

BAB 3

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama menjalani masa magang di PT Rintis Sejahtera, tanggung jawab utama penulis adalah mengembangkan sistem pengelolaan data berbasis web untuk mengefisienkan kerja staf di perusahaan. Pengerjaan ini dilakukan di bawah arahan dan bimbingan Bapak Iyan Perdana, Bapak Adrian, Bapak Stevan Del Arisandi dan Ibu Yunita Wirawan serta bantuan dari tim IT *E-Payment* yang lain. Semua koordinasi dilakukan secara luring dari *design* web, dan *review* web. Selain pengembangan web tersebut, penulis juga diberikan beberapa kerjaan sampingan seperti membuat presentasi, dan memasukkan data XML.

Kegiatan ini melibatkan kolaborasi dengan berbagai divisi untuk memastikan bahwa web ini berjalan lancar dan efektif serta memenuhi kebutuhan operasional. Proses pengembangan dilakukan dengan memanfaatkan PHP *Native* sebagai *Back-End*, HTML/CSS/JS sebagai *Front-End*, dan MySQL sebagai databasenya. Departemen IT *E-Payment* hanya memantau perkembangan tugas setiap anggota tim dengan memantau melewati Github. Penamaan *branch* tidak disesuaikan dengan jenis pekerjaan, sehingga siapapun yang mendapat akses untuk mengedit ke Github dapat mengubah kode atau data di dalamnya. Setelah pengerjaan *branch* sudah baik maka *branch* tersebut penulis melakukan penggabungan (*merge*) ke dalam *branch* utama penulis.[5, 6]

3.2 Tugas yang Dilakukan

Kegiatan selama program magang dibagi menjadi dua tahap. Tahap pertama berfokus pada pembelajaran dan analisis awal, mendalami pemahaman PHP *native* sebagai *backend*, HTML/CSS/JS sebagai *frontend*, serta mempelajari sistem kerja *switching* dalam jaringan ATM dan sistem keamanan transaksi *E-Payment*. Selain itu, dilakukan peninjauan terhadap sistem dan aplikasi terdahulu untuk mengetahui kelebihan, kendala, serta hal-hal yang perlu diperhatikan atau dihindari.

Tahap kedua adalah tahap perencanaan dan pengembangan. Pada tahap ini, dilakukan pertemuan dengan pihak divisi IT *E-Payment* PT Rintis untuk menyampaikan ide dan memastikan kebutuhan sistem. Selanjutnya dilakukan analisis kebutuhan dengan memperhatikan aspek fungsi, operasional, dan

keamanan. Hasil analisis tersebut digunakan sebagai dasar untuk:

1. Merancang dan mengembangkan komponen web yang bersifat dinamis dan dapat digunakan kembali (*reusable*).
2. Mengimplementasikan fitur *export file* ke dalam format Excel dan PDF untuk kebutuhan pelaporan dan dokumentasi.
3. Melakukan *System Integration Training* (SIT) untuk memastikan kestabilan dan kelancaran sistem saat digunakan di *server* perusahaan.

3.3 Uraian Pelaksanaan Magang

3.3.1 Proyek yang dibuat

Tabel 3.1 menunjukkan rincian linimasa kegiatan yang dilakukan setiap minggunya selama masa magang di PT Rintis Sejahtera.



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

| Minggu Ke - | Pekerjaan yang dilakukan |
|-------------|---|
| 1 | Membuat rencana dan presentasi rencana pembuatan proyek, pengaturan lingkungan pengembangan menggunakan Github, Ionic React, dan Firebase. Pembuatan prototipe di Figma. Pembuatan fitur utamanya beserta CRUD. |
| 2 | Mengganti <i>database</i> dan bahasa pemrograman dengan menggunakan MySQL sebagai <i>database</i> dan PHP sebagai bahasa pemrogramannya. Penambahan fitur-fitur yang diinginkan oleh staf. |
| 3 | Merancang antarmuka <i>user (user interface)</i> khususnya untuk entitas seperti bank dan <i>biller</i> , termasuk desain visual, pembuatan <i>form input</i> , dan pengujian fungsi dasar. |
| 4 | Melakukan pengecekan akhir dan menyerahkan program kepada staf serta mengevaluasi <i>improvement</i> yang dapat diimplementasikan. |
| 5 | Memulai transisi ke proyek validasi JSON dan mengembangkan logika dasar validasi struktur dan format <i>input</i> JSON. |
| 6 | Mengembangkan fitur <i>edit</i> dan <i>delete</i> data serta memperluas validasi JSON berdasarkan skenario data nyata. |
| 7 | Mengimplementasikan fitur <i>resubmit</i> dan melakukan <i>debugging</i> terhadap <i>error</i> struktur JSON. |
| 8 | Perbaikan UI/UX dan pemisahan validasi data <i>acquirer</i> dan <i>issuer</i> agar lebih modular. |
| 9 | Penambahan fitur <i>delete</i> entitas dan laporan eror otomatis pada hasil validasi. |
| 10 | Implementasi penuh fitur <i>resubmit</i> untuk menyederhanakan mekanisme edit data. |
| 11 | Validasi ditingkatkan agar mampu mendeteksi lebih dari satu <i>error</i> sekaligus. |
| 12 | Instalasi <i>software</i> pendukung dan penyusunan dokumentasi serta materi presentasi internal. |
| 13 | Penyempurnaan tampilan tabel validasi JSON dan riset CPM dan MPM. |
| 14 | Presentasi internal dan implementasi ekspor data ke Excel. |
| 15 | Penyelesaian fitur <i>multi-step validation</i> dan perapihan dokumentasi teknis. |
| 16 | Perubahan indeks JSON menjadi dinamis serta <i>debugging</i> final validasi CPM dan MPM. |

3.4 Analisis Dan Perancangan Sistem

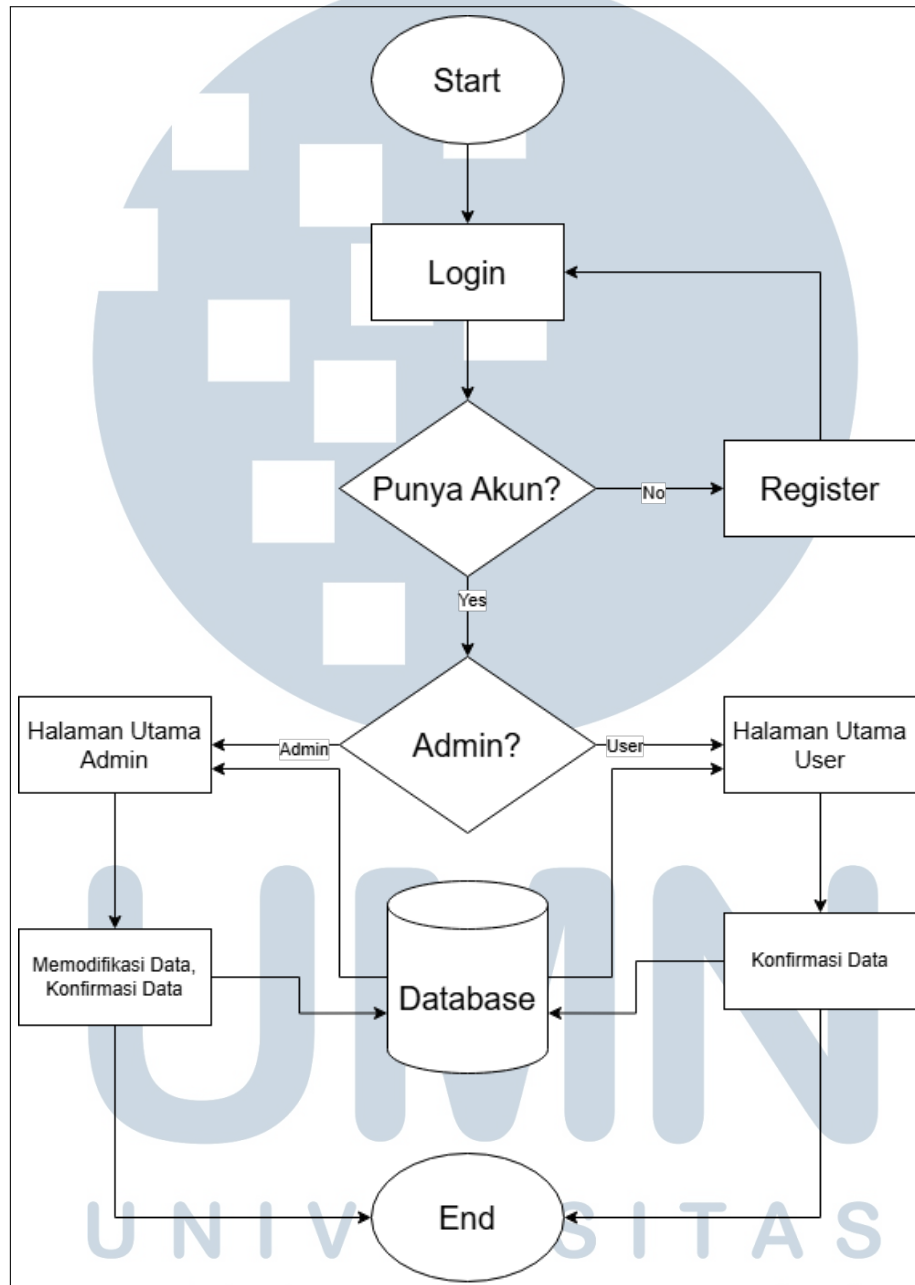
Dalam pelaksanaan kegiatan magang, perancangan sistem untuk kedua proyek yaitu sistem CRUD dan sistem validasi data JSON dengan fitur *multi-step validation* saling berkaitan dan saling mendukung dalam satu kesatuan sistem yang terintegrasi. Meskipun keduanya memiliki ruang lingkup teknis yang berbeda, keduanya dikembangkan untuk saling melengkapi dan memastikan sistem dapat memenuhi kebutuhan *user* secara menyeluruh. Untuk menjamin sistem berfungsi dengan baik dan sesuai dengan kebutuhan, penulis melakukan analisis kebutuhan secara menyeluruh dan merancang sistem secara sistematis dengan bantuan berbagai jenis diagram yang umum digunakan dalam pengembangan perangkat lunak.

Pada proyek utama, fokus pengembangan diarahkan pada penyempurnaan proses validasi melalui fitur *single-step validation* dan *multi-step validation* serta integrasi ekspor hasil ke format Excel. Berdasarkan analisis kebutuhan, sistem harus mampu melakukan validasi bertahap sesuai jenis data seperti *Customer Presented Mode* (CPM) dan *Merchant Presented Mode* (MPM) serta menghasilkan *output* yang dapat diunduh *user* dalam bentuk file Excel. Untuk menggambarkan fitur ini.

Penulis juga menggunakan *Flowchart* untuk menggambarkan alur kerja programnya secara sederhana agar mendapat gambaran koneksi webnya kemana saja. Walaupun penulis sudah membuat diagram lainnya dari pihak perusahaan tidak memperbolehkan penulis untuk menunjukkannya.

Melalui pendekatan analisis dan perancangan sistematis ini, penulis memastikan bahwa sistem yang dikembangkan tidak hanya terstruktur dengan baik, tetapi juga fleksibel untuk pengembangan lebih lanjut dan efektif dalam memenuhi kebutuhan *user*. Diagram-diagram yang penulis buat menjadi landasan penting dalam proses implementasi teknis serta pemeliharaan sistem di masa mendatang.

3.4.1 Flowchart Proyek



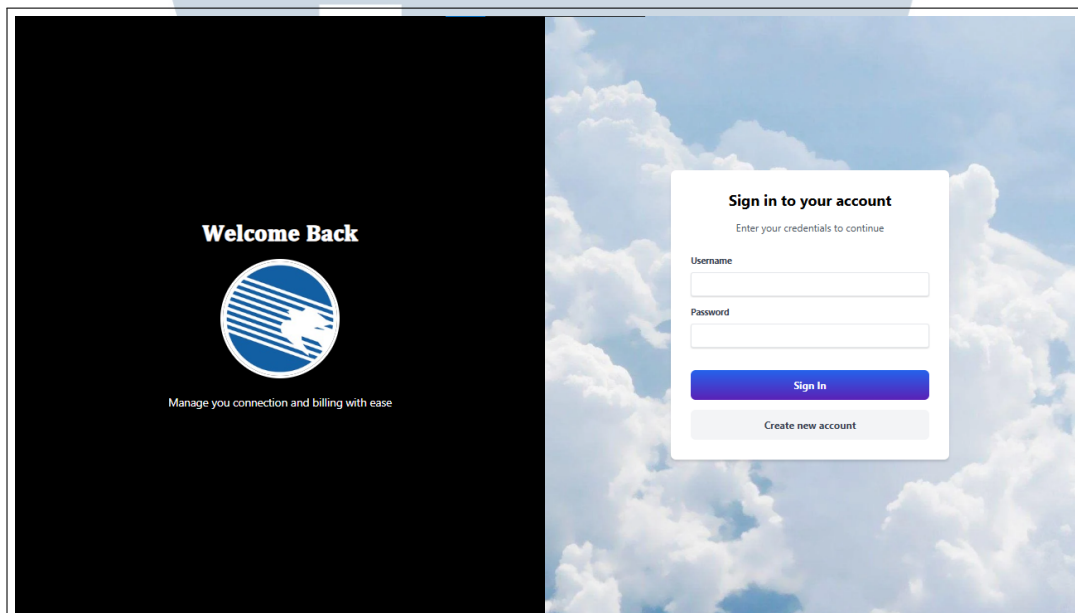
Gambar 3.1. Flowchart Sistem

Flowchart ini menjelaskan Alur sistem dimulai dari halaman *login* atau *register*, di mana pengguna (baik *admin* maupun *user* biasa) harus terlebih dahulu melakukan autentikasi. Setelah *login*, sistem membedakan akses berdasarkan peran: *admin* dapat mengelola data dalam web, dan mengelola akun pengguna,

sedangkan *user* biasa fokus pada *input* dan validasi data JSON dalam skenario CPM, MPM, atau *multi-step validation*. Setiap *input* JSON divalidasi secara *real-time* jika gagal, pengguna dapat melihat detail *error* lalu *resubmit* setelah memperbaiki datanya; jika berhasil, data disimpan ke *database* MySQL dan dapat diekspor ke Excel. Seluruh aktivitas berakhir dengan *logout*, yang menghentikan sesi pengguna. Alur ini memastikan sistem bersifat aman, modular, dan mendukung kebutuhan operasional divisi IT *E-Payment* di PT Rintis Sejahtera.[7]

3.4.2 Hasil Implementasi

A Login



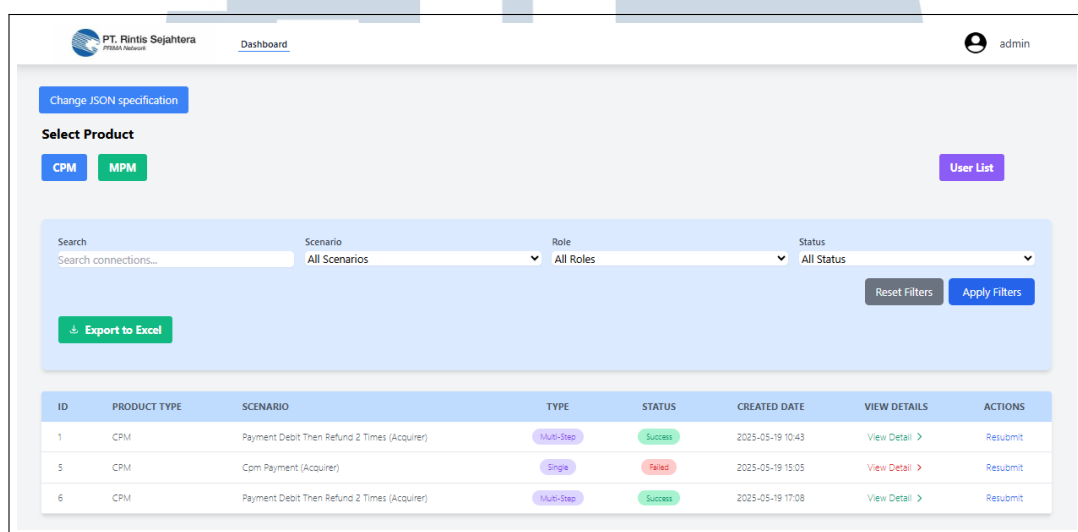
Gambar 3.2. Halaman *Login*

Halaman *login* yang dapat dilihat di Gambar 3.2 merupakan pintu gerbang autentikasi untuk web Rintis. Halaman ini menampilkan dua kolom. Pada bagian kiri terdapat latar belakang hitam yang menampilkan logo Rintis dalam wadah lingkaran dengan *border* putih, dilengkapi judul “*Welcome Back*” dan *tagline* tentang kemudahan mengelola koneksi dan *billing*. Bagian ini akan tersembunyi pada tampilan *mobile* untuk optimasi ruang dan pada bagian kanan ada *box* putih yang menunjukkan *sign in*.

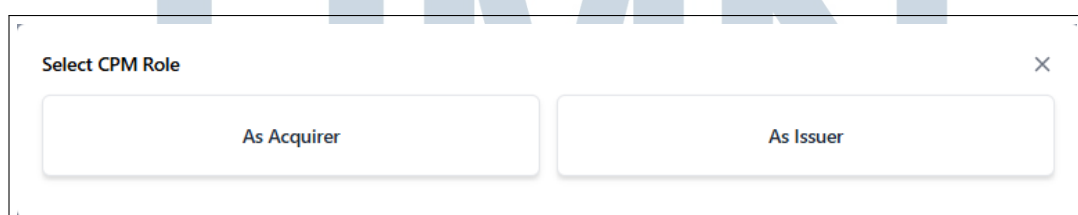
Di bagian atas *form* terdapat judul “*Sign in to your account*” dan *subtitle* yang mengarahkan *user* untuk memasukkan kredensial mereka. *Form* memiliki

dua input utama: *username* dan *password*, keduanya dilengkapi label yang jelas dan validasi *required*. Sistem autentikasi bekerja dengan mengecek kredensial *user* di *database*. *Password* diverifikasi menggunakan fungsi untuk membandingkan dengan data yang tersimpan. Setelah verifikasi sistem menyimpannya di *database*. [8][9]

B Halaman Utama atau *Index*



Gambar 3.3. Halaman Utama atau *Index*



Gambar 3.4. Tampilan CPM

Halaman pada Gambar 3.3 merupakan halaman utama dari *website* PrimaSwitch yang dirancang untuk membantu *user* dalam mengelola dan memantau berbagai skenario transaksi pembayaran. *Dashboard* ini menyajikan antarmuka yang komprehensif untuk menangani dua jenis produk utama, yaitu CPM pada Gambar 3.4 dan MPM. Melalui halaman ini, *user* dapat mengakses berbagai fitur penting yang berkaitan dengan pengelolaan transaksi secara efisien dan aman.

Bagian utama dari *dashboard* menampilkan tabel data transaksi yang berisi informasi rinci mengenai setiap transaksi. Informasi tersebut meliputi ID transaksi, jenis produk, skenario yang digunakan, tipe transaksi, status transaksi, tanggal pembuatan, serta opsi untuk melihat detail dan melakukan tindakan lanjutan terhadap transaksi tersebut.

Untuk keperluan analisis atau pelaporan, sistem menyediakan fitur ekspor data dalam bentuk tombol “*Export to Excel*” yang memungkinkan *user* mengunduh data yang ada di tabel *list*. File ekspor ini akan diberi nama otomatis dengan format excel. Selama proses ekspor berlangsung, sistem menampilkan status pemrosesan (*loading*), memberikan notifikasi keberhasilan atau kegagalan, dan mempertahankan filter yang sedang aktif. Setelah berhasil, file akan langsung terunduh secara otomatis.

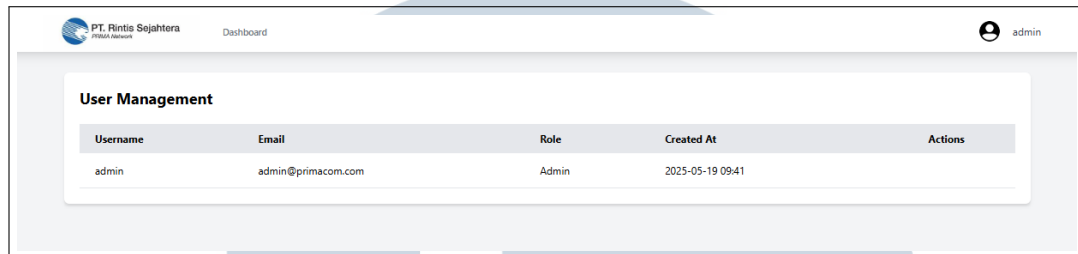
Dashboard juga dilengkapi dengan berbagai modal dan *pop-up* interaktif untuk memudahkan *user*. Beberapa di antaranya adalah modal skenario produk untuk memilih jenis skenario, modal input untuk memasukkan parameter transaksi, modal detail yang menyajikan informasi transaksi secara lengkap, serta modal *multi-step* untuk skenario yang memerlukan proses bertahap.

Gambar 3.3 menunjukkan bahwa *dashboard* dilengkapi dengan sistem filter yang komprehensif yang memungkinkan *user* untuk menyaring data berdasarkan beberapa kriteria: kotak pencarian untuk mencari koneksi spesifik, *filter* skenario untuk memilih jenis transaksi (CPM *Payment*, CPM *Reversal*, MPM *Static QRIS 26*, MPM *Dynamic QRIS 26*, dll), *filter* peran untuk memilih antara *Acquirer* atau *Issuer*, dan filter status untuk melihat transaksi berdasarkan kondisi (*Success*, *Failed*, *Not Tested*).

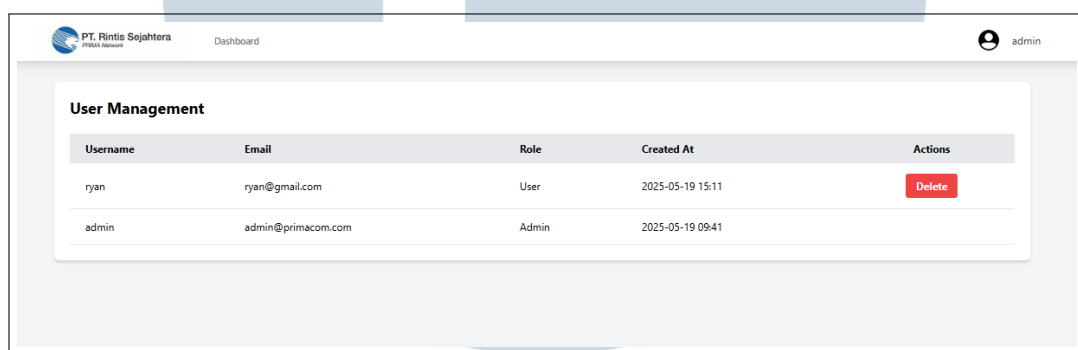
Dalam hal navigasi dan akses, *dashboard* menyediakan tombol pemilih produk (CPM/MPM), akses ke daftar *user* yang hanya tersedia untuk admin, serta sistem *pagination* untuk mempermudah penelusuran data dalam jumlah besar.

Dari sisi keamanan dan validasi, sistem menerapkan berbagai mekanisme perlindungan, termasuk validasi input untuk memastikan data sesuai format yang diharapkan, pengecekan peran *user* untuk membatasi akses berdasarkan hak, serta sanitasi data guna mencegah serangan injeksi atau eksploitasi keamanan lainnya. Seluruh fitur ini menjadikan *dashboard* PrimaSwitch sebagai pusat kendali yang andal dan aman untuk pengelolaan transaksi.

C *User List*



Gambar 3.5. Halaman *User List* Jika Tidak Ada *User*



Gambar 3.6. Halaman *User List* Jika Ada *User*

Halaman *Admin User Management* merupakan antarmuka khusus dalam sistem PrimaSwitch yang dirancang untuk administrator dalam mengelola data *user*. Halaman ini hanya dapat diakses oleh *user* yang memiliki peran sebagai *admin*, yang ditampilkan pada Gambar 3.5. Fitur ini menjadi bagian penting dari sistem karena memberikan kontrol penuh terhadap manajemen akun *user*, dengan tetap mengutamakan keamanan dan integritas data.

Dari sisi fitur keamanan, sistem menerapkan *role-based access control* untuk memastikan bahwa hanya *admin* yang memiliki akses ke halaman ini. Untuk mencegah kesalahan fatal, sistem tidak mengizinkan admin menghapus akun mereka sendiri terlihat pada Gambar 3.6.

Halaman ini memiliki komponen utama berupa Halaman *user* yang menampilkan daftar lengkap seluruh *user* dalam sistem. Tabel ini mencakup kolom seperti *Username*, *Email*, *Role*, Tanggal Pembuatan, serta kolom Aksi yang menyediakan tombol hapus untuk setiap *user*, kecuali untuk akun admin yang sedang *login*.

Adapun alur kerja halaman dimulai ketika admin mengakses halaman sebagai *user*. Sistem kemudian memverifikasi status admin *user*. Jika *user* bukan admin, maka akan langsung diarahkan ke halaman utama (*index*). Setelah berhasil diverifikasi, sistem menampilkan daftar *user* dalam bentuk tabel. Admin dapat memilih untuk menghapus *user* dengan menekan tombol *Delete*, yang kemudian akan memunculkan dialog konfirmasi. Setelah konfirmasi diberikan, sistem akan menghapus data *user* beserta log aktivitasnya, dan notifikasi hasil operasi akan ditampilkan sesuai dengan keberhasilan atau kegagalan proses tersebut.

D Profile

The screenshot displays the 'Profile Settings' interface within a web dashboard. The header includes the company logo 'PT. Rintis Sejahtera PRIMA Network' and a 'Dashboard' link. A user profile icon and the name 'admin' are in the top right. The main content area has a light green background. A white form titled 'Profile Settings' is centered, containing a 'Profile Picture' placeholder, a 'Username' field with the value 'admin', an 'Email' field with 'admin@primacom.com', and two password fields labeled 'Current Password' and 'New Password'. A blue 'Save Changes' button is at the bottom of the form.

Gambar 3.7. Halaman *Profile*

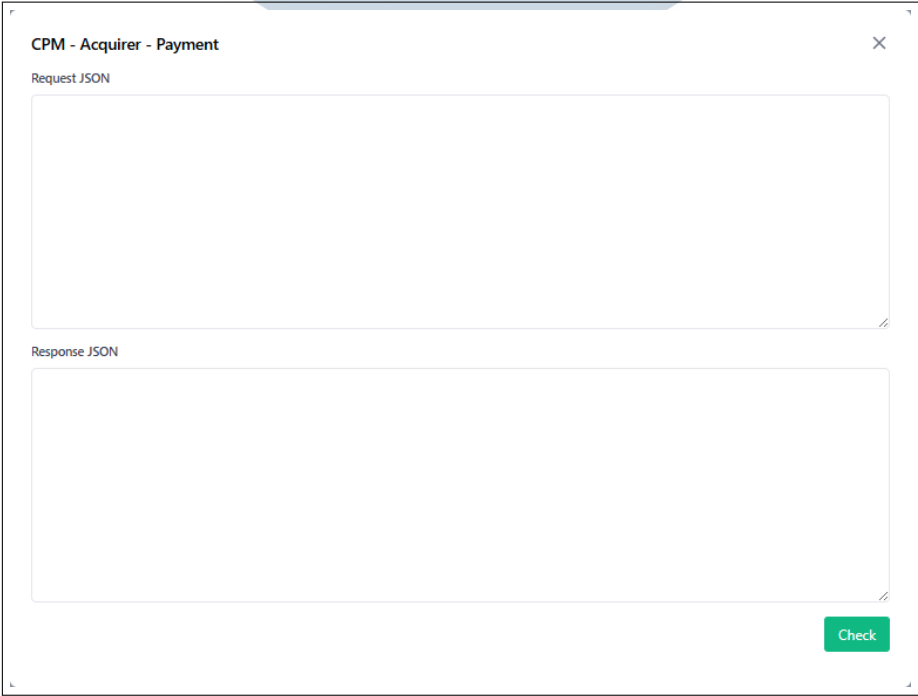
Halaman profil merupakan antarmuka *user* dalam sistem PrimaSwitch yang dirancang untuk mengelola informasi akun pribadi. Halaman ini hanya dapat diakses oleh *user* yang telah *login*, dan menyediakan fitur untuk memperbarui data profil serta mengubah kata sandi. Tampilan halaman ini dapat dilihat pada Gambar 3.7.

Fitur utama halaman profil meliputi beberapa elemen penting. Pertama, terdapat foto profil yang menampilkan gambar *default user* dalam format lingkaran dengan *border*, berukuran 24x24 piksel. Kedua, bagian informasi akun memungkinkan *user* untuk melihat dan mengubah nama *user* (*username*) serta alamat email. Semua *field* input dalam bagian ini memiliki penataan teks yang rata tengah (*text-center*).

Ketiga, pada bagian manajemen kata sandi, tersedia *field* untuk memasukkan kata sandi saat ini dan kata sandi baru. Fitur ini juga dilengkapi dengan tombol *toggle* yang memungkinkan *user* menampilkan atau menyembunyikan karakter kata sandi, serta proses validasi sebelum perubahan disimpan. Selain itu, terdapat sistem notifikasi yang memberikan umpan balik berupa pesan sukses dengan latar belakang hijau atau pesan kesalahan dengan latar belakang merah, yang akan muncul setelah *user* melakukan aksi pembaruan profil.

Alur kerja halaman profil dimulai saat *user* mengakses halaman, kemudian sistem akan memverifikasi status *login*. Setelah berhasil diverifikasi, data profil yang ada akan ditampilkan dalam bentuk formulir. *User* dapat melakukan perubahan informasi sesuai kebutuhan, dan sistem akan memvalidasi setiap input yang dimasukkan. Jika validasi berhasil, perubahan akan disimpan ke dalam *database*, dan sistem akan menampilkan notifikasi yang sesuai dengan hasil operasi.

E JSON Validation



The image shows a web application window titled "CPM - Acquirer - Payment". Inside the window, there are two large, empty text input areas. The top one is labeled "Request JSON" and the bottom one is labeled "Response JSON". At the bottom right of the window, there is a green button labeled "Check".

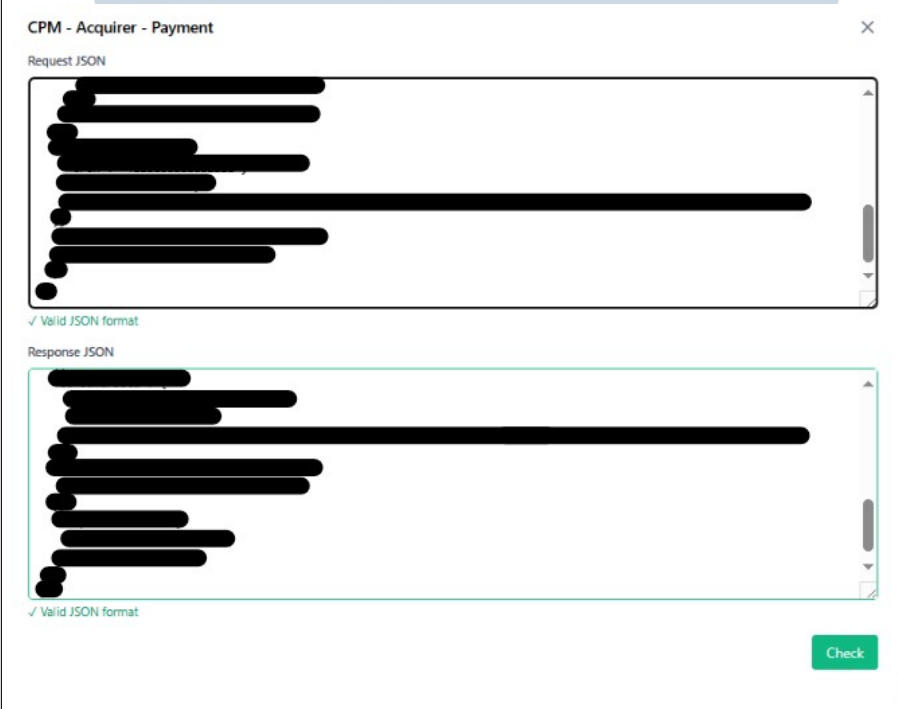
Gambar 3.8. *JSON Request and Response*

PrimaSwitch menyediakan antarmuka untuk mengelola dan memvalidasi transaksi melalui format JSON yang terstruktur. Sistem ini terdiri dari dua area

input utama yang dirancang untuk memudahkan *user* dalam memasukkan dan memvalidasi data transaksi, seperti yang ditampilkan pada Gambar 3.8.

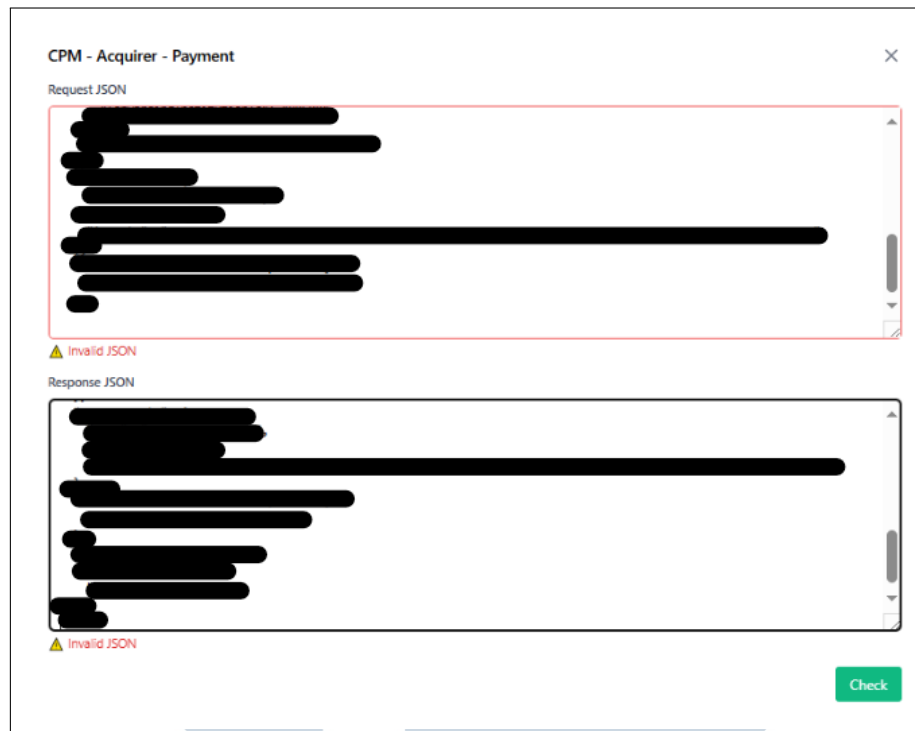
Area pertama adalah *Request JSON* yang terletak di bagian atas *form*. Area ini berfungsi sebagai tempat untuk memasukkan data permintaan transaksi dalam format JSON. *Text area* ini dilengkapi dengan latar belakang putih dan menggunakan *font monospace* untuk meningkatkan keterbacaan kode.

Di bawahnya, terdapat area *Response JSON* yang memiliki karakteristik serupa dengan area *Request*. Area ini digunakan untuk memasukkan data respons yang diharapkan dari transaksi. Kedua area input ini dilengkapi dengan sistem validasi *realtime* yang akan secara otomatis memeriksa format JSON setiap kali *user* memasukkan atau mengubah data.

The image shows a web application window titled "CPM - Acquirer - Payment". Inside, there are two text input areas. The first is labeled "Request JSON" and contains several lines of redacted text (black bars). Below it, a green checkmark and the text "Valid JSON format" are visible. The second area is labeled "Response JSON" and also contains redacted text. Below it, another green checkmark and the text "Valid JSON format" are visible. At the bottom right of the window, there is a green button labeled "Check".

Gambar 3.9. *Valid JSON Format*

UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.10. *Invalid JSON Format*

Sistem validasi bekerja secara proaktif dengan memberikan umpan balik visual kepada *user*. Ketika format JSON valid, sistem akan menampilkan tanda centang hijau disertai teks "*Valid JSON format*", dan *border* input akan berubah menjadi hijau seperti terlihat di Gambar 3.9. Sebaliknya, jika terdapat kesalahan format, sistem akan menampilkan tanda peringatan dengan teks "*Invalid JSON*" dan *border* merah, memudahkan *user* untuk segera mengidentifikasi dan memperbaiki kesalahan yang dapat dilihat di Gambar 3.10.

Di bagian bawah *form*, terdapat tombol "*Check*" berwarna hijau yang berfungsi untuk memulai proses validasi lengkap. Ketika tombol ini ditekan, sistem akan melakukan validasi menyeluruh terhadap kedua JSON yang dimasukkan dan menampilkan hasil validasi dalam bentuk *popup*. *Popup* ini memberikan informasi detail tentang hasil validasi dan langkah selanjutnya yang dapat diambil.

User cukup memasukkan JSON *request* sesuai format yang ditentukan, kemudian memasukkan JSON *response* yang diharapkan. Sistem akan secara otomatis memvalidasi format kedua JSON tersebut secara *realtime*. Setelah yakin dengan input yang dimasukkan, *user* dapat mengklik tombol *Check* untuk memulai proses validasi lengkap. Hasil validasi akan ditampilkan secara jelas dan terstruktur, memungkinkan *user* untuk dengan mudah memahami status validasi

dan mengambil tindakan yang diperlukan.[3][4][10]

F Change JSON Specification

| Field | Type | Min | Max | Allowed Characters | |
|------------|--------|--------|-----|--------------------|----------------|
| [Redacted] | String | Number | 1 | 11 | Numeric only |
| [Redacted] | String | Number | 1 | 11 | Numeric only |
| [Redacted] | String | Number | | | Any characters |
| [Redacted] | String | Number | | | Any characters |
| [Redacted] | String | Number | | | Any characters |

Gambar 3.11. *Change JSON Specification Pop Up*

Change JSON Specification merupakan fitur kunci dalam sistem PrimaSwitch yang memungkinkan administrator untuk mengelola dan menyesuaikan spesifikasi format JSON untuk berbagai jenis transaksi. Fitur ini diimplementasikan sebagai tombol interaktif yang memunculkan modal dialog untuk mengonfigurasi berbagai aspek validasi JSON, seperti yang ditampilkan pada Gambar 3.11.

Sistem ini bekerja dengan mengumpulkan informasi *field* dari berbagai sumber dalam aplikasi, termasuk *validator.js*, *multistep.js*, dan konfigurasi yang ada di sistem. Ketika tombol “*Change JSON specification*” diklik, sebuah modal akan muncul menampilkan tabel komprehensif yang berisi daftar semua *field* yang tersedia beserta pengaturannya.

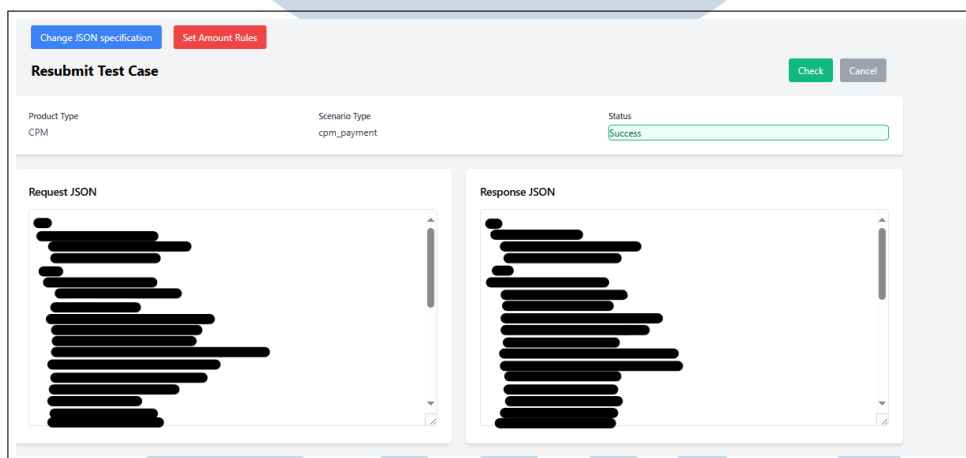
Setiap *field* dalam tabel dapat dikonfigurasi dengan tiga parameter utama: tipe data (*Type*), panjang minimum (*Min*), dan panjang maksimum (*Max*), serta karakter yang diizinkan (*Allowed Characters*). Sistem menyimpan preferensi

ini menggunakan `localStorage browser`, memungkinkan pengaturan bertahan bahkan setelah halaman di-*refresh* atau *browser* ditutup.

Fitur ini juga dilengkapi dengan sistem validasi *real-time* yang memeriksa apakah JSON yang dimasukkan sesuai dengan spesifikasi yang telah ditentukan. Jika ditemukan ketidaksesuaian, sistem akan menampilkan pesan *error* yang spesifik menunjukkan di mana letak kesalahannya. Hal ini sangat membantu dalam proses *debugging* dan memastikan format data yang konsisten.

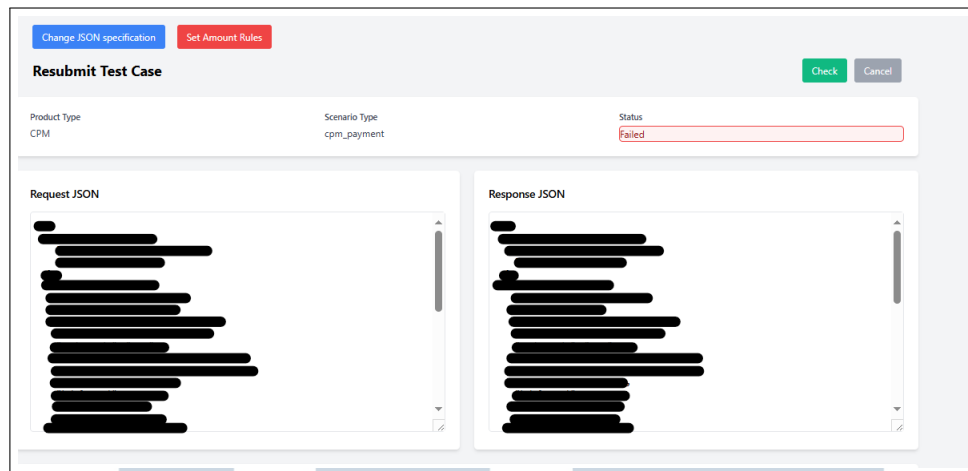
Untuk meningkatkan kegunaan, sistem ini juga menyertakan fungsi “*Set Amount Rules*” yang memungkinkan *user* untuk menentukan aturan khusus terkait validasi jumlah transaksi. Fitur ini penting terutama untuk memastikan konsistensi dalam transaksi keuangan dan mencegah kesalahan input yang dapat berakibat fatal. Dengan adanya sistem ini, administrator dapat dengan mudah menyesuaikan dan memperbarui aturan validasi sesuai dengan kebutuhan bisnis yang berkembang.[10]

G Halaman Resubmit (Single Step)



Gambar 3.12. Halaman *Resubmit Single Step Success*

UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.13. Halaman *Resubmit Single Step Failed*

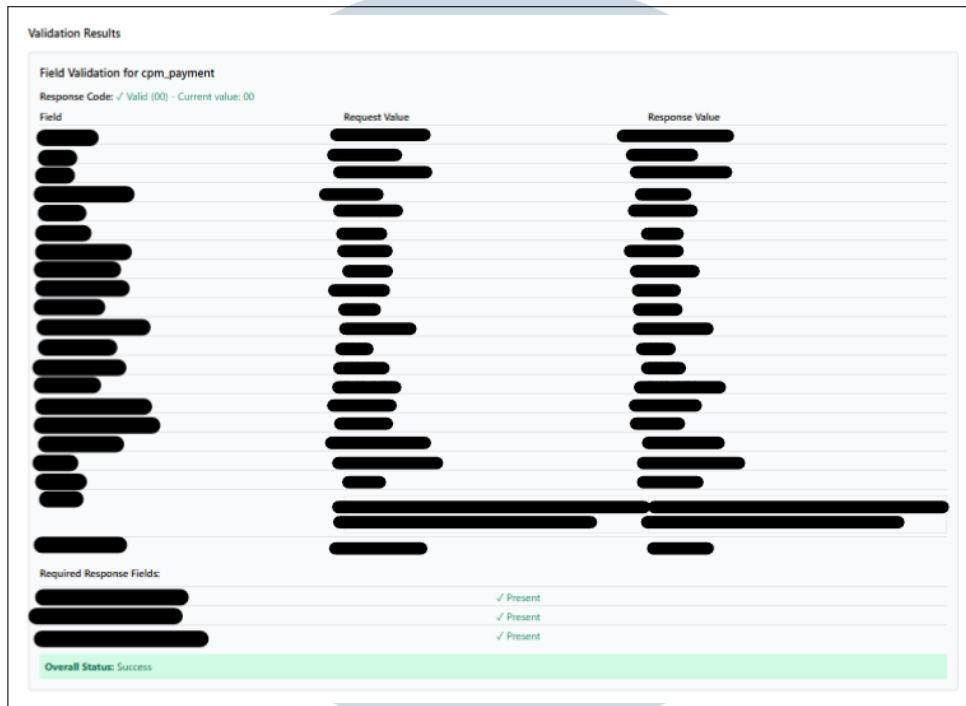
Halaman *resubmit* merupakan komponen penting dalam sistem PrimaSwitch yang memungkinkan *user* untuk memodifikasi dan memvalidasi kasus uji baik untuk transaksi CPM maupun MPM. Sistem ini dilengkapi dengan validasi JSON yang komprehensif untuk *request* dan *response*.

Halaman ini menyediakan dua editor JSON utama yang diposisikan berdampingan. Editor *Request JSON* berada di sisi kiri untuk memasukkan data permintaan, sementara Editor *Response JSON* di sisi kanan untuk data respons. Kedua editor dilengkapi dengan fitur validasi sintaks *real-time* yang memberikan umpan balik visual berupa *border* hijau untuk JSON yang valid dan merah untuk yang *invalid*. Sistem juga dilengkapi dengan fitur *auto-formatting* saat melakukan *paste* data, memudahkan pembacaan dan pengeditan JSON.

Validasi dilakukan secara berlapis dan *real-time*. Saat *user* mengetik atau menempelkan JSON, sistem langsung melakukan pengecekan sintaks. Validasi lebih mendalam mencakup verifikasi format *field* dan pengecekan *field* yang wajib ada. Setiap perubahan langsung memicu *feedback* visual dan pesan validasi yang muncul di bawah masing-masing editor, memberikan panduan langsung kepada *user*.

Proses *submission* dimulai saat *user* menekan tombol *Check*. Sistem akan melakukan validasi menyeluruh terhadap kedua JSON. Hasil validasi ditampilkan dalam *popup* yang informatif, menunjukkan status validasi dan daftar *error* jika ada. *User* diberikan dua opsi: kembali mengedit data atau melanjutkan *submission*. Status validasi ditampilkan dengan warna yang berbeda - hijau untuk sukses yang ditunjukkan di Gambar 3.12, dan merah untuk gagal yang ditunjukkan di Gambar 3.13, serta abu-abu untuk belum diuji.

G.1 Penanganan Hasil



The screenshot displays a 'Validation Results' window. At the top, it says 'Field Validation for cpm_payment' and 'Response Code: ✓ Valid (00) - Current value: 00'. Below this is a table with three columns: 'Field', 'Request Value', and 'Response Value'. The table contains 15 rows of data, all of which are redacted with black bars. At the bottom left, there is a section 'Required Response Fields:' with three items, each followed by a green checkmark and the word 'Present'. At the bottom right, there is a green bar with the text 'Overall Status: Success'.

| Field | Request Value | Response Value |
|------------|---------------|----------------|
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |
| [Redacted] | [Redacted] | [Redacted] |

Required Response Fields:

- ✓ Present
- ✓ Present
- ✓ Present

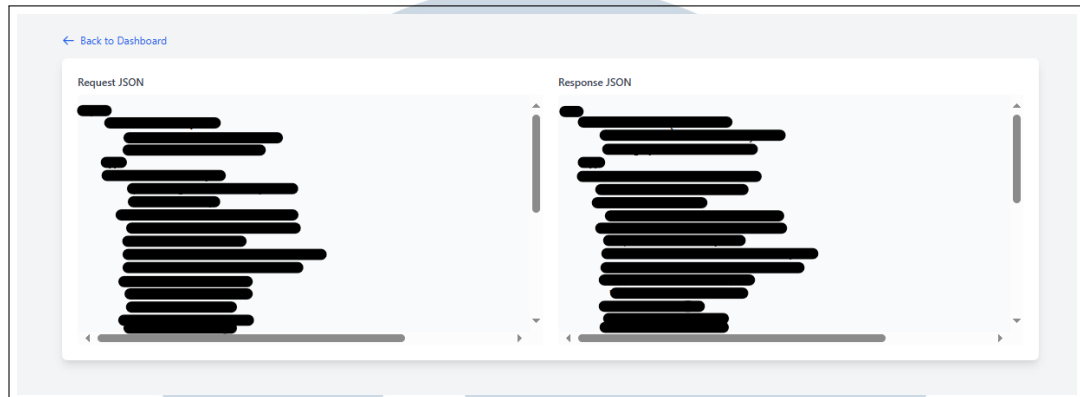
Overall Status: Success

Gambar 3.14. Halaman *Resubmit Single Step Validation Results*

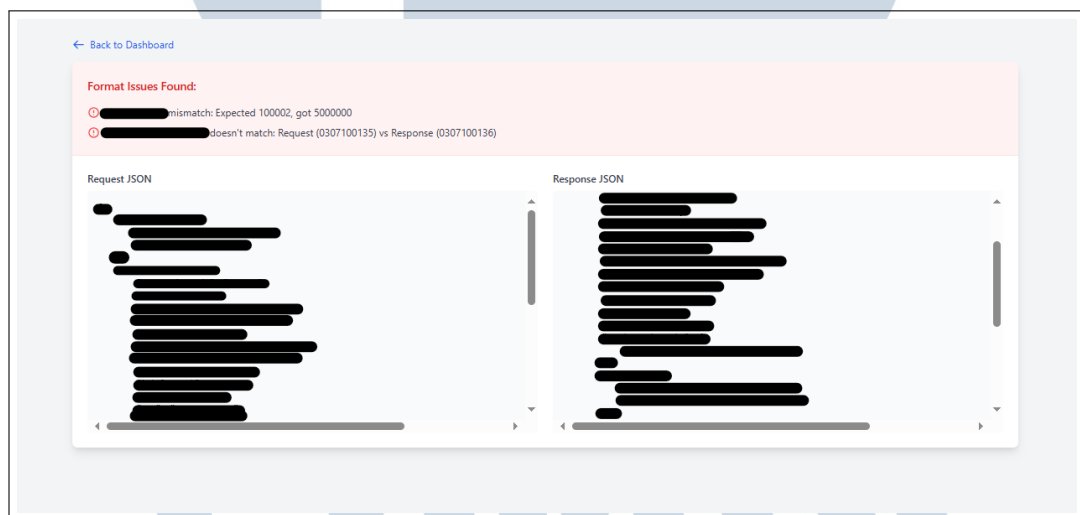
Sistem memberikan *feedback* yang jelas untuk setiap hasil *submission*. Jika berhasil, notifikasi hijau muncul dan halaman akan otomatis dialihkan ke halaman indeks setelah 3 detik. Jika terjadi *error*, notifikasi merah muncul dengan detail kesalahan, dan *user* dapat melakukan modifikasi dan *submit* ulang. Status kasus uji akan diperbarui sesuai dengan hasil validasi seperti yang ditampilkan pada Gambar 3.14.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A

H Halaman View Detail (Single Step)



Gambar 3.15. *View Detail Single Step*



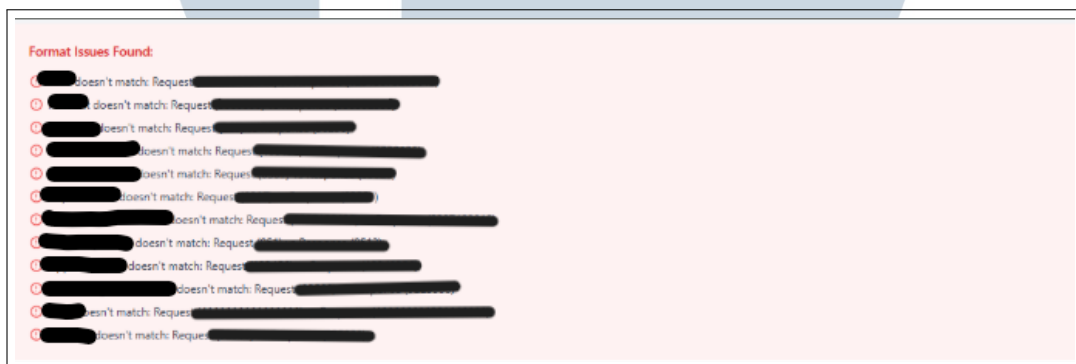
Gambar 3.16. *View Detail Error Single Step*

Fitur *View Detail* dalam sistem PrimaSwitch merupakan komponen penting yang menyediakan antarmuka untuk membandingkan dan menganalisis data transaksi secara mendalam. Fitur ini memungkinkan *user* untuk memeriksa format JSON dari *request* dan *response*, serta melakukan validasi terhadap struktur dan isi data. Tampilan fitur ini dapat dilihat pada Gambar 3.15.

Antarmuka *View Detail* terdiri dari beberapa komponen utama. Pertama adalah bagian navigasi dan *header*, di mana sistem menyediakan tombol “*Back to Dashboard*” yang memudahkan *user* kembali ke halaman utama. Tombol ini dilengkapi dengan ikon panah untuk meningkatkan kemudahan dan intuitivitas dalam navigasi.

Selanjutnya, terdapat area validasi format yang ditampilkan apabila sistem mendeteksi masalah dalam format data transaksi. Bagian ini muncul dengan latar belakang merah muda dan menyajikan daftar lengkap permasalahan yang ditemukan, masing-masing disertai ikon peringatan dan deskripsi detail untuk memudahkan *user* dalam memahami serta menangani kesalahan tersebut, seperti yang dapat dilihat di Gambar 3.16.

Selain itu, halaman ini dilengkapi dengan panel perbandingan JSON yang terdiri dari dua panel utama. Panel pertama menampilkan *Request JSON*, yaitu data permintaan transaksi yang dikirim ke sistem. Panel kedua menampilkan *Response JSON*, yaitu data respons yang diterima dari sistem. Kedua panel ini mempermudah *user* dalam melakukan analisis dan identifikasi perbedaan atau kesalahan dalam proses transaksi.



Gambar 3.17. *Format Issues Found Single Step*

Fungsionalitas format yang ditunjukkan pada Gambar 3.17 bertujuan untuk menunjukkan hasil pemeriksaan otomatis format JSON. *Format issues* menunjukkan beberapa *error* yang termasuk validasi *field* wajib, pengecekan tipe data, validasi panjang karakter dan juga pemeriksaan konsistensi antara *request* dan *response*.

Untuk penanganan *error*, sistem dilengkapi dengan mekanisme penanganan *error* yang komprehensif, termasuk *redirect* otomatis ke halaman *login* jika *session* tidak valid, penanganan kesalahan format JSON, notifikasi visual untuk setiap jenis *error* dan log *error* untuk keperluan *debugging*.

I Halaman Resubmit (Multi Step)

The screenshot shows a web interface for resubmitting a multi-step test case. At the top, there are three buttons: 'Change JSON specification' (blue), 'Set Amount Rules' (red), and 'Resubmit Multi-Step Test Case' (black). To the right of these buttons are 'Check' (green) and 'Cancel' (grey) buttons. Below the buttons is a form with three sections: 'Product Type' with value 'CPM', 'Scenario Type' with value 'Payment Debit then Refund 2 Times', and 'Overall Status' with a red 'Failed' label. Below these are three numbered steps: 1. 'Request1 and Response1' with subtext 'cpm_payment_refund2', 2. 'Request2 and Response2' with subtext 'cpm_authorization', and 3. 'Request3 and Response3' with subtext 'cpm_settlement'.

Gambar 3.18. Halaman *Resubmit Multi-Step*

Multi-Step Test Case merupakan fitur kompleks dalam sistem PrimaSwitch yang dirancang untuk menguji skenario transaksi yang melibatkan beberapa tahapan secara berurutan. Fitur ini sangat penting untuk memastikan integritas dan konsistensi alur transaksi yang kompleks, seperti skenario *Payment Debit then Refund 2 Times*, yang terdiri dari tiga pasang *request-response*. Tampilan dan fungsi fitur ini dapat dilihat pada Gambar 3.18.

Setiap *test case multi-step* terdiri dari beberapa komponen utama. Pertama adalah Informasi Status, yang mencakup *Product Type* untuk menunjukkan jenis produk (CPM atau MPM), *Scenario Type* untuk menampilkan jenis skenario yang diuji, serta *Overall Status* yang menunjukkan status keseluruhan pengujian, apakah *Success*, *Failed*, atau *Not Tested*. Selanjutnya adalah tahapan *request/response*, yang mencakup *Request1* dan *Response1* untuk transaksi pembayaran awal, *Request2* dan *Response2* untuk otorisasi, serta *Request3* dan *Response3* untuk proses *settlement*.

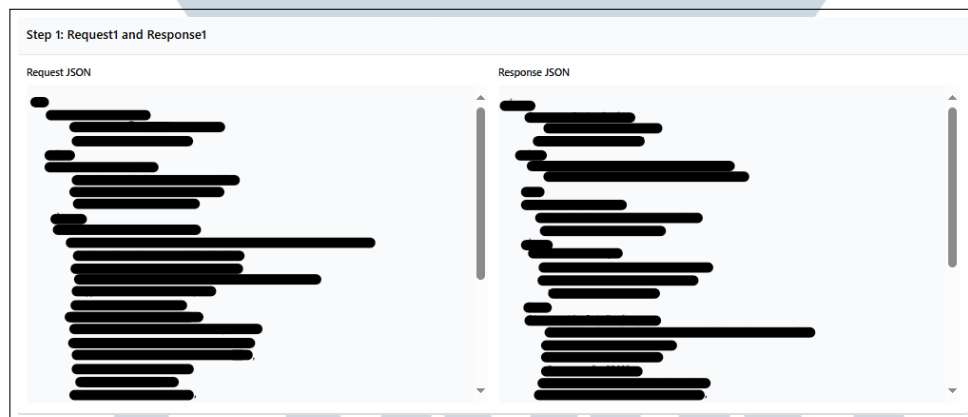
Fitur ini juga dilengkapi dengan validasi komprehensif di setiap tahap pengujian. Validasi mencakup pemeriksaan format JSON untuk setiap *request* dan *response*, pengecekan terhadap *field* wajib, validasi relasi antar tahapan, serta pemeriksaan konsistensi data di antara semua tahapan transaksi. Hal ini memastikan bahwa alur transaksi tidak hanya berjalan secara teknis, tetapi juga sesuai dengan ketentuan dan logika bisnis yang berlaku.

Dalam pengoperasiannya, *user* dapat menggunakan tombol aksi seperti *Change JSON Specification* untuk mengubah spesifikasi format JSON dan menyesuaikan aturan validasi serta batasan *field*, *Set Amount Rules* untuk mengatur

jumlah transaksi dengan batas minimum, maksimum, dan format nominal yang sesuai, serta tombol *Check* untuk memulai proses validasi atau *Cancel* untuk membatalkan proses dan kembali ke *dashboard*.

Alur fitur ini dimulai dengan *user* memilih skenario *multi-step* yang ingin diuji, lalu mengisi data untuk setiap pasangan *request-response* sesuai dengan alur transaksi. Setelah itu, sistem melakukan validasi di setiap tahap, memeriksa konsistensi antar tahap, dan memproses keseluruhan *test case*. Hasil dari proses ini ditampilkan melalui status yang menunjukkan keberhasilan atau kegagalan di setiap tahap maupun keseluruhan. Status *Success* ditampilkan dengan warna hijau, *Failed* dengan warna merah, dan *Not Tested* dengan warna abu-abu, sehingga memudahkan *user* dalam memantau dan mengevaluasi hasil pengujian secara menyeluruh.

J Halaman View Detail (Multi Step)

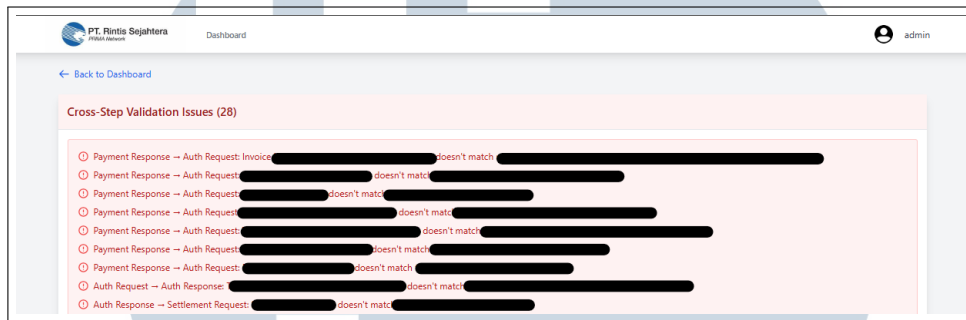


Gambar 3.19. Halaman *View Detail Multi-Step Step 1*

Fitur *Compare Multi-Step* memungkinkan *user* untuk melihat dan menganalisis detail lengkap dari skenario pengujian yang terdiri dari beberapa langkah berurutan. Saat membuka halaman *compare_multi.php*, sistem akan menampilkan semua langkah dalam grup *multi-step* beserta JSON *request* dan *response* untuk setiap langkahnya, seperti yang terlihat pada Gambar 3.19.

Fitur ini dilengkapi dengan validasi otomatis yang memeriksa format dan konsistensi data antar langkah. Sistem akan memvalidasi hubungan antar langkah (*cross-step relationships*) untuk memastikan nilai-nilai penting seperti *Message ID*, *Transaction ID*, *STAN*, dan *Amount* tetap konsisten di seluruh alur transaksi. Jika ditemukan perbedaan atau ketidaksesuaian, sistem akan menampilkan pesan *error* yang detail.

Setiap langkah ditampilkan dalam panel terpisah yang berisi informasi seperti nama langkah, nomor urut, status, dan JSON *request-response*. *User* dapat melihat perbandingan format JSON untuk setiap langkah, dengan *highlight* pada *field* yang berbeda atau tidak sesuai. Data JSON ditampilkan dalam format yang mudah dibaca dengan indentasi yang tepat.



Gambar 3.20. Halaman *View Detail Multi-Step Cross-Step Validation Issues*

Untuk memudahkan analisis, sistem mengelompokkan *issue* yang ditemukan menjadi dua kategori: *issue format* (terkait struktur dan tipe data) dan *issue cross-step* (terkait konsistensi data antar langkah). Total jumlah *issue* ditampilkan di bagian atas halaman sebagai ringkasan cepat dari hasil validasi, seperti yang ditunjukkan pada Gambar 3.20.

Tampilan halaman dioptimalkan dengan fitur *collapse/expand* untuk setiap langkah, memungkinkan *user* fokus pada langkah tertentu. *JSON viewer* dilengkapi dengan batasan tinggi maksimum dan *scrollbar* untuk menangani data yang panjang. Semua pesan *error* dan *warning* ditampilkan dengan warna dan ikon yang intuitif untuk memudahkan identifikasi masalah.

| From Step | To Step | Type | Description |
|-----------|---------|------------------|--|
| step_A | step_B | request-response | Payment Request → Payment Response |
| step_B | step_C | response-request | Payment Response → Auth Request |
| step_C | step_D | request-response | Auth Request → Auth Response |
| step_D | step_E | response-request | Auth Response → Settlement Request |
| step_E | step_F | request-response | Settlement Request → Settlement Response |

Gambar 3.21. Relasi Antara *Request* dan *Response*

Sistem juga mencatat relasi antar langkah dalam basis data, memungkinkan validasi yang lebih kompleks seperti memeriksa keterkaitan antara langkah *authorization* dengan *settlement* atau *payment* dengan *reversal*. Relasi ini ditampilkan dalam bentuk visual yang mudah dipahami, membantu *user* memahami alur lengkap dari skenario *multi-step*, seperti yang terlihat pada Gambar 3.21.

3.5 Pengujian Sistem

Metode pengujian yang digunakan dalam proyek ini adalah *Black Box Testing*. *Black Box Testing* merupakan metode pengujian perangkat lunak yang fokus pada fungsionalitas sistem tanpa mengetahui struktur internal atau kode program. Pengujian dilakukan dengan cara memberikan input dan mengamati output yang dihasilkan, untuk memverifikasi apakah sistem telah berjalan sesuai dengan yang diharapkan berdasarkan spesifikasi fungsional.

Tabel 3.2. Hasil Pengujian Sistem

| No | Fitur yang Diuji | Skenario Pengujian | Hasil yang Diharapkan | Status |
|----|----------------------------|--|---|--------|
| 1 | Login dan Role Akses | User dapat masuk ke sistem dan diarahkan ke halaman sesuai peran (<i>admin/user</i>) | User berhasil login dan <i>role-based redirect</i> bekerja sesuai fungsinya | Sesuai |
| 2 | Input JSON Valid | Input JSON lengkap dan sesuai struktur (CPM/MPM) | Sistem menampilkan hasil validasi sukses dan status " <i>Valid</i> " | Sesuai |
| 3 | Input JSON Tidak Valid | Input JSON dengan field kosong atau struktur tidak sesuai | Sistem menampilkan pesan <i>error</i> dan lokasi kesalahan dalam tampilan <i>detail</i> | Sesuai |
| 4 | Fitur <i>Edit</i> JSON | Klik tombol <i>edit</i> untuk memperbarui konten JSON | JSON tersimpan ulang dan hasil validasi diperbarui sesuai perubahan | Sesuai |
| 5 | Fitur <i>Resubmit</i> JSON | Kirim ulang JSON yang telah diperbaiki | Sistem memvalidasi ulang file JSON dan menampilkan status hasil baru | Sesuai |

Bersambung ke halaman berikutnya

Tabel 3.2 – Lanjutan dari halaman sebelumnya

| No | Fitur yang Diuji | Skenario Pengujian | Hasil yang Diharapkan | Status |
|----|----------------------------------|--|---|--------------|
| 6 | <i>Multi-step Validation</i> | JSON melewati validasi bertahap untuk format CPM dan MPM | Sistem menampilkan status per langkah validasi dan informasi lengkap | Sesuai |
| 7 | <i>User list/Multiple User</i> | <i>User list</i> menunjukkan berbagai <i>user</i> yang <i>login</i> dan <i>admin</i> diberikan opsi untuk menghapus <i>user</i> tersebut | <i>Admin</i> diberikan fitur untuk menghapus data <i>user</i> lain | Sesuai |
| 8 | <i>View Detail</i> | Menampilkan hasil kesalahan validasi JSON | <i>User</i> dapat melihat ringkasan kesalahan dengan <i>field</i> dan baris <i>detail</i> | Sesuai |
| 9 | <i>Change JSON Specification</i> | <i>User</i> diberikan opsi untuk mengubah format JSON | <i>User</i> dapat mengubah format JSONnya di <i>singlestep</i> dan juga <i>multistep</i> | Sesuai |
| 10 | <i>Change Total Amount</i> | <i>User</i> diberikan opsi untuk mengubah peraturan <i>total amount</i> dan juga <i>amount fee</i> | Jika <i>total amount</i> atau <i>amount fee</i> berbeda maka keluar <i>warning</i> | Sesuai |
| 11 | <i>Export Data ke Excel</i> | Data yang telah difilter dapat diekspor | Data berhasil diekspor dalam format Excel sesuai hasil <i>filter</i> | Sesuai |
| 12 | <i>Export Data ke PDF</i> | Data yang telah difilter dapat diekspor | Tidak berhasil di implementasi | Tidak Sesuai |

Bersambung ke halaman berikutnya

Tabel 3.2 – Lanjutan dari halaman sebelumnya

| No | Fitur yang Diuji | Skenario Pengujian | Hasil yang Diharapkan | Status |
|----|---------------------|---|---|--------|
| 13 | <i>Your Profile</i> | <i>User</i> dapat membuka <i>profile settings</i> dan mengubah data | <i>User</i> dapat mengubah informasi yang berada di <i>your profile</i> | Sesuai |

3.6 Kendala dan Solusi yang Ditemukan

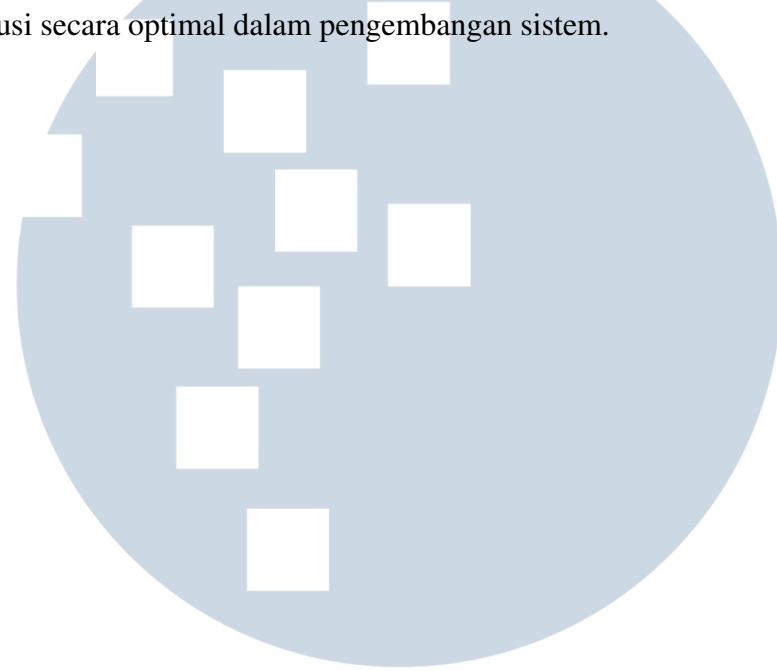
Selama menjalani kegiatan magang dari bulan Februari hingga Juni, penulis menghadapi sejumlah kendala yang cukup memengaruhi jalannya proses pengembangan sistem, baik pada proyek sampingan (pengelolaan data bank, biller, dan spesifikasi) maupun proyek utama (validasi data JSON). Salah satu tantangan utama yang penulis hadapi adalah terbatasnya jumlah sumber daya manusia yang terlibat. Dalam praktiknya, sebagian besar proses mulai dari perancangan, pengembangan, hingga pengujian sistem dilakukan secara individu. Padahal, idealnya proyek semacam ini ditangani oleh tim yang terdiri dari 4 orang dengan pembagian peran yang lebih spesifik, seperti *Back-End*, *Front-End*, dokumentasi, dan *Quality Assurance*. Hal ini tentu menambah beban kerja dan membutuhkan manajemen waktu yang lebih ketat.

Kendala lain yang sering penulis temui adalah munculnya error atau kesalahan sistem yang tidak disertai dengan pesan yang jelas. Ini menyulitkan proses *debugging*, karena tidak semua error langsung menunjukkan sumber permasalahan. Beberapa bahkan berasal dari kesalahan konfigurasi kecil atau dependensi eksternal yang sulit dideteksi, terutama saat awal proses integrasi sistem. Situasi ini menuntut pemahaman teknis yang lebih mendalam dan sering kali memerlukan pendekatan *trial-and-error* yang cukup memakan waktu.

Penerapan fitur-fitur kompleks seperti *multi-step validation* pada proyek JSON juga menjadi tantangan tersendiri. Fitur ini memiliki alur logika yang panjang dan saling bergantung antar modul, sehingga kesalahan kecil pada satu bagian dapat memengaruhi keseluruhan proses. Tantangan ini semakin besar karena keterbatasan dokumentasi awal penulis harus memahami struktur sistem secara mandiri dari awal, yang tentu memerlukan waktu dan ketelitian ekstra.

Sebagai peserta magang yang belum memiliki pengalaman kerja profesional sebelumnya, penulis menghadapi tantangan dalam memahami keseluruhan alur

sistem juga menjadi tantangan yang cukup besar. Dibutuhkan waktu dan usaha lebih untuk memahami struktur data, alur sistem, relasi antar-entitas, serta format JSON dan tahapan validasinya. Oleh karena itu, fase orientasi dan eksplorasi di awal masa magang terasa sangat krusial dan cukup menyita waktu sebelum dapat berkontribusi secara optimal dalam pengembangan sistem.



UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA