

## BAB 3

### PELAKSANAAN KERJA MAGANG

#### 3.1 Kedudukan dan Koordinasi

##### 3.1.1 Kedudukan

Selama pelaksanaan kerja magang di PT Inovasi Daya Solusi, posisi yang ditempati adalah sebagai *Full Stack Developer Intern* yang ditempatkan pada *Engineer Team* di bawah divisi *Technology & Digital Transformation*. Dalam posisi ini, tugas utama yang diberikan adalah mengembangkan *Sistem Deposit* yang mencakup pengembangan *backend* menggunakan Java dan *frontend* menggunakan Next.js.

Pembimbing lapangan selama pelaksanaan kerja magang adalah Bapak Vincent Pribadi yang menjabat sebagai *Head of Strategic Partnership*. Pembimbing lapangan bertugas memberikan arahan teknis, dan membantu dalam menyelesaikan kendala yang dihadapi selama pelaksanaan kerja magang.

##### 3.1.2 Koordinasi

Koordinasi selama pelaksanaan kerja magang dilakukan dengan mengikuti metodologi *Agile* yang diterapkan oleh tim. Proses koordinasi meliputi beberapa tahapan, yaitu *Grooming*, *Sprint Planning*, dan pelaksanaan tugas harian.

Pada tahap *Grooming*, tim melakukan diskusi mengenai fitur-fitur yang akan dikembangkan dan melakukan estimasi kompleksitas pekerjaan. Selanjutnya pada tahap *Sprint Planning*, tugas-tugas dibagi kepada anggota tim untuk dikerjakan dalam periode *sprint* tertentu. Pelaksanaan tugas harian didukung oleh beberapa *tools*, yaitu Jira untuk manajemen proyek, Google Chat untuk komunikasi tim, GitLab untuk *version control*, Postman untuk pengujian API, dan DBeaver untuk manajemen *database*.

#### 3.2 Tugas yang Dilakukan

Selama periode magang dari Juni 2025 hingga Januari 2026, berbagai tugas telah dikerjakan. Tabel 3.1 menunjukkan rincian tugas yang dilakukan selama periode tersebut.

Tabel 3.1. Tugas yang dilakukan selama magang

No	Bulan	Nama Proyek	Jenis Pekerjaan
1	Juni - Juli 2025	Sistem Deposit	<i>Frontend &amp; Backend</i>
2	Agustus 2025	Dashboard SOB - AI Assistant	<i>Frontend</i>
3	Agustus 2025	VA - VA List	<i>Frontend</i>
4	September 2025	VA - VA Missing Transaction	<i>Frontend</i>
5	September 2025	M&M - Template Management	<i>Frontend</i>
6	Oktober 2025	Digital Goods - Retry Notes Transaction	<i>Frontend</i>
7	Oktober 2025	IMS - Enhancement Sales Invoice	<i>Frontend</i>
8	November 2025	Ototrans - Export Feature	<i>Frontend</i>
9	November 2025	Digital Goods - Supplier Onboarding	<i>Frontend</i>
10	Desember 2025	Digital Goods - Revamp Fee Product	<i>Frontend</i>
11	Desember 2025	AI Assistant - Marketing Agent	<i>Frontend &amp; Backend</i>

Berdasarkan Tabel 3.1, proyek utama yang menjadi fokus pembahasan dalam laporan ini adalah pengembangan *Sistem Deposit* yang dikerjakan pada bulan Juni hingga Juli 2025.

### 3.3 Uraian Pelaksanaan Kerja

#### 3.3.1 Proses Pelaksanaan

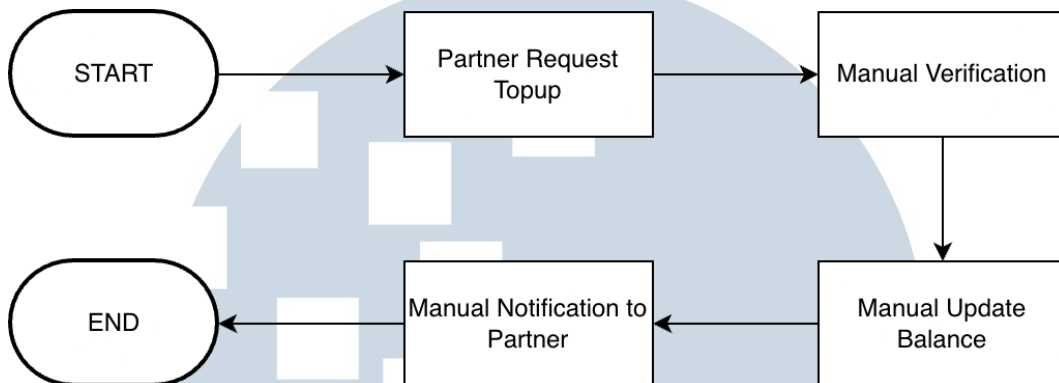
Pengembangan *Sistem Deposit* dilaksanakan melalui beberapa tahapan, yaitu analisis kebutuhan sistem, perancangan sistem, implementasi *backend*, dan implementasi *frontend*.

#### A Analisis Kebutuhan Sistem

##### A.1 Latar Belakang Sistem

Sebelum pengembangan *Sistem Deposit*, proses pengelolaan saldo *partner* dan *merchant* di PT Inovasi Daya Solusi masih dilakukan secara manual.

Gambar 3.1 menunjukkan perbandingan alur proses antara sistem lama dan sistem baru yang dikembangkan.



Gambar 3.1. Alur sistem lama

Pada sistem lama, ketika *partner* atau *merchant* ingin melakukan *top up* atau *withdraw* saldo, mereka harus menghubungi admin. Admin kemudian mencatat permintaan tersebut di sistem terpisah. Setelah itu, admin melakukan verifikasi pembayaran secara manual dan memperbarui saldo di *database*. Proses terakhir adalah admin memberitahu *partner* atau *merchant* bahwa transaksi telah selesai diproses.

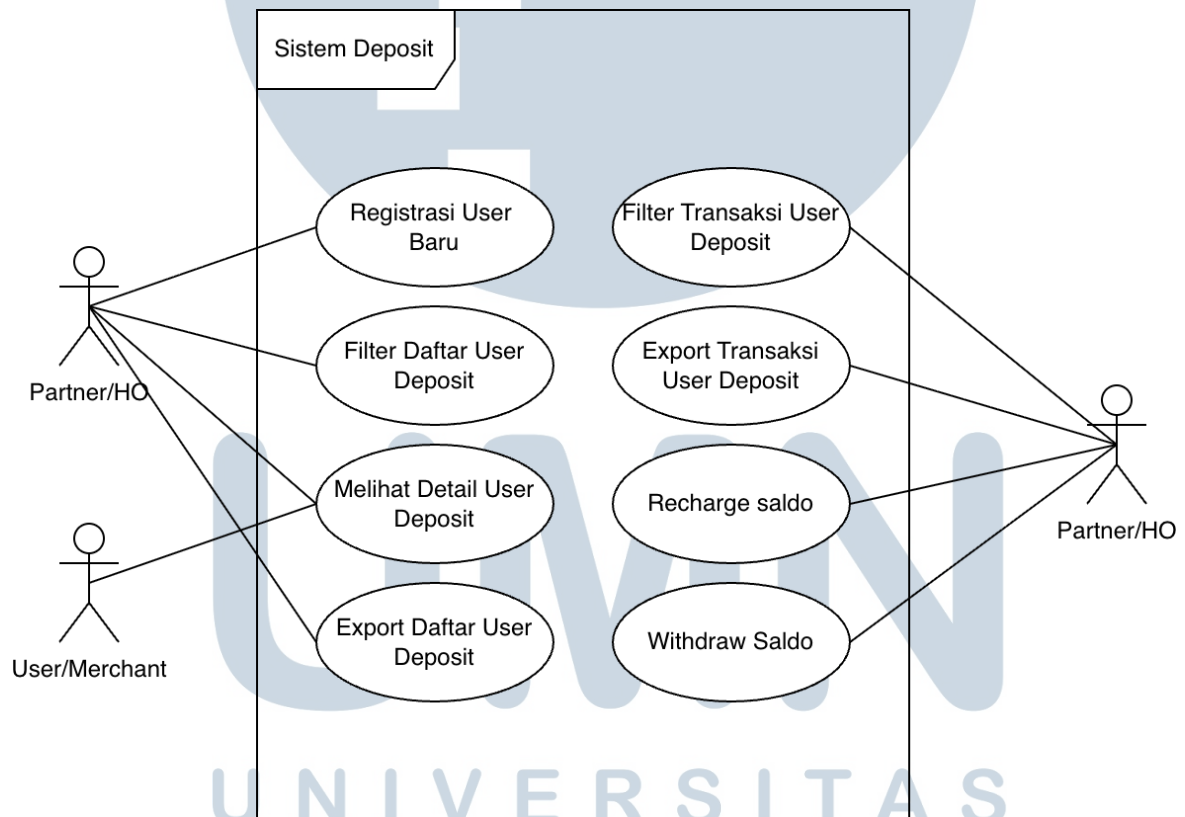
Sistem lama memiliki beberapa keterbatasan, antara lain:

- Proses lambat dan tidak efisien karena bergantung pada ketersediaan admin
- Rentan terhadap *human error* dalam pencatatan dan verifikasi
- Tidak ada transparansi *real-time* untuk *partner* dan *merchant*
- Sulit melakukan *tracking* riwayat transaksi
- Tidak ada mekanisme rekonsiliasi otomatis

Berdasarkan keterbatasan tersebut, dikembangkan *Sistem Deposit* yang mengotomatisasi seluruh proses pengelolaan saldo. Pada sistem baru, *partner* dapat secara mandiri melakukan *recharge* saldo melalui *Virtual Account* yang dibuat otomatis oleh sistem. Verifikasi pembayaran dilakukan secara otomatis melalui mekanisme *callback* dari *Payment Gateway*. Sistem juga menyediakan fitur *History Transaction* yang memungkinkan *partner* untuk melihat seluruh riwayat transaksi dengan fitur *filter* dan *export*.

*Sistem Deposit* merupakan sistem yang dikembangkan untuk mengelola saldo *partner* dan *merchant* di PT Inovasi Daya Solusi. Sistem ini melayani dua model bisnis utama. Model bisnis pertama adalah PPOB (*Payment Point Online Bank*), yaitu *partner* melakukan *top up* deposit, kemudian *partner* atau *merchant* melakukan transaksi PPOB, lalu sistem mengurangi saldo, dan pembayaran diteruskan ke *supplier*. Model bisnis kedua adalah ekspedisi, yaitu cabang *merchant* menyetorkan dana ke *Head Office* secara terpusat.

Berdasarkan analisis kebutuhan, sistem memiliki dua aktor utama, yaitu *Partner* atau *Head Office* (HO) dan *Merchant*. *Partner* atau HO dapat mengakses fitur pada halaman *Partner Deposit* dan *User Deposit*, sedangkan *Merchant* hanya dapat mengakses fitur pada halaman *User Deposit*.



Gambar 3.2. Use case diagram Sistem Deposit

Berdasarkan Gambar 3.2, aktor *Partner* atau HO memiliki akses untuk melihat daftar *merchant*, mendaftarkan *merchant* baru, melihat detail *merchant*, mengecek saldo *merchant*, melihat transaksi *partner*, melakukan *recharge* saldo melalui pembuatan VA, melakukan *withdraw* saldo ke rekening bank, dan meng-export data. Sementara itu, aktor *Merchant* memiliki akses untuk melihat detail

*merchant* dan mengecek saldo *merchant*.

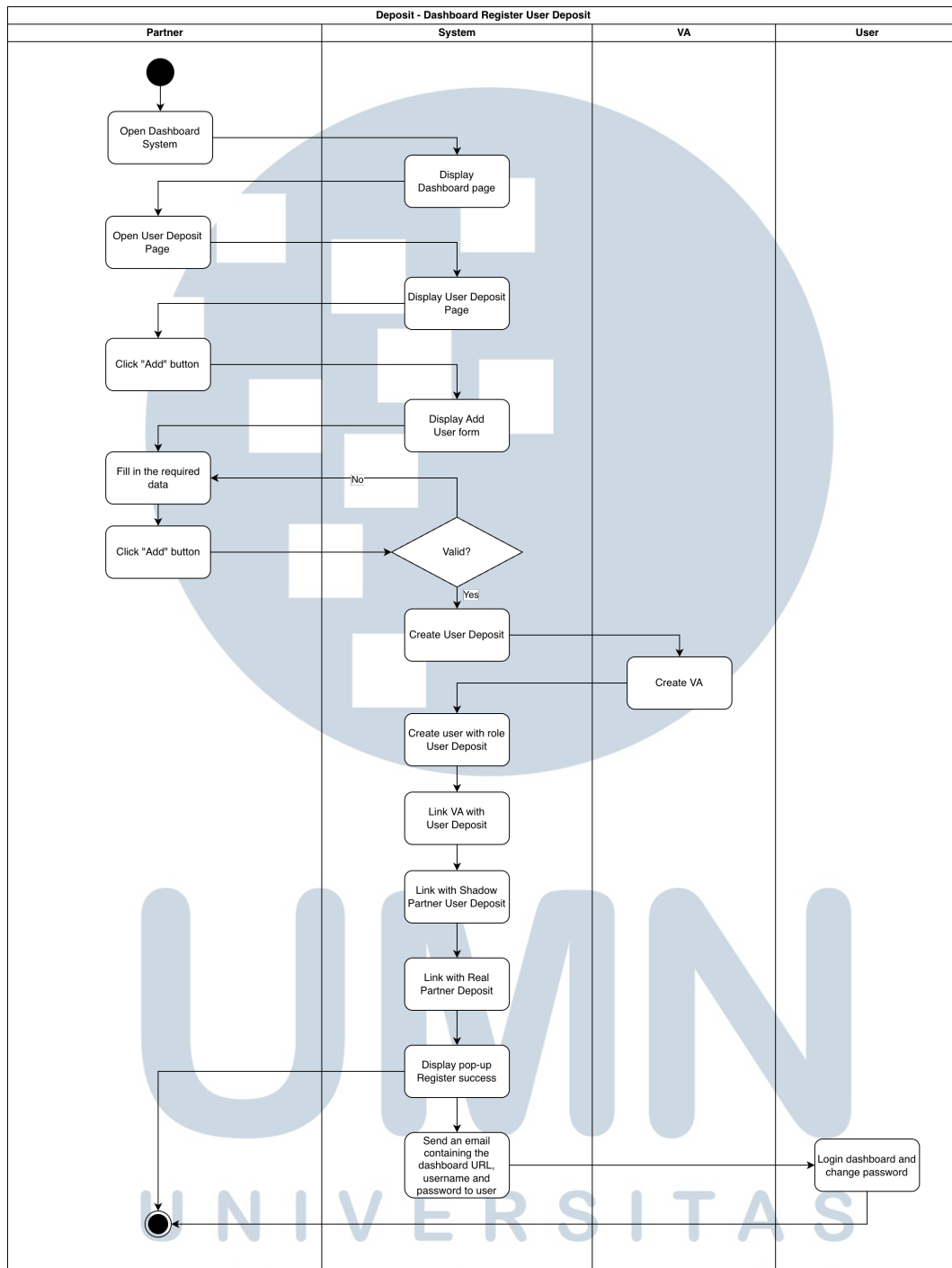
## **B Perancangan Sistem**

Perancangan sistem dilakukan dengan membuat *Activity Diagram* dan *Sequence Diagram* untuk setiap fitur utama. *Activity Diagram* menggambarkan alur aktivitas dalam sistem dari perspektif proses bisnis dengan menunjukkan urutan aktivitas yang dilakukan oleh aktor dan sistem, serta keputusan yang diambil dalam setiap proses. *Sequence Diagram* menggambarkan interaksi antara *User*, *Dashboard (frontend)*, *API (backend)*, *Database*, dan layanan eksternal secara lebih teknis.

### **B.1 Registrasi User Baru**

*Activity Diagram* untuk fitur registrasi *user* baru ditunjukkan pada Gambar 3.3.



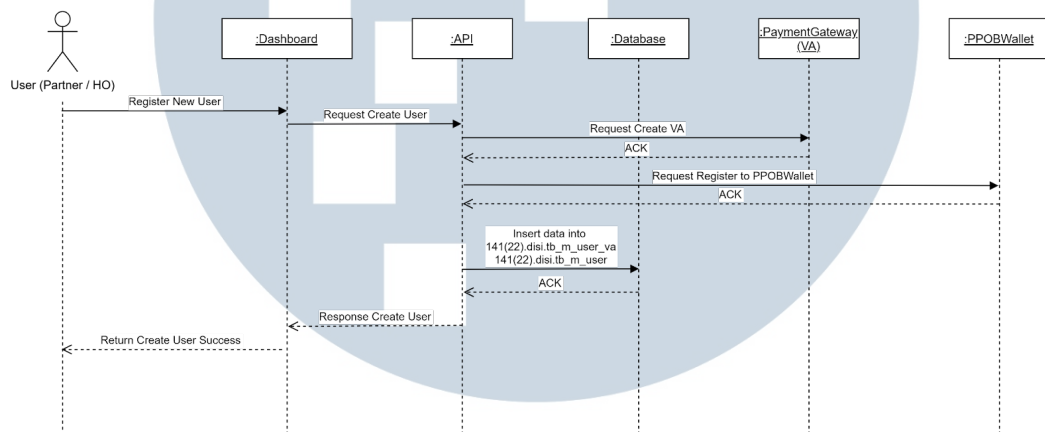


Gambar 3.3. Activity diagram registrasi user baru

Berdasarkan Gambar 3.3, proses registrasi dimulai dari *Partner* atau HO yang mengisi formulir pendaftaran dengan data *merchant*, informasi bank account, dan pemilihan bank VA. Sistem kemudian melakukan validasi data. Jika data tidak valid, sistem akan menampilkan pesan kesalahan dan meminta *user*

untuk mengisi ulang formulir. Jika data valid, sistem akan mengirim *request* ke *PaymentGateway* untuk membuat *Virtual Account*. Apabila pembuatan VA gagal, sistem menampilkan pesan error dan proses berakhir. Jika VA berhasil dibuat, sistem melanjutkan dengan mengirim *request* ke *PPOBWallet* untuk mendaftarkan *wallet*. Setelah *wallet* berhasil didaftarkan, data *merchant* disimpan ke *Database* pada tabel *disi.tb\_m\_user\_va* dan *disi.tb\_m\_user*. Sistem kemudian menampilkan notifikasi bahwa *merchant* berhasil didaftarkan.

*Sequence Diagram* untuk fitur registrasi *user* baru ditunjukkan pada Gambar 3.4.



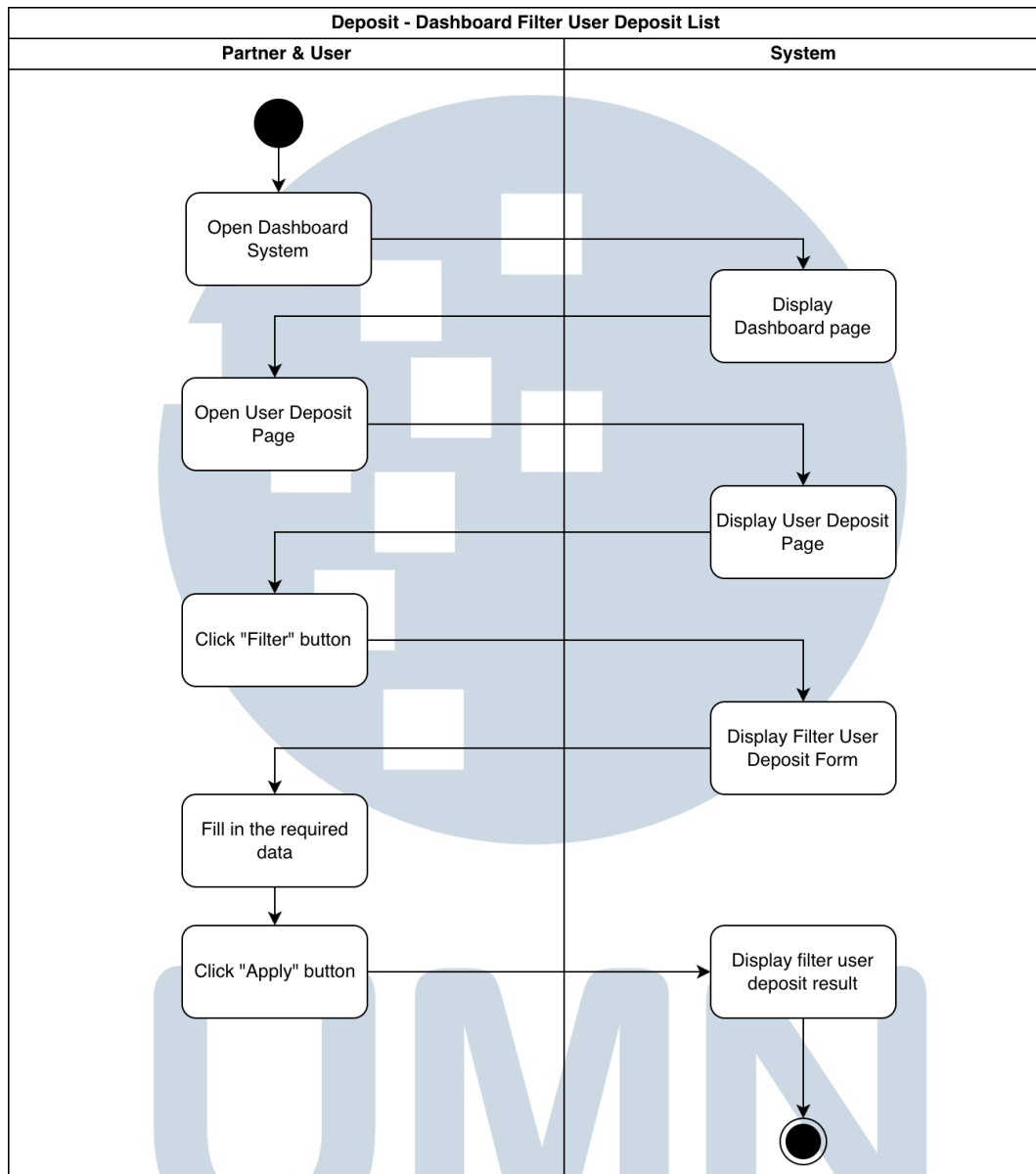
Gambar 3.4. *Sequence diagram* registrasi *user* baru

Berdasarkan Gambar 3.4, alur registrasi *user* baru dimulai ketika *User* (*Partner* atau *HO*) mengakses halaman registrasi dan mengisi formulir pendaftaran *merchant* baru. *Dashboard* mengirimkan *request* ke *API* untuk membuat *user* baru. *API* kemudian mengirimkan *request* ke *PaymentGateway* untuk membuat *Virtual Account* (VA). Setelah VA berhasil dibuat, *API* mengirimkan *request* ke *PPOBWallet* untuk mendaftarkan *user* ke sistem *wallet*. Selanjutnya data *user* disimpan ke *Database* pada tabel *disi.tb\_m\_user\_va* dan *disi.tb\_m\_user*. Setelah proses selesai, *Dashboard* menampilkan notifikasi bahwa *user* berhasil dibuat.

## B.2 Filter Daftar User Deposit

*Activity Diagram* untuk fitur *filter* daftar *User Deposit* ditunjukkan pada Gambar 3.5.



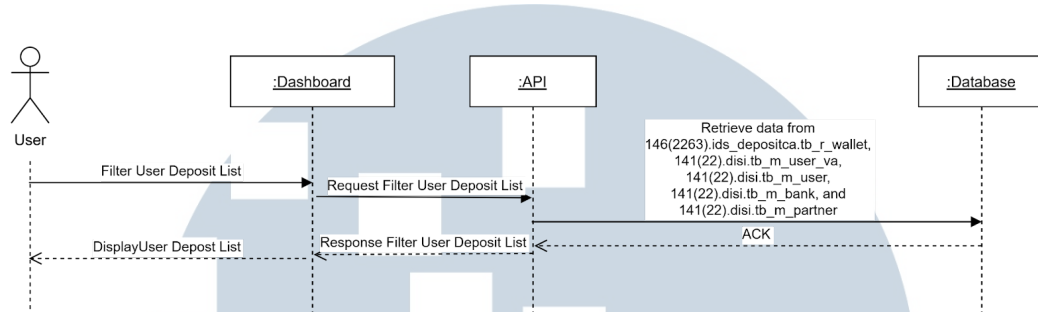


Gambar 3.5. Activity diagram filter daftar user deposit

Berdasarkan Gambar 3.5, proses dimulai ketika *User* memasukkan kriteria *filter* seperti *merchant ID*, nama *merchant*, atau status, kemudian menekan tombol "Search" atau "Filter". *Dashboard* mengirimkan *request* ke *API* untuk mengambil data sesuai *filter*. *API* melakukan *query* ke *Database* dengan mengakses tabel yang relevan. Sistem mengambil data yang sesuai dengan kriteria *filter* dari *Database*. Data yang diperoleh kemudian dikembalikan ke *API* dan diteruskan ke *Dashboard*. *Dashboard* menampilkan daftar *merchant* yang sesuai dengan kriteria *filter* kepada *User* dengan sistem paginasi.



*Sequence Diagram* untuk fitur *filter* daftar *User Deposit* ditunjukkan pada Gambar 3.6.



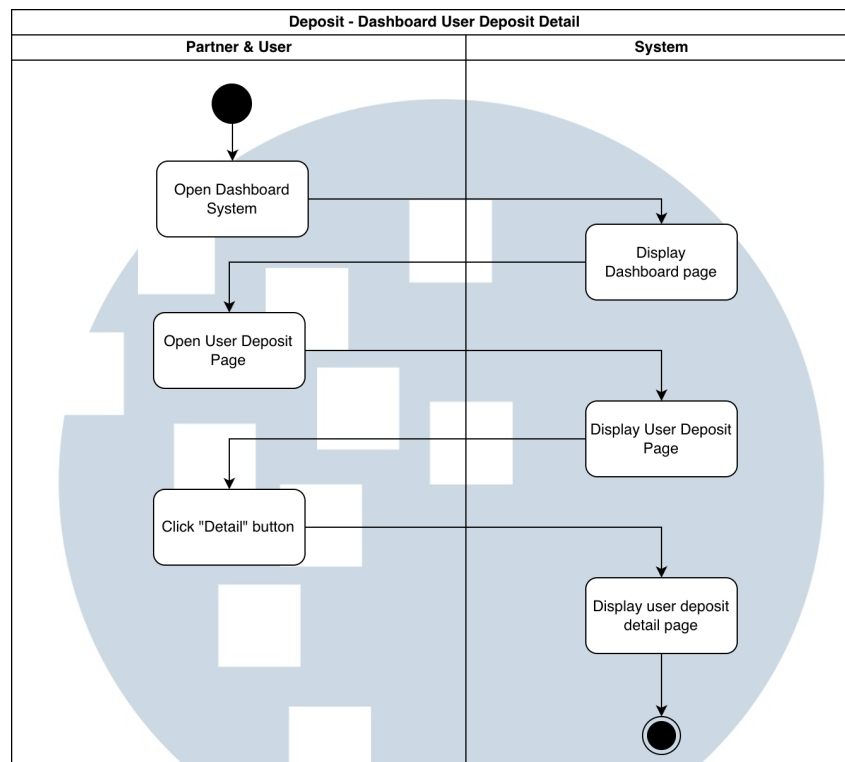
Gambar 3.6. *Sequence diagram filter* daftar *user deposit*

Berdasarkan Gambar 3.6, alur *filter* daftar *User Deposit* dimulai ketika *User* memasukkan kriteria *filter* pada halaman *User Deposit*. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data sesuai *filter*. *API* mengambil data dari *Database* dengan mengakses tabel *ids\_depositca.tb\_r\_wallet*, *disi.tb\_m\_user\_va*, *disi.tb\_m\_user*, *disi.tb\_m\_bank*, dan *disi.tb\_m\_partner*. Data yang diperoleh kemudian dikembalikan ke *Dashboard* untuk ditampilkan kepada *User*.

### B.3 Melihat Detail User Deposit

*Activity Diagram* untuk fitur melihat detail *User Deposit* ditunjukkan pada Gambar 3.7.

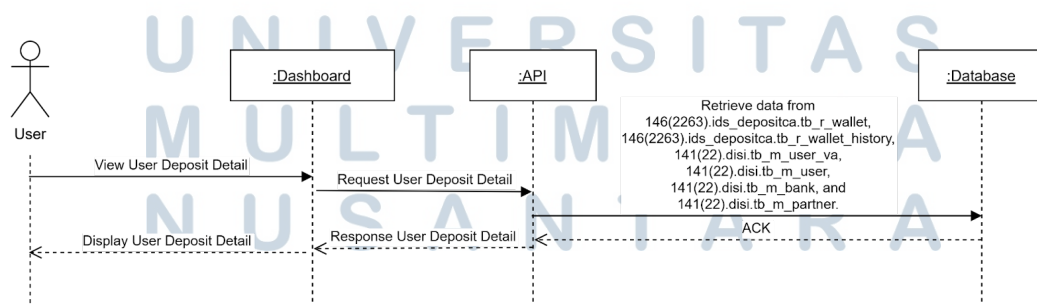




Gambar 3.7. Activity diagram melihat detail user deposit

Berdasarkan Gambar 3.7, proses dimulai ketika *User* mengklik salah satu *merchant* dari daftar. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data detail *merchant*. *API* melakukan *query* ke *Database* untuk mengambil data detail *merchant* beserta riwayat transaksi. Data yang diperoleh kemudian dikembalikan ke *API* dan diteruskan ke *Dashboard*. *Dashboard* menampilkan halaman detail *merchant* yang berisi informasi lengkap *merchant* dan riwayat transaksi kepada *User*.

*Sequence Diagram* untuk fitur melihat detail *User Deposit* ditunjukkan pada Gambar 3.8.



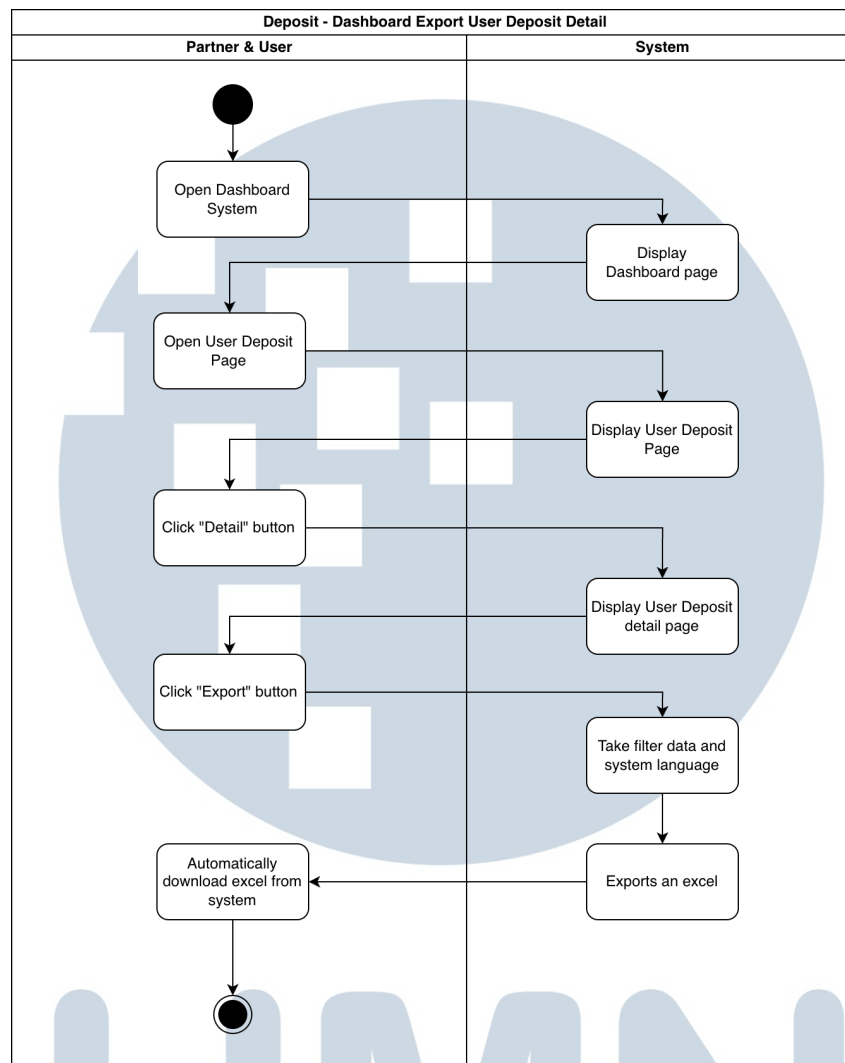
Gambar 3.8. Sequence diagram melihat detail user deposit

Berdasarkan Gambar 3.8, alur melihat detail *User Deposit* dimulai ketika *User* memilih salah satu *merchant* untuk melihat detailnya. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data detail *merchant*. *API* mengambil data dari *Database* dengan mengakses tabel `ids_depositca.tb_r_wallet`, `ids_depositca.tb_r_wallet_history`, `disi.tb_m_user_va`, `disi.tb_m_user`, `disi.tb_m_bank`, dan `disi.tb_m_partner`. Data yang diperoleh kemudian dikembalikan ke *Dashboard* untuk ditampilkan kepada *User*.

#### **B.4 Export Daftar User Deposit**

*Activity Diagram* untuk fitur *export* daftar *User Deposit* ditunjukkan pada Gambar 3.9.

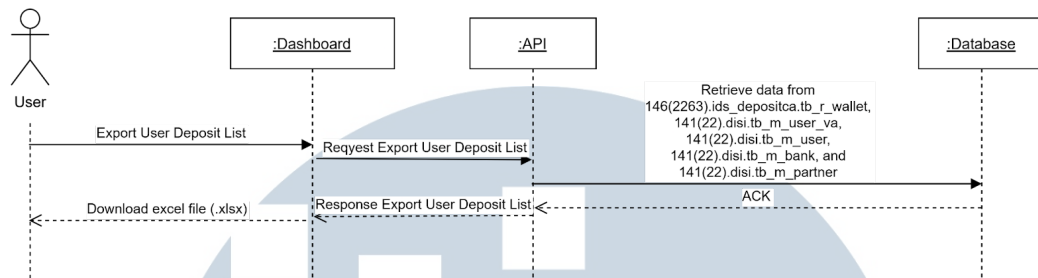




Gambar 3.9. Activity diagram export daftar user deposit

Berdasarkan Gambar 3.9, proses dimulai ketika *User* menekan tombol *export*. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data yang akan di-*export*. *API* melakukan *query* ke *Database* untuk mengambil data sesuai dengan kriteria *filter* yang sedang aktif (jika ada). Data yang diperoleh kemudian dikembalikan ke *API* dan diteruskan ke *Dashboard*. *Dashboard* memproses data dan mengkonversinya ke format Excel (.xlsx) di sisi *frontend*. Setelah file Excel berhasil dibuat, *browser* memulai proses *download* dan file Excel tersimpan di perangkat *User*.

*Sequence Diagram* untuk fitur *export* daftar *User Deposit* ditunjukkan pada Gambar 3.10.



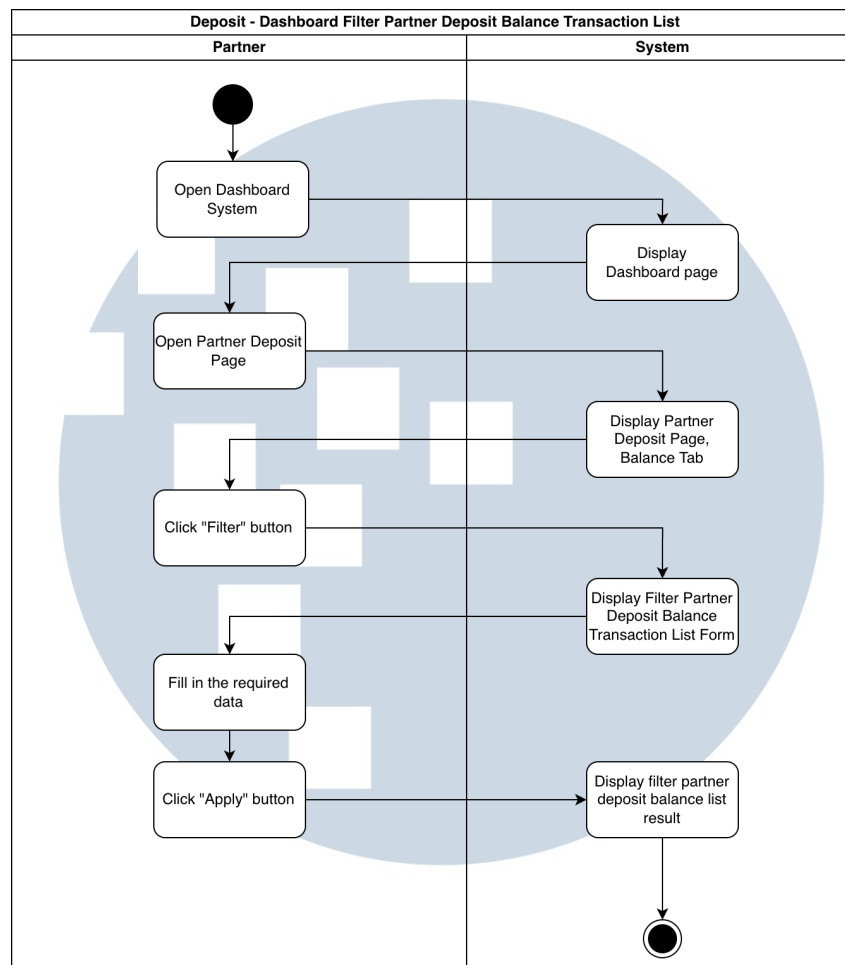
Gambar 3.10. *Sequence diagram export daftar user deposit*

Berdasarkan Gambar 3.10, alur *export* daftar *User Deposit* dimulai ketika *User* menekan tombol *export* pada halaman *User Deposit*. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data yang akan di-*export*. *API* mengambil data dari *Database* dengan mengakses tabel *ids\_depositca.tb\_r\_wallet*, *disi.tb\_m\_user\_va*, *disi.tb\_m\_user*, *disi.tb\_m\_bank*, dan *disi.tb\_m\_partner*. Setelah data diperoleh, *API* mengembalikan *response* dan *Dashboard* mengunduh *file* Excel (.xlsx) ke perangkat *User*.

## B.5 Filter Transaksi Partner Deposit

*Activity Diagram* untuk fitur *filter* transaksi *Partner Deposit* ditunjukkan pada Gambar 3.11.

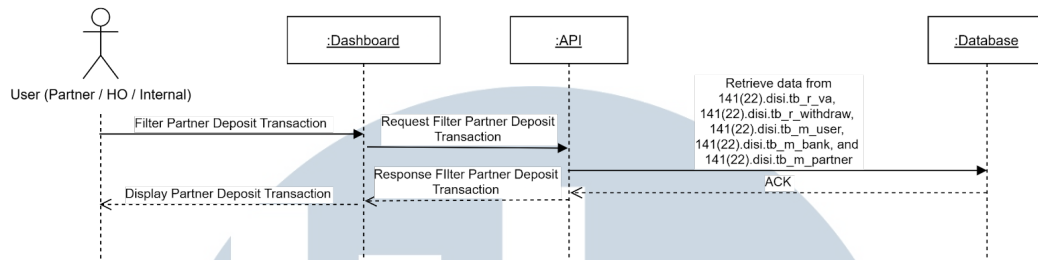
UMN  
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA



Gambar 3.11. Activity diagram filter transaksi partner deposit

Berdasarkan Gambar 3.11, proses dimulai ketika *User* (*Partner* atau *HO*) memasukkan kriteria *filter* seperti rentang tanggal, jenis transaksi (*recharge* atau *withdraw*), atau status transaksi, kemudian menekan tombol "Search" atau "Filter". *Dashboard* mengirimkan *request* ke *API* untuk mengambil data transaksi sesuai *filter*. *API* melakukan *query* ke *Database* dengan mengakses tabel yang relevan. Sistem mengambil data transaksi yang sesuai dengan kriteria *filter* dari *Database*. Data yang diperoleh kemudian dikembalikan ke *API* dan diteruskan ke *Dashboard*. *Dashboard* menampilkan daftar transaksi yang sesuai dengan kriteria *filter* kepada *User* dengan sistem paginasi.

*Sequence Diagram* untuk fitur *filter* transaksi *Partner Deposit* ditunjukkan pada Gambar 3.12.



Gambar 3.12. *Sequence diagram filter transaksi partner deposit*

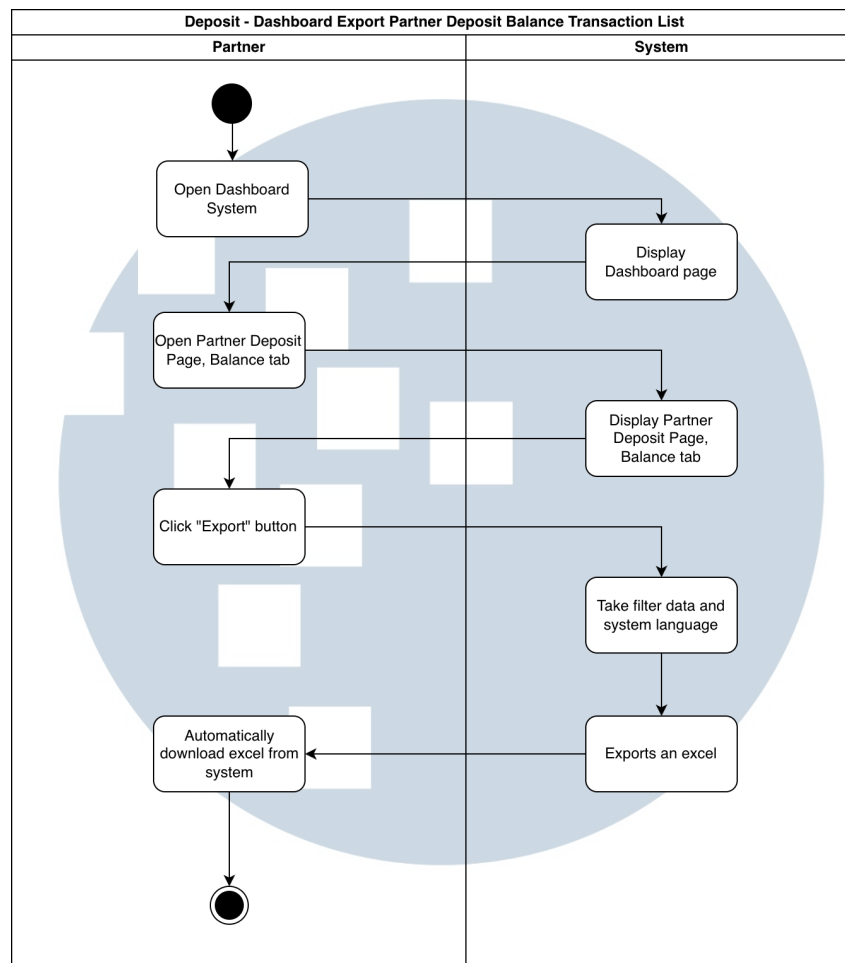
Berdasarkan Gambar 3.12, alur *filter* transaksi *Partner Deposit* dimulai ketika *User* (*Partner* atau *HO*) memasukkan kriteria *filter* pada halaman *Partner Deposit*. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data transaksi sesuai *filter*. *API* mengambil data dari *Database* dengan mengakses tabel *disi.tb\_r\_va*, *disi.tb\_r\_withdraw*, *disi.tb\_m\_user*, *disi.tb\_m\_bank*, dan *disi.tb\_m\_partner*. Data yang diperoleh kemudian dikembalikan ke *Dashboard* untuk ditampilkan kepada *User*.

## B.6 Export Transaksi Partner Deposit

*Activity Diagram* untuk fitur *export* transaksi *Partner Deposit* ditunjukkan pada Gambar 3.13.

UMN  
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

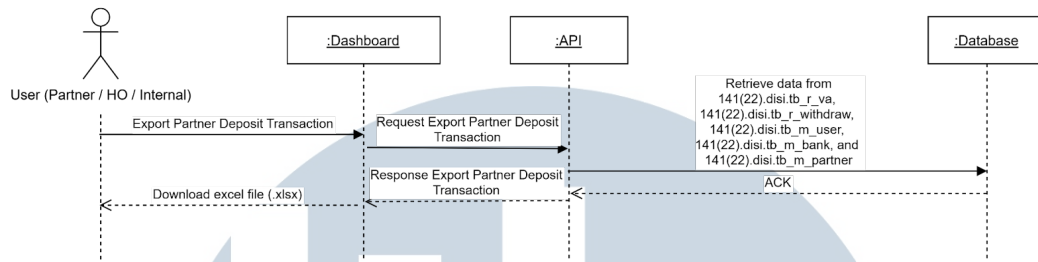




Gambar 3.13. Activity diagram export transaksi partner deposit

Berdasarkan Gambar 3.13, proses dimulai ketika *User* menekan tombol *export*. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data transaksi yang akan di-*export*. *API* melakukan *query* ke *Database* untuk mengambil data transaksi sesuai dengan kriteria *filter* yang sedang aktif (jika ada). Data yang diperoleh kemudian dikembalikan ke *API* dan diteruskan ke *Dashboard*. *Dashboard* memproses data dan mengkonversinya ke format Excel (.xlsx) di sisi *frontend*. Setelah file Excel berhasil dibuat, *browser* memulai proses *download* dan file Excel tersimpan di perangkat *User*.

*Sequence Diagram* untuk fitur *export* transaksi *Partner Deposit* ditunjukkan pada Gambar 3.14.



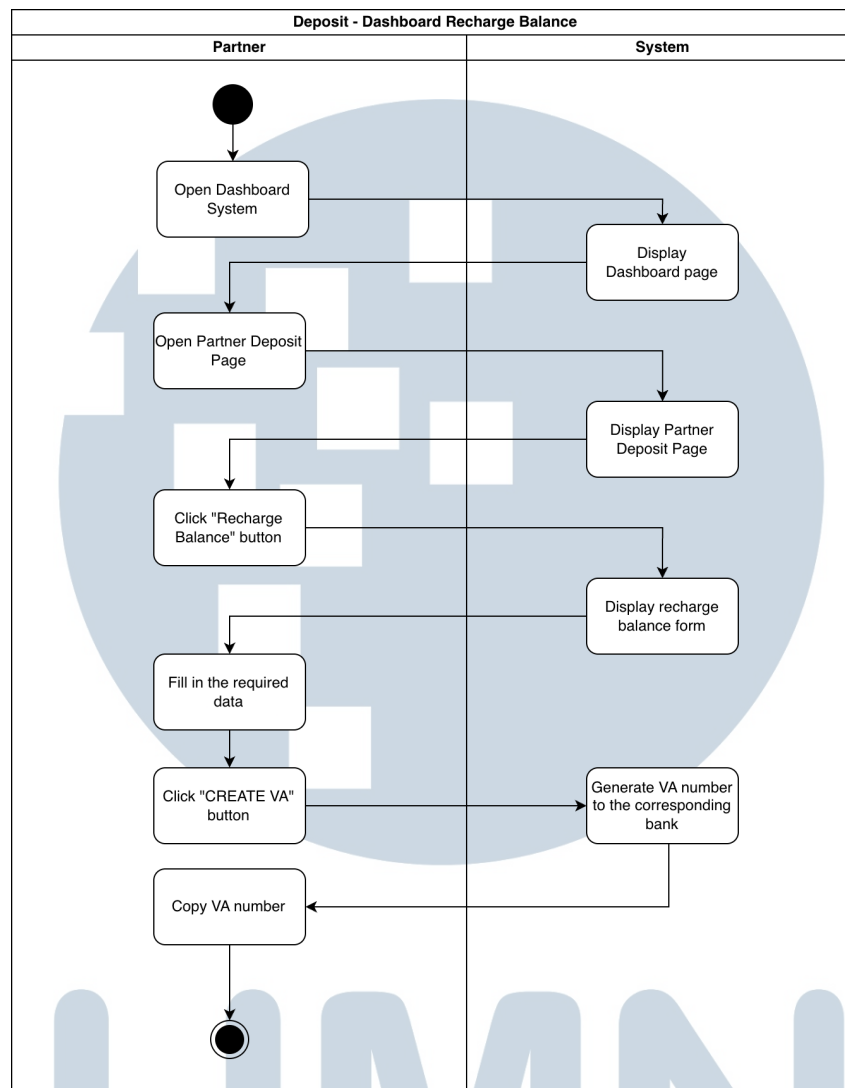
Gambar 3.14. *Sequence diagram export transaksi partner deposit*

Berdasarkan Gambar 3.14, alur *export* transaksi *Partner Deposit* dimulai ketika *User* menekan tombol *export* pada halaman *Partner Deposit*. *Dashboard* mengirimkan *request* ke *API* untuk mengambil data yang akan di-*export*. *API* mengambil data dari *Database* dengan mengakses tabel *disi.tb\_r\_va*, *disi.tb\_r\_withdraw*, *disi.tb\_m\_user*, *disi.tb\_m\_bank*, dan *disi.tb\_m\_partner*. Setelah data diperoleh, *API* mengembalikan *response* dan *Dashboard* mengunduh *file Excel (.xlsx)* ke perangkat *User*.

## B.7 Recharge Saldo (Create VA)

*Activity Diagram* untuk fitur *recharge* saldo ditunjukkan pada Gambar 3.15.

UMN  
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

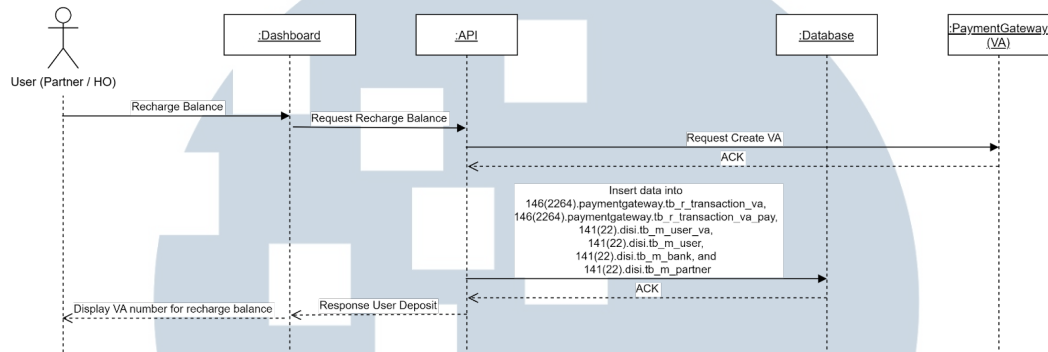


Gambar 3.15. Activity diagram recharge saldo

Berdasarkan Gambar 3.15, proses dimulai ketika *User* (*Partner* atau HO) memasukkan nominal dan memilih bank VA, kemudian menekan tombol submit. *Dashboard* mengirimkan *request* ke *API* untuk membuat VA. *API* kemudian mengirimkan *request* ke *PaymentGateway* untuk membuat nomor VA. Sistem melakukan pengecekan apakah VA berhasil dibuat. Jika pembuatan VA gagal, sistem menampilkan pesan error kepada *User* dan proses dibatalkan. Jika VA berhasil dibuat, data VA disimpan ke *Database*. Setelah data tersimpan, *Dashboard* menampilkan nomor VA yang dapat digunakan oleh *User* untuk melakukan pembayaran. *User* kemudian melakukan pembayaran melalui bank menggunakan nomor VA yang telah dibuat. Setelah pembayaran diterima oleh *PaymentGateway*, sistem secara otomatis memperbarui saldo *partner* di *Database* melalui mekanisme

*callback.*

*Sequence Diagram* untuk fitur *recharge* saldo ditunjukkan pada Gambar 3.16.



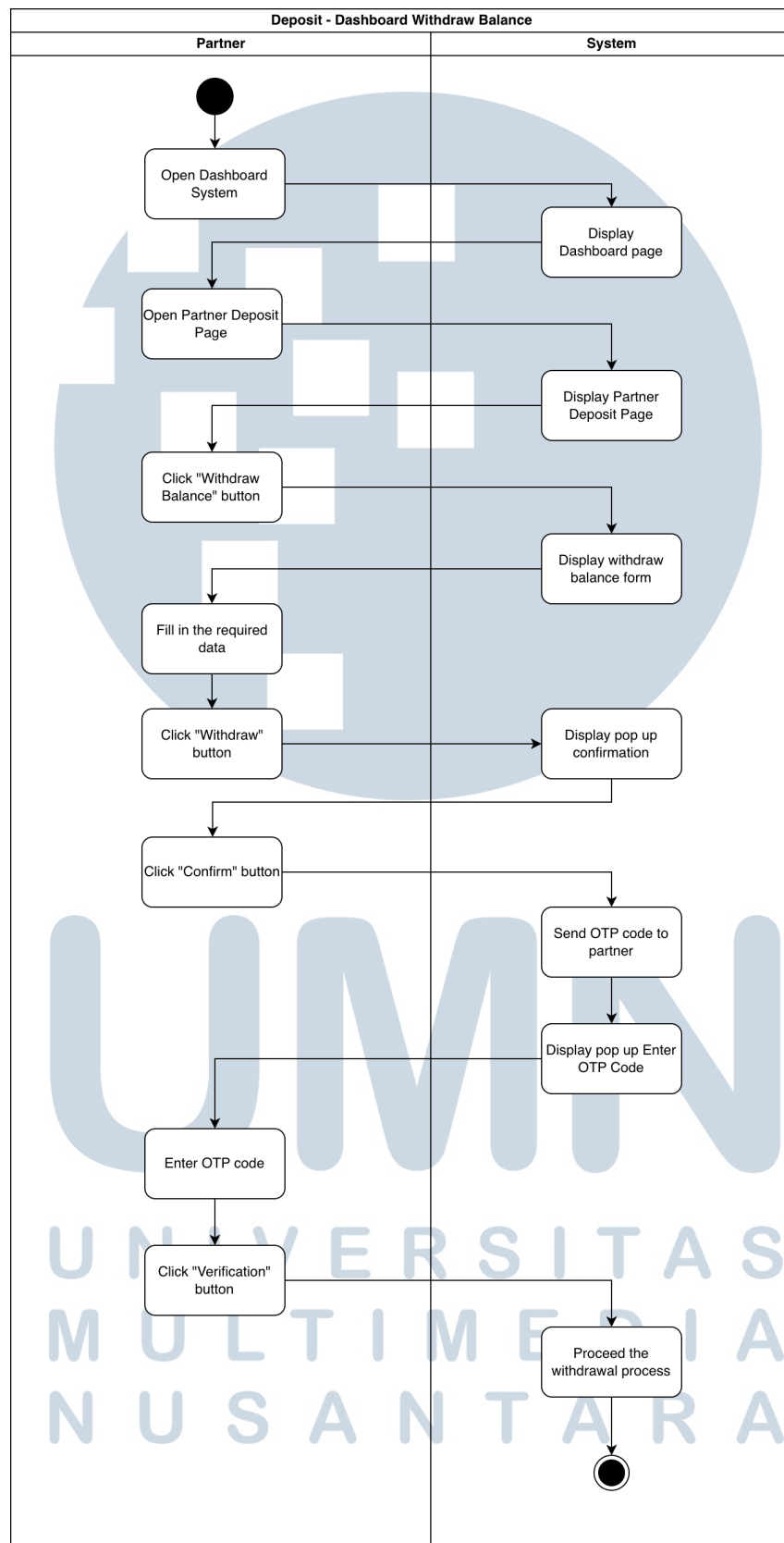
Gambar 3.16. *Sequence diagram recharge* saldo

Berdasarkan Gambar 3.16, alur *recharge* saldo dimulai ketika *User* (*Partner* atau *HO*) mengakses fitur *recharge* dan memasukkan nominal serta memilih bank *VA*. *Dashboard* mengirimkan *request* ke *API* untuk membuat *VA*. *API* kemudian mengirimkan *request* ke *PaymentGateway* untuk membuat nomor *VA*. Setelah *VA* berhasil dibuat, data disimpan ke *Database* pada tabel `paymentgateway.tb_r_transaction_va`, `paymentgateway.tb_r_transaction_va_pay`, `disi.tb_m_user_va`, `disi.tb_m_user`, `disi.tb_m_bank`, dan `disi.tb_m_partner`. *Dashboard* kemudian menampilkan nomor *VA* yang dapat digunakan oleh *User* untuk melakukan pembayaran.

## B.8 Withdraw Saldo

*Activity Diagram* untuk fitur *withdraw* saldo ditunjukkan pada Gambar 3.17.

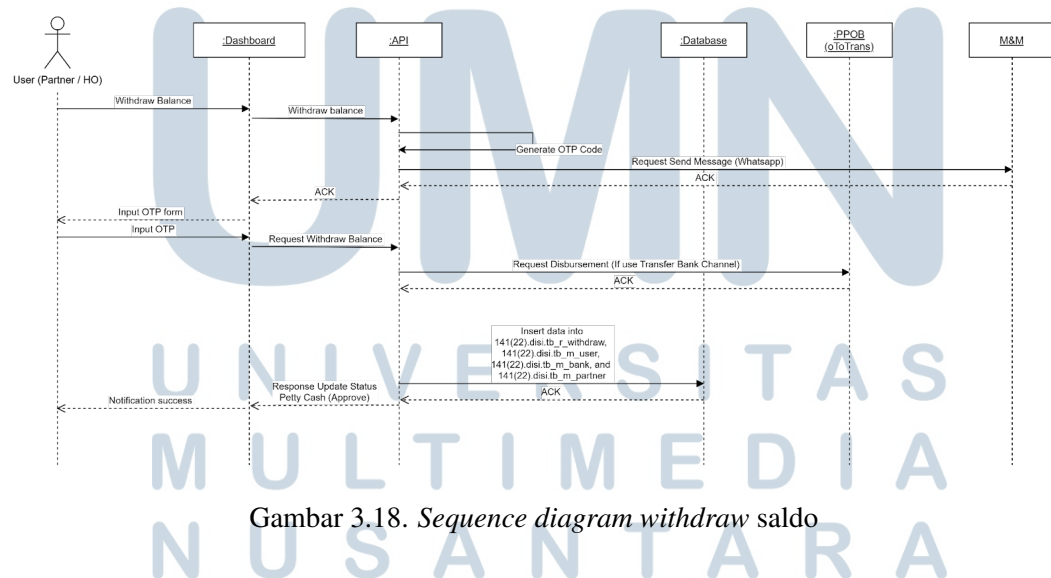
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA



Gambar 3.17. Activity diagram withdraw saldo

Berdasarkan Gambar 3.17, proses dimulai ketika *User* (*Partner* atau *HO*) memasukkan nominal penarikan dan menekan tombol submit. *Dashboard* mengirimkan *request* ke *API* untuk memulai proses *withdraw*. *API* kemudian membuat kode OTP secara otomatis. Setelah OTP dibuat, *API* mengirimkan *request* ke layanan M&M untuk mengirimkan OTP melalui WhatsApp. Sistem melakukan pengecekan apakah OTP berhasil dikirim. Jika pengiriman OTP gagal, sistem menampilkan pesan error dan proses dibatalkan. Jika OTP berhasil terkirim, *Dashboard* menampilkan formulir untuk memasukkan OTP kepada *User*. *User* memasukkan kode OTP yang diterima melalui WhatsApp dan menekan tombol submit. *Dashboard* mengirimkan *request* ke *API* untuk memvalidasi OTP dan memproses *withdraw*. Sistem melakukan validasi OTP. Jika OTP tidak valid, sistem menampilkan pesan "OTP tidak valid" dan meminta *User* untuk memasukkan kembali OTP yang benar. Jika OTP valid, sistem melakukan pengecekan *channel withdraw*. Jika menggunakan *channel* transfer bank, *API* mengirimkan *request* ke *PPOB (oToTrans)* untuk melakukan *disbursement*. Sistem melakukan pengecekan apakah *disbursement* berhasil. Jika gagal, sistem menampilkan pesan error. Jika berhasil, data transaksi *withdraw* disimpan ke *Database*. Setelah proses selesai, *Dashboard* menampilkan notifikasi bahwa *withdraw* berhasil dilakukan.

*Sequence Diagram* untuk fitur *withdraw* saldo ditunjukkan pada Gambar 3.18.



Gambar 3.18. *Sequence diagram withdraw* saldo

Berdasarkan Gambar 3.18, alur *withdraw* saldo dimulai ketika *User* (*Partner* atau *HO*) mengakses fitur *withdraw* dan memasukkan nominal penarikan. *Dashboard* mengirimkan *request* ke *API* untuk memulai proses *withdraw*. *API*

kemudian membuat kode OTP dan mengirimkan *request* ke layanan M&M untuk mengirimkan OTP melalui WhatsApp. Setelah OTP terkirim, *Dashboard* menampilkan formulir untuk memasukkan OTP.

Setelah *User* memasukkan OTP, *Dashboard* mengirimkan *request* ke *API* untuk memvalidasi OTP dan memproses *withdraw*. Jika menggunakan *channel* transfer bank, *API* mengirimkan *request* ke *PPOB (oToTrans)* untuk melakukan *disbursement*. Data transaksi kemudian disimpan ke *Database* pada tabel *disi.tb\_r\_withdraw*, *disi.tb\_m\_user*, *disi.tb\_m\_bank*, dan *disi.tb\_m\_partner*. Setelah proses selesai, *Dashboard* menampilkan notifikasi bahwa *withdraw* berhasil dilakukan.

### C Implementasi Backend

Implementasi *backend* dilakukan menggunakan bahasa pemrograman Java dengan arsitektur REST API. Tabel 3.2 menunjukkan daftar *endpoint* API yang dikembangkan untuk *Sistem Deposit*.

Tabel 3.2. Daftar *Endpoint* API Sistem Deposit

No	Fitur	Endpoint	Method	Deskripsi
1	Registrasi <i>User</i>	/userdeposit/createuser	POST	Membuat <i>user</i> baru
2	<i>Filter User Deposit</i>	/listmerchant	GET	Mengambil daftar <i>merchant</i>
3	Detail <i>User Deposit</i>	/userdeposit/headermerchantdetail	GET	Mengambil <i>header</i> detail
4	Detail <i>User Deposit</i>	/userdeposit/bodymerchantdetail	GET	Mengambil riwayat transaksi
5	<i>Filter Transaksi</i>	/partnerdeposit/partnerrealtrx	GET	Mengambil transaksi <i>partner</i>
6	<i>Recharge Saldo</i>	/partnerdeposit/createvanumber	POST	Membuat nomor VA
Lanjut pada halaman berikutnya				



Tabel 3.2 Daftar *Endpoint* API Sistem Deposit (lanjutan)

No	Fitur	Endpoint	Method	Deskripsi
7	<i>Withdraw</i> Saldo	/partnerdeposit/otpwithdrawpartner	POST	Membuat OTP
8	<i>Withdraw</i> Saldo	/partnerdeposit/issendotpwithdrawpartner	POST	Memperbarui status OTP
9	<i>Withdraw</i> Saldo	/partnerdeposit/withdrawpartner	POST	Melakukan <i>withdraw</i>

Berdasarkan Tabel 3.2, terdapat 9 *endpoint* API yang dikembangkan untuk mendukung fitur-fitur utama pada *Sistem Deposit*. Berikut adalah spesifikasi detail dari beberapa *endpoint* utama.

### C.1 Create User (POST /portal/userdeposit/createuser)

*Endpoint* ini digunakan untuk mendaftarkan *user* atau *merchant* baru ke dalam sistem. Tabel 3.3 menunjukkan spesifikasi *request* dan Tabel 3.4 menunjukkan spesifikasi *response* dari *endpoint* ini.

Tabel 3.3. Spesifikasi *request create user*

Parameter	Tipe Data	Wajib	Deskripsi
trxdate	String	M	Tanggal transaksi
merchantid	String	M	ID <i>merchant</i>
description	String	M	Deskripsi <i>merchant</i>
partnerId	String	M	ID <i>partner</i>
address	String	M	Alamat <i>merchant</i>
phone	String	M	Nomor telepon
email	String	M	Alamat <i>email</i>
bankaccount	String	M	Nomor rekening bank
bankcode	String	M	Kode bank
bankname	String	M	Nama bank
province	String	M	Provinsi
city	String	M	Kota
urbanvillage	String	M	Kelurahan
postalcode	Integer	M	Kode pos
bankva	String	M	Kode bank VA

Tabel 3.4. Spesifikasi *response create user*

Parameter	Tipe Data	Wajib	Deskripsi
responseCode	String	M	Kode <i>response</i>
responseMessage	String	M	Pesan <i>response</i>
trxdate	String	O	Tanggal transaksi
merchantid	String	O	ID <i>merchant</i>
partnerId	String	O	ID <i>partner</i>
bankva	String	O	Kode bank VA
vanumber	String	O	Nomor <i>Virtual Account</i>

### C.2 Create VA Number (POST /portal/partnerdeposit/createvanumber)

*Endpoint* ini digunakan untuk membuat nomor *Virtual Account* (VA) yang dapat digunakan oleh *partner* untuk melakukan *recharge* saldo. Tabel 3.5 menunjukkan spesifikasi *request* dan Tabel 3.6 menunjukkan spesifikasi *response* dari *endpoint* ini.

Tabel 3.5. Spesifikasi *request create VA number*

Parameter	Tipe Data	Wajib	Deskripsi
merchantid	String	M	ID <i>merchant</i>
partnerid	String	M	ID <i>partner</i>
bankva	String	M	Kode bank VA

Tabel 3.6. Spesifikasi *response create VA number*

Parameter	Tipe Data	Wajib	Deskripsi
responseCode	String	M	Kode <i>response</i>
responseMessage	String	M	Pesan <i>response</i>
bankva	String	O	Kode bank VA
vanumber	String	O	Nomor <i>Virtual Account</i>
vaexpired	String	O	Waktu kadaluarsa VA

### C.3 Withdraw Partner (POST /portal/partnerdeposit/withdrawpartner)

*Endpoint* ini digunakan oleh *partner* untuk melakukan penarikan saldo ke rekening bank. Tabel 3.7 menunjukkan spesifikasi *request* dan Tabel 3.8

menunjukkan spesifikasi *response* dari *endpoint* ini.

Tabel 3.7. Spesifikasi *request withdraw partner*

Parameter	Tipe Data	Wajib	Deskripsi
codeOTP	String	M	Kode OTP untuk verifikasi
merchantid	String	M	ID <i>merchant</i>
partnerid	String	M	ID <i>partner</i>
amount	String	M	Jumlah penarikan
trackingref	String	M	Nomor referensi transaksi
description	String	O	Deskripsi penarikan

Tabel 3.8. Spesifikasi *response withdraw partner*

Parameter	Tipe Data	Wajib	Deskripsi
responseCode	String	M	Kode <i>response</i>
responseMessage	String	M	Pesan <i>response</i>
trxdate	String	M	Tanggal transaksi

#### Keterangan:

- M = *Mandatory* (Wajib)
- O = *Optional* (Opsional)

Fitur *export* pada halaman *User Deposit* dan *Partner Deposit* tidak memerlukan *endpoint* terpisah karena proses *export* dilakukan di sisi *frontend* dengan memanfaatkan data yang sudah diambil dari *endpoint filter*.

## D Implementasi Frontend

Implementasi *frontend* dilakukan menggunakan *framework* Next.js dengan bahasa pemrograman TypeScript. Terdapat dua halaman utama yang dikembangkan untuk *Sistem Deposit*.

### D.1 Halaman User Deposit

Halaman *User Deposit* merupakan halaman yang digunakan oleh *Partner* atau HO dan *Merchant* untuk mengelola data *merchant*. Fitur-fitur yang tersedia

pada halaman ini meliputi melihat daftar *merchant* dengan fitur *filter* dan paginasi, mendaftarkan *merchant* baru (khusus *Partner* atau HO), melihat detail *merchant* beserta riwayat transaksi, mengecek saldo *merchant*, dan meng-*export* data ke *file* Excel.

## D.2 Halaman Partner Deposit

Halaman *Partner Deposit* merupakan halaman yang digunakan oleh *Partner* atau HO untuk mengelola saldo deposit. Fitur-fitur yang tersedia pada halaman ini meliputi melihat informasi saldo *real*, *shadow*, *pending*, dan *debt*, melihat daftar transaksi dengan fitur *filter* dan paginasi, melakukan *recharge* saldo melalui pembuatan *Virtual Account* (VA), melakukan *withdraw* saldo ke rekening bank dengan verifikasi OTP, dan meng-*export* data transaksi ke *file* Excel.

### D.2.1 Sub-menu History Transaction

Pada halaman *Partner Deposit*, terdapat sub-menu *History Transaction* yang menampilkan riwayat lengkap dari seluruh transaksi yang dilakukan oleh *partner*, baik transaksi *recharge* maupun *withdraw*. Sub-menu ini memberikan transparansi penuh kepada *partner* mengenai seluruh aktivitas keuangan yang terjadi pada akun deposit mereka.

Fitur-fitur yang tersedia pada sub-menu *History Transaction* meliputi:

1. **Filter Transaksi:** *Partner* dapat melakukan *filter* berdasarkan rentang tanggal, jenis transaksi (*recharge* atau *withdraw*), dan status transaksi (berhasil, *pending*, atau gagal). Fitur *filter* ini memudahkan *partner* untuk mencari transaksi spesifik tanpa harus melihat seluruh riwayat transaksi.
2. **Detail Transaksi:** Setiap baris transaksi menampilkan informasi detail seperti nomor referensi transaksi, tanggal dan waktu transaksi, jenis transaksi (*recharge* atau *withdraw*), nominal transaksi, status transaksi (berhasil, *pending*, atau gagal), dan keterangan tambahan jika ada.
3. **Paginasi:** Daftar transaksi ditampilkan dengan sistem paginasi untuk memudahkan navigasi ketika jumlah transaksi cukup banyak. *Partner* dapat memilih jumlah data yang ditampilkan per halaman dan berpindah antar halaman dengan mudah.

4. **Export Data:** *Partner* dapat meng-*export* data transaksi yang telah di-*filter* ke dalam format Excel (.xlsx) untuk keperluan pelaporan, analisis, atau rekonsiliasi keuangan lebih lanjut. Fitur *export* ini memproses data di sisi *frontend* sehingga dapat dilakukan dengan cepat.

Dengan adanya sub-menu *History Transaction*, *partner* dapat dengan mudah melakukan *monitoring* terhadap seluruh aktivitas transaksi deposit mereka, melakukan rekonsiliasi keuangan secara mandiri, dan memantau arus kas secara detail tanpa perlu menghubungi admin untuk meminta laporan transaksi.

## E Pengujian Sistem

Pengujian sistem dilakukan menggunakan metode *Black Box Testing* untuk memastikan bahwa setiap fitur berfungsi sesuai dengan spesifikasi yang telah ditentukan. *Black Box Testing* berfokus pada pengujian fungsionalitas sistem tanpa memperhatikan struktur internal kode program. Pengujian dilakukan dengan berbagai skenario, termasuk skenario *positive testing* (input valid) dan *negative testing* (input tidak valid atau kondisi error).

Tabel 3.9 menunjukkan hasil pengujian *Black Box* terhadap fitur-fitur utama Sistem Deposit.

Tabel 3.9. Hasil *Black Box Testing* Sistem Deposit

No	Fitur yang Diuji	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Registrasi <i>Merchant</i>	Mengisi formulir dengan data valid dan menekan tombol submit	<i>Merchant</i> berhasil terdaftar, VA dibuat, dan notifikasi sukses ditampilkan	Berhasil
2	Registrasi <i>Merchant</i>	Mengisi formulir dengan data tidak lengkap	Sistem menampilkan pesan validasi pada field yang kosong	Berhasil
Lanjut pada halaman berikutnya				

Tabel 3.9 Hasil *Black Box Testing* Sistem Deposit (lanjutan)

No	Fitur yang Diuji	Skenario Pengujian	Hasil yang Diharapkan	Status
3	Registrasi <i>Merchant</i>	Mengisi formulir dengan email tidak valid	Sistem menampilkan pesan "Format email tidak valid"	Berhasil
4	<i>Filter User Deposit</i>	Memasukkan kriteria <i>filter</i> dan menekan tombol cari	Daftar <i>merchant</i> yang sesuai kriteria ditampilkan	Berhasil
5	<i>Filter User Deposit</i>	Memasukkan kriteria <i>filter</i> yang tidak ada datanya	Sistem menampilkan pesan "Data tidak ditemukan"	Berhasil
6	<i>Detail User Deposit</i>	Memilih salah satu <i>merchant</i> dari daftar	Detail <i>merchant</i> dan riwayat transaksi ditampilkan	Berhasil
7	<i>Export User Deposit</i>	Menekan tombol <i>export</i> pada halaman <i>User Deposit</i>	<i>File Excel</i> berhasil diunduh dengan data yang sesuai	Berhasil
8	<i>Export User Deposit</i>	Menekan tombol <i>export</i> ketika tidak ada data	Sistem menampilkan pesan "Tidak ada data untuk di-export"	Berhasil
9	<i>Recharge Saldo</i>	Memasukkan nominal dan memilih bank VA, lalu submit	Nomor VA berhasil dibuat dan ditampilkan	Berhasil
10	<i>Recharge Saldo</i>	Memasukkan nominal di bawah minimum	Sistem menampilkan pesan "Nominal minimal adalah Rp X"	Berhasil
Lanjut pada halaman berikutnya				



Tabel 3.9 Hasil *Black Box Testing* Sistem Deposit (lanjutan)

No	Fitur yang Diuji	Skenario Pengujian	Hasil yang Diharapkan	Status
11	<i>Recharge</i> Saldo	Tidak memilih bank VA	Sistem menampilkan pesan "Pilih bank VA terlebih dahulu"	Berhasil
12	<i>Withdraw</i> Saldo	Memasukkan nominal <i>withdraw</i> dan submit	OTP terkirim dan formulir input OTP ditampilkan	Berhasil
13	<i>Withdraw</i> Saldo	Memasukkan OTP yang benar dan submit	<i>Withdraw</i> berhasil diproses dan saldo berkurang	Berhasil
14	<i>Withdraw</i> Saldo	Memasukkan OTP yang salah	Sistem menampilkan pesan "OTP tidak valid"	Berhasil
15	<i>Withdraw</i> Saldo	Memasukkan nominal melebihi saldo	Sistem menampilkan pesan "Saldo tidak mencukupi"	Berhasil
16	<i>Withdraw</i> Saldo	Memasukkan nominal di bawah minimum	Sistem menampilkan pesan "Nominal minimal withdraw adalah Rp X"	Berhasil
19	<i>Export Partner Deposit</i>	Menekan tombol <i>export</i> pada halaman transaksi	<i>File</i> Excel berhasil diunduh dengan data transaksi	Berhasil
20	<i>History Transaction</i>	Mengakses sub-menu <i>History Transaction</i>	Riwayat transaksi ditampilkan dengan paginasi	Berhasil
Lanjut pada halaman berikutnya				



Tabel 3.9 Hasil *Black Box Testing* Sistem Deposit (lanjutan)

No	Fitur yang Diuji	Skenario Pengujian	Hasil yang Diharapkan	Status
21	<i>History Transaction</i>	Melakukan <i>filter</i> berdasarkan jenis transaksi	Transaksi sesuai jenis yang dipilih ditampilkan	Berhasil
22	<i>History Transaction</i>	Melakukan <i>export</i> data dari <i>History Transaction</i>	<i>File</i> Excel berhasil diunduh	Berhasil
23	Paginasi	Navigasi ke halaman selanjutnya	Data pada halaman selanjutnya ditampilkan dengan benar	Berhasil
24	Paginasi	Navigasi ke halaman sebelumnya	Data pada halaman sebelumnya ditampilkan dengan benar	Berhasil
25	Paginasi	Mengubah jumlah data per halaman	Jumlah data yang ditampilkan sesuai dengan pilihan	Berhasil

Berdasarkan Tabel 3.9, seluruh fitur yang diuji telah berfungsi dengan baik sesuai dengan spesifikasi yang ditentukan. Pengujian mencakup skenario *positive testing* yang menguji fungsi normal sistem dengan input yang valid, serta skenario *negative testing* yang menguji bagaimana sistem menangani input yang tidak valid atau kondisi error. Hasil pengujian menunjukkan bahwa Sistem Deposit dapat menangani berbagai kondisi dengan baik dan memberikan *response* yang sesuai kepada *user*, baik dalam kondisi normal maupun kondisi error. Sistem juga mampu menampilkan pesan kesalahan yang informatif ketika *user* melakukan kesalahan input, sehingga membantu *user* untuk memperbaiki kesalahan tersebut.

### 3.3.2 Kendala yang Ditemukan

Selama pelaksanaan pengembangan *Sistem Deposit*, kendala utama yang dihadapi adalah pengembangan REST API menggunakan bahasa pemrograman Java merupakan pengalaman pertama. Hal ini menyebabkan diperlukannya waktu

tambahan untuk mempelajari sintaks, struktur proyek, dan *best practice* dalam pengembangan API menggunakan Java.

### 3.3.3 Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala tersebut, beberapa langkah solusi telah dilakukan. Pertama, mempelajari *codebase* yang sudah ada di perusahaan sebagai referensi untuk memahami struktur proyek dan pola pengembangan yang digunakan. Kedua, melakukan diskusi dengan pembimbing lapangan, yaitu Bapak Vincent Pribadi, untuk mendapatkan arahan mengenai arsitektur sistem dan standar *coding* yang diterapkan di perusahaan. Ketiga, mencari referensi dari internet melalui tutorial, dokumentasi resmi Java, dan forum seperti Stack Overflow untuk memperdalam pemahaman teknis. Keempat, melakukan pengujian setiap *endpoint* menggunakan Postman sebelum diintegrasikan dengan *frontend* untuk memastikan API berfungsi dengan baik.

