokumen yang relevan dengan modul *Payment* CC.

    b) Setelah seluruh kriteria penerimaan terpenuhi, perubahan diintegrasikan ke lingkungan pengujian dan disiapkan untuk proses *deployment* sesuai dengan prosedur yang berlaku di tim pengembangan.

Koordinasi rutin selama proses pengembangan dilakukan melalui beberapa mekanisme berikut:

1. *Meeting* berkala untuk membahas pembaruan progres, penyesuaian prioritas pengembangan, serta mitigasi risiko teknis.
2. Komunikasi harian melalui media *chat* yang berfungsi sebagai sarana koordinasi singkat dan menyerupai praktik *daily stand-up* dalam pengembangan berbasis *Agile*.
3. Pendampingan teknis melalui diskusi terarah dan proses *code review* untuk menjaga standar kualitas implementasi serta keselarasan arsitektur sistem.

Dengan pengaturan kedudukan, penerapan pendekatan pengembangan berbasis *Agile*, serta alur koordinasi kerja yang terstruktur, proses pengembangan fitur *Payment* CC pada GMMS dapat berjalan secara iteratif, terarah, dan terdokumentasi dengan baik, sehingga mampu memenuhi kebutuhan bisnis serta standar pengembangan sistem di lingkungan Departemen *Technology, Product, & Data*.

3.2 Tugas yang Dilakukan

Selama periode pelaksanaan magang di PT Kompas Media Nusantara, kegiatan dilaksanakan pada Departemen *Technology, Product, & Data* dengan tugas dan tanggung jawab sebagai *Software Engineer*. Tugas utama yang dilakukan berfokus pada pengembangan dan penyempurnaan fitur *Payment* CC dalam modul *Purchase Order* (PO) pada sistem GMMS berbasis ASP.NET MVC. Fitur ini dikembangkan untuk mendukung proses pencatatan dan konfirmasi pembayaran menggunakan kartu kredit yang terhubung langsung dengan dokumen PO.

Secara umum, pekerjaan yang dilakukan meliputi analisis kebutuhan sistem, perancangan alur kerja, serta implementasi antarmuka dan logika pemrosesan data.

Kegiatan yang dikerjakan mencakup pembangunan hubungan antara halaman PO *Detail* dan modul *Payment* CC, sehingga sistem dapat secara otomatis membuat data pembayaran berdasarkan informasi PO yang tersedia. Selain itu, pengembangan juga dilakukan pada halaman *Payment CC Index* dan *Payment Detail* yang berfungsi menampilkan daftar transaksi pembayaran, dilengkapi dengan fitur pencarian, *filter* tanggal, serta pengaturan jumlah baris untuk mendukung kemudahan pengguna dalam menelusuri data.

Dalam proses pengembangan, pekerjaan juga mencakup integrasi dengan API *(Application Programming Interface)* internal dan basis data SQL *Server* untuk memastikan setiap transaksi tersimpan dan dapat diakses secara konsisten. Tahapan pengujian dilakukan untuk memastikan bahwa fungsi-fungsi seperti penambahan, pengeditan, dan penghapusan *invoice*; penambahan dan penghapusan *attachment* pada masing-masing *invoice*; pencetakan dokumen pembayaran; serta validasi status dokumen berjalan sesuai ketentuan sistem.

Melalui kegiatan tersebut, kontribusi diberikan dalam peningkatan efisiensi proses bisnis perusahaan melalui otomasi dan penyelarasan data antara modul PO dan *Payment* CC, sehingga proses pencatatan pembayaran menjadi lebih cepat, terpusat, dan akurat.

3.3 Uraian Pelaksanaan Magang

Adapun uraian kegiatan magang yang dilakukan di PT Kompas Media Nusantara disajikan secara terstruktur berdasarkan pembagian per minggu, sebagaimana dijelaskan pada tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Mengikuti kegiatan *onboarding The Intern Batch 3–2025*, melakukan instalasi dan konfigurasi awal *framework* Nuxt, mempelajari konsep dasar *view entry point*, *components*, *pages*, dan *layouts*, serta menyelesaikan administrasi magang dan penyesuaian lingkungan pengembangan.

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

Minggu Ke -	Pekerjaan yang dilakukan
2	Mempelajari modul *Assets*, *Public*, dan *Styling* pada Nuxt, kemudian memulai pengembangan peta interaktif berbasis *Leaflet* untuk data demonstrasi dan kerusuhan menggunakan JSON API Unimind, termasuk *refactor* UI, penguatan pipeline transformasi data, dan penerapan *marker clustering*.
3	Menyempurnakan peta Unimind dengan penyesuaian *layout* dan *responsive behavior*, *hardening* pipeline (deduplikasi titik, *spiderfy*, *logging*), serta mulai berorientasi pada sistem GMMS dengan mempelajari struktur proyek API dan pengujian *endpoint* melalui Swagger UI.
4	Melakukan *build* dan *run* proyek GMMS di *localhost*, mengatasi error koneksi database hingga login berhasil, mempelajari pola *routing* ASP.NET *MVC*, serta memulai audit dan implementasi tombol *Payment* CC di halaman PO *Detail*.
5	Menyempurnakan logika *enable/disable* tombol aksi, khususnya *Payment* CC, agar sesuai aturan *workstate* dan alur bisnis pada modul PO, serta mengikuti kegiatan *gathering* dan *onboarding offline* tim Sobat HR Kompas.
6	Mengembangkan `PaymentController` pada proyek API untuk mengambil data dari `dbo.PaymentList`, membuat `PaymentController` pada proyek *Website*, mengintegrasikan halaman `UI_PaymentCCList.cshtml`, melakukan *debugging* tampilan, serta menambahkan dropdown *Rows* dan *setup OpenVPN* untuk akses jaringan internal.
7	Melanjutkan pengujian dan penyesuaian halaman *Payment* CC, memverifikasi `/gmmsapi/Payment/list` dan `/gmmsapi/Payment/save`, menyiapkan dan menguji `PaymentModel`, `PaymentViewModel`, dan `PaymentFilterModel`, serta merapikan struktur *Model–View–Controller* agar data dari API dapat ditampilkan dengan benar.

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

Minggu Ke -	Pekerjaan yang dilakukan
8	Mengembangkan fitur *Payment* CC *Detail* dengan menyesuaikan model dan *controller* terhadap tabel *PaymentTable*, memperkaya `PaymentModel` (detail *vendor* dan penerima), memperbaiki *parsing* JSON, serta mengintegrasikan prosedur `PaymentImportSave` ke dalam alur aplikasi.
9	Menyusun *layout Payment Detail* (Request Info, PO Info, *Payment Term*, *Items*, dan *Attachments*), mengintegrasikan API `/Payment/invoiceList` menggunakan AJAX, membangun halaman *Payment Add/Edit* dan *Payment Print*, serta menyempurnakan fitur *Add Invoice* (dropdown UOM dan *currency*, validasi, `PaymentInvoiceSave`, dan *delete invoice*).
10	Merevisi halaman *Payment Print* agar seragam dengan *Print PO*, menyelaraskan alur unggah *attachment* di *Payment Detail* dengan modul PO, mengidentifikasi kebutuhan prosedur `PaymentDocumentSave` dan `PaymentDocumentDelete`, serta menyesuaikan modul *Payment* dan tabel `PaymentXDocument` agar siap diintegrasikan dengan prosedur tersebut.
11	Melakukan *debugging* fitur *Add Attachment* di *Payment Detail* (perbaikan *documentRead*, validasi `PaymentInvoiceID`, dan pengiriman token ke API *documentSave*), menguji penyimpanan dokumen melalui `PaymentDocumentSave`, memperbaiki fungsi unggah dan *delete attachment*, serta memastikan daftar *attachment* dapat ditampilkan sesuai struktur baru.
12	Menstabilkan fitur *attachment* dan *invoice* pada modul *Payment* CC, termasuk penyesuaian tampilan *attachment* per *invoice*, penampilan *Next Payment* dari kolom `NextSchedule`, penyempurnaan fitur *edit invoice*, perbaikan URL *download attachment*, penambahan modal konfirmasi, serta penyelarasan dropdown dan mekanisme *select/unselect invoice*.

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

Minggu Ke -	Pekerjaan yang dilakukan
13	Menambahkan dropdown pemilihan *invoice* di modal *Add Attachment* beserta peringatan jika belum dipilih, merapikan validasi dan tampilan pesan error, menambahkan tombol *reset* pada modal *Add/Edit Invoice*, serta melakukan serangkaian verifikasi dan penyesuaian alur *download attachment*, termasuk analisis kebutuhan dan struktur path file dari database.
14	Melakukan *review* modul *Payment* CC dengan fokus pada kesiapan fitur *download attachment*, termasuk pengecekan konsistensi data, kestabilan UI, serta validasi alur *Add/Edit Invoice*.
15	Melakukan pengecekan struktur fungsi *download attachment*, meninjau kembali integrasi halaman *Payment Edit*, serta memastikan konsistensi tampilan dan navigasi pada halaman *Payment List* dan *Payment Detail*.
16	Melakukan pengecekan ulang modul *Payment* CC secara fungsional, termasuk validasi input, konsistensi tampilan, serta peninjauan alur proses dari sisi pengguna.
17	Melakukan pengecekan dan revisi tampilan halaman *Payment Edit* dan *Payment Print*, serta melakukan konsultasi terkait penyesuaian alur sistem sesuai dengan proses bisnis.
18	Melakukan penyesuaian akhir modul *Payment* CC, termasuk penerapan kondisi *read only* sementara dan pengecekan final kestabilan sistem menjelang penutupan magang.

Dari uraian kegiatan tersebut, pelaksanaan kerja magang berlangsung secara bertahap. Kegiatan diawali dengan proses *onboarding*, penyiapan lingkungan kerja, serta pengenalan teknologi yang digunakan, termasuk Nuxt dan pengembangan peta interaktif berbasis *Leaflet*. Selanjutnya, fokus kegiatan beralih pada pengembangan dan penyempurnaan modul *Payment* CC pada sistem GMMS, yang mencakup implementasi dan penyesuaian *controller*, model, dan *view*, integrasi API, pengelolaan data *invoice* dan *attachment*, serta penyesuaian tampilan dan alur sistem. Seluruh rangkaian kegiatan tersebut menunjukkan proses adaptasi teknis yang bertahap serta pemahaman terhadap alur bisnis dan standar pengembangan

yang diterapkan di lingkungan industri.

3.3.1 User Requirement dan Perangkat Penunjang

Pada subbab ini dijelaskan kebutuhan pengguna (*user requirement*) serta perangkat penunjang yang digunakan dalam pengembangan fitur *Payment* CC pada sistem GMMS. Fitur ini berfungsi untuk mendukung proses pencatatan, verifikasi, dan pelacakan pembayaran yang berasal dari modul PO agar seluruh transaksi tercatat secara akurat, transparan, dan terintegrasi dengan sistem perusahaan.

A User Requirement

Analisis kebutuhan pengguna dilakukan untuk memastikan bahwa fitur *Payment* CC dapat berfungsi secara menyeluruh mulai dari pemicu aksi pada halaman PO *Detail* hingga proses penyimpanan dan penampilan data pada halaman *Payment Detail*. Hasil identifikasi kebutuhan pengguna adalah sebagai berikut:

1. Pemicu Aksi *Payment* CC dari PO *Detail*: Sistem menyediakan tombol *Payment* CC pada halaman PO *Detail* untuk memproses pembayaran berbasis kartu kredit. Tombol ini hanya aktif ketika status dokumen (*Workstate*) berada pada kondisi *Approval*. Setelah pengguna memilih tombol tersebut dan mengisi data pengulangan pembayaran (*Recurring Type* dan *Recurring Every*), sistem secara otomatis membentuk data *Payment* baru yang terhubung dengan PO terkait.

2. Halaman *Payment* CC *List*: Halaman ini menampilkan daftar seluruh data *Payment* yang tersimpan pada sistem. Pengguna dapat melakukan pencarian berdasarkan kata kunci, memfilter berdasarkan rentang tanggal, serta mengatur jumlah baris data per halaman. Data diambil melalui *endpoint* `/Payment/list` yang terhubung ke fungsi `dbo.PaymentList`.

3. Halaman *Payment Detail*: Setelah pengguna memilih salah satu data pada daftar *Payment*, sistem akan menampilkan halaman *Payment Detail*. Halaman ini menyajikan informasi *header* meliputi *Request Info*, PO *Info*, dan *Payment Term Info* serta daftar *invoice* terkait yang diperoleh melalui `/Payment/invoiceList`. Pada halaman ini, pengguna dapat menambahkan, mengedit, maupun menghapus *invoice*, menambahkan *attachment* per

invoice melalui tombol aksi yang tersedia, serta melakukan pengeditan data pembayaran dan mencetak dokumen.

4. Pengelolaan Data *Invoice*: Sistem menyediakan *form Add Invoice* untuk menambahkan data transaksi pembayaran kartu kredit. Pada *form* ini, pengguna dapat mengisi sejumlah informasi, termasuk *Invoice Number*, *Invoice Date*, *Payment Date*, *Card Number*, *Card Holder*, *Bank Issuer*, *Quantity*, *UOM*, *Currency*, *Total Price*, dan *Receipt Number*. Seluruh isian wajib mengikuti validasi yang telah disediakan agar data tersimpan dengan benar. Setelah pengguna menekan tombol *Save*, data akan dikirim melalui `/Payment/invoiceSave` untuk kemudian disimpan ke dalam `PaymentInvoiceTable`. Selain menambah data, pengguna juga dapat menghapus baris *invoice* melalui tombol *DEL* yang memanggil `/Payment/invoiceDelete`. Tersedia pula tombol *Reset* untuk mengosongkan seluruh isian *form* sebelum disimpan.

5. Fitur Pencetakan Dokumen: Pengguna dapat mencetak dokumen pembayaran melalui tombol *Print*, yang akan menampilkan data *Payment* dalam *format* siap cetak sesuai standar GMMS. Data yang ditampilkan telah melalui proses pengambilan dan validasi di sisi *server*, sehingga informasi *invoice*, *header* pembayaran, dan rincian transaksi yang tercetak tetap konsisten dan akurat.

6. Konsistensi dan Tampilan Antarmuka: Antarmuka *Payment* CC dirancang mengikuti standar tampilan modul lain pada GMMS, mencakup konsistensi gaya tabel, warna status dokumen, serta penempatan tombol aksi. Sistem juga menampilkan informasi jumlah data yang tersedia, termasuk *RowCount* dan *PageCount*, sehingga pengguna dapat memantau jumlah transaksi yang ditampilkan dengan lebih jelas.

B Perangkat Penunjang

Dalam proses pengembangan dan pengujian fitur *Payment* CC, digunakan beberapa perangkat dan teknologi sebagai berikut:

1. *Framework* ASP.NET MVC: Versi 5.2.0 digunakan sebagai kerangka utama pengembangan sistem berbasis web dengan pola *Model–View–Controller*, yang memisahkan logika bisnis, antarmuka pengguna, dan kontrol alur data.

2. Bahasa Pemrograman C#: Digunakan pada sisi *server* untuk mengelola logika aplikasi, pemanggilan API internal, serta penghubung antara sistem dan basis data SQL *Server*.

3. Database Microsoft SQL *Server*: Berfungsi sebagai penyimpanan utama data *Payment* CC. Basis data digunakan untuk menyimpan informasi dokumen pembayaran, *invoice*, dan lampiran pendukung, serta diakses melalui fungsi dan prosedur tersimpan untuk kebutuhan pengambilan dan pengelolaan data.

4. HTML, CSS, JavaScript, dan jQuery: Digunakan untuk mengembangkan antarmuka pengguna pada halaman *Index*, *Detail*, *Edit*, dan *Print*, serta untuk menangani interaksi dinamis seperti pencarian, *filter* tanggal, dan *pagination*.

5. Bootstrap 3.4.1: Dipakai sebagai *front-end framework* untuk memastikan tampilan halaman responsif dan konsisten dengan modul lain di GMMS.

6. Visual Studio 2015 dan SQL Server Management Studio (SSMS): Visual Studio digunakan untuk pengembangan, kompilasi, dan *debugging* aplikasi ASP.NET MVC, sedangkan SSMS digunakan untuk pengujian kueri, pembuatan tabel, dan prosedur tersimpan.

7. Swagger UI: Dimanfaatkan sebagai alat bantu pengujian dan verifikasi layanan API internal yang digunakan dalam modul *Payment* CC.

8. OpenVPN: Digunakan untuk mengakses jaringan internal Kompas secara aman saat bekerja dari luar kantor (*remote access*) ke *server* pengembangan GMMS.

Dengan kebutuhan pengguna dan perangkat penunjang tersebut, pengembangan fitur *Payment* CC dapat dilaksanakan secara terarah dan terintegrasi, sehingga mendukung kelancaran proses bisnis dan meningkatkan efisiensi pengelolaan transaksi pembayaran dalam sistem GMMS.

3.3.2 Perancangan Sistem

Bagian ini menjelaskan perancangan sistem yang menjadi dasar pengembangan fitur *Payment* CC pada GMMS. Perancangan dilakukan dengan meninjau komponen-komponen utama, meliputi gambaran umum sistem, arsitektur aplikasi, alur proses bisnis, serta rancangan basis data yang memastikan integrasi berjalan konsisten dengan modul yang telah ada.

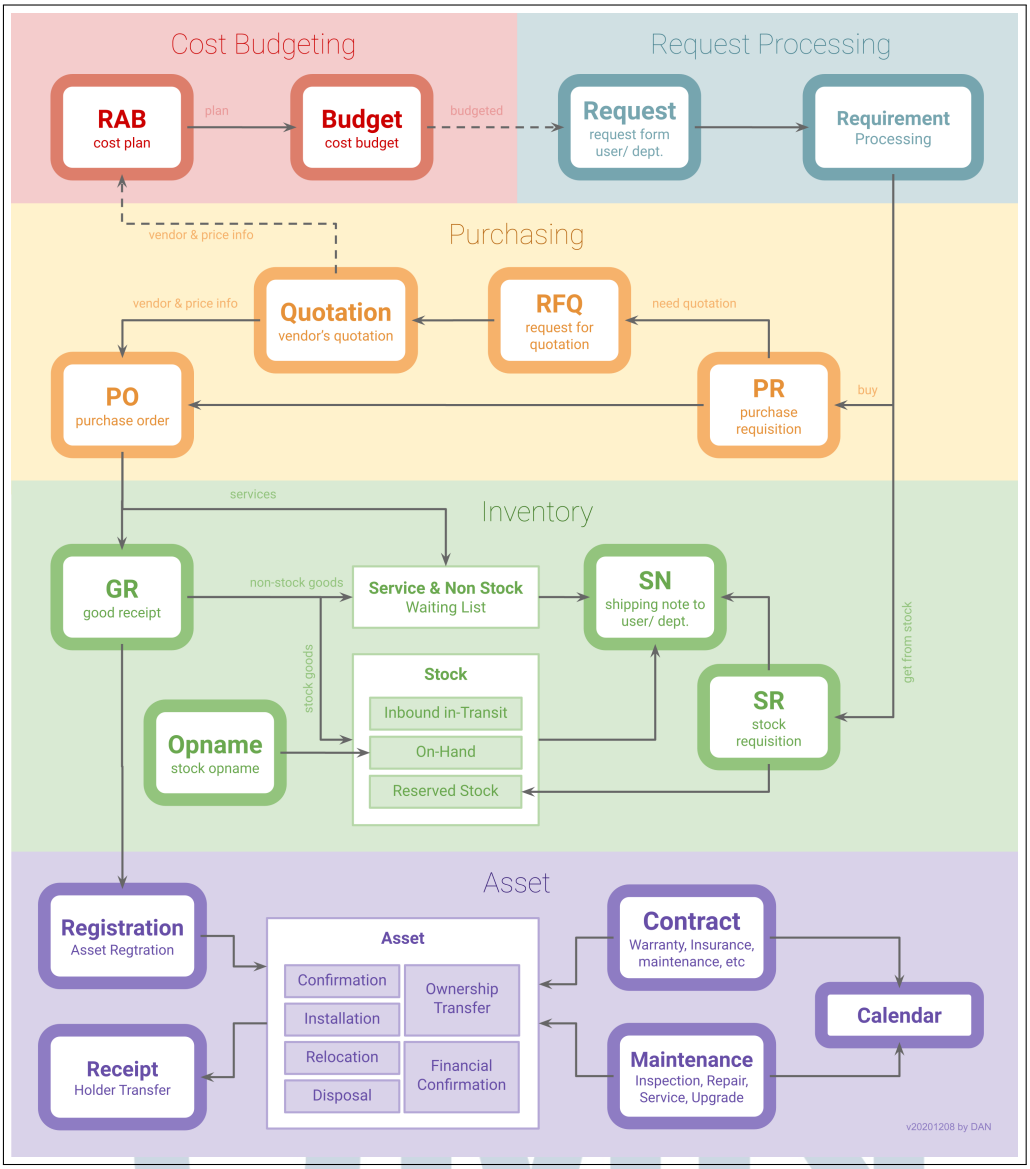
Fokus perancangan adalah memastikan fitur *Payment* CC terhubung dengan modul *Purchase Order* (PO) tanpa mengubah struktur inti GMMS. Tahapan perancangan disusun secara berurutan, dimulai dari gambaran umum sistem untuk menjelaskan konteks dan posisi fitur dalam ekosistem GMMS, diikuti arsitektur sistem yang menggambarkan alur komunikasi *model–view–controller* antara pengguna, *website*, API, dan basis data. Selanjutnya, perancangan alur proses bisnis disajikan melalui *sequence diagram* dan *activity diagram* sebagai representasi interaksi dan langkah operasional utama. Bagian terakhir adalah perancangan basis data yang mencakup penyusunan *class diagram* serta struktur *class* untuk mendefinisikan entitas, atribut, dan relasinya.

Seluruh tahapan disusun secara iteratif agar setiap komponen dapat diuji, disesuaikan, dan memenuhi standar pengembangan di Departemen *Technology, Product, & Data*. Perancangan ini menjadi landasan utama sebelum memasuki tahap implementasi, mencakup integrasi antarmuka, logika aplikasi, dan koneksi basis data.

A Gambaran Umum Sistem

GMMS merupakan sistem internal yang digunakan oleh PT Kompas Media Nusantara untuk mengelola kegiatan operasional perusahaan secara terpusat, mulai dari perencanaan anggaran hingga pengelolaan aset. GMMS terdiri atas beberapa modul utama yang saling terhubung, yaitu *Cost Budgeting*, *Request Processing*, *Purchasing*, *Inventory*, dan *Asset Management*. Fitur *Payment* CC dikembangkan sebagai bagian dari modul *Purchasing* untuk mencatat dan memantau pembayaran berbasis kartu kredit terhadap dokumen PO. Fitur ini berfungsi mempercepat proses verifikasi pembayaran serta memastikan seluruh transaksi tercatat secara akurat dan terpusat di sistem.

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 3.1. Gambaran umum alur sistem GMMS

Gambar 3.1 memperlihatkan alur utama sistem GMMS yang terdiri atas lima lapisan proses bisnis sebagai berikut:

1. Lapisan *Cost Budgeting*: Tahap awal perencanaan biaya yang dimulai dari pembuatan Rencana Anggaran Biaya (RAB) dan dilanjutkan ke tahap penyusunan *Budget*. Anggaran ini menjadi dasar dalam pembuatan permintaan pada tahap berikutnya.

2. Lapisan *Request Processing*: Tahapan pengajuan permintaan dari pengguna atau departemen terkait yang akan diteruskan ke bagian *Purchasing* untuk diproses lebih lanjut.

3. Lapisan *Purchasing*: Mencakup serangkaian dokumen seperti *Purchase Requisition* (PR), *Request for Quotation* (RFQ), *Quotation*, dan PO. Setelah PO diterbitkan, proses pembayaran dapat dilakukan melalui fitur *Payment* CC yang digunakan untuk mencatat transaksi berbasis kartu kredit serta menautkannya dengan data PO dan *vendor* terkait.

4. Lapisan *Inventory*: Mengelola penerimaan barang dan pencatatan stok melalui dokumen *Good Receipt* (GR), *Stock Opname*, serta *Shipping Note* (SN). Seluruh barang dan jasa yang diterima akan dicatat dan dipantau agar sesuai dengan data pesanan yang telah disetujui.

5. Lapisan *Asset Management*: Mengelola aset perusahaan mulai dari pendaftaran, pemeliharaan, perpindahan kepemilikan, hingga penghapusan aset. Dokumen yang digunakan antara lain *Registration*, *Contract*, dan *Maintenance* untuk memastikan seluruh aset tercatat secara lengkap.

Melalui integrasi antar-modul tersebut, GMMS membentuk satu kesatuan sistem pengelolaan yang mencakup seluruh tahapan kegiatan perusahaan, mulai dari perencanaan, pengadaan, penerimaan, hingga pencatatan aset. Fitur *Payment* CC memperluas fungsi modul *Purchasing* dengan menambahkan proses pencatatan dan verifikasi pembayaran, sehingga meningkatkan transparansi dan akurasi dalam pengelolaan transaksi keuangan perusahaan.

B Arsitektur Sistem

Arsitektur sistem GMMS dirancang dengan menerapkan pola MVC yang memisahkan logika aplikasi, pengelolaan data, dan tampilan pengguna. Pola ini diimplementasikan melalui dua proyek utama, yaitu *Website* (ASP.NET MVC) sebagai antarmuka pengguna dan API (*Web* API) sebagai pengelola logika bisnis serta jembatan ke basis data. Pendekatan ini memastikan struktur pengembangan yang modular, mudah diuji, dan dapat dikembangkan tanpa mengganggu modul lain dalam GMMS.

1. Lapisan *View*: Lapisan *View* berfungsi sebagai antarmuka pengguna pada modul *Payment* CC. Implementasi dilakukan melalui halaman *Index.cshtml*, *Detail.cshtml*, *AddEdit.cshtml*, dan *Print.cshtml* untuk menampilkan daftar data, detail dokumen, pengelolaan *Payment*, serta tampilan cetak. Setiap interaksi pengguna pada lapisan ini diteruskan ke *Controller* untuk diproses

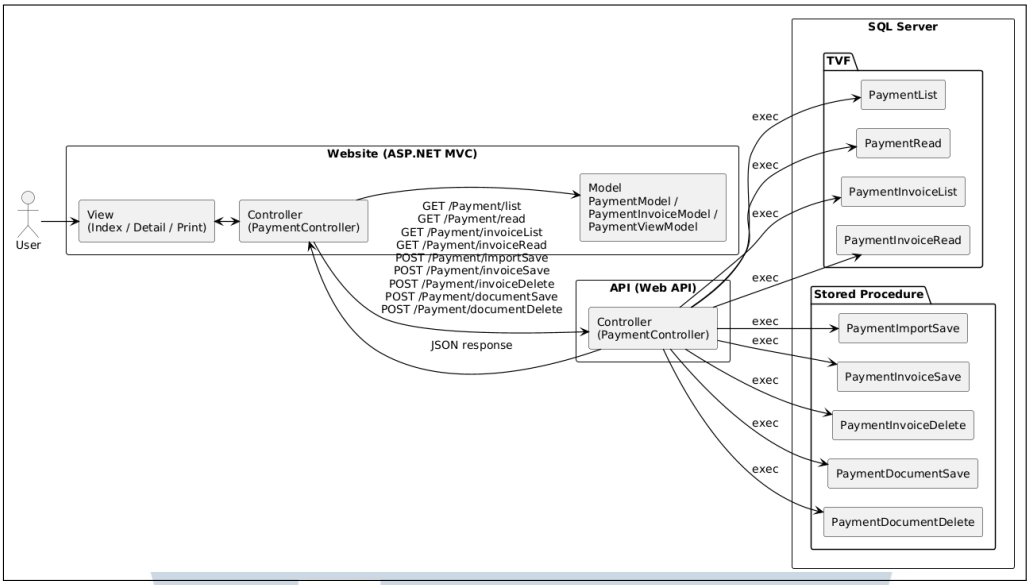
sesuai logika bisnis, dengan penerapan *workstate guard* guna mengatur ketersediaan aksi berdasarkan status dokumen.

2. Lapisan *Controller* pada *Website*: Lapisan ini mengatur seluruh alur komunikasi dari sisi klien. *PaymentController* menerima permintaan dari pengguna, menyiapkan data melalui *Model*, lalu mengirimkannya ke API menggunakan permintaan `HTTP GET/POST` dalam *format* JSON. *Endpoint* yang digunakan meliputi: `GET /Payment/list`, `GET /Payment/read`, `GET /Payment/invoiceList`, `GET /Payment/invoiceRead`, `POST /Payment/importSave`, `POST /Payment/invoiceSave`, `POST /Payment/invoiceDelete`, `POST /Payment/documentSave`, dan `POST /Payment/documentDelete`.

3. Lapisan *Controller* pada API: *Controller* di proyek API bertugas memproses permintaan dari *Website* dan berinteraksi langsung dengan basis data SQL *Server*. Setiap permintaan dieksekusi melalui fungsi tabel TVF *(Table-Valued Function)* atau *Stored Procedure* sesuai jenis operasinya, kemudian hasilnya dikembalikan dalam *format* JSON berisi struktur {`status, message, data`}.

4. Lapisan *Model* dan Basis Data: Lapisan *Model* pada *Website* (*PaymentModel*, *PaymentInvoiceModel*, *PaymentViewModel*) merepresentasikan data yang diperoleh dari API dan digunakan sebagai wadah data untuk ditampilkan kembali oleh *View*. Di sisi basis data, TVF digunakan untuk proses pembacaan seperti `PaymentList`, `PaymentRead`, `PaymentInvoiceList`, dan `PaymentInvoiceRead`. Sedangkan SP digunakan untuk operasi manipulasi data seperti `PaymentImportSave`, `PaymentInvoiceSave`, `PaymentInvoiceDelete`, `PaymentDocumentSave`, dan `PaymentDocumentDelete`. Seluruh prosedur tersebut dieksekusi (`exec`) oleh *Controller* API.

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 3.2. Alur komunikasi MVC pada GMMS modul *Payment* CC

Gambar 3.2 menggambarkan alur komunikasi arsitektur *Model-View-Controller* (MVC) pada sistem GMMS, khususnya pada modul *Payment* CC yang berfungsi untuk pengelolaan bukti pembayaran kartu kredit. Modul ini tidak melakukan proses transaksi atau pembayaran secara langsung, melainkan menangani pencatatan, pengelolaan, dan dokumentasi data pembayaran yang telah dilakukan. Alur komunikasi sistem pada modul ini dapat dijelaskan sebagai berikut:

1. Pengguna berinteraksi dengan sistem melalui *View* pada modul *Payment* CC, seperti melihat daftar bukti pembayaran, membuka detail dokumen, menambah atau mengelola data *invoice*, mengunggah atau menghapus dokumen pendukung (*attachment*), serta mencetak dokumen sebagai arsip administrasi.

2. Setiap interaksi pada *View* diteruskan ke *Website Controller* (*PaymentController*) pada aplikasi berbasis ASP.NET MVC. Controller ini berperan sebagai pengelola alur permintaan (*request handling*) dan penghubung antara antarmuka pengguna dengan layanan API.

3. *Website Controller* memanggil *endpoint* API sesuai dengan kebutuhan proses, antara lain:

   - `GET /Payment/list`
   - `GET /Payment/read`

- `GET /Payment/invoiceList`
- `GET /Payment/invoiceRead`
- `POST /Payment/importSave`
- `POST /Payment/invoiceSave`
- `POST /Payment/invoiceDelete`
- `POST /Payment/documentSave`
- `POST /Payment/documentDelete`

4. *API Controller* menerima permintaan dari aplikasi *Website* dan mengeksekusi logika bisnis dengan memanggil fungsi tabel (*Table Valued Function*/TVF) atau *Stored Procedure* pada SQL *Server*, seperti `PaymentList`, `PaymentRead`, `PaymentInvoiceList`, `PaymentInvoiceSave`, dan `PaymentDocumentSave`.

5. SQL *Server* memproses permintaan tersebut dan mengembalikan hasil eksekusi ke *API Controller*. Data hasil pemrosesan ini kemudian dikonversi ke dalam format JSON sebagai respons API.

6. Respons JSON dikirim kembali oleh API ke *Website Controller* untuk diproses lebih lanjut.

7. *Website Controller* memetakan data JSON ke dalam *Model* yang sesuai, seperti `PaymentModel`, `PaymentInvoiceModel`, atau `PaymentViewModel`, sebelum data tersebut diteruskan ke *View* untuk ditampilkan kepada pengguna.

Arsitektur ini memastikan pemisahan tanggung jawab yang jelas antar lapisan sistem. *View* berfungsi sebagai antarmuka pengguna, *Controller* mengelola alur permintaan dan logika aplikasi, serta *Model* berperan sebagai representasi data. API bertindak sebagai penghubung antara aplikasi *Website* dan SQL *Server*, sedangkan SQL *Server* menangani pemrosesan data melalui TVF dan *Stored Procedure*. Dengan pendekatan ini, modul *Payment* CC mendukung kebutuhan administrasi dan audit bukti pembayaran secara terstruktur, konsisten,dan mudah dikembangkan.

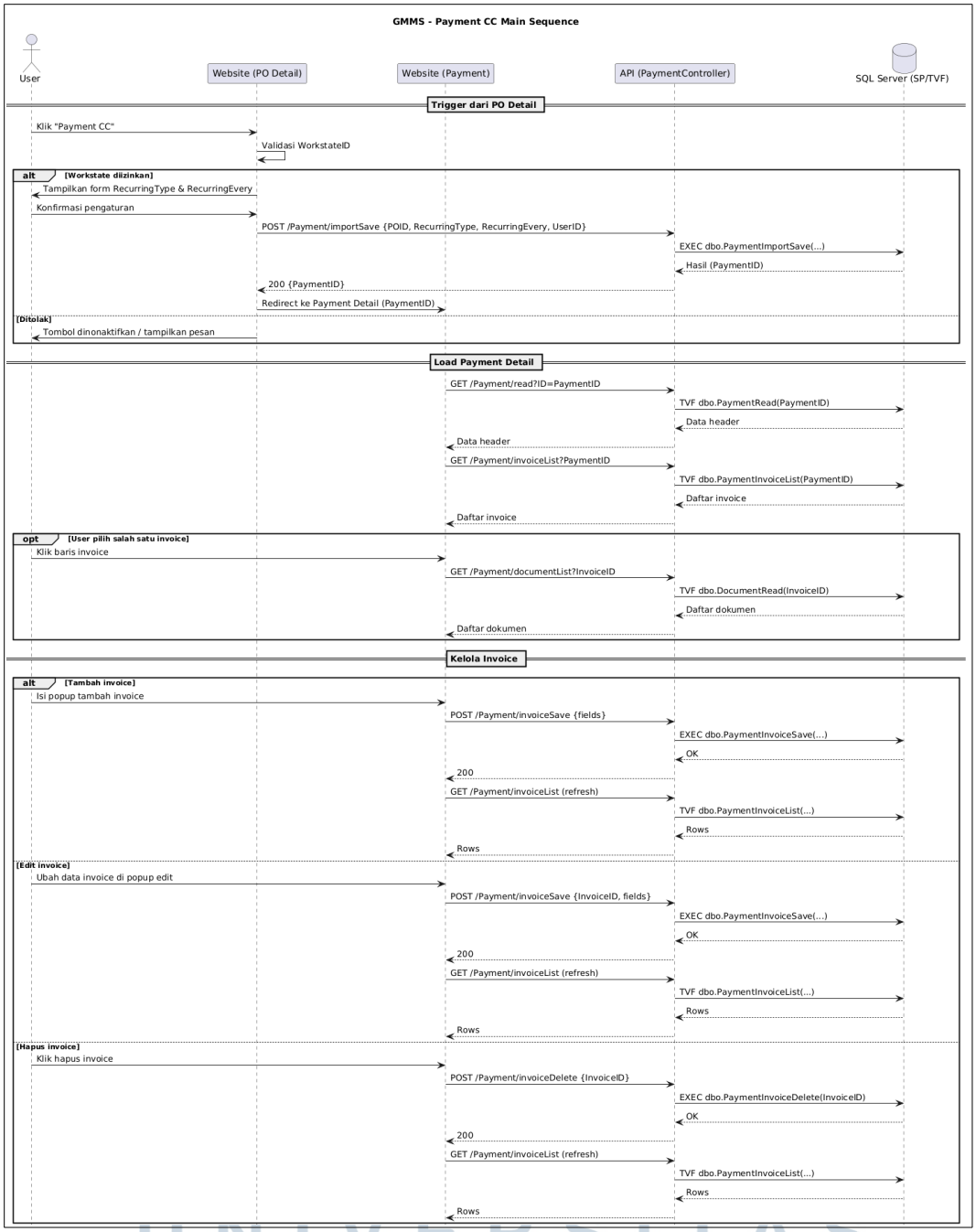
C Perancangan Alur Proses Bisnis

Perancangan alur proses bisnis pada modul *Payment* CC bertujuan untuk menggambarkan secara komprehensif urutan interaksi antara pengguna, sistem *Website*, API, dan basis data. Alur ini memastikan bahwa seluruh proses, mulai dari pemicu di halaman PO hingga pengelolaan *invoice*, *attachment*, serta pencetakan dokumen, berjalan secara sistematis dan sesuai aturan bisnis modul *Purchasing* GMMS.

Proses pada modul *Payment* CC tetap terintegrasi dengan modul PO, di mana pemilihan tombol *Payment* CC hanya dapat dilakukan apabila dokumen PO berada pada `WorkstateID` yang diizinkan. Mekanisme validasi ini menjamin bahwa pembentukan data pembayaran dilakukan secara terkendali dan sesuai dengan kebijakan perusahaan.

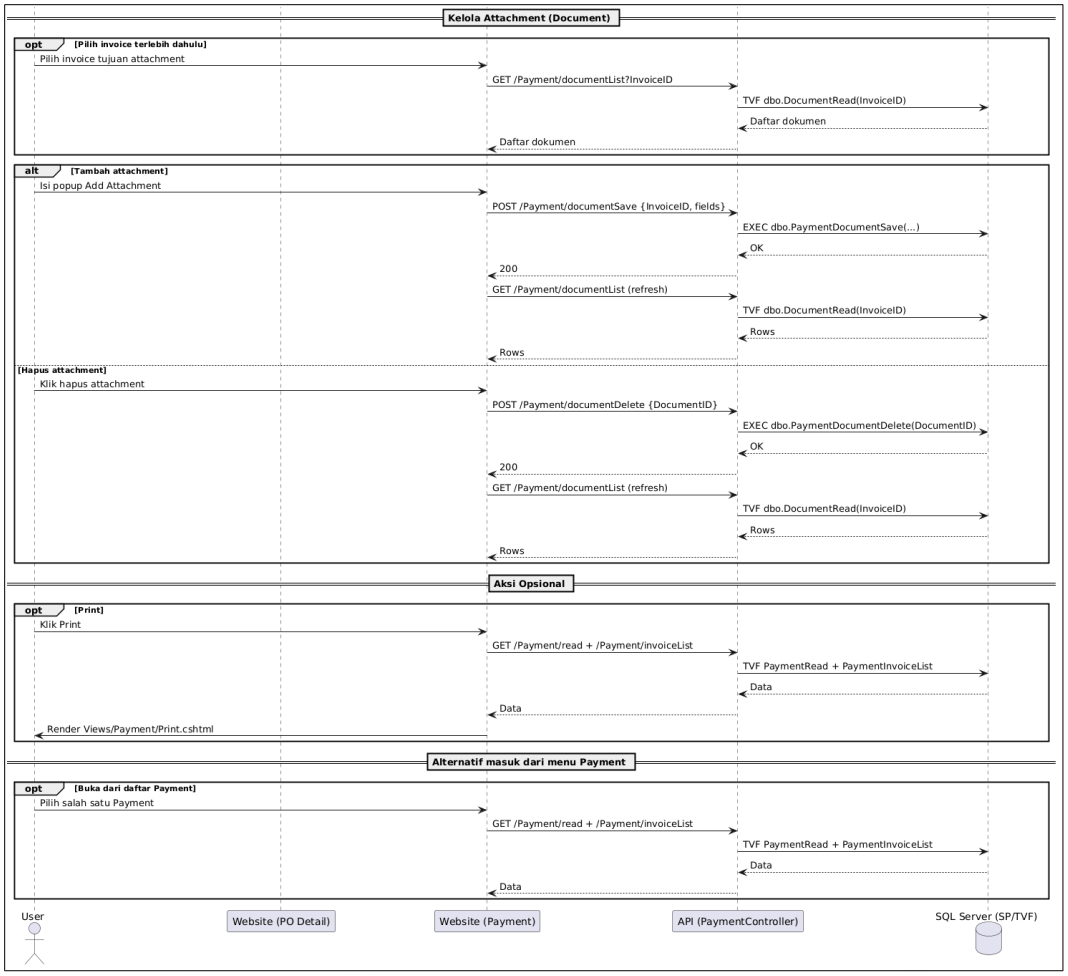
Bagian ini diawali oleh diagram urutan proses bisnis (*sequence diagram*) yang menunjukkan hubungan langsung antar komponen sistem, kemudian dilanjutkan dengan diagram aktivitas yang menguraikan alur dari masing-masing fitur secara lebih detail.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.3. Diagram urutan proses bisnis modul *Payment* CC pada GMMS

UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 3.3. Diagram urutan proses bisnis modul *Payment* CC pada GMMS (lanjutan)

Gambar 3.3 menggambarkan keseluruhan alur mulai dari pemicu pembuatan data *Payment* CC, proses pemuatan halaman, pengelolaan *invoice*, pengelolaan *attachment*, hingga aksi opsional seperti *Print* dan akses melalui menu daftar *Payment*.

1. Pemicu dari modul PO: Proses dimulai ketika pengguna mengakses halaman PO *Detail* dan menekan tombol *Payment* CC. Sistem melakukan validasi `WorkstateID` untuk memastikan bahwa status dokumen PO berada pada kondisi *Approval*. Selain validasi status dokumen, sistem juga menerapkan aturan bisnis untuk mencegah terjadinya *double payment*. Setiap dokumen *Purchase Order* (PO) hanya diperbolehkan memiliki satu data *Payment* CC sebagai bukti pembayaran. Ketika tombol *Payment* CC ditekan, sistem terlebih dahulu melakukan pengecekan apakah PO tersebut sudah memiliki data *Payment* yang terdaftar. Apabila data *Payment* telah ada, maka sistem

tidak mengizinkan pembuatan data *Payment* baru untuk PO yang sama. Jika valid, sistem menampilkan formulir pengaturan pengulangan pembayaran (`RecurringType` dan `RecurringEvery`). Setelah pengguna mengonfirmasi, data dikirim ke API melalui `/Payment/importSave`. API menjalankan prosedur tersimpan `dbo.PaymentImportSave` untuk membentuk entri *Payment* baru di basis data, kemudian pengguna diarahkan ke halaman *Payment Detail*.

2. Pemuatan data pada halaman *Payment Detail*: Halaman *Payment Detail* memuat dua jenis data utama yaitu *header* pembayaran yang diambil melalui fungsi TVF `dbo.PaymentRead` dan daftar *invoice* melalui TVF `dbo.PaymentInvoiceList`. Sistem kemudian menampilkan informasi seperti nomor PO, *vendor*, tanggal, nilai, status dokumen, dan daftar *invoice* yang telah tersimpan.

3. Pemilihan *invoice* untuk memuat dokumen (*attachment*): Ketika pengguna memilih salah satu baris *invoice*, sistem mengirim permintaan ke API `/Payment/documentList` untuk mengambil daftar dokumen. API kemudian menjalankan TVF `dbo.DocumentRead` untuk membaca seluruh *attachment* terkait *invoice*. Hasilnya ditampilkan pada bagian *Attachment* di halaman *Payment Detail*.

4. Pengelolaan *invoice* (tambah, edit, hapus): Pengguna dapat mengelola *invoice* melalui tiga aksi utama:

   - Tambah: Sistem mengirim data ke API `/Payment/invoiceSave`, lalu prosedur `dbo.PaymentInvoiceSave` menyimpan data *invoice* baru.
   - *Edit*: Perubahan data dikirim kembali melalui `/Payment/invoiceSave` dengan parameter `InvoiceID`.
   - Hapus: API memanggil `dbo.PaymentInvoiceDelete` untuk menghapus *invoice*.

   Setelah tiap operasi berhasil, daftar *invoice* diperbarui dengan memanggil kembali `PaymentInvoiceList`.

5. Pengelolaan *attachment* (tambah dan hapus): Setiap *attachment* wajib terhubung pada salah satu *invoice*, sehingga pengguna harus memilih *invoice* sebelum menambah dokumen. Fitur ini mencakup:

- Tambah *attachment*: Data dikirim melalui `/Payment/documentSave`, dan API menjalankan `dbo.PaymentDocumentSave`.
- Hapus *attachment*: Sistem memanggil `/Payment/documentDelete`, yang menjalankan `dbo.PaymentDocumentDelete`.

Setelah aksi dilakukan, daftar dokumen diperbarui melalui TVF `dbo.DocumentRead`.

6. Aksi opsional pencetakan dokumen: Ketika pengguna memilih opsi *Print*, sistem memuat ulang data `PaymentRead` dan `PaymentInvoiceList` untuk memastikan informasi yang ditampilkan adalah versi terbaru. Hasilnya dirender melalui `Views/Payment/Print.cshtml` dalam format siap cetak.

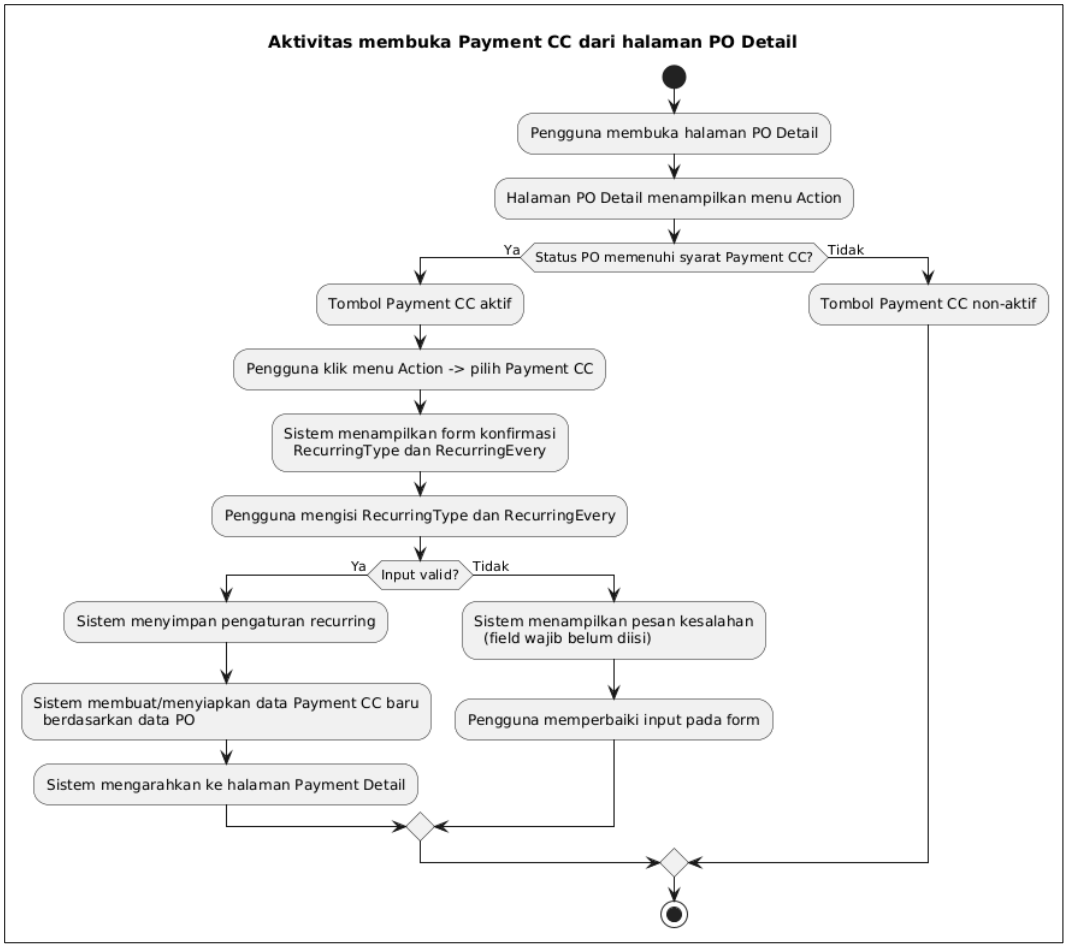
7. Akses melalui menu daftar Payment: Modul *Payment* CC juga dapat diakses melalui daftar *Payment*. Ketika pengguna memilih salah satu dokumen, sistem memuat data dengan prosedur yang sama seperti pada saat menampilkan halaman *Payment Detail*.

C.1 Activity Diagram Modul Payment CC

Bagian ini menampilkan rangkaian *activity diagram* yang menggambarkan alur kerja modul *Payment* CC secara lebih rinci. Diagram dipisahkan berdasarkan fungsi utama agar setiap proses dapat ditelusuri secara jelas dan sistematis, mulai dari pemicu pembuatan *Payment* CC, navigasi halaman daftar *Payment*, pengelolaan *invoice*, pengelolaan *attachment*, hingga proses pencetakan dokumen.

1. Aktivitas membuka modul *Payment* CC dari PO *Detail*

   Sebelum modul *Payment* CC dibuka, sistem memeriksa status PO untuk menentukan apakah menu tersebut tersedia bagi pengguna. Jika status memenuhi syarat, pengguna dapat memilih menu *Action* untuk membuka formulir pengaturan *recurring* yang berisi parameter seperti *RecurringType* dan *RecurringEvery*. Setelah pengaturan divalidasi, sistem membuat data *Payment* CC baru dan mengarahkan pengguna ke halaman *Payment Detail*. Alur aktivitas ini dapat dilihat pada Gambar 3.4.

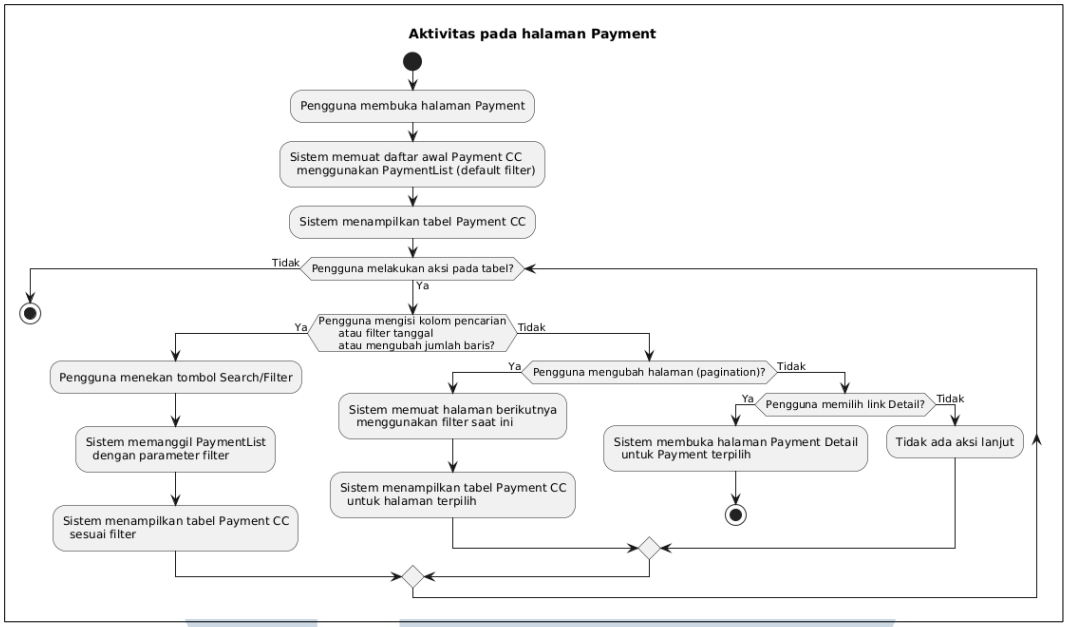


Gambar 3.4. Diagram aktivitas membuka modul *Payment* CC dari PO *Detail*

2. Aktivitas pada halaman *Payment*

   Setelah modul dibuka, pengguna diarahkan ke halaman daftar *Payment* yang menampilkan seluruh data *Payment* CC yang tersedia. Sistem memuat data awal menggunakan fungsi `PaymentList`, dan pengguna dapat memanfaatkan fitur pencarian, *filter* tanggal, pengaturan jumlah baris, dan *pagination*. Saat salah satu baris dipilih, sistem menampilkan detail dokumen pada halaman *Payment Detail*. Alur aktivitas tersebut dapat dilihat pada Gambar 3.5.

UNIVERSITAS MULTIMEDIA NUSANTARA

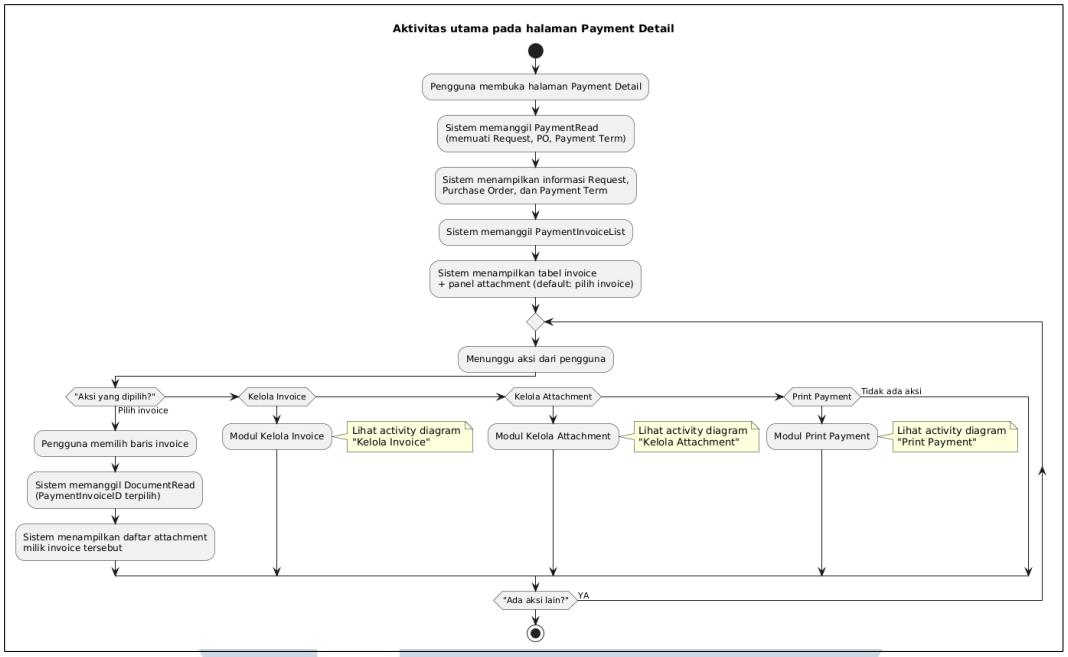


Gambar 3.5. Diagram aktivitas pada halaman *Payment*

3. Aktivitas utama pada halaman *Payment Detail*

   Halaman *Payment Detail* menampilkan informasi lengkap dokumen pembayaran, termasuk data *Request*, PO, dan *Payment Term* yang diperoleh melalui fungsi `PaymentRead`. Daftar *invoice* ditampilkan menggunakan `PaymentInvoiceList`. Pengguna dapat memilih *invoice* untuk melihat *attachment*, mengelola *invoice*, mengelola *attachment*, atau melakukan pencetakan dokumen. Alur aktivitas tersebut dapat dilihat pada Gambar 3.6.

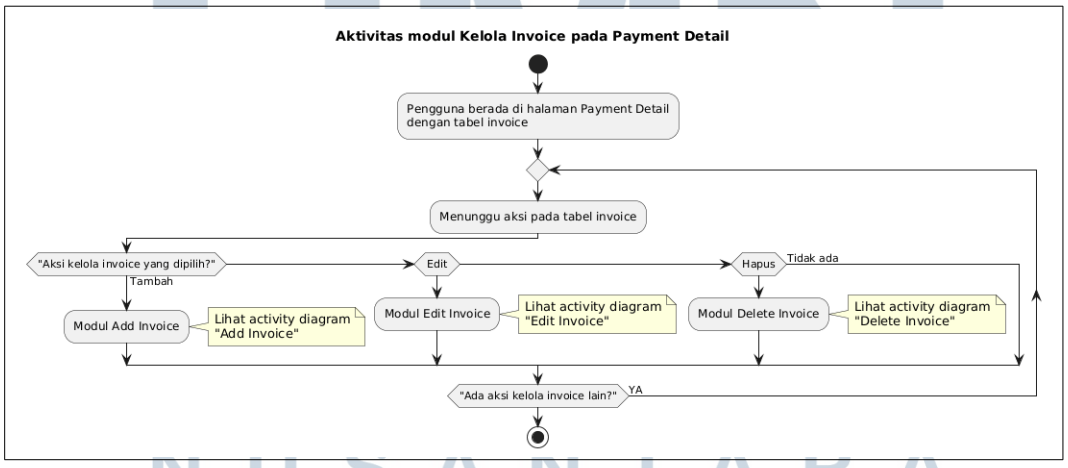
UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.6. Diagram aktivitas utama pada halaman *Payment Detail*

4. Aktivitas modul Kelola *Invoice*

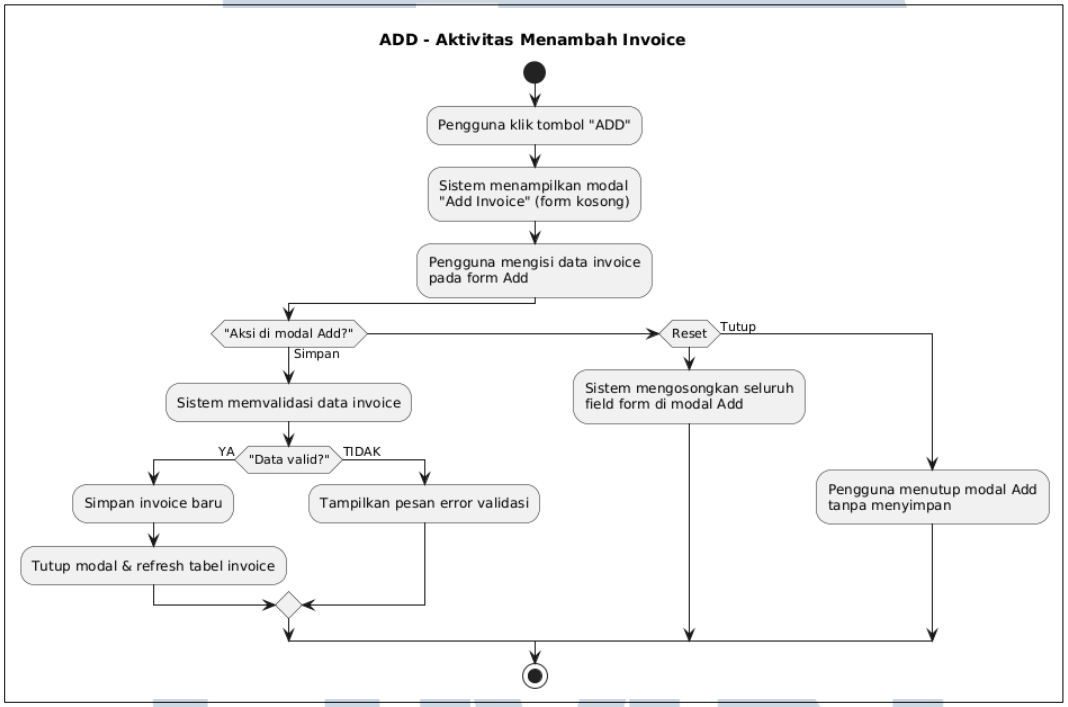
   Modul ini menyediakan tiga fungsi utama, yaitu menambah, mengedit, dan menghapus *invoice*. Sistem menunggu interaksi pengguna pada tabel *invoice* sebelum memunculkan proses yang sesuai. Rangkaian aktivitas pada modul ini dapat dilihat pada Gambar 3.7.



Gambar 3.7. Diagram aktivitas modul Kelola *Invoice*

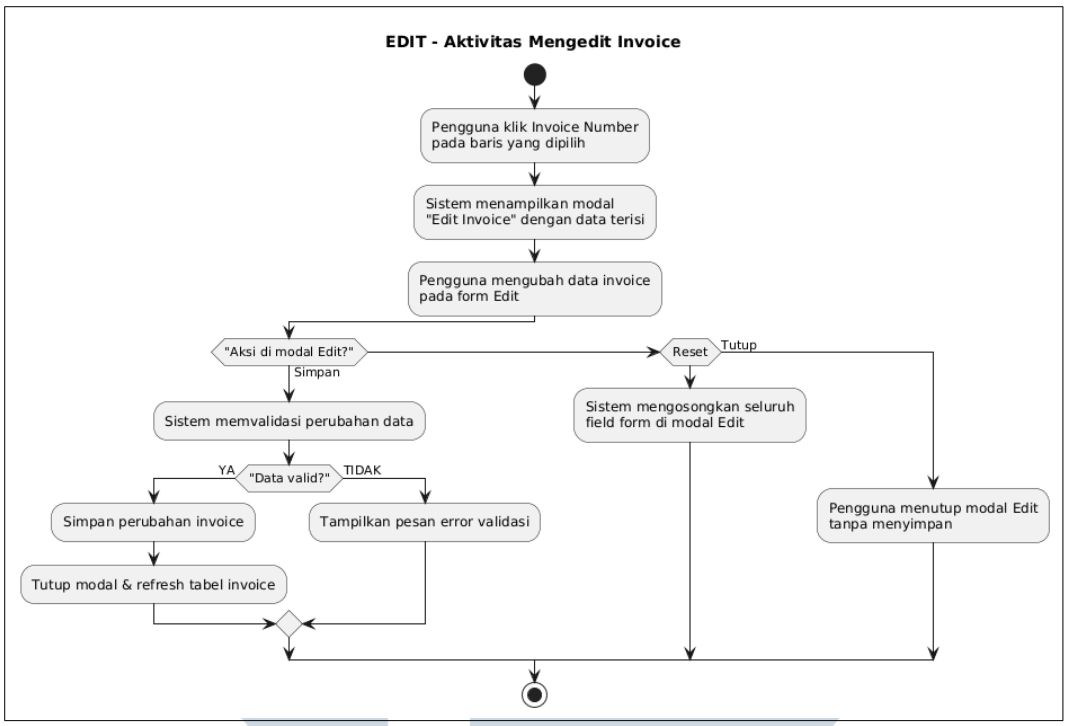
Rincian aktivitas dijelaskan sebagai berikut:

a) Menambah *Invoice*: Ketika pengguna memilih tombol *ADD*, sistem menampilkan modal *Add Invoice* dalam keadaan kosong. Pengguna mengisi informasi seperti nomor *invoice*, tanggal, pemegang kartu, nilai transaksi, bank, dan data pendukung lainnya. Setelah tombol *Save* dipilih, sistem melakukan validasi sebelum menyimpan data dan memperbarui tabel *invoice*. Alur proses ini dapat dilihat pada Gambar 3.8.



Gambar 3.8. Diagram aktivitas *Add Invoice*

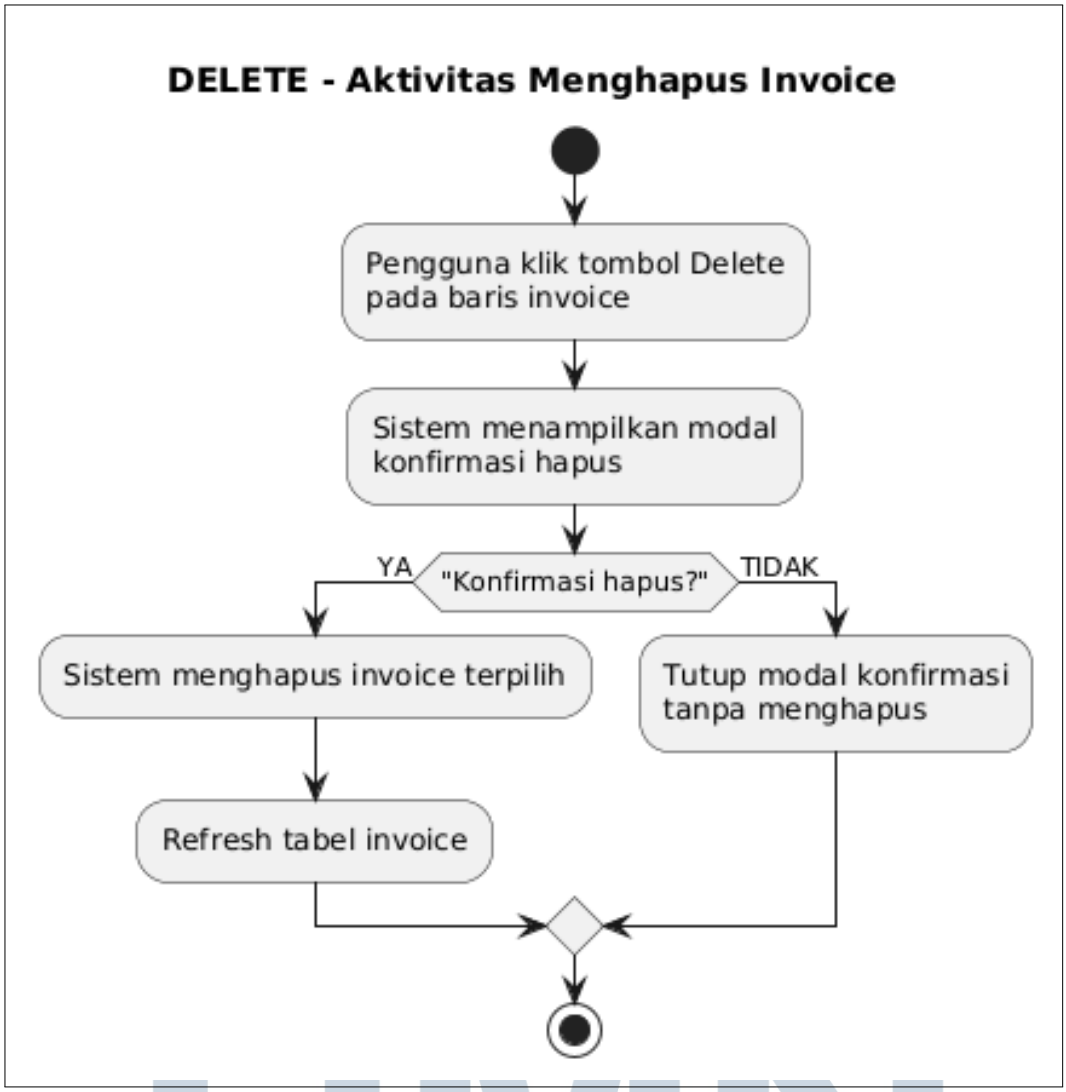
b) Mengedit *Invoice*: Pengguna dapat membuka modal *Edit Invoice* dengan memilih *Invoice Number*. Sistem menampilkan data awal yang dapat diperbarui pengguna. Aksi yang tersedia meliputi *Save*, *Reset*, dan *Close*. Jika validasi berhasil, data disimpan dan tabel diperbarui tanpa perlu memuat ulang halaman. Alur proses ini dapat dilihat pada Gambar 3.9.



Gambar 3.9. Diagram aktivitas *Edit Invoice*

c) Menghapus *Invoice*: Saat pengguna memilih tombol *Delete*, sistem menampilkan modal konfirmasi untuk memastikan bahwa proses penghapusan dilakukan dengan benar. Jika pengguna menyetujui, sistem menghapus data melalui API dan memperbarui tabel *invoice*. Alur proses ini dapat dilihat pada Gambar 3.10.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

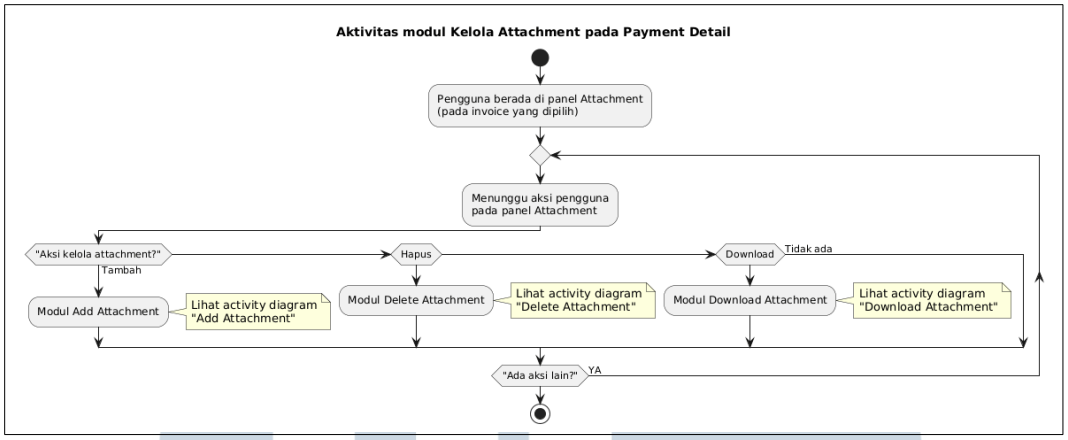


Gambar 3.10. Diagram aktivitas *Delete Invoice*

5. Aktivitas modul Kelola *Attachment*

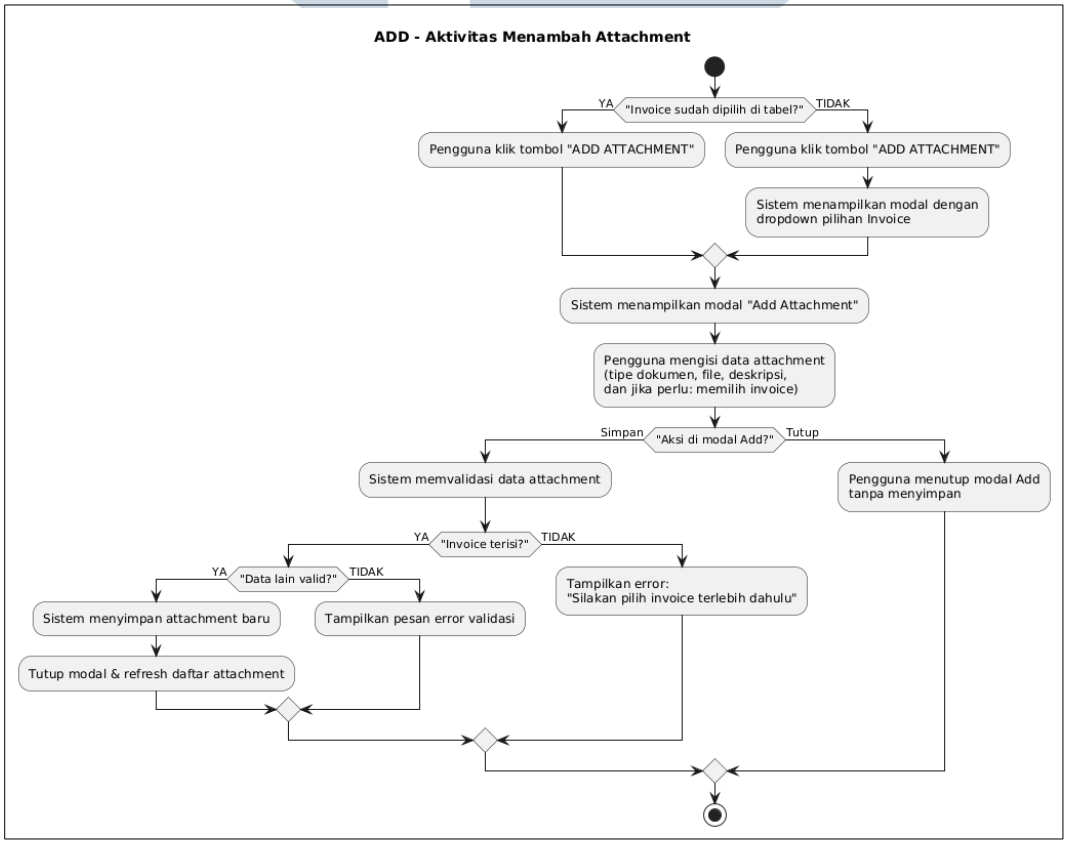
   Panel *Attachment* hanya aktif jika pengguna memilih salah satu *invoice*. Sistem kemudian menampilkan daftar *attachment* yang terkait. Pengguna dapat menambah, menghapus, ataupun mengunduh *attachment*. Alur lengkap modul ini dapat dilihat pada Gambar 3.11.

UNIVERSITAS MULTIMEDIA NUSANTARA



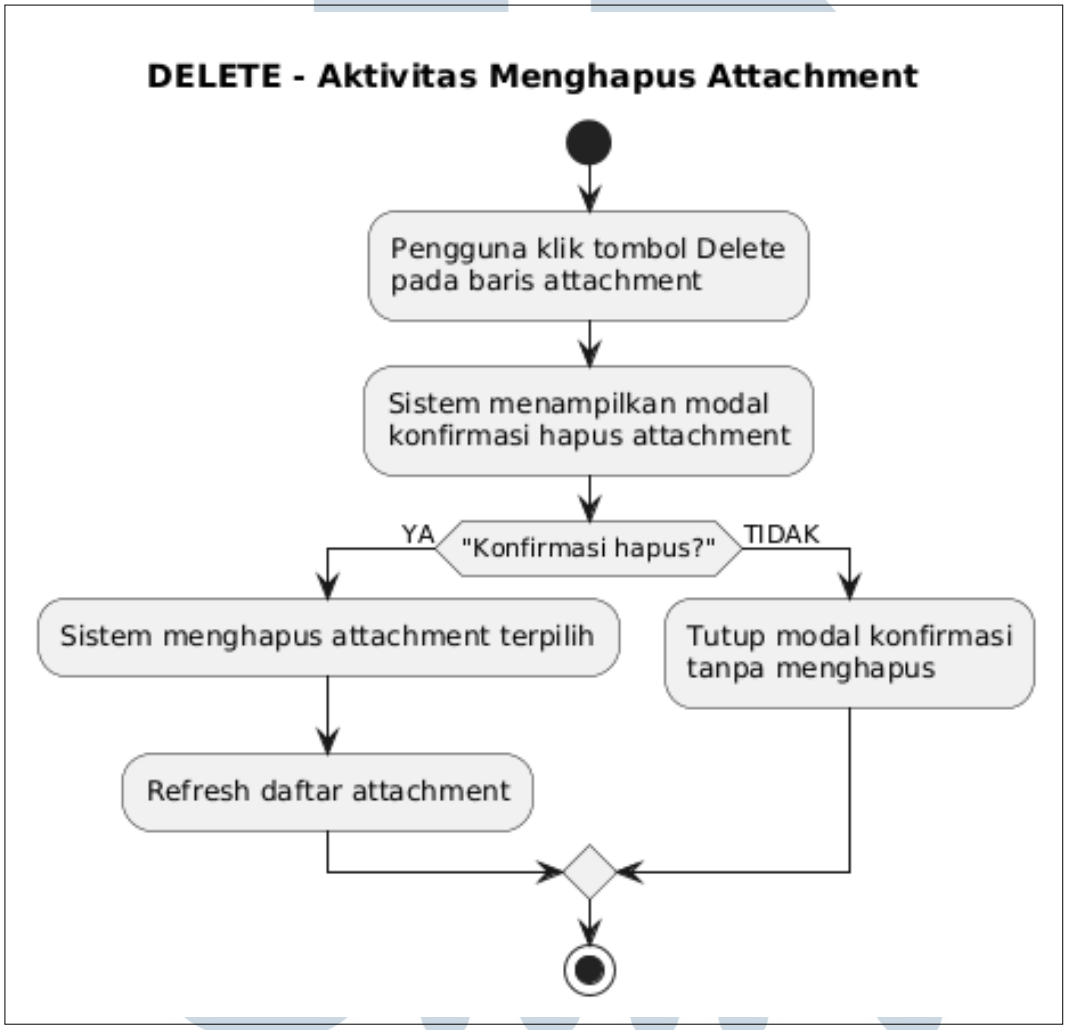
Gambar 3.11. Diagram aktivitas modul Kelola *Attachment*

a) Menambah *Attachment*: Pengguna membuka modal *Add Attachment* untuk memilih jenis dokumen, file, deskripsi, dan *invoice* tujuan. Sistem melakukan validasi saat tombol *Save* dipilih sebelum menyimpan data. Alur penambahan *attachment* dapat dilihat pada Gambar 3.12.



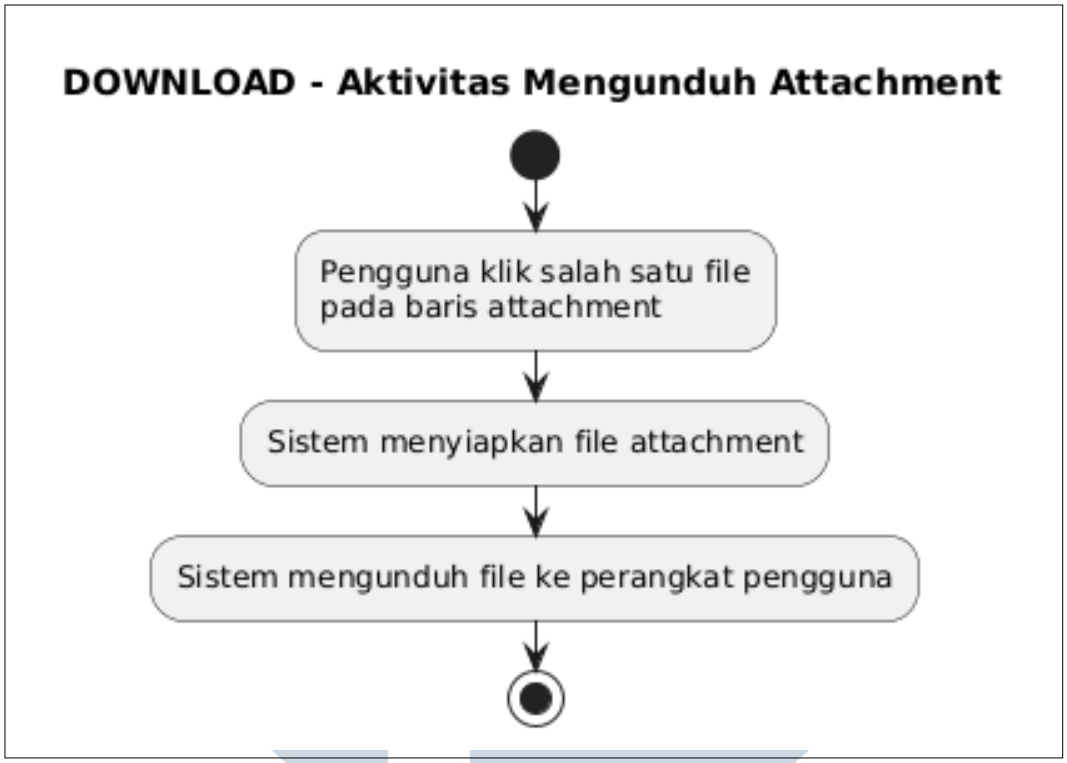
Gambar 3.12. Diagram aktivitas *Add Attachment*

b) Menghapus *Attachment*: Sistem menampilkan modal konfirmasi ketika pengguna memilih tombol *Delete*. Jika pengguna menyetujui, sistem menghapus file dan memperbarui daftar *attachment*. Alur proses ini dapat dilihat pada Gambar 3.13.



Gambar 3.13. Diagram aktivitas *Delete Attachment*

c) Mengunduh *Attachment*: Pengguna dapat mengunduh file dengan memilih nama *attachment* yang tersedia pada tabel. Sistem kemudian menyiapkan file melalui API dan memulai proses pengunduhan tanpa memerlukan konfirmasi tambahan. Alur pengunduhan *attachment* dapat dilihat pada Gambar 3.14.

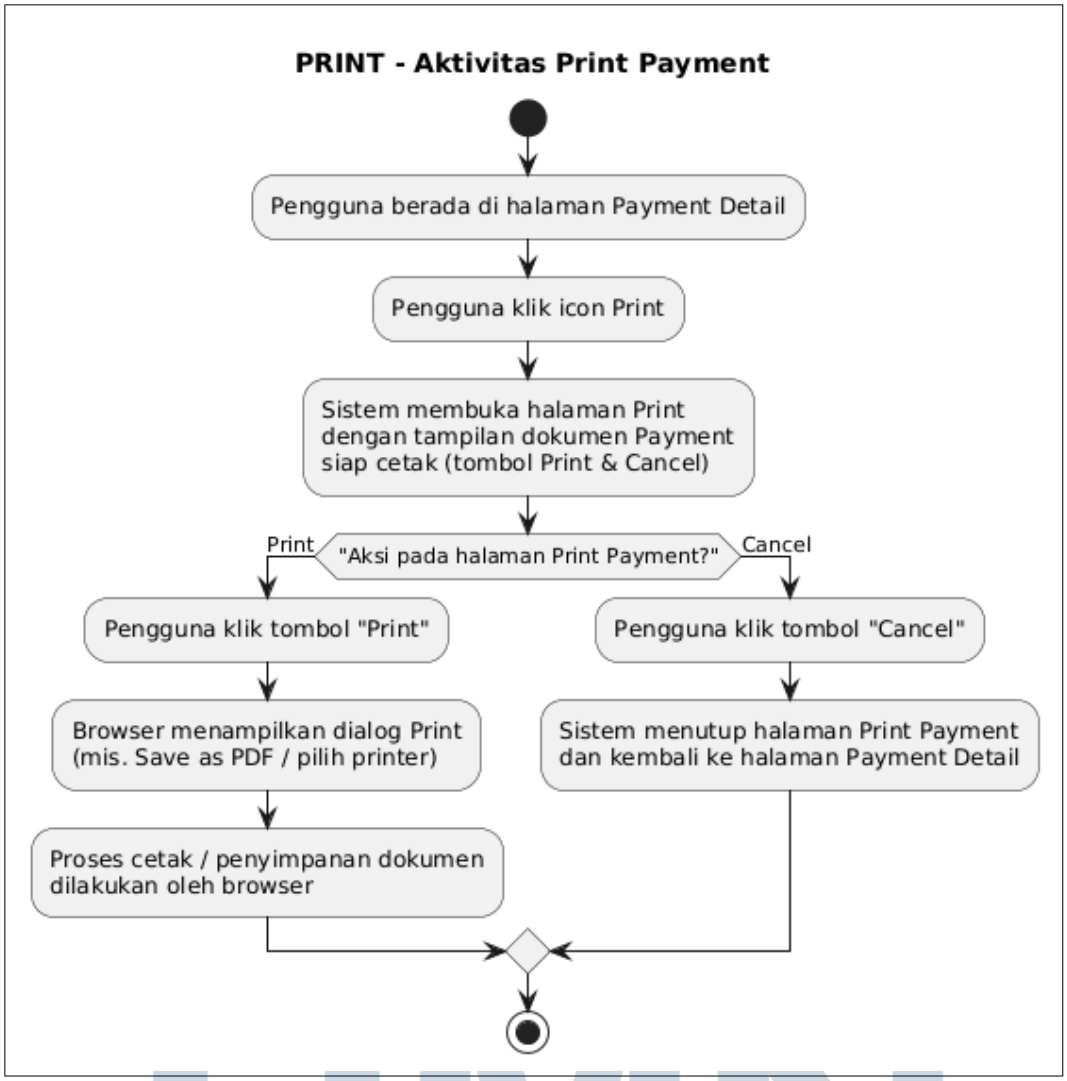


Gambar 3.14. Diagram aktivitas *Download Attachment*

6. Aktivitas mencetak dokumen *Payment*

   Ketika pengguna memilih ikon *Print*, sistem membuka halaman *Print Payment* yang menampilkan tampilan dokumen siap cetak. Pengguna dapat memilih tombol *Print* untuk membuka dialog cetak atau memilih *Cancel* untuk kembali ke halaman sebelumnya. Alur pencetakan dapat dilihat pada Gambar 3.15.

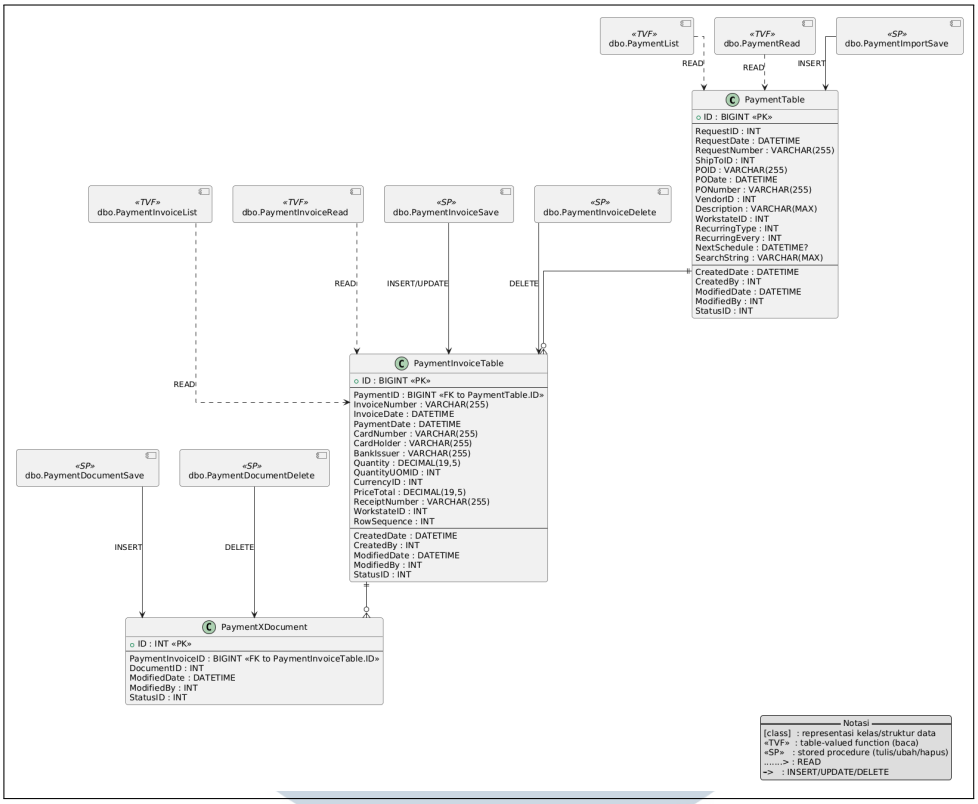
UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.15. Diagram aktivitas *Print Payment*

D Perancangan Basis Data

Perancangan struktur *class* pada modul *Payment* CC terdiri dari tiga *class* utama, yaitu `PaymentTable`, `PaymentInvoiceTable`, dan `PaymentXDocument`. Ketiga *class* ini merepresentasikan objek-objek penting dalam proses pengelolaan pembayaran, mulai dari informasi *header* pembayaran, rincian *invoice*, dan lampiran dokumen. Hubungan antara *class* dirancang untuk mencerminkan alur data aktual dalam sistem, seperti keterhubungan antara `PaymentTable` dengan *class* `PaymentInvoiceTable`, serta relasi antara `PaymentInvoiceTable` dan `PaymentXDocument`. Gambaran lengkap struktur *class* tersebut ditunjukkan pada Gambar 3.16.



Gambar 3.16. Diagram struktur *class* modul *Payment* CC pada GMMS

Struktur pada Gambar 3.16 juga menampilkan komponen eksternal berupa *table-valued function* (TVF) dan *stored procedure* (SP) yang digunakan untuk operasi baca maupun tulis pada basis data. Meskipun bukan bagian dari *class* inti, keberadaan TVF dan SP ditampilkan untuk memperjelas aliran data yang terjadi antara operasi basis data dan *class* utama dalam modul. Dengan rancangan ini, hubungan antar objek dan alur proses menjadi lebih mudah dipahami sekaligus konsisten dengan arsitektur *Model–View–Controller* yang diterapkan pada GMMS.

D.1 Entitas Class dan Atribut Utama

Tiga *class* utama pada modul *Payment* CC dirancang untuk merepresentasikan objek-objek bisnis yang terlibat dalam proses pengelolaan pembayaran kartu kredit, mulai dari *header* pembayaran, rincian *invoice*, hingga lampiran dokumen pendukung. Setiap *class* memiliki atribut dan relasi kunci yang mencerminkan hubungan antardata dalam sistem, sebagaimana ditunjukkan pada Gambar 3.16. Struktur *class* ini juga selaras dengan alur operasi sistem yang menggunakan *table-valued function* (TVF) untuk membaca data dan *stored*

procedure (SP) untuk proses simpan, ubah, dan hapus.

1. `PaymentTable` (*Header* Pembayaran)
   Class ini menjadi pusat dari seluruh proses transaksi pembayaran karena menyimpan berbagai informasi penting, mulai dari data permintaan pembayaran, referensi dokumen PO, hingga keterkaitan dengan entitas *vendor* dan lokasi pengiriman. Dalam modul *Payment* CC, *class* `PaymentTable` merepresentasikan satu kesatuan pembayaran yang kemudian menjadi titik awal hubungan menuju detail *invoice* melalui `PaymentInvoiceTable`. *Class* ini menggunakan `ID` sebagai *primary key*, sementara atribut seperti `RequestID`, `ShipToID`, dan `VendorID` digunakan sebagai referensi terhadap data pada modul lain. Struktur lengkap *class* ditunjukkan pada Tabel 3.2.

Tabel 3.2. Struktur *class* `PaymentTable`

Nama Kolom	Tipe Data	Default	Keterangan
ID	BIGINT (PK)	Identity (1,1)	Kunci utama tabel
RequestID	INT	0	Nomor permintaan pembayaran
RequestDate	DATETIME	getdate()	Tanggal permintaan dibuat
RequestNumber	VARCHAR(255)	”	Nomor referensi dokumen
ShipToID	INT	0	Tujuan pengiriman
POID	VARCHAR(255)	”	ID referensi PO
PODate	DATETIME	getdate()	Tanggal dokumen PO
PONumber	VARCHAR(255)	”	Nomor PO terkait
VendorID	INT	0	Pemasok atau *vendor* terkait
Description	VARCHAR(MAX)	”	Deskripsi pembayaran
WorkstateID	INT	0	Status proses pembayaran
RecurringType	INT	0	Jenis pembayaran berulang
RecurringEvery	INT	0	Interval pembayaran berulang
NextSchedule	DATETIME	getdate()	Jadwal pembayaran berikutnya
SearchString	VARCHAR(MAX)	”	Indeks pencarian cepat
CreatedDate / CreatedBy	DATETIME / INT	getdate() / 0	Waktu dan pengguna pembuat
ModifiedDate / ModifiedBy	DATETIME / INT	getdate() / 0	Waktu dan pengguna pengubah
StatusID	INT	1	Status aktif atau nonaktif

2. `PaymentInvoiceTable` (Detail *Invoice*)

   Class ini digunakan untuk merepresentasikan rincian setiap *invoice* yang tercatat pada satu objek `PaymentTable`. Informasi yang tersimpan meliputi nomor faktur, tanggal transaksi, tanggal pembayaran, serta data terkait kartu kredit dan tanda terima. Melalui atribut `PaymentID`, *class* ini terhubung langsung dengan objek induknya, yaitu `PaymentTable`. Hubungan ini memungkinkan sistem menautkan beberapa *invoice* ke dalam satu transaksi pembayaran. Struktur lengkap *class* ditampilkan pada Tabel 3.3.

Tabel 3.3. Struktur *class* `PaymentInvoiceTable`

Nama Kolom	Tipe Data	Default	Keterangan
ID	BIGINT (PK)	Identity (1,1)	Kunci utama tabel
PaymentID	BIGINT (FK)	0	Relasi ke *PaymentTable.ID*
InvoiceNumber	VARCHAR(255)	”	Nomor faktur pembayaran
InvoiceDate / PaymentDate	DATETIME	getdate()	Tanggal faktur dan pembayaran
CardNumber / CardHolder / BankIssuer	VARCHAR(255)	”	Informasi kartu kredit
Quantity	DECIMAL(19,5)	1.00	Jumlah transaksi
QuantityUOMID / CurrencyID	INT	0	Satuan dan mata uang
PriceTotal	DECIMAL(19,5)	1.00	Nilai total pembayaran
ReceiptNumber	VARCHAR(255)	”	Nomor tanda terima
WorkstateID	INT	0	Status proses data
CreatedDate / CreatedBy	DATETIME / INT	getdate() / 0	Waktu dan pengguna pembuat
ModifiedDate / ModifiedBy	DATETIME / INT	getdate() / 0	Waktu dan pengguna pengubah
RowSequence / StatusID	INT	1	Urutan baris dan status aktif

3. `PaymentXDocument` (Relasi Lampiran Dokumen)

   Class ini digunakan untuk merepresentasikan lampiran dokumen yang dikaitkan dengan suatu *invoice*. Informasi yang dicatat meliputi identitas dokumen, keterhubungan dengan objek `PaymentInvoiceTable`, waktu perubahan, serta status dokumen. Melalui atribut `PaymentInvoiceID`, *class*

ini memungkinkan satu *invoice* memiliki lebih dari satu dokumen pendukung. Struktur lengkap *class* ditunjukkan pada Tabel 3.4.

Tabel 3.4. Struktur *class* `PaymentXDocument`

Nama Kolom	Tipe Data	Default	Keterangan
ID	INT (PK)	Identity (1,1)	Kunci utama tabel
PaymentInvoiceID	BIGINT (FK)	0	Relasi ke *PaymentInvoiceTable.ID*
DocumentID	INT	0	ID dokumen lampiran
ModifiedDate / ModifiedBy	DATETIME / INT	getdate() / 0	Waktu dan pengguna pengubah
StatusID	INT	1	Status aktif atau nonaktif

D.2 Relasi Antar Class

Relasi antar *class* dalam modul *Payment* CC dirancang untuk memastikan keterhubungan data antara objek pembayaran, *invoice*, *item*, dan dokumen pendukung. Secara ringkas, relasi dan kardinalitasnya adalah sebagai berikut:

1. `Payment` → `PaymentInvoice`: 1–* (satu objek pembayaran memiliki banyak objek *invoice*).
2. `PaymentInvoice` → `PaymentDocument`: 1–* (satu objek *invoice* memiliki banyak objek lampiran dokumen).
3. Atribut `POID`, `PONumber`, dan `PODate` pada *class* `Payment` digunakan untuk menyimpan referensi ke objek PO yang terkait.

D.3 Pemetaan Akses Data ke API

Meskipun perancangan pada lapisan aplikasi menggunakan *class diagram*, seluruh akses data pada modul *Payment* CC dimediasi melalui lapisan *Web API*. Lapisan ini berfungsi sebagai penghubung antara objek aplikasi dan basis data dengan memanfaatkan *table-valued function* (TVF) untuk operasi baca data serta *stored procedure* (SP) untuk operasi penyimpanan dan penghapusan data.

1. TVF yang digunakan untuk operasi baca data meliputi `dbo.PaymentList`, `dbo.PaymentRead`, `dbo.PaymentInvoiceList`, dan `dbo.PaymentInvoiceRead`.

2. SP yang digunakan untuk operasi penyimpanan dan penghapusan data meliputi `dbo.PaymentImportSave`, `dbo.PaymentInvoiceSave`, `dbo.PaymentInvoiceDelete`, `dbo.PaymentDocumentSave`, dan `dbo.PaymentDocumentDelete`.

Kombinasi antara perancangan berbasis *class* dan pemetaan fungsi basis data melalui API ini memastikan bahwa modul *Payment* CC tetap konsisten dengan pendekatan *object-oriented* pada lapisan aplikasi, sekaligus tetap kompatibel dengan mekanisme penyimpanan data yang telah tersedia di GMMS.

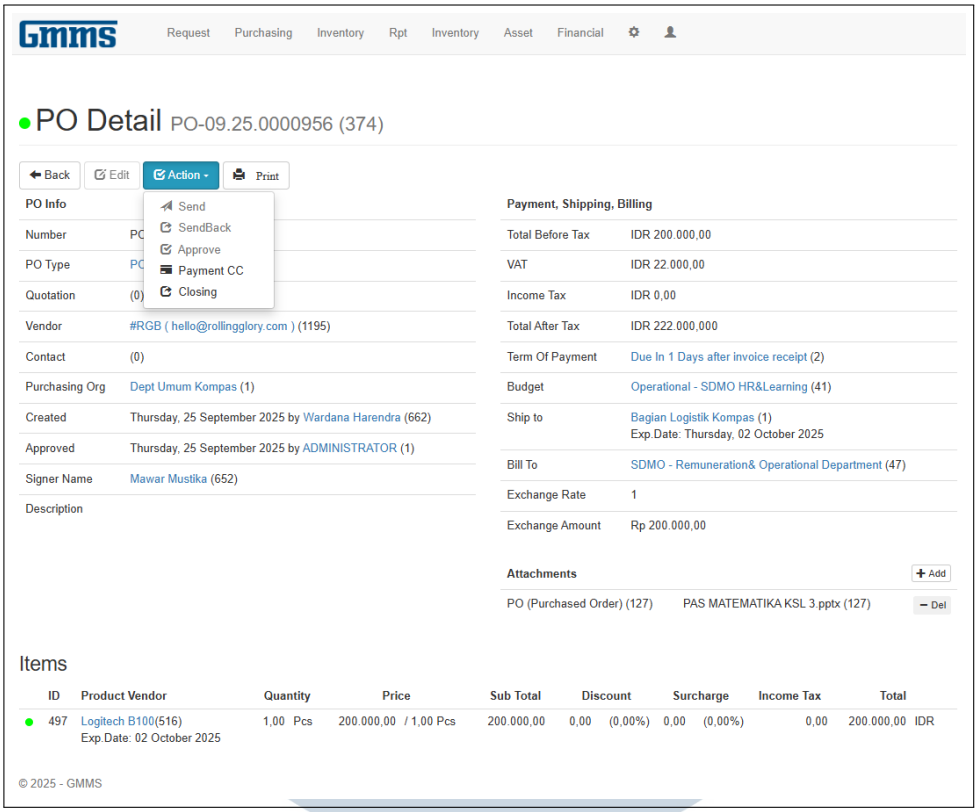
3.3.3 Implementasi

Implementasi pada modul *Payment* CC dilakukan berdasarkan rancangan yang telah dijabarkan pada tahap perancangan sistem. Seluruh antarmuka dibangun menggunakan teknologi berbasis web, yaitu *HTML*, *CSS*, *Bootstrap 3.4.1*, dan *jQuery*, serta menerapkan pola arsitektur *Model–View–Controller* yang digunakan secara konsisten pada sistem GMMS.

Tujuan utama implementasi ini adalah untuk menyediakan antarmuka yang informatif, interaktif, dan mudah digunakan, sehingga mendukung seluruh alur proses bisnis mulai dari pemicu pembuatan *Payment* CC, pengelolaan *invoice* dan lampiran, hingga tampilan cetak dokumen. Setiap halaman dan formulir dirancang agar memiliki keselarasan visual dan fungsional dengan modul-modul lain seperti PO dan *Request Processing* dalam GMMS.

A Dropdown Aksi di Halaman PO

Implementasi dimulai dari halaman PO *Detail*, di mana opsi *Payment* CC disediakan pada menu *Action*. Opsi ini hanya aktif apabila status PO berada pada kondisi *Approval* sesuai aturan bisnis yang berlaku. Ketika opsi dipilih, sistem menampilkan *form* konfirmasi pengaturan pengulangan pembayaran. Tampilan implementasi ditunjukkan pada Gambar 3.17.

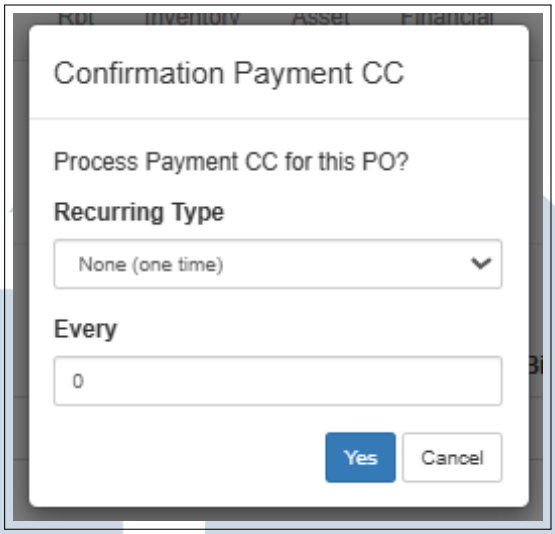


Gambar 3.17. Tampilan menu *Action* pada halaman PO dengan opsi *Payment* CC

B Form Konfirmasi Pengaturan Pengulangan Pembayaran

Setelah pengguna memilih opsi *Payment* CC, sistem menampilkan modal *pop-up* untuk menentukan nilai `RecurringType` dan `RecurringEvery`. *Form* ini merupakan bagian penting sebelum data baru dibentuk melalui API. Implementasinya dapat dilihat pada Gambar 3.18.

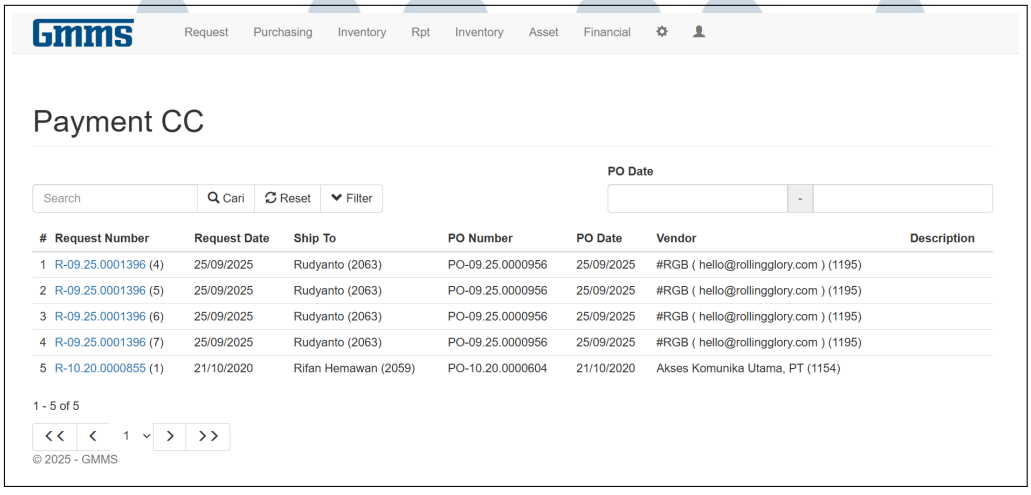
UNIVERSITAS MULTIMEDIA NUSANTARA



Gambar 3.18. *Form* konfirmasi pengaturan pengulangan pembayaran (*Recurring Type* dan *Recurring Every*)

C Halaman Daftar Payment (Payment List)

Halaman *Payment List* diimplementasikan untuk menampilkan seluruh dokumen *Payment* CC yang tersimpan pada sistem. Antarmuka menyediakan fitur pencarian, *filter* tanggal, kontrol jumlah baris, dan navigasi *pagination*. Tampilan halaman ditunjukkan pada Gambar 3.19.



Gambar 3.19. Tampilan halaman daftar Payment CC (Payment Index)

Komponen penting pada halaman ini meliputi:

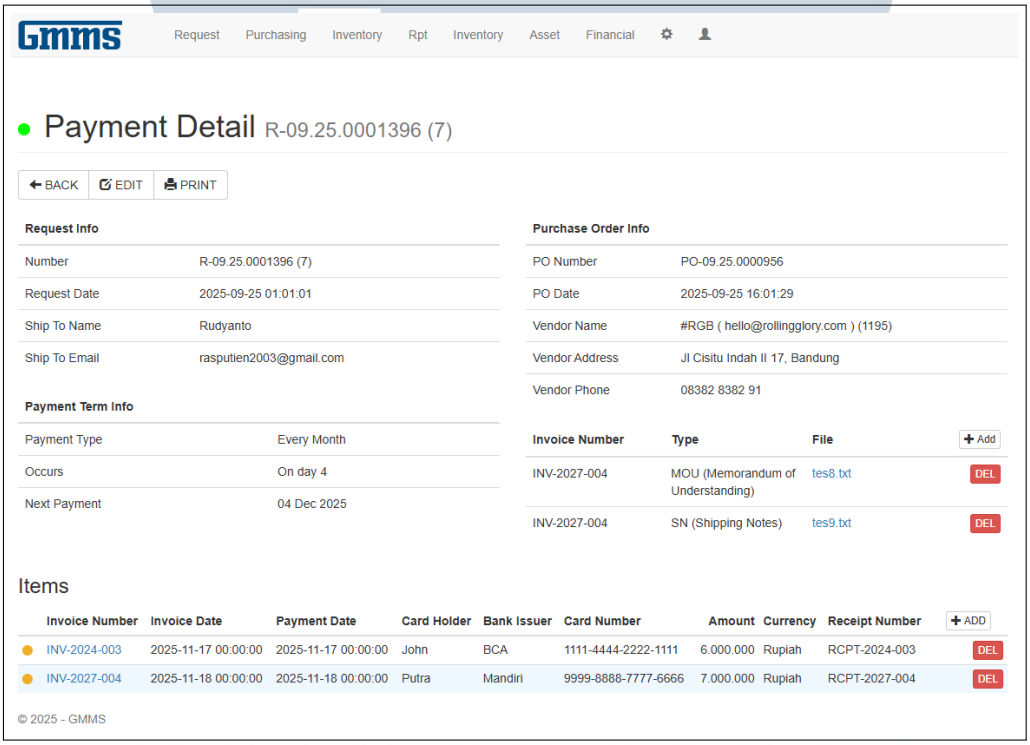
- Kolom pencarian dan *filter* tanggal.

- Tabel data berisi `Request Number`, `Request Date`, `PO Number`, `Vendor`, dan `Description`.

- Navigasi *pagination* dan pengaturan jumlah baris tampilan.

D Halaman Detail Payment (Payment Detail)

Halaman *Payment Detail* menampilkan informasi lengkap dokumen pembayaran, mulai dari data *header*, detail PO, hingga daftar *invoice* dan dokumen pendukung. Data pada halaman ini dimuat melalui pemanggilan fungsi `PaymentRead` dan `PaymentInvoiceList`. Tampilan implementasi halaman *Payment Detail* ditunjukkan pada Gambar 3.20.



Gambar 3.20. Tampilan halaman detail *Payment* CC (*Payment Detail*)

Elemen utama yang disajikan pada halaman ini meliputi:

- Informasi *header* dokumen pembayaran, seperti `Request Number`, `Request Date`, dan `Payment Term`.

- Informasi PO, meliputi `PO Number`, `PO Date`, serta data *Vendor*.

- Tabel daftar *invoice* yang menampilkan `Invoice Number`, `Invoice Date`, `Payment Date`, `Card Holder`, `Amount`, `Currency`, dan `Receipt Number`.
- Bagian *Attachment* yang menampilkan daftar dokumen pendukung berdasarkan *invoice*, lengkap dengan jenis dokumen dan nama berkas.
- Tombol aksi untuk menambah dan menghapus *invoice*, menambah dan menghapus *attachment*, serta mencetak dokumen pembayaran.

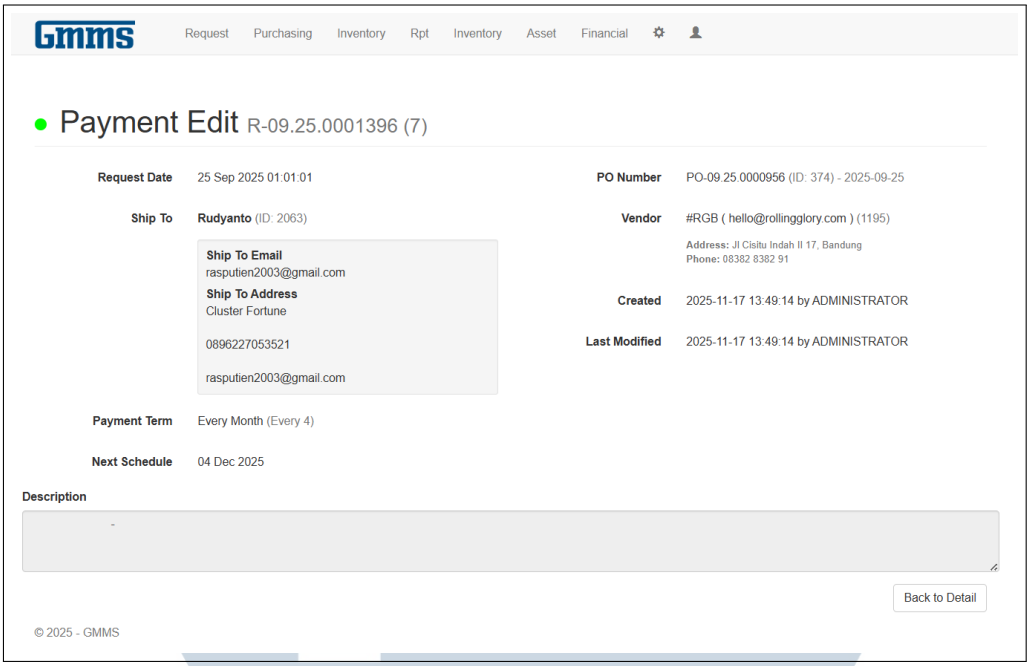
E Halaman Pengeditan Payment (Payment Edit)

Halaman *Payment Edit* digunakan sebagai halaman pengelolaan dokumen pembayaran yang telah terbentuk pada sistem. Meskipun dinamakan sebagai halaman *edit*, pada tahap implementasi ini sebagian besar informasi ditampilkan dalam kondisi *read-only* dan tidak dapat diubah secara langsung oleh pengguna.

Pembatasan tersebut diterapkan karena data pembayaran bersumber dari modul (PO) dan telah melalui tahapan validasi sebelumnya. Perubahan langsung pada halaman *Payment Edit* berpotensi menyebabkan ketidaksesuaian data antara modul *Payment* dan modul PO, sehingga sistem hanya menyediakan fasilitas peninjauan informasi untuk menjaga konsistensi dan integritas data.

Data yang ditampilkan pada halaman ini diperoleh melalui pemanggilan fungsi `PaymentRead`, yang menampilkan informasi PO, *vendor*, *ship to*, ketentuan pembayaran, serta jadwal pembayaran berikutnya. Tampilan implementasi halaman *Payment Edit* ditunjukkan pada Gambar 3.21.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.21. Tampilan halaman *Payment Edit* pada modul *Payment* CC

Informasi yang disajikan pada halaman ini meliputi:

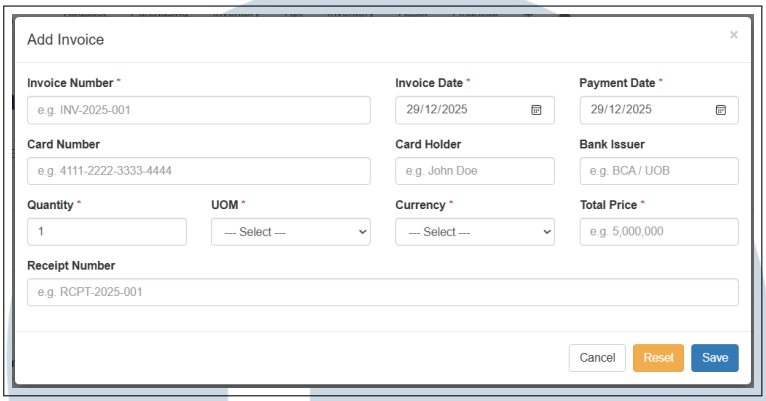
- Informasi utama dokumen pembayaran, seperti `Request Number`, `Request Date`, `PO Number`, dan `Vendor`.
- Informasi tujuan pengiriman (`Ship To`) beserta detail alamat dan kontak pendukung.
- Ketentuan pembayaran, termasuk `Payment Term` dan `Next Schedule`.
- Bagian `Description` yang disediakan dalam bentuk kotak teks *read-only* sebagai penjelasan tambahan terkait dokumen pembayaran.

F Form Tambah Invoice (Add Invoice)

Form *Add Invoice* ditampilkan dalam bentuk modal *pop-up* ketika pengguna memilih tombol `Add Invoice` pada halaman *Payment Detail*. Form ini digunakan untuk menambahkan data *invoice* baru yang terkait dengan dokumen pembayaran (*Payment* CC).

Seluruh kolom pada form berada dalam kondisi aktif dan dapat diisi oleh pengguna. Selain tombol `Save` dan `Cancel`, form ini juga menyediakan tombol `Reset` yang berfungsi untuk mengosongkan seluruh isian dan mengembalikan

form ke kondisi awal. Tampilan implementasi form *Add Invoice* ditunjukkan pada Gambar 3.22.



Gambar 3.22. Tampilan *form pop-up* tambah *invoice* (*Add Invoice*)

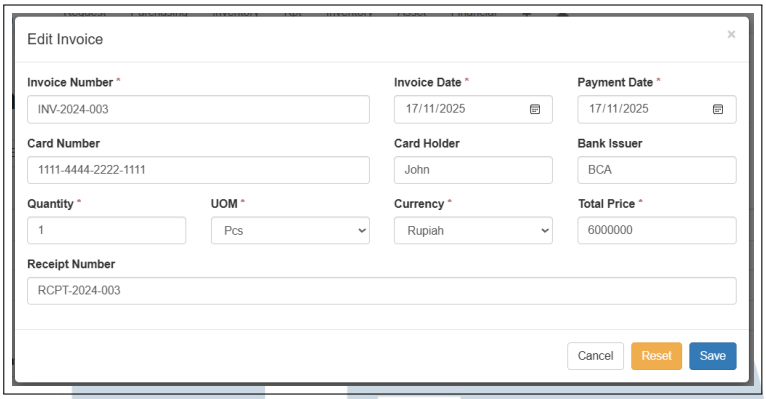
Kolom yang disediakan pada form ini meliputi:

- Informasi utama *invoice*, yaitu `Invoice Number`, `Invoice Date`, dan `Payment Date`.
- Informasi kartu pembayaran, meliputi `Card Number`, `Card Holder`, dan `Bank Issuer`.
- Detail transaksi, yaitu `Quantity`, `UOM`, `Currency`, dan `Total Price`.
- Informasi pendukung berupa `Receipt Number`.

G Form Edit Invoice

Form *Edit Invoice* ditampilkan dalam bentuk modal *pop-up* ketika pengguna memilih salah satu `Invoice Number` yang telah tersimpan pada daftar *invoice*. Form ini digunakan untuk melakukan penyesuaian atau koreksi terhadap data *invoice* yang sudah ada.

Pada saat form dibuka, seluruh kolom akan terisi secara otomatis berdasarkan data *invoice* yang dipilih. Pengguna dapat memperbarui nilai pada kolom tertentu sesuai kebutuhan. Form *Edit Invoice* juga dilengkapi dengan tombol `Reset` yang berfungsi untuk mengosongkan seluruh isian dan mengembalikan form ke kondisi awal. Tampilan implementasi form *Edit Invoice* ditunjukkan pada Gambar 3.23.



Gambar 3.23. Tampilan *form pop-up* edit *invoice*

Kolom yang disediakan pada form ini meliputi:

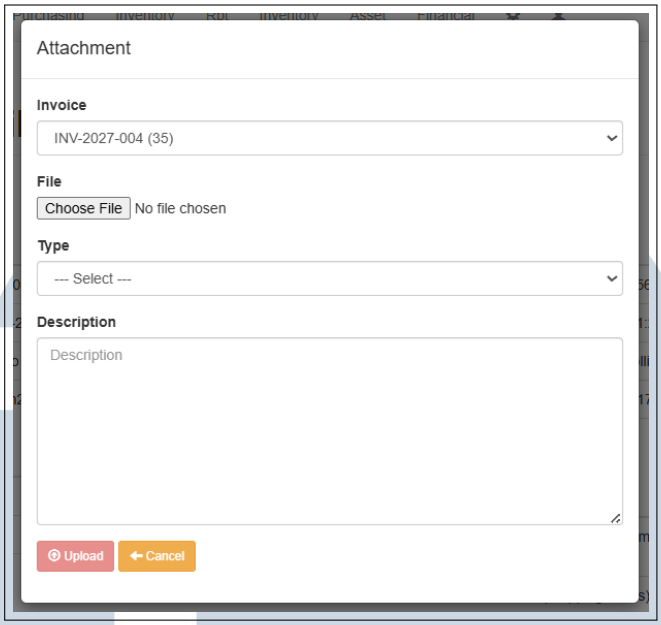
- Informasi utama *invoice*, yaitu `Invoice Number`, `Invoice Date`, dan `Payment Date`.
- Informasi kartu pembayaran, meliputi `Card Number`, `Card Holder`, dan `Bank Issuer`.
- Detail transaksi, yaitu `Quantity`, `UOM`, `Currency`, dan `Total Price`.
- Informasi pendukung berupa `Receipt Number`.

H Form Tambah Lampiran (Add Attachment)

Form *Add Attachment* digunakan untuk mengunggah dokumen pendukung yang dikaitkan dengan suatu *invoice*. Form ini ditampilkan dalam bentuk modal *pop-up* ketika pengguna memilih aksi penambahan lampiran pada halaman *Payment Detail*.

Pada tahap awal, pengguna diwajibkan memilih *invoice* tujuan melalui komponen `Invoice`. Pemilihan *invoice* ini berfungsi untuk memastikan bahwa setiap dokumen yang diunggah terasosiasi secara tepat dengan data *invoice* yang bersangkutan.

Setelah *invoice* dipilih, pengguna dapat mengunggah berkas lampiran serta melengkapi informasi pendukung lainnya. Tampilan implementasi form *Add Attachment* ditunjukkan pada Gambar 3.24.



Gambar 3.24. *Form pop-up* tambah lampiran (*Add Attachment*)

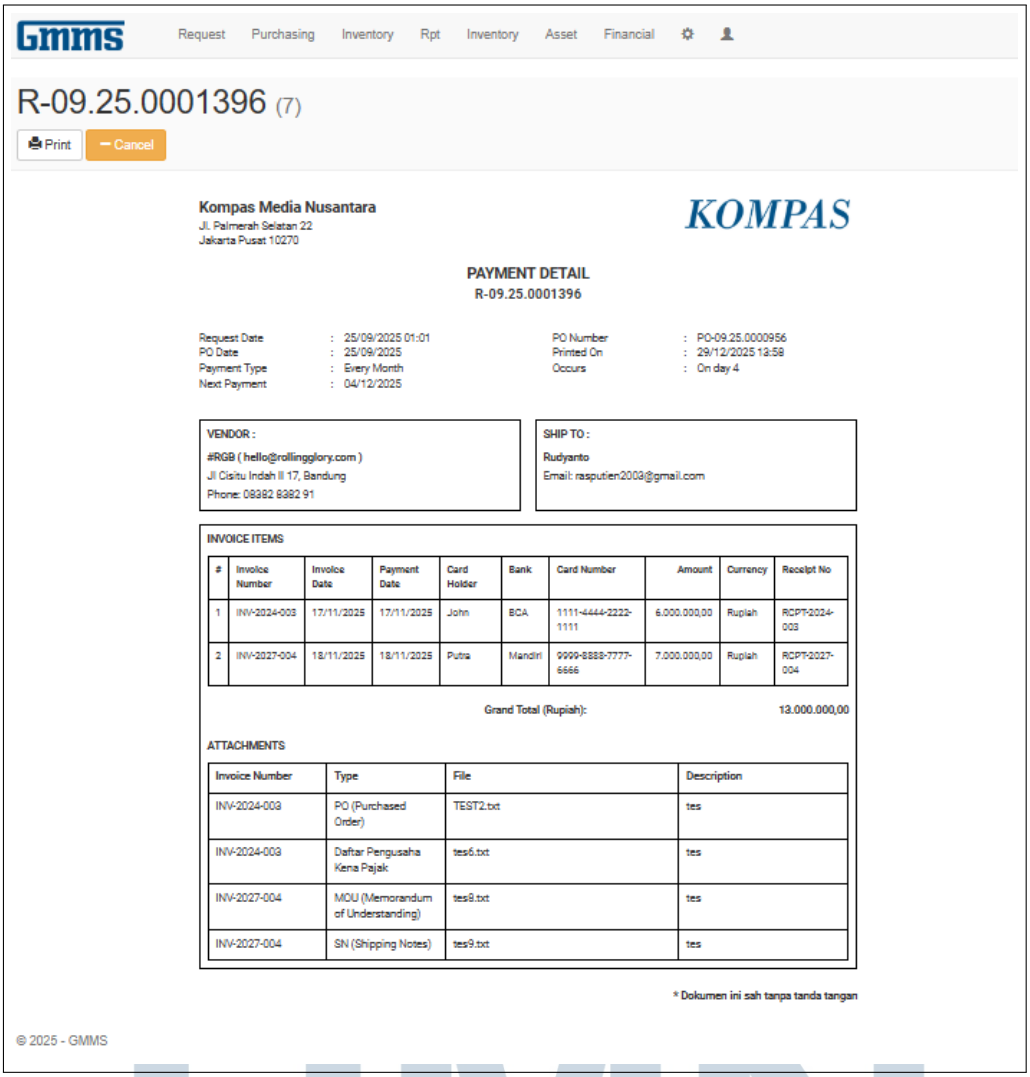
Komponen yang disediakan pada form ini meliputi:

- `Invoice`: pilihan *invoice* yang akan dikaitkan dengan lampiran.
- `File`: unggahan dokumen pendukung.
- `Type`: kategori atau jenis dokumen lampiran.
- `Description`: keterangan tambahan terkait lampiran.

I Halaman Cetak Dokumen (Print View)

Halaman *Print View* menyediakan tampilan khusus untuk pencetakan dokumen *Payment* CC dalam format siap cetak. Sebelum ditampilkan, data dimuat ulang melalui pemanggilan fungsi `PaymentRead`, `PaymentInvoiceList`, dan data lampiran terkait untuk memastikan informasi yang dicetak merupakan data terbaru dan konsisten dengan basis data.

Halaman ini dirancang menyerupai dokumen resmi perusahaan, dengan struktur yang terorganisasi dan mudah dibaca. Tampilan implementasi halaman cetak ditunjukkan pada Gambar 3.25.



Gambar 3.25. Tampilan halaman cetak dokumen *Payment* CC (*Print View*)

Struktur utama halaman cetak meliputi:

- Bagian *header* dokumen yang menampilkan identitas perusahaan, nomor dokumen *Payment*, serta informasi waktu pencetakan.
- Informasi ringkas dokumen pembayaran, seperti `Request Date`, `PO Number`, `Payment Type`, dan jadwal pembayaran berikutnya.
- Informasi pihak terkait, meliputi data *Vendor* dan *Ship To* yang disajikan dalam bentuk blok informasi terpisah.
- Bagian `Invoice Items` yang menampilkan daftar *invoice*, termasuk `Invoice Number`, tanggal *invoice*, informasi kartu pembayaran, nilai transaksi, mata uang, dan `Receipt Number`.

- Informasi `Grand Total` yang menampilkan total keseluruhan nilai transaksi *invoice* dalam satuan mata uang yang digunakan.
- Bagian `Attachments` yang menampilkan daftar dokumen pendukung, dikelompokkan berdasarkan *invoice*, beserta jenis dokumen, nama berkas, dan deskripsi singkat.
- Pada bagian akhir halaman ditampilkan keterangan bahwa dokumen hasil cetak bersifat informatif dan dihasilkan secara sistem, sehingga tidak memerlukan tanda tangan basah.

Seluruh halaman dalam modul *Payment* CC terhubung mengikuti alur proses bisnis yang telah dijelaskan pada bagian sebelumnya. Urutan implementasi alur adalah sebagai berikut:

1. Pengguna membuka halaman PO *Detail* dan memilih menu *Payment* CC.
2. Sistem menampilkan *form* pengaturan pengulangan dan membentuk data baru.
3. Halaman *Payment Detail* terbuka secara otomatis untuk menampilkan data yang baru dibuat.
4. Pengguna dapat menambah *invoice*, mengunggah lampiran, mengedit data, atau mencetak dokumen.
5. Pengguna kembali ke halaman *Payment List* untuk melihat seluruh riwayat data.

Implementasi antarmuka ini memastikan bahwa seluruh rangkaian interaksi berjalan konsisten, mudah dipahami, dan mendukung efisiensi dalam pengelolaan pembayaran berbasis kartu kredit pada sistem GMMS.

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kegiatan magang di PT Kompas Media Nusantara, berbagai kendala teknis dan nonteknis muncul selama proses pengembangan dan implementasi fitur *Payment* CC pada sistem GMMS. Kendala-kendala tersebut timbul pada tahap adaptasi sistem, konfigurasi proyek, integrasi antar modul, serta pengujian sistem. Bagian ini memaparkan kendala utama yang dihadapi beserta solusi yang diterapkan untuk mengatasinya.

3.4.1 Kendala

1. Kesulitan memahami struktur sistem GMMS: Pada tahap awal, kesulitan muncul dalam memahami arsitektur GMMS karena dokumentasi teknis yang terbatas serta kondisi sistem yang telah berjalan cukup lama. GMMS memiliki modul terintegrasi seperti PO, *Payment*, dan *Approval Workflow* yang saling bergantung melalui tabel dan logika bisnis yang kompleks. Proses adaptasi terhadap kode dan struktur data memerlukan waktu tambahan untuk memahami dependensi antar modul.
2. Perbedaan versi dan ketergantungan pustaka proyek: Saat melakukan kompilasi proyek ASP.NET MVC di *Visual Studio 2015*, muncul peringatan `Found conflicts between different versions of the same dependent assembly`. Masalah ini terjadi karena perbedaan versi pustaka antara proyek API dan *Website*, serta ketidaksesuaian dengan versi `.NET Framework` yang digunakan.
3. Ketidaksesuaian struktur data antara *Website* dan API: Ditemukan perbedaan nama kolom dan tipe data antara model di proyek *Website* dengan struktur hasil keluaran API. Hal ini menyebabkan kesalahan konversi dan nilai `null` pada saat pemanggilan *endpoint* seperti `/Payment/read` dan `/Payment/invoiceList`.
4. Masalah komunikasi antara *Website* dan API: Beberapa permintaan dari *Website* ke API gagal karena konfigurasi *base URL* berbeda antara lingkungan pengembangan lokal (`localhost`) dan *server* uji coba (`gmms.kgmedia.id`). Kondisi ini menyebabkan respons dari API tidak terbaca oleh `Ajax` di sisi klien.
5. Keterbatasan akses jaringan internal: Beberapa pengujian hanya dapat dilakukan melalui jaringan internal Kompas, sehingga saat bekerja dari luar kantor, sistem tidak dapat dijalankan tanpa koneksi VPN ke jaringan perusahaan. Hal ini menghambat efisiensi waktu saat *debugging* dan pengujian fitur.

3.4.2 Solusi

Untuk mengatasi kendala-kendala tersebut, dilakukan beberapa langkah penyelesaian sebagai berikut:

1. Studi mandiri dan analisis kode eksisting: Pemahaman terhadap struktur proyek GMMS diperoleh dengan menelusuri *controller*, *model*, dan *view* pada setiap modul secara menyeluruh. File seperti `PaymentController.cs`, `PaymentViewModel.cs`, dan `FormModel.cs` dipelajari untuk memahami alur data dan pemetaan antar model, sehingga mempercepat proses adaptasi terhadap sistem.

2. Sinkronisasi versi referensi dan konfigurasi proyek: Konflik pustaka diatasi dengan menyelaraskan referensi antar proyek (`API.csproj` dan `Website.csproj`) serta memperbarui file `Web.config` dan `packages.config` agar sesuai dengan `.NET Framework 4.8`. Dependensi pihak ketiga juga disesuaikan melalui `NuGet Package Manager` untuk memastikan kompatibilitas penuh.

3. Penyelarasan model dan tipe data API–*Website*: Seluruh model seperti `PaymentModel`, `PaymentInvoiceModel`, dan `PaymentViewModel` diperbarui agar sesuai dengan hasil keluaran API. Properti yang berpotensi kosong diberi tipe `nullable`, dan pengecekan `null` ditambahkan sebelum proses binding di *View*.

4. Standarisasi konfigurasi koneksi API: Variabel global seperti `varGlobalGMMSWebUrl` dan `varGlobalGMMSAPIUrl` diterapkan agar rute permintaan konsisten di semua lingkungan. Mekanisme deteksi *environment* ditambahkan agar sistem otomatis menyesuaikan URL sesuai mode pengujian.

5. Penggunaan VPN untuk akses jaringan internal: Dengan konfigurasi `OpenVPN` ke jaringan internal Kompas, akses terhadap sistem dapat dilakukan dari luar kantor sehingga memungkinkan pengujian API `/gmmsapi/Payment/list` dan `/gmmsapi/Payment/save` secara langsung.

UNIVERSITAS MULTIMEDIA NUSANTARA