BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama menjalani program magang di PT Infosys Solusi Terpadu, posisi yang ditempati adalah *Junior Backend Developer* dengan status *intern*. Posisi tersebut berada di bawah *Retail Banking Division* dan pembimbing lapangan adalah *Tech Lead Backend Developer* yang bertanggung jawab atas supervisi langsung. Dalam struktur tim pengembangan aplikasi Hibank, pembimbing lapangan berperan sebagai atasan yang melakukan supervisi terhadap pelaksanaan tugas harian. Tim pengembangan terdiri dari 27 anggota dengan komposisi peran yang dapat berubah sesuai kebutuhan proyek, meliputi *Project Manager*, *Backend Developer*, *Frontend Developer*, *Quality Assurance*, dan fungsi pendukung lainnya.

Untuk keperluan koordinasi harian, WhatsApp digunakan sebagai media komunikasi informal, sedangkan Asana dimanfaatkan untuk pengelolaan tugas, pelaporan progres, dan dokumentasi. Pengendalian versi kode dilakukan menggunakan Git dengan *repository* yang dikelola pada GitLab dan diakses melalui jaringan VPN (*Virtual Private Network*) internal perusahaan. Proses pengembangan kode mengikuti pola *branching* konvensional, seperti `feature/` untuk pengembangan fitur baru dan `hotfix/` untuk perbaikan *bug*. Proses integrasi kode dilakukan secara bertahap. Kode diuji terlebih dahulu pada *environment* SIT (*System Integrated Testing*) oleh tim QA (*Quality Assurance*) internal, kemudian dipromosikan ke *environment* UAT (*User Acceptance Testing*) untuk validasi oleh tim QA Hibank, dan setelah lolos dapat dipromosikan ke *environment production*.

3.2 Tugas yang Dilakukan

Tugas yang dilaksanakan selama periode 2 magang adalah sebagai berikut.

1. Membuat *backend* untuk fitur Hi-Komunitas menggunakan Java dengan *framework* Spring Boot dan basis data PostgreSQL. Pekerjaan ini mencakup pembuatan 15 *endpoint* API (*Application Programming Interface*) baru serta modifikasi dan integrasi pada 20 *endpoint* API lama untuk mendukung alur fungsionalitas *shared wallet*.

2. Perbaikan *bug* (*bug fixing*) pada berbagai layanan *backend* yang telah

ada, baik yang berkaitan langsung dengan integrasi fitur Hi-Komunitas maupun *microservice* lain yang memerlukan pemeliharaan sesuai dengan permasalahan yang ditemukan oleh pihak QA Hibank ataupun QA tim internal.

3. Peningkatan kualitas kode melalui pengembangan *unit test* dengan JUnit dan Mockito untuk meminimalisir risiko regresi dan meningkatkan reliabilitas sistem.

Berikut adalah rincian *endpoint* yang dikerjakan, yang terbagi menjadi pengembangan API baru dan modifikasi API lama.

3.2.1 Pengembangan 15 API Baru

API berikut dibangun dari nol untuk menangani siklus hidup *Shared Wallet*.

1. `POST /hikomunitas/create/inquiry`
2. `POST /hikomunitas/create/posting`
3. `POST /hikomunitas/member/inquiry`
4. `POST /hikomunitas/member/posting`
5. `POST /hikomunitas/member/update/inquiry`
6. `POST /hikomunitas/member/update/posting`
7. `POST /hikomunitas/invitation/cancel`
8. `GET /hikomunitas/member/list`
9. `GET /hikomunitas/member/detail`
10. `PUT /hikomunitas/update-name`
11. `GET /hikomunitas/list`
12. `GET /hikomunitas/detail`
13. `GET /hikomunitas/statement`
14. `POST /hikomunitas/invitation/accept`
15. `POST /hikomunitas/invitation/decline`

3.2.2 Modifikasi 20 API Eksisting

Logika API berikut dimodifikasi agar dapat memproses data dari sumber Hi-Komunitas.

1. `POST /user/registration/non-cif`
2. `POST /user/mpin/create/inquiry`
3. `POST /user/mpin/create/posting`
4. `GET /transaction/history/pdf`
5. `POST /payment/bpjs/inquiry`
6. `POST /payment/bpjs/posting`
7. `POST /payment/pdam/inquiry`
8. `POST /payment/pdam/posting`
9. `POST /payment/pln-nontaglis/inquiry`
10. `POST /payment/pln-nontaglis/posting`
11. `POST /payment/pln-postpaid/inquiry`
12. `POST /payment/pln-postpaid/posting`
13. `POST /payment/telco-postpaid/inquiry`
14. `POST /payment/telco-postpaid/posting`
15. `POST /purchase/ewallet/inquiry`
16. `POST /purchase/ewallet/posting`
17. `POST /purchase/pln/inquiry`
18. `POST /purchase/pln/posting`
19. `POST /purchase/telco/inquiry`
20. `POST /purchase/telco/posting`

3.3 Uraian Pelaksanaan Magang

Rangkaian kegiatan magang dialokasikan secara proporsional pada tiga aspek utama pengembangan layanan *backend* Hi-Komunitas, yaitu implementasi fitur baru, penanganan *bug* kritikal (baik di lingkungan *Production* maupun UAT), serta pengembangan *Unit Test* yang komprehensif. Ketiga aktivitas ini dijalankan secara sinergis guna menjaga stabilitas sistem dan kualitas kode (*code quality*). Rincian pekerjaan mingguan selengkapnya dapat dilihat pada Tabel 3.1.

Tabel 3.1. Detail Pekerjaan yang Dilakukan

Minggu Ke-	**Deskripsi Pekerjaan dan Aktivitas**
1	Melakukan analisis FSD Hi-Komunitas melakukan *environment setup* serta perbaikan *minor* pada fitur *Dropdown* Urutan di sistem lama agar sesuai spesifikasi UI (*User Interface*).
2	Melakukan konfigurasi awal model data *database schema* untuk proyek Hi-Komunitas dan mempelajari arsitektur *microservices* eksisting.
3	Finalisasi penyiapan *development environment* melakukan kloning repositori serta analisis awal untuk integrasi sistem Hi-Komunitas dengan layanan inti *Core Banking*.
4	Memperbaiki *bug* gambar identitas pengguna yang tidak tersimpan ke tabel *opening account* pasca validasi serta menambahkan *field response* tambahan pada API.
5	Menyelesaikan perbaikan isu produksi penyesuaian nama alias dan akun mengizinkan karakter garis miring pada *form* registrasi dan memunculkan kembali *Recent List* serta memulai inisiasi API *Create Shared Wallet*.
6	Mengembangkan REST API (*Representational State Transfer Application Programming Interface*) inti untuk pembuatan *shared wallet Inquiry* dan *Posting* serta memulai implementasi *API Add Member* beserta validasi nomor ponsel dan rekening.
7	Menyelesaikan API *Add Member* melakukan *code review* dan perbaikan logika validasi termasuk memastikan notifikasi terkirim ke anggota yang baru ditambahkan.

Tabel 3.1. Detail Pekerjaan yang Dilakukan (Lanjutan)

Minggu Ke-	**Deskripsi Pekerjaan dan Aktivitas**
8	Mengimplementasikan fitur *Audit Trails* pada seluruh layanan Hi-Komunitas melakukan perbaikan ulang pada isu sebelumnya dan memperbaiki isu struk transaksi *Shareout Receipt*.
9	Melakukan *enhancement* pada *Payment Service* untuk transaksi Hi-Komunitas menangani isu kritikal dimana mutasi rekening tercatat duplikat saat pembukaan rekening Hi-Nabung.
10	Menyelesaikan implementasi *Payment Service* membantu perbaikan di QRIS *Service* serta melakukan *debugging* pada fitur *Add Invitation* yang gagal menghapus undangan lama akibat kesalahan parameter pada *layer repository*.
11	Melakukan perbaikan pada *Account Service* pasca pengujian QA serta menyusun dokumentasi teknis terkait perubahan API dan struktur tabel.
12	Memperbaiki isu *database mismatch* pada lingkungan pengembangan serta menyesuaikan tipe data (*data type*) untuk tampilan saldo agar presisi.
13	Mengoptimalkan *API Member List Request* serta memberikan dukungan teknis *Support* QA untuk siklus pengujian integrasi.
14	Melanjutkan dukungan QA memperbaiki logika pengecekan *limit* anggota dan saldo utama pada *Payment* dan *Purchase Service*.
15	Melanjutkan *support* QA dan *fixing* pada *Payment* dan *Purchase Service* terkait pengecekan *limit* dan saldo serta melakukan merge pembaruan *Audit Trail* ke *main branch*.
16	Melakukan *setup manifest variable* untuk *deployment* serta menyelesaikan implementasi fitur *download transaction* pada Hi-Komunitas.
17	Melakukan pemantauan hasil *deployment* awal memperbaiki format *limit* transaksi yang salah serta membuat dokumentasi teknis terkait perubahan API.

Tabel 3.1. Detail Pekerjaan yang Dilakukan (Lanjutan)

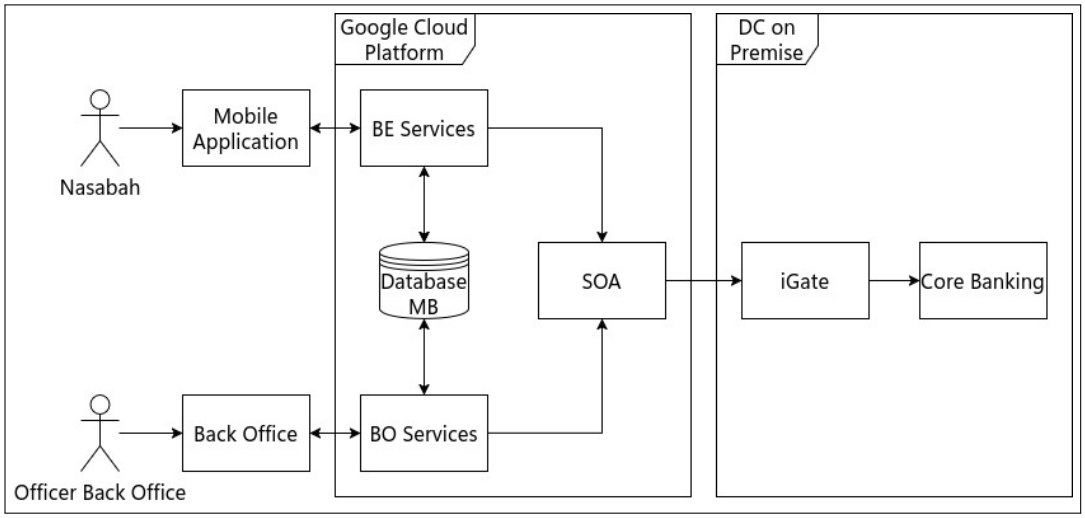
Minggu Ke-	Deskripsi Pekerjaan dan Aktivitas
18	Menyelesaikan perbaikan isu transaksi QRIS gagal akibat nomor referensi kosong perbaikan tanggal kedaluwarsa dan memperbaiki notifikasi transaksi.
19	Mengimplementasikan *Unit Test* untuk fitur *login* guna meningkatkan *code coverage* serta melakukan persiapan rilis ke lingkungan UAT.
20	Memberikan dukungan pengujian *Testing Support* untuk fitur OTP serta melakukan analisis *log error* pada skenario kegagalan *login*.
21	Melanjutkan dukungan pengujian untuk skenario *error login* dan validasi keamanan memastikan penanganan *exception* berjalan sesuai standar.
22	Menambahkan *Audit Trails* pada layanan *Virtual Card* serta memulai implementasi *Unit Test* untuk skenario transaksi QRIS.
23	Melanjutkan pengembangan *Unit Test* QRIS untuk mencakup berbagai skenario *positive and negative case* serta melakukan refaktorisasi kode pengujian.
24	Menyelesaikan seluruh implementasi *Unit Test* QRIS melakukan finalisasi kode sebelum fokus dialihkan pada penyusunan laporan akhir magang.

3.3.1 Pembuatan Layanan *Backend* Hi-Komunitas

Subbab ini menguraikan tahapan teknis dalam pembangunan layanan *backend* Hi-Komunitas secara sistematis. Pembahasan diawali dengan Gambaran Umum Arsitektur Sistem guna memberikan konteks terkait lingkungan pengembangan aplikasi Hibank, dilanjutkan dengan penjabaran *User Requirements* yang menjadi dasar spesifikasi fitur. Selanjutnya, alur logika dan struktur data dimodelkan dalam tahap Perancangan Sistem menggunakan standar UML, dan diakhiri dengan bagian Implementasi yang menampilkan antarmuka aplikasi sebagai representasi visual dari fungsionalitas *backend* yang telah terintegrasi.

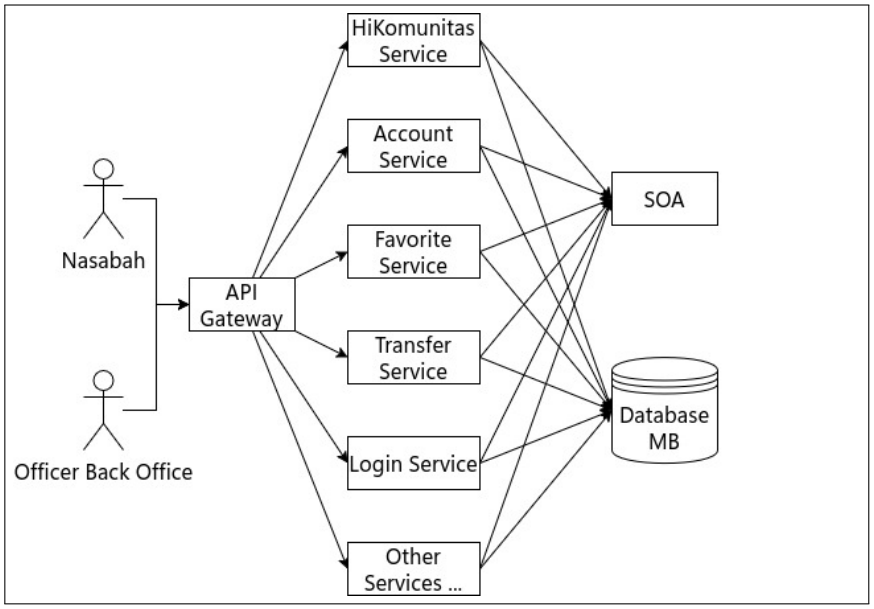
A Gambaran Umum Arsitektur Sistem

Arsitektur sistem aplikasi Hibank dirancang dengan pendekatan terdistribusi, di mana setiap komponen memiliki tanggung jawab spesifik sesuai domainnya. Sistem terbagi menjadi dua lingkungan utama, yaitu Google Cloud Platform (GCP) yang menjalankan berbagai layanan *microservice* aplikasi, serta *Data Center* (DC) *on-premise* yang menyimpan sistem inti perbankan (*core banking*) dan data sensitif. Kedua lingkungan ini dihubungkan melalui *middleware Service Orchestration API* (SOA), yang berfungsi sebagai jembatan komunikasi aman menuju iGate, seperti ditunjukkan pada Gambar 3.1.



Gambar 3.1. Skema Arsitektur Umum Sistem Aplikasi Hibank
Sumber [11]

Di domain GCP, layanan *backend* dikembangkan dengan arsitektur *microservices* yang terpisah menurut fungsinya, seperti Hi-Komunitas *Service*, *Account Service*, dan *Favorite Service*. Komunikasi antar layanan dilakukan melalui *API Gateway* untuk menjaga keamanan, standarisasi, dan kontrol alur data, seperti ditunjukkan pada Gambar 3.2. Pendekatan ini memudahkan penambahan fitur baru tanpa mengganggu layanan lain serta meningkatkan efisiensi dalam proses *debugging*, pemantauan, dan skalabilitas sistem.



Gambar 3.2. Skema Arsitektur *Microservice*
Sumber [11]

Selain pemisahan layanan berbasis fungsi, arsitektur sistem juga menerapkan pola transaksi dua tahap (*two-phase transaction pattern*). Penerapan pola ini bertujuan vital untuk mencegah inkonsistensi data (*data inconsistency*) yang mungkin terjadi akibat kegagalan jaringan di tengah proses. Oleh karena itu, mekanisme transaksi krusial dibagi menjadi dua fase distingtif sebagai berikut.

1. Fase *Inquiry* (Validasi) adalah tahap awal di mana sistem melakukan pemeriksaan terhadap data yang dikirimkan pengguna, termasuk pengecekan kelengkapan informasi, status entitas terkait, serta validasi aturan bisnis tanpa melakukan perubahan apa pun pada data di sistem. Fase ini sepenuhnya bersifat *read-only* dan berfungsi memastikan bahwa permintaan pengguna memenuhi syarat untuk diproses lebih lanjut.

2. Fase *Posting* (Eksekusi) adalah tahap final di mana sistem menerapkan perubahan yang sebelumnya telah divalidasi. Fase ini dimulai dengan proses otorisasi keamanan, seperti verifikasi MPIN (*Mobile Personal Identification Number*). Jika otorisasi berhasil, sistem akan melanjutkan eksekusi dan melakukan pembaruan data secara permanen (*commit*) pada *database* maupun *Core Banking*.

Kombinasi antara infrastruktur *microservices* yang terisolasi dan mekanisme transaksi defensif di atas membentuk fondasi teknis yang andal. Kestabilan

arsitektur ini menjadi prasyarat mutlak untuk dapat mengakomodasi kompleksitas aturan bisnis dan spesifikasi fungsional yang akan dijabarkan pada analisis kebutuhan pengguna berikut.

B User Requirements

Analisis kebutuhan pengguna didasarkan pada dokumen *Functional Specification Document* (FSD) versi final, yaitu ”Enhancement Hi-Komunitas V1.0”. Meskipun dokumen referensi menggunakan nama *”Enhancement”*, seluruh implementasi teknis yang dilakukan selama periode magang merupakan pengembangan awal (*initial development*) untuk membangun fondasi layanan *backend* dari dasar.

Sistem *backend* dirancang untuk memenuhi kebutuhan fungsional berikut guna mendukung ekosistem Hi-Komunitas finansial.

1. Pengguna pemilik dapat membuat dan mengelola hingga tiga *shared wallet* yang terintegrasi dengan sistem perbankan utama, termasuk memantau saldo dan daftar anggota secara terpusat.

2. Pemilik memiliki kendali penuh atas manajemen keanggotaan, meliputi kemampuan mengundang hingga lima anggota (baik melalui kontak maupun *input* manual), validasi status nasabah otomatis oleh sistem, serta wewenang untuk memberhentikan akses anggota.

3. Pemilik berwenang menetapkan aturan penggunaan dana yang spesifik bagi setiap anggota, mencakup pembatasan jenis transaksi, penetapan *limit* nominal berkala, serta penentuan masa berlaku akses.

4. Anggota yang telah bergabung dapat melakukan transaksi finansial sesuai batasan yang diberikan pemilik, dengan hak akses data yang dibatasi hanya pada riwayat transaksi pribadi demi menjaga privasi antar anggota.

5. Sistem menjamin keamanan dan transparansi melalui validasi MPIN untuk setiap perubahan data krusial, serta penyediaan fitur *monitoring* dan notifikasi transaksi secara *real-time* bagi pemilik maupun anggota terkait.

Kebutuhan fungsional yang telah dijabarkan di atas selanjutnya diterjemahkan ke dalam model teknis pada tahap perancangan sistem. Pemodelan visual dilakukan untuk memastikan seluruh logika bisnis dan batasan sistem dapat diimplementasikan secara presisi pada arsitektur aplikasi.

C Perancangan Sistem

Tahap perancangan sistem bertujuan untuk menerjemahkan kebutuhan pengguna ke dalam model teknis yang lebih spesifik. Mengingat sistem ini dibangun dengan pendekatan berorientasi objek (*Object-Oriented Programming*) dan arsitektur *microservices*, pemodelan dilakukan menggunakan standar UML (*Unified Modeling Language*). Perancangan ini difokuskan pada visualisasi logika bisnis (*business logic*) dan interaksi teknis antar-komponen untuk seluruh fitur yang telah didefinisikan dalam lingkup pengembangan layanan Hi-Komunitas.

C.1 Use Case Diagram

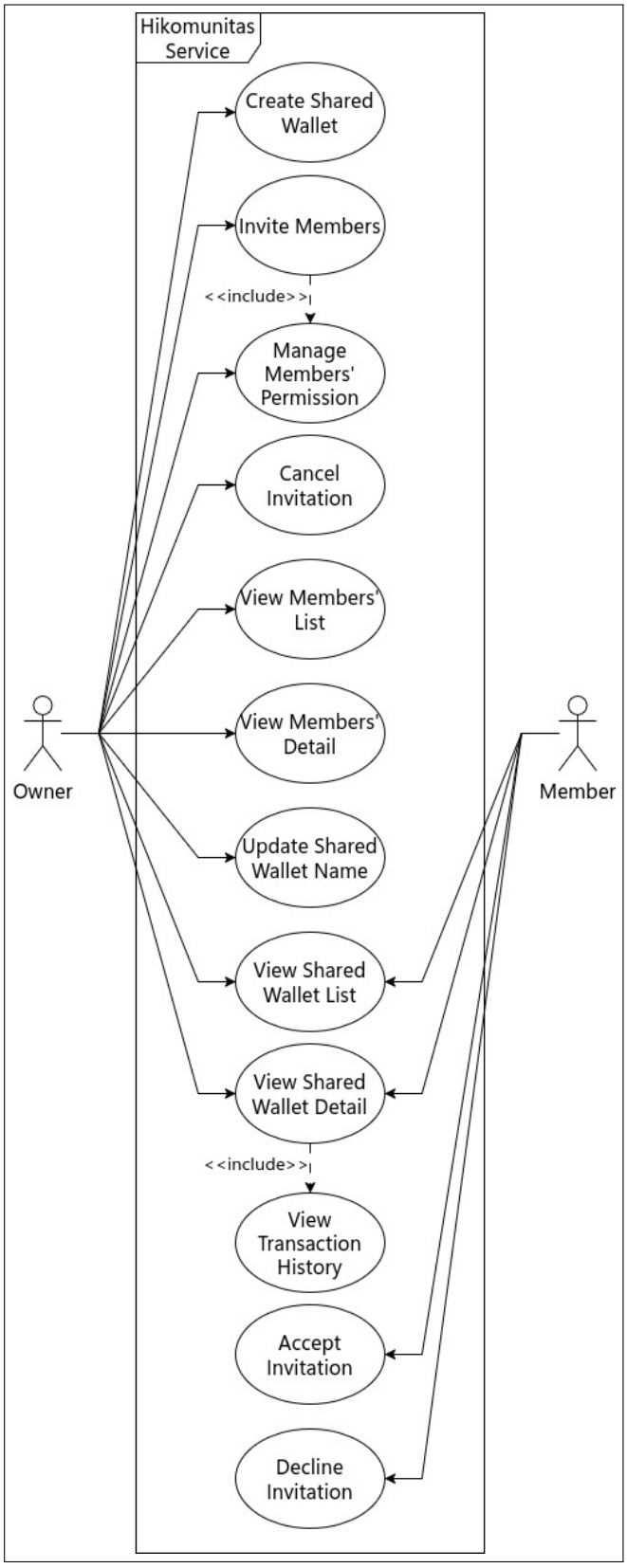
Perancangan fungsionalitas sistem dimodelkan menggunakan *Use Case Diagram* untuk memvisualisasikan interaksi tingkat tinggi antara aktor pengguna dan batasan sistem (*system boundary*). Diagram ini disusun berdasarkan spesifikasi *Unified Modeling Language* (UML) untuk mendefinisikan kebutuhan perilaku (*behavioral requirements*) layanan Hi-Komunitas.

Terdapat dua aktor utama yang berinteraksi dengan sistem sebagai berikut.

1. Aktor *Owner* bertanggung jawab atas pembuatan dan pengelolaan *shared wallet* yang mencakup aktivitas seperti *create shared wallet*, *invite members*, *cancel invitation*, *manage members' permissions*, *view members' list*, *view members' details*, dan *update shared wallet name*.

2. Aktor *Member* merupakan pengguna yang diundang untuk bergabung ke dalam *shared wallet* dan dapat melakukan tindakan seperti *accept invitation* dan *decline invitation*.

Selain fungsi spesifik di atas, kedua aktor juga memiliki akses umum untuk menampilkan informasi Hi-Komunitas melalui fungsi *view shared wallet list*, *view shared wallet details*, dan *view transaction history*. Hubungan hierarkis antara aktor dan sistem divisualisasikan secara lengkap pada Gambar 3.3.



Gambar 3.3. *Use Case* Diagram Layanan Hi-Komunitas

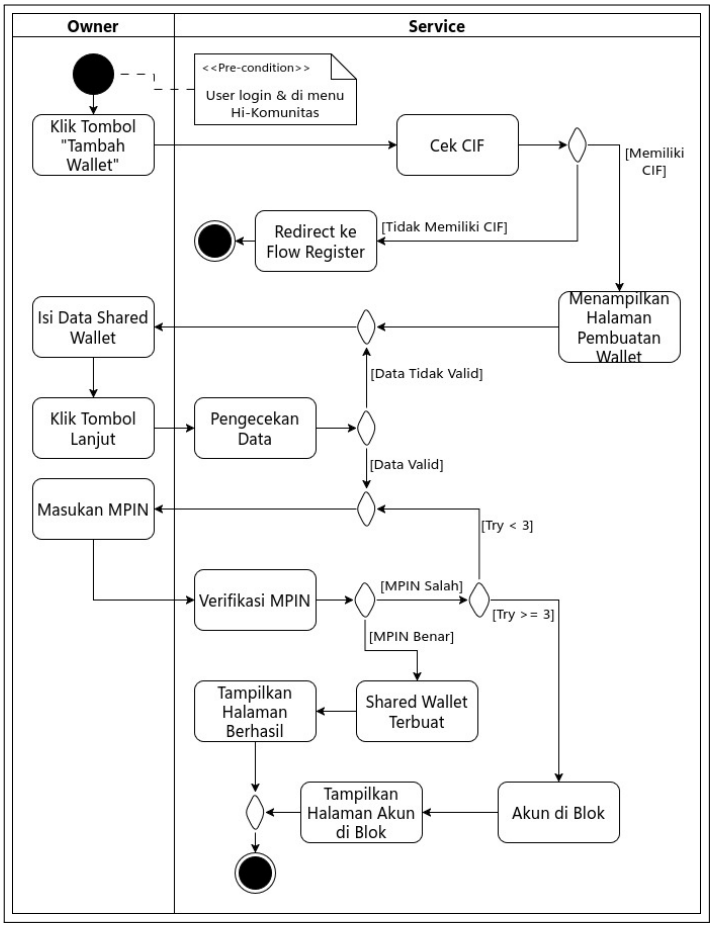
Sumber [11]

C.2 *Activity Diagram*

Perancangan alur logika prosedural dimodelkan menggunakan *Activity Diagram* untuk memvisualisasikan urutan proses bisnis dan respon sistem secara mendetail. Diagram ini memanfaatkan notasi *swimlanes* guna memisahkan batasan tanggung jawab antara inisiasi tindakan pengguna (*User*) dan logika pemrosesan data oleh sistem.

1. *Create Shared Wallet (Inquiry)*
   Tahap inisiasi menerima *input* data pembuatan *shared wallet* dari pengguna dan memvalidasi kepemilikan *Customer Information File* (CIF). Data yang dinyatakan valid akan disimpan sementara ke dalam penyimpanan memori (*cache*), tanpa melakukan penulisan data permanen ke basis data utama, sebagaimana bagian awal alur pada Gambar 3.4.

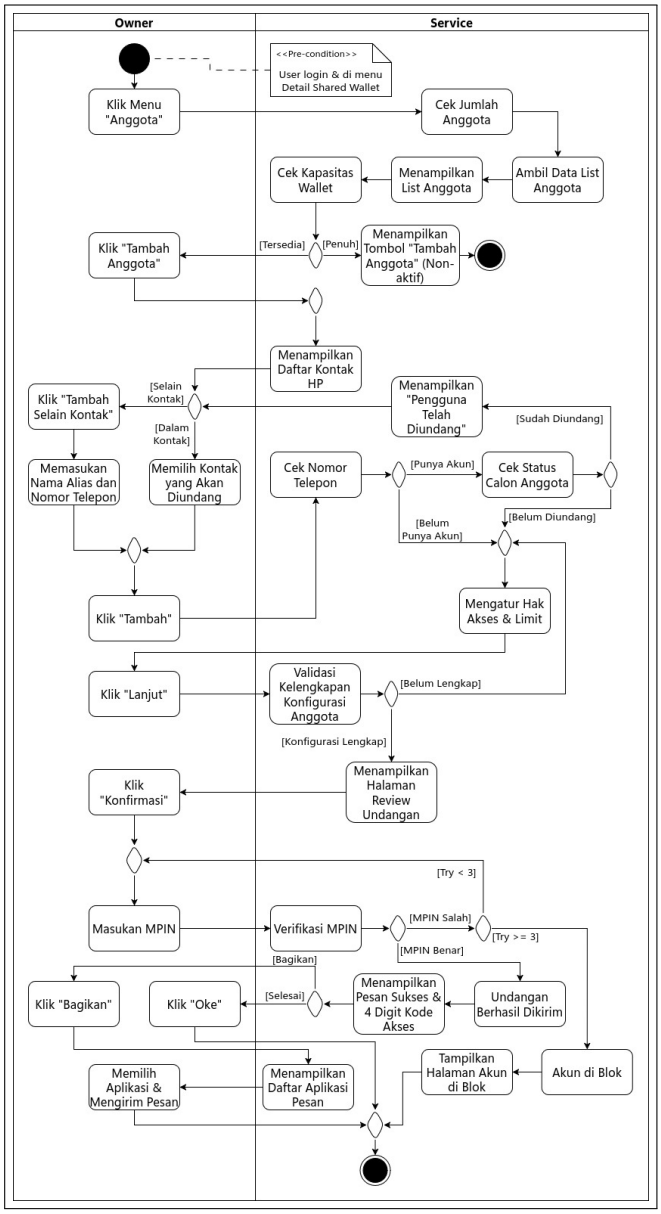
2. *Create Shared Wallet (Posting)*
   Tahap eksekusi dijalankan setelah pengguna berhasil melakukan otentikasi MPIN. Sistem mengambil kembali data parameter yang tersimpan di *cache*, kemudian memprosesnya untuk disimpan secara permanen ke dalam basis data guna membentuk *shared wallet* baru. Keseluruhan alur mekanisme *caching* hingga persistensi data ini divisualisasikan secara utuh pada Gambar 3.4.



Gambar 3.4. *Activity Diagram Create Shared Wallet*
Sumber [11]

3. *Invite Members (Inquiry)*
   Proses dimulai dengan memvalidasi ketersediaan kapasitas *slot shared wallet* dan validitas nomor telepon calon anggota. Pemilik kemudian menentukan konfigurasi hak akses serta *limit* saldo. Data yang telah tervalidasi disimpan sementara ke dalam *cache*, tanpa melakukan perubahan permanen pada basis data, sebagaimana bagian awal alur pada Gambar 3.5.

4. *Invite Members (Posting)*
   Tahap eksekusi dilakukan setelah pemilik berhasil melakukan verifikasi keamanan menggunakan MPIN. Sistem mengambil kembali data konfigurasi dari *cache*, kemudian menyimpannya secara permanen ke basis data dan memicu pengiriman notifikasi undangan ke calon anggota. Gabungan kedua fase proses ini ditampilkan secara utuh pada Gambar 3.5.

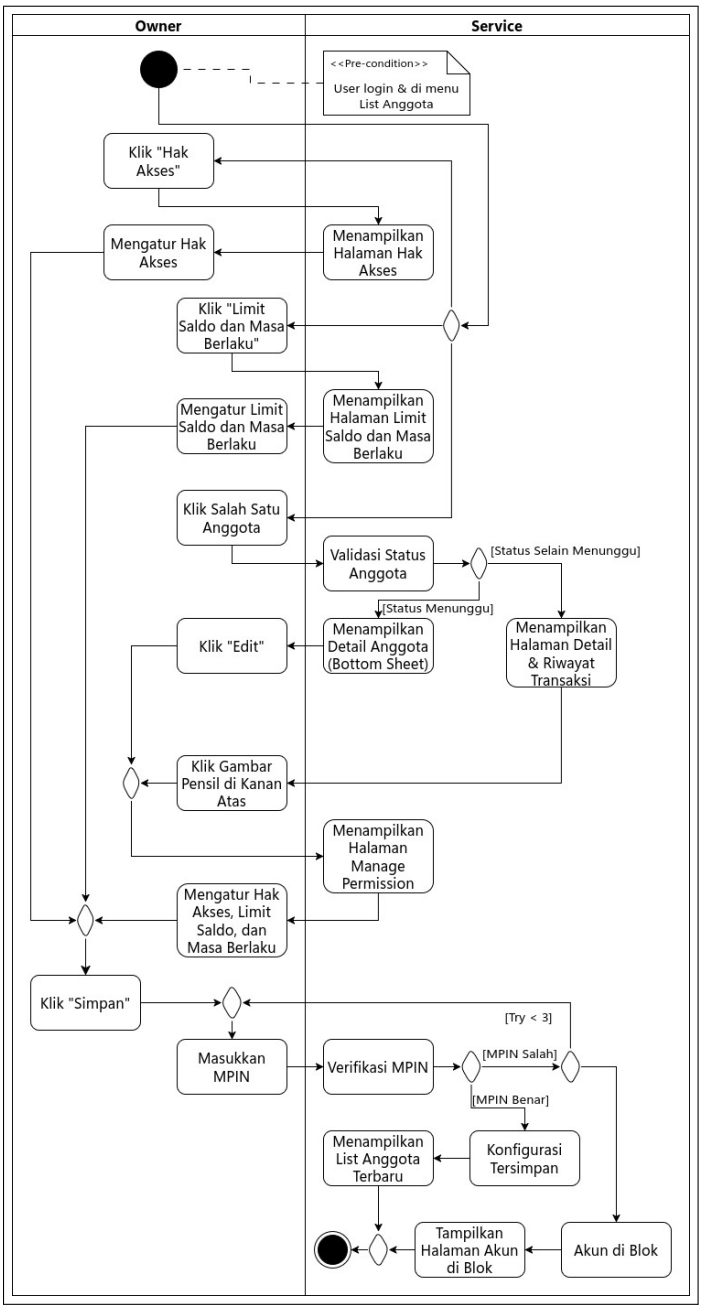


Gambar 3.5. *Activity Diagram Invite Members*
Sumber [11]

5. *Manage Members' Permissions (Inquiry)*
   Pemilik *shared wallet* memulai proses dengan menyesuaikan hak akses atau *limit* saldo anggota. Sistem memvalidasi *input* konfigurasi baru tersebut, dan jika data valid, sistem menyimpannya sementara ke dalam *cache* tanpa langsung mengubah data di basis data utama, sebagaimana terlihat pada alur awal Gambar 3.6.

6. *Manage Members' Permissions (Posting)*

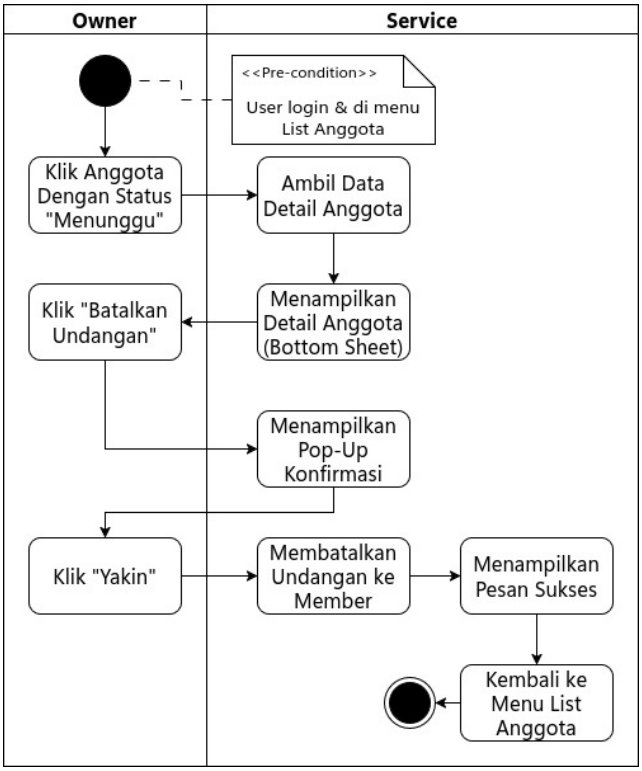
   Tahap eksekusi dilakukan setelah pemilik melakukan otentikasi ulang menggunakan MPIN. Sistem mengambil kembali data konfigurasi dari *cache*, kemudian memperbarui pengaturan anggota secara permanen di basis data dan mengirimkan notifikasi perubahan. Seluruh alur proses dari penyesuaian hingga penyimpanan ini divisualisasikan pada Gambar 3.6.



Gambar 3.6. *Activity Diagram Manage Members' Permissions*
Sumber [11]

7. *Cancel Invitation*

Jika undangan telah terkirim namun belum direspon oleh penerima, pemilik *shared wallet* dapat menarik kembali atau membatalkan undangan tersebut. Alur pembatalan undangan ini akan menghapus data permintaan dari sistem, seperti yang dapat dilihat pada Gambar 3.7.

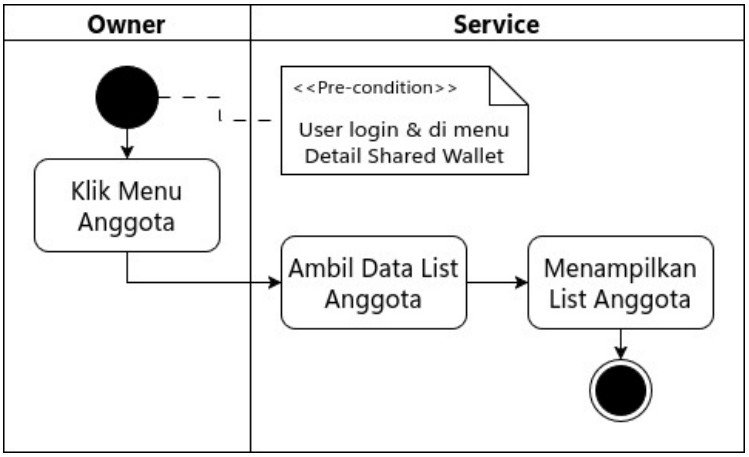


Gambar 3.7. *Activity Diagram Cancel Invitation*

Sumber [11]

8. *View Members' List*

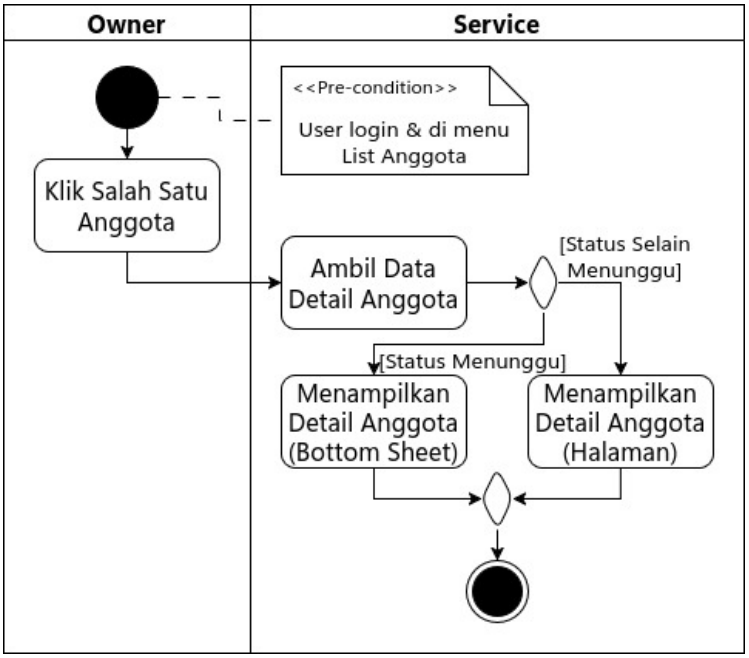
Di dalam menu detail *shared wallet*, pemilik dapat mengakses daftar seluruh anggota yang tergabung dalam grup tersebut. Sistem akan menampilkan nama serta status keanggotaan masing-masing pengguna, sebagaimana terlihat pada Gambar 3.8.



Gambar 3.8. *Activity Diagram View Members' List*
Sumber [11]

9. *View Members' Details*

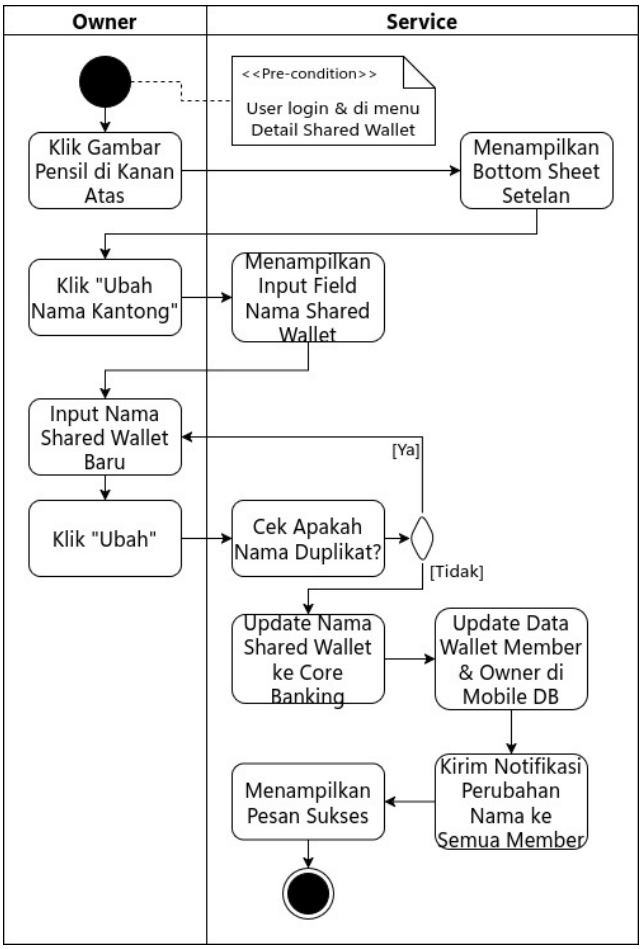
Pemilik dapat mengakses rincian informasi anggota lain. Sistem membedakan tampilan antarmuka berdasarkan status anggota dengan status 'Menunggu' akan ditampilkan dalam format ringkas (*Bottom Sheet*), sedangkan anggota aktif ditampilkan pada halaman penuh beserta riwayat aktivitasnya, seperti pada Gambar 3.9.



Gambar 3.9. *Activity Diagram View Members' Details*
Sumber [11]

10. *Update Shared Wallet Name*

Pemilik memiliki otoritas penuh untuk memperbarui nama *shared wallet* setelah *input* lolos validasi format dan duplikasi. Perubahan ini memicu proses sinkronisasi data ke sistem *Core Banking* serta pembaruan informasi pada akun seluruh anggota yang terhubung demi menjaga konsistensi data, sebagaimana digambarkan pada Gambar 3.10.

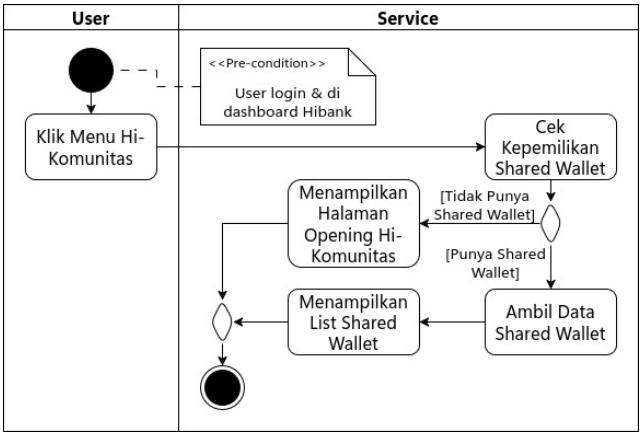


Gambar 3.10. *Activity Diagram Update Shared Wallet Name*

Sumber [11]

11. *View Shared Wallet List*

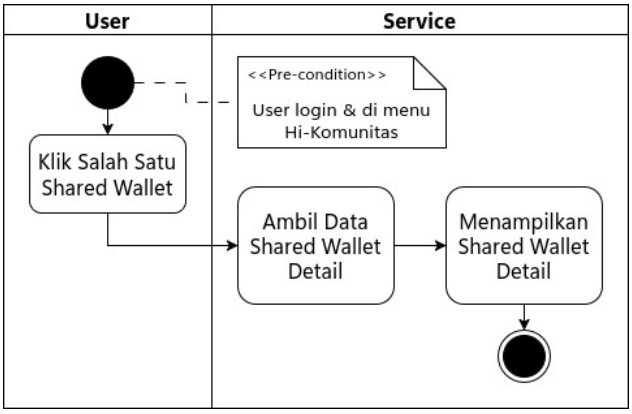
Pengguna dapat melihat daftar seluruh *shared wallet* yang dimilikinya maupun *shared wallet* dimana ia terdaftar sebagai anggota. Sistem akan menampilkan daftar tersebut berdasarkan data kepemilikan dan keanggotaan yang tersimpan, seperti ditunjukkan pada Gambar 3.11.



Gambar 3.11. *Activity Diagram View Shared Wallet List*
Sumber [11]

12. *View Shared Wallet Details*

    Untuk melihat informasi lebih rinci seperti saldo terkini dan ringkasan aktivitas, pengguna dapat memilih salah satu *shared wallet* dari daftar. Alur sistem untuk menampilkan detail informasi *shared wallet* ini dapat dilihat pada Gambar 3.12.

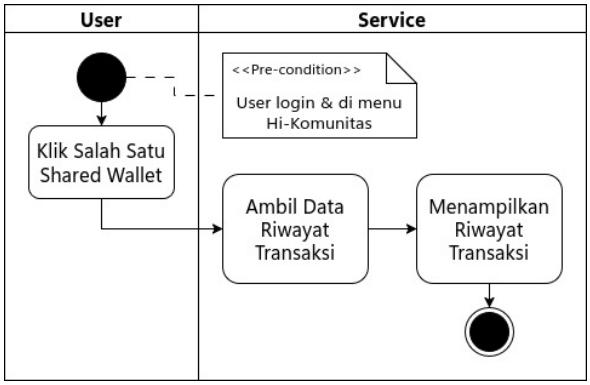


Gambar 3.12. *Activity Diagram View Shared Wallet Details*
Sumber [11]

13. *View Transaction History*

    Fitur ini memfasilitasi pengguna untuk meninjau rekam jejak transaksi keuangan. Sistem menerapkan mekanisme kontrol akses berbasis peran, di mana pengguna (*Member*) dibatasi hanya dapat mengakses riwayat transaksi pribadi, sedangkan pemilik *shared wallet* (*Owner*) diberikan privilese penuh untuk memantau seluruh aktivitas transaksi yang dilakukan oleh seluruh

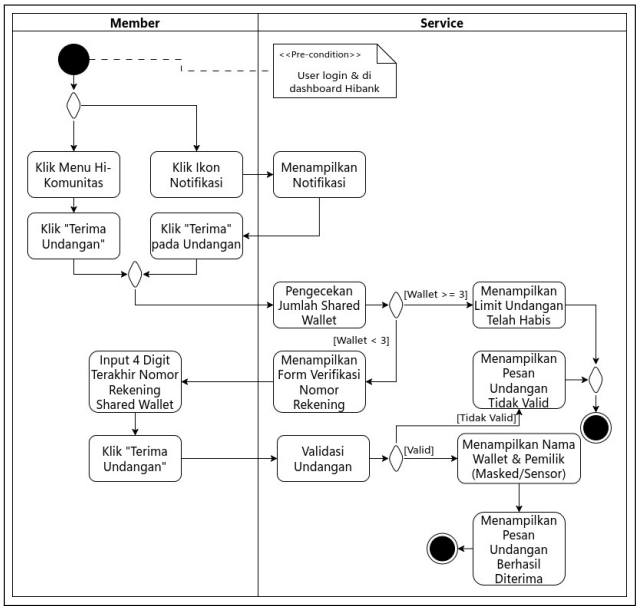
anggota dalam *shared wallet* tersebut.



Gambar 3.13. *Activity Diagram View Transaction History*
Sumber [11]

14. *Accept Invitation*

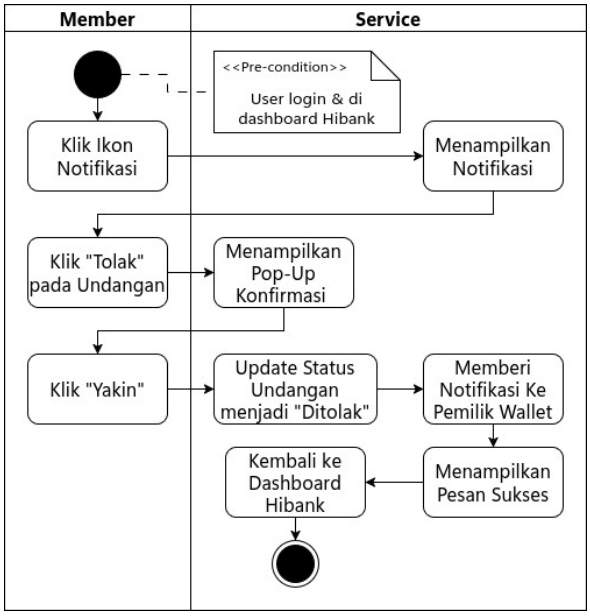
    Proses penerimaan undangan mensyaratkan verifikasi keamanan yang ketat. Sistem akan memvalidasi jumlah *shared wallet* yang dimiliki pengguna (maksimal 3) serta meminta *input* 4 digit terakhir nomor rekening *shared wallet* sebagai verifikasi. Apabila seluruh validasi tersebut berhasil, sistem akan langsung memproses peresmian keanggotaan, seperti terlihat pada Gambar 3.14.



Gambar 3.14. *Activity Diagram Accept Invitation*
Sumber [11]

15. *Decline Invitation*

    Apabila pengguna memutuskan untuk tidak bergabung, sistem menyediakan opsi untuk menolak undangan tersebut. Proses ini akan memperbarui status undangan menjadi *rejected* di dalam basis data dan memberikan umpan balik status kepada pengguna, sebagaimana digambarkan pada Gambar 3.15.



Gambar 3.15. *Activity Diagram Decline Invitation*
Sumber [11]

C.3 *Sequence Diagram*

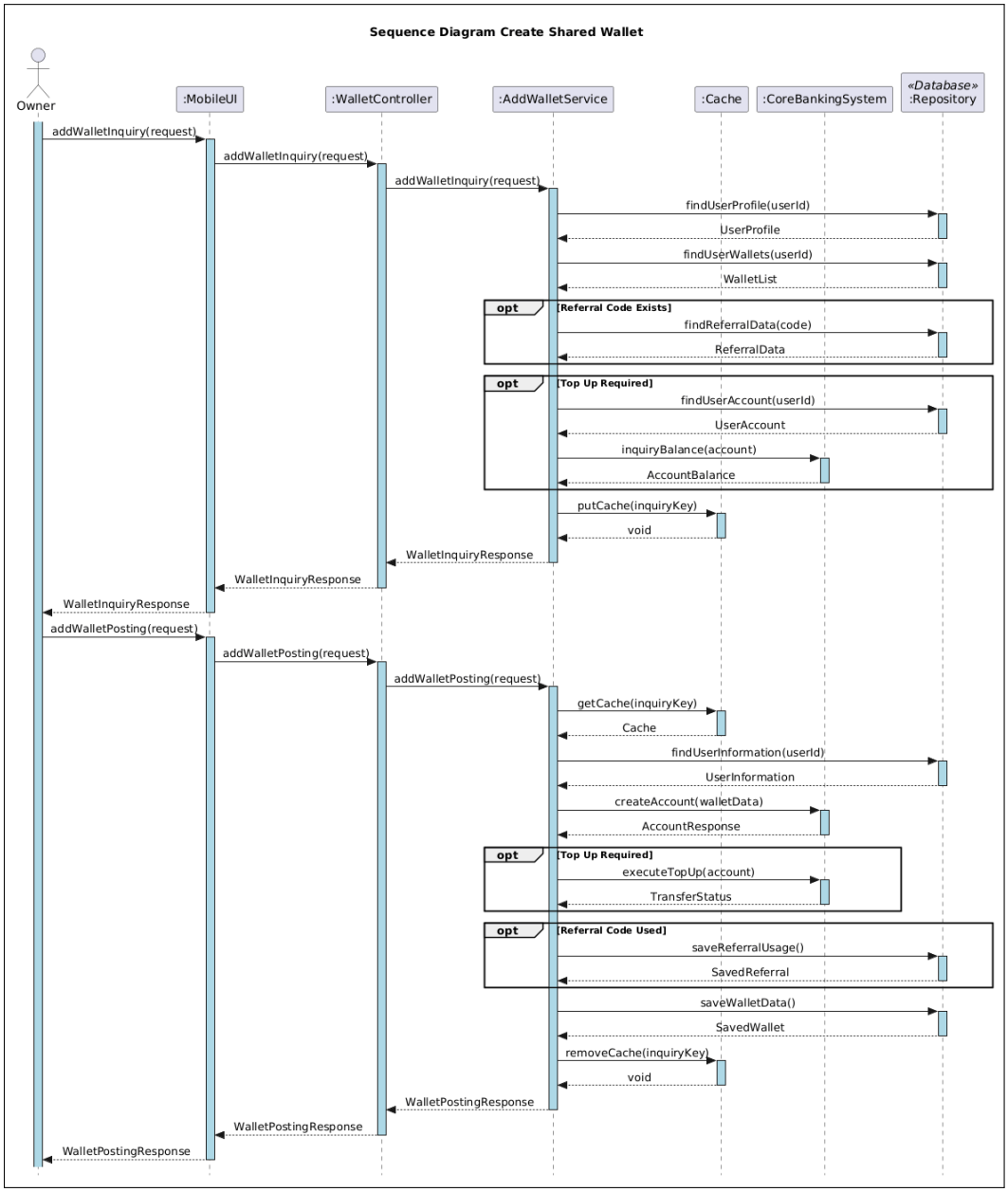
Detail interaksi antar objek dalam sistem dimodelkan menggunakan *Sequence Diagram* untuk menggambarkan skenario teknis berdasarkan urutan waktu (*time sequence*). Diagram ini memvisualisasikan aliran pertukaran pesan (*message passing*) antara aktor dan komponen sistem, mulai dari inisiasi *request* hingga terbentuknya *response*, guna memvalidasi logika eksekusi pada setiap fitur. Berikut adalah rincian diagram urutan untuk layanan Hi-Komunitas.

1. *Create Shared Wallet (Inquiry)*

   Proses inisiasi dimulai dengan memvalidasi kepemilikan *Customer Information File* (CIF) dan kelayakan data *input* pengguna. Apabila data valid, sistem akan menyimpannya sementara ke dalam memori (*cache*) dan mengembalikan respons sukses untuk melanjutkan ke tahap otorisasi, sebagaimana bagian awal interaksi pada Gambar 3.16.

2. *Create Shared Wallet (Posting)*

   Setelah pengguna berhasil melakukan otorisasi MPIN, sistem mengambil kembali data yang tersimpan di *cache*. Data tersebut kemudian digunakan untuk mengajukan pembukaan rekening ke sistem perbankan inti dan menyimpan struktur Hi-Komunitas secara permanen ke basis data. Seluruh rangkaian interaksi objek dari validasi hingga penyimpanan ini digambarkan pada Gambar 3.16.



Gambar 3.16. *Sequence Diagram Create Shared Wallet*

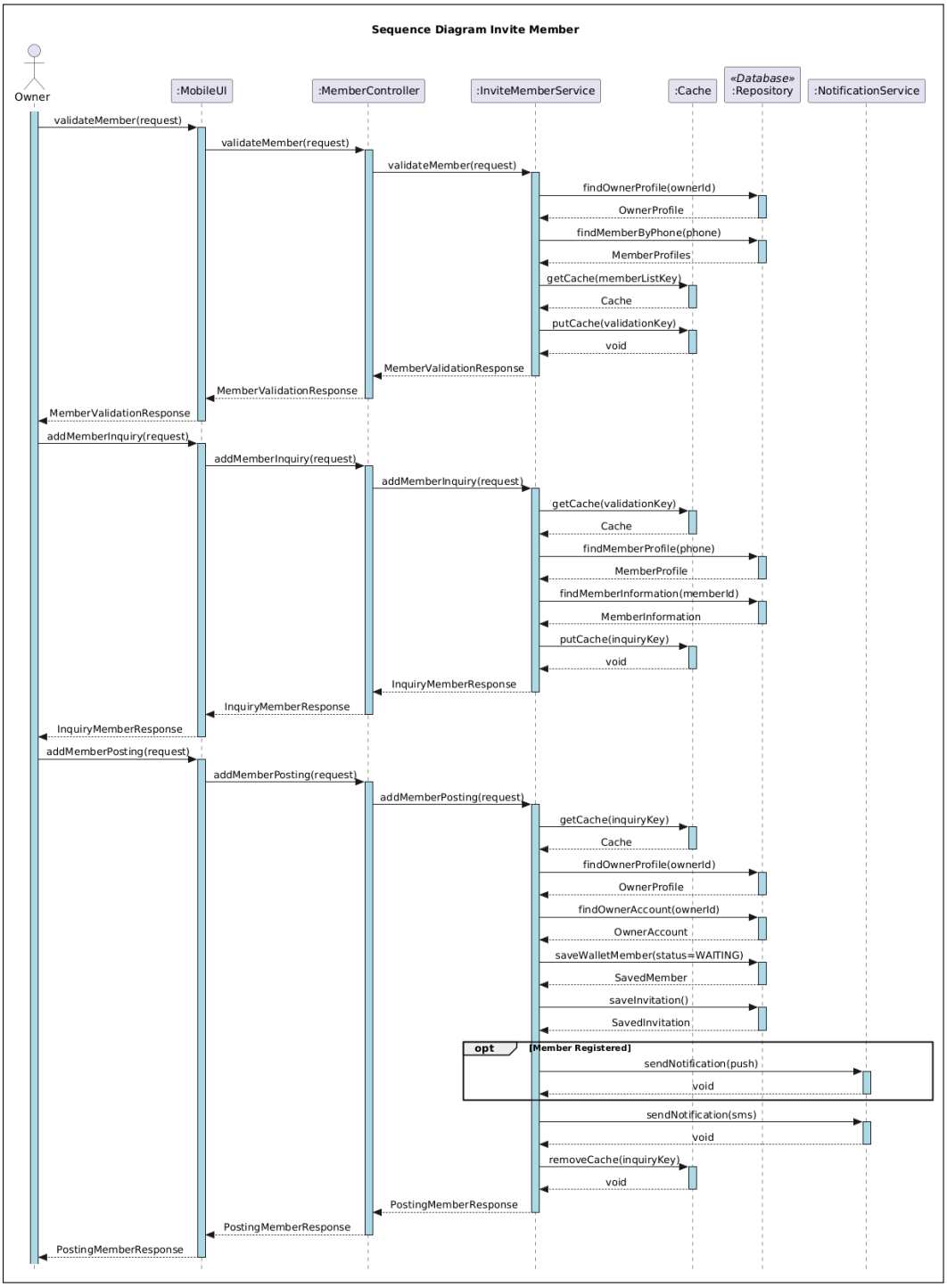
Sumber [11]

3. *Invite Members (Inquiry)*

   Alur penambahan anggota diawali dengan validasi kepemilikan *shared wallet* dan pengecekan batas maksimal kapasitas anggota. Sistem memverifikasi validitas nomor telepon calon anggota, dan jika data valid, sistem menyimpannya ke dalam memori sementara (*cache*) sebagai persiapan untuk tahap eksekusi, sebagaimana digambarkan pada bagian awal interaksi Gambar 3.17.

4. *Invite Members (Posting)*

   Tahap eksekusi dijalankan setelah pengguna melakukan konfirmasi. Sistem mengambil kembali parameter data dari *cache*, kemudian membentuk data keanggotaan baru dengan status ”menunggu persetujuan” secara permanen di basis data serta mengirimkan notifikasi undangan ke calon anggota terkait. Seluruh rangkaian interaksi objek ini ditunjukkan secara utuh pada Gambar 3.17.

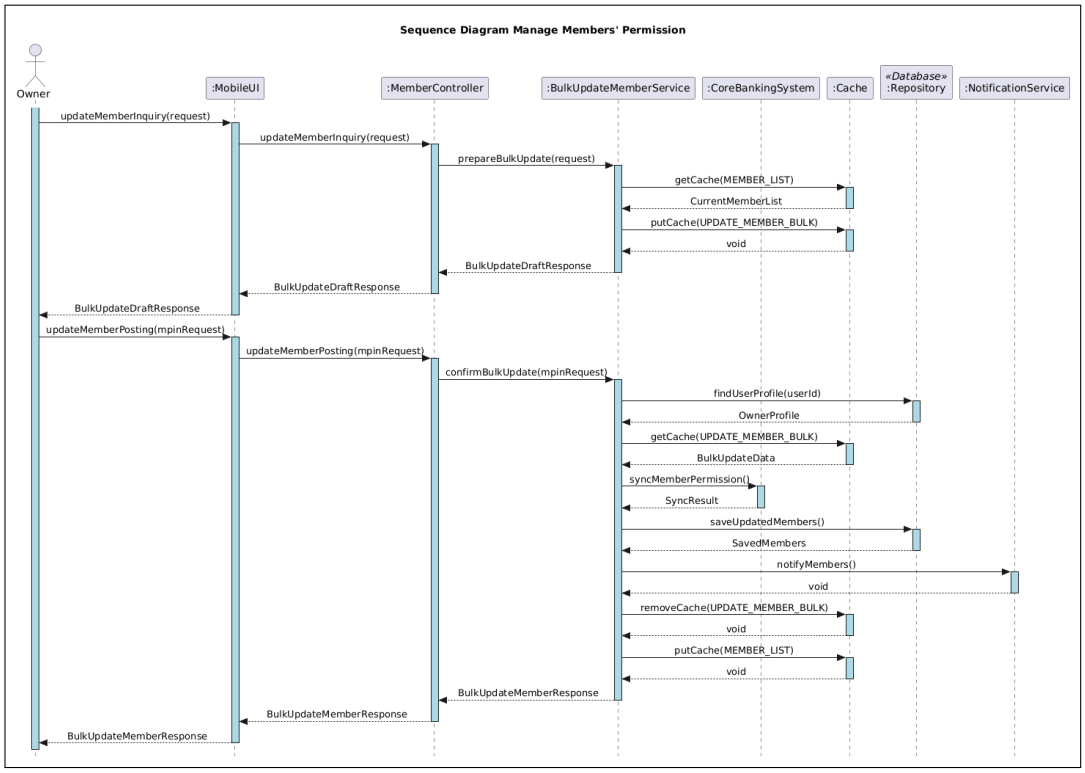


Gambar 3.17. *Sequence Diagram Invite Members*

Sumber [11]

5. *Manage Members' Permissions (Inquiry)*
   Proses pengelolaan dimulai saat pemilik melakukan perubahan hak akses atau *limit* transaksi anggota. Sistem memvalidasi konfigurasi baru tersebut dan menyimpannya sebagai draf sementara di memori (*cache*) untuk menunggu konfirmasi, sebagaimana digambarkan pada bagian awal interaksi Gambar 3.18.

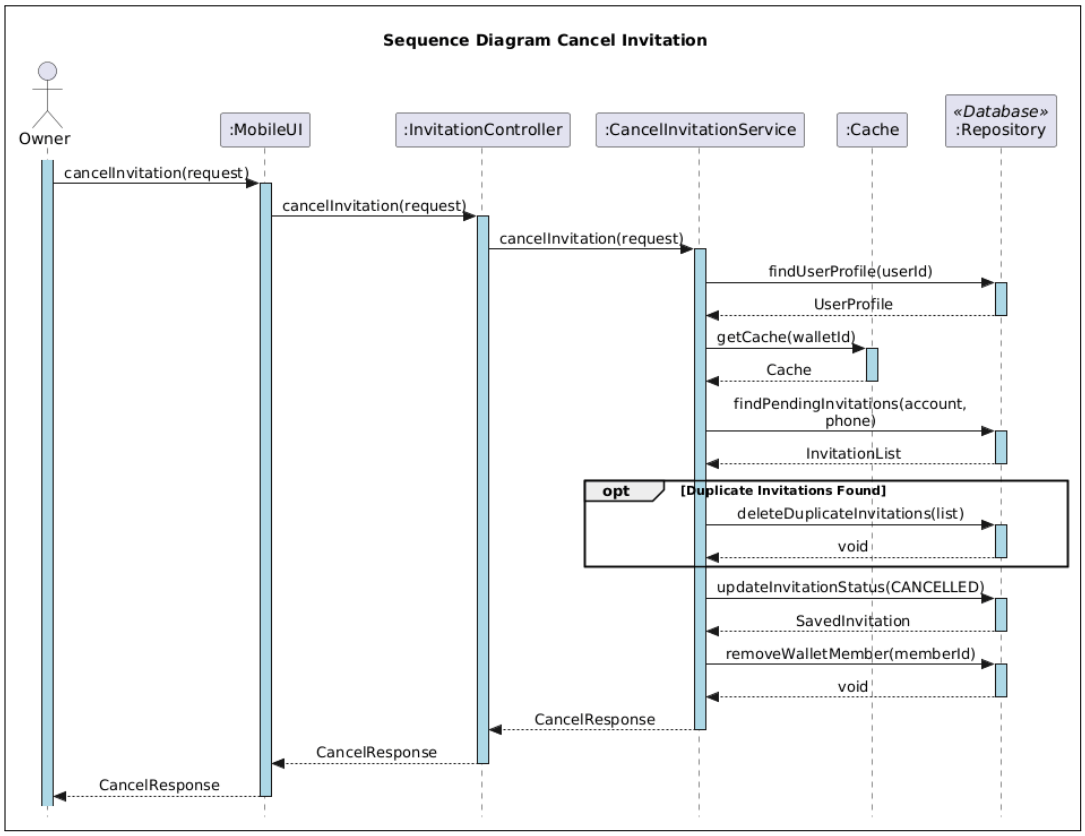
6. *Manage Members' Permissions (Posting)*
   Tahap eksekusi dipicu setelah pemilik memberikan konfirmasi final. Sistem mengambil kembali konfigurasi draf dari *cache*, memperbarui pengaturan pada entitas anggota secara permanen di basis data, melakukan sinkronisasi ke sistem perbankan jika diperlukan, serta mengirimkan notifikasi kepada anggota terdampak. Seluruh rangkaian interaksi ini ditunjukkan secara utuh pada Gambar 3.18.



Gambar 3.18. *Sequence Diagram Manage Members' Permissions*
Sumber [11]

7. *Cancel Invitation*

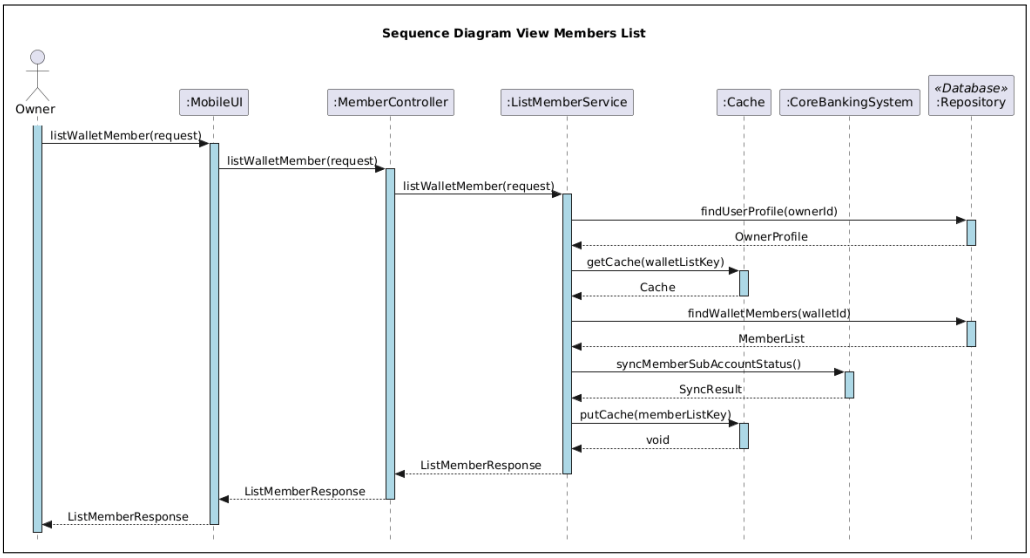
   Pembatalan undangan yang belum direspon dilakukan dengan memastikan validitas permintaan dari pemilik *shared wallet* yang sah. Sistem melacak status undangan aktif, kemudian menonaktifkannya beserta data keanggotaan terkait untuk mencegah penggunaan ulang tautan atau notifikasi undangan tersebut, seperti digambarkan pada Gambar 3.19.



Gambar 3.19. *Sequence Diagram Cancel Invitation*
Sumber [11]

8. *View Members' List*

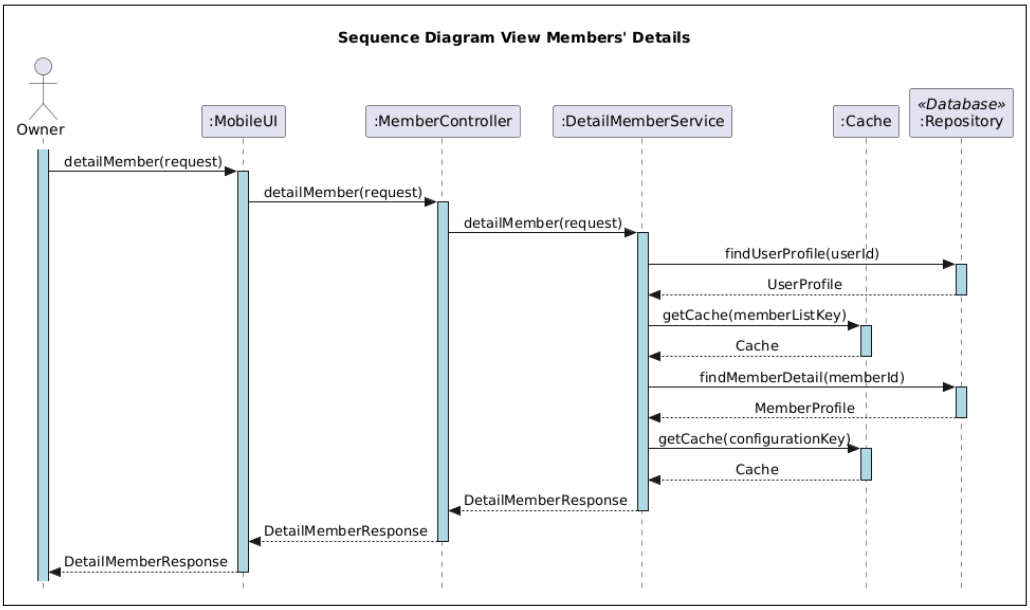
   Penampilan daftar anggota melibatkan pengambilan seluruh data entitas yang terasosiasi dengan *shared wallet*. Sistem melakukan validasi integritas untuk menyaring data duplikat atau profil non-aktif, kemudian menyinkronkan status terkini sebelum menyajikan daftar anggota yang valid kepada pengguna, sebagaimana ditunjukkan pada Gambar 3.20.



Gambar 3.20. *Sequence Diagram View Members' List*
Sumber [11]

9. *View Members' Details*

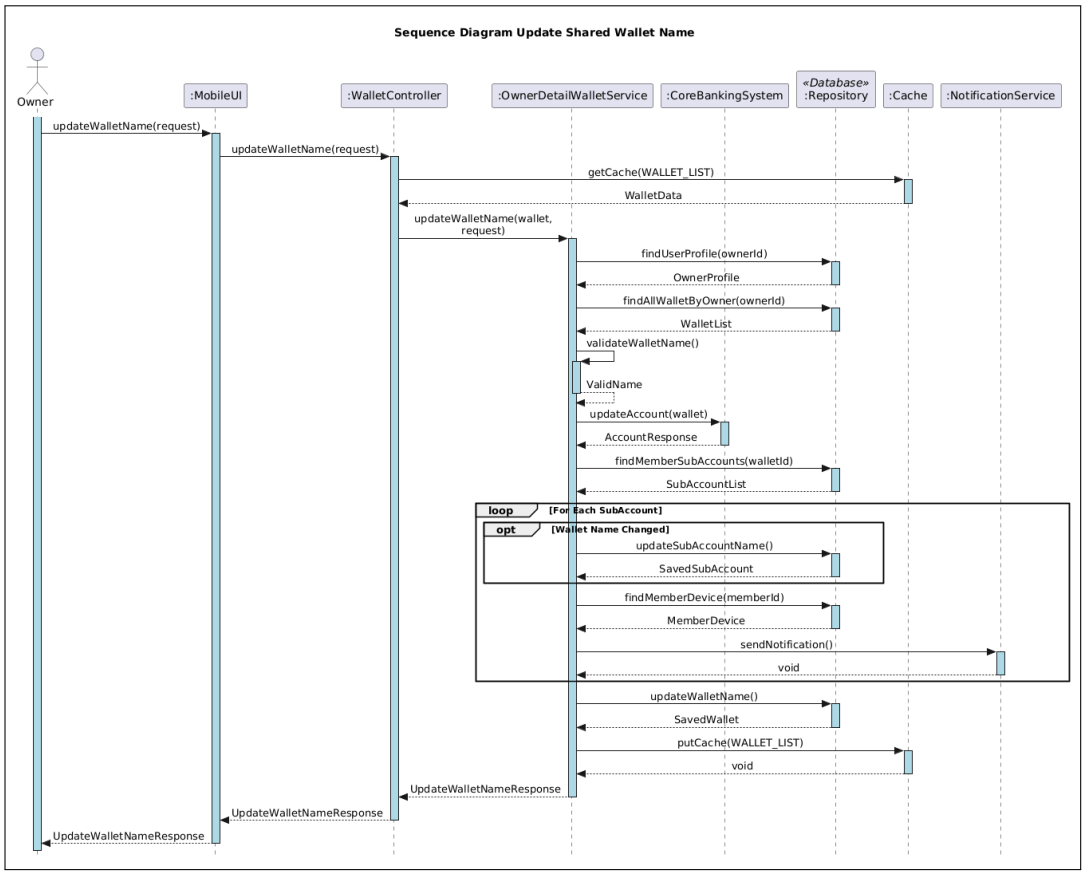
   Saat pemilik meninjau konfigurasi anggota tertentu, sistem mengambil detail data anggota beserta parameter batas transaksi yang berlaku. Sebelum informasi ditampilkan, sistem menerapkan mekanisme perlindungan data dengan menyamarkan informasi sensitif guna menjaga privasi pengguna, seperti terlihat pada Gambar 3.21.



Gambar 3.21. *Sequence Diagram View Members' Details*
Sumber [11]

10. *Update Shared Wallet Name*

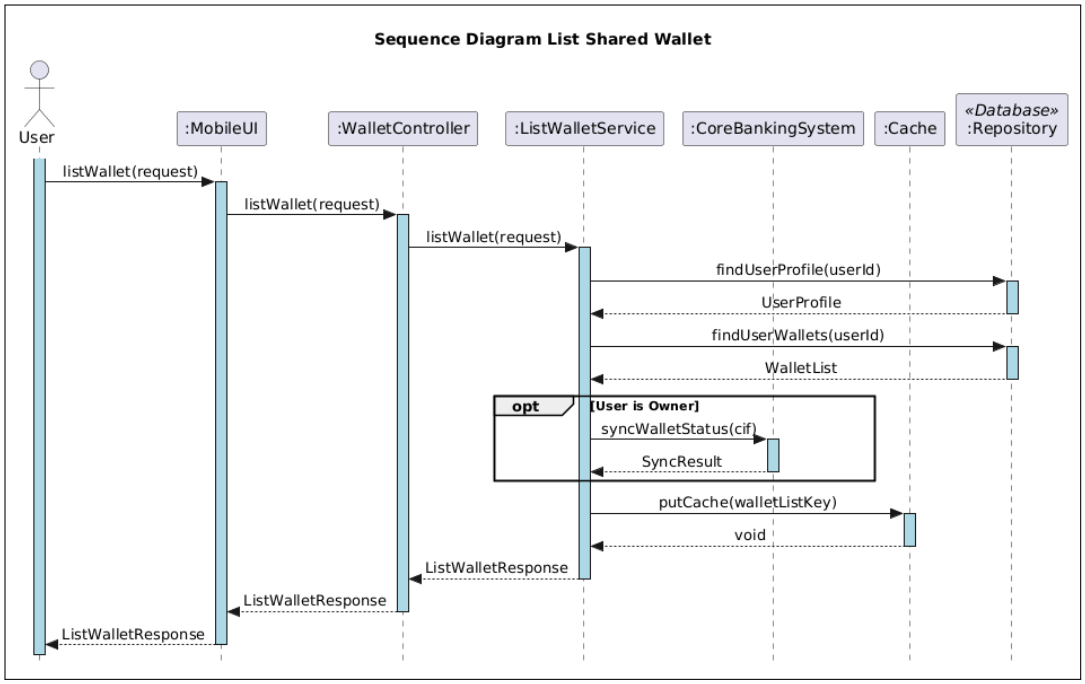
Proses pengubahan nama *shared wallet* memvalidasi *input* nama baru sesuai ketentuan sistem. Perubahan yang lolos validasi akan disinkronkan ke sistem perbankan inti dan diperbarui pada seluruh akun turunan anggota untuk menjamin konsistensi informasi di seluruh perangkat pengguna, sebagaimana digambarkan pada Gambar 3.22.



Gambar 3.22. *Sequence Diagram Update Shared Wallet Name*
Sumber [11]

11. *View Shared Wallet List*

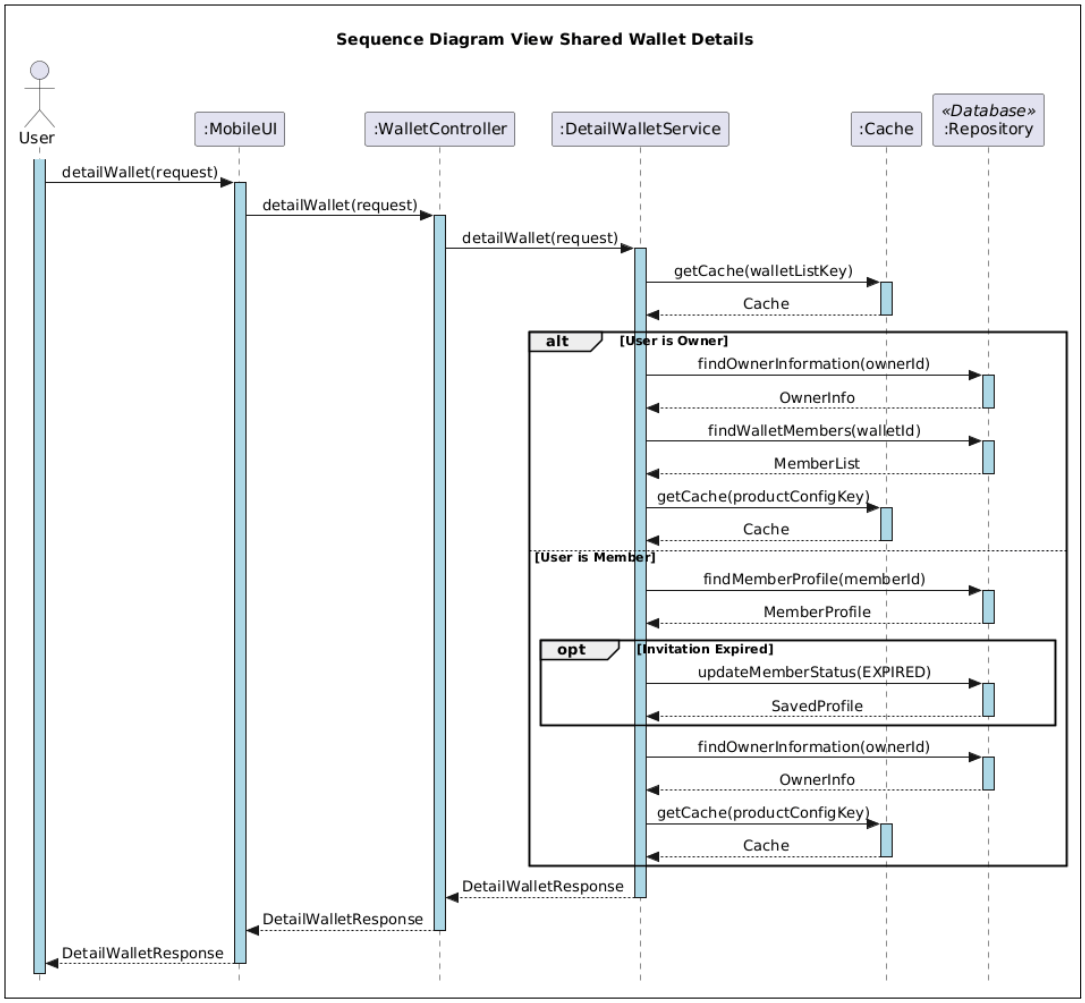
Sistem menampilkan daftar *shared wallet* berdasarkan relasi kepemilikan dan keanggotaan pengguna. Bagi pengguna dengan peran pemilik, sistem menyinkronkan status rekening dan menampilkan saldo aktual, sedangkan bagi anggota, sistem menampilkan sisa batas transaksi yang tersedia sesuai hak akses masing-masing, seperti ditunjukkan pada Gambar 3.23.



Gambar 3.23. *Sequence Diagram View Shared Wallet List*
Sumber [11]

12. *View Shared Wallet Details*

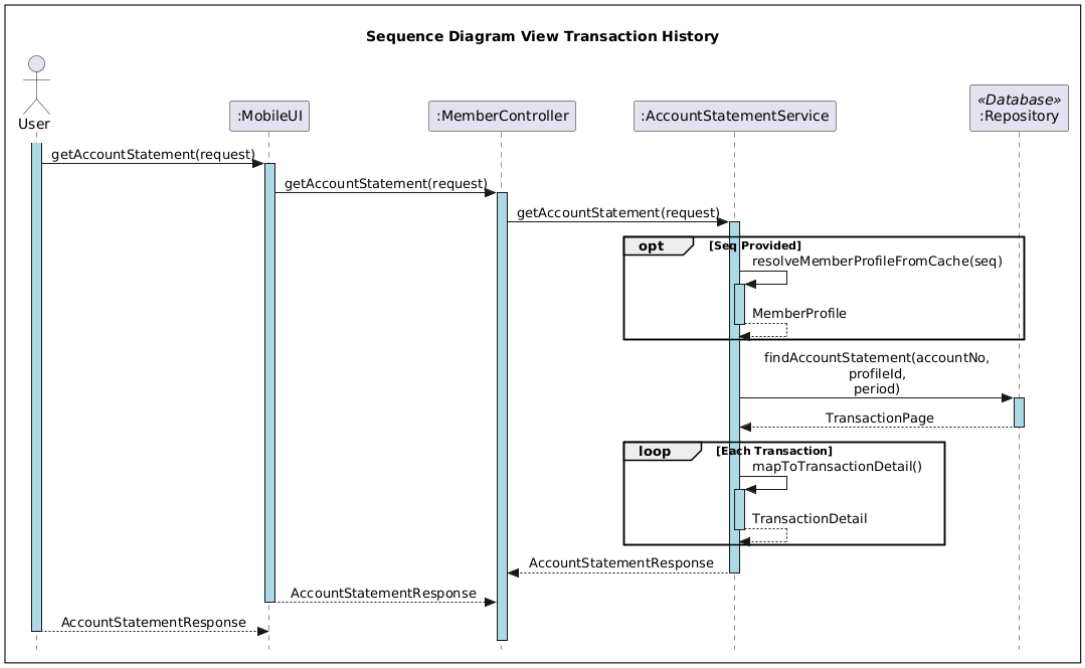
    Penampilan detail *shared wallet* menerapkan logika percabangan berdasarkan peran pengguna. Pemilik diberikan akses informasi penuh mencakup saldo total dan daftar anggota, sementara anggota hanya menerima informasi terbatas yang relevan dengan hak aksesnya setelah melalui validasi status keanggotaan, sebagaimana terlihat pada Gambar 3.24.



Gambar 3.24. *Sequence Diagram View Shared Wallet Details*
Sumber [11]

13. *View Transaction History*

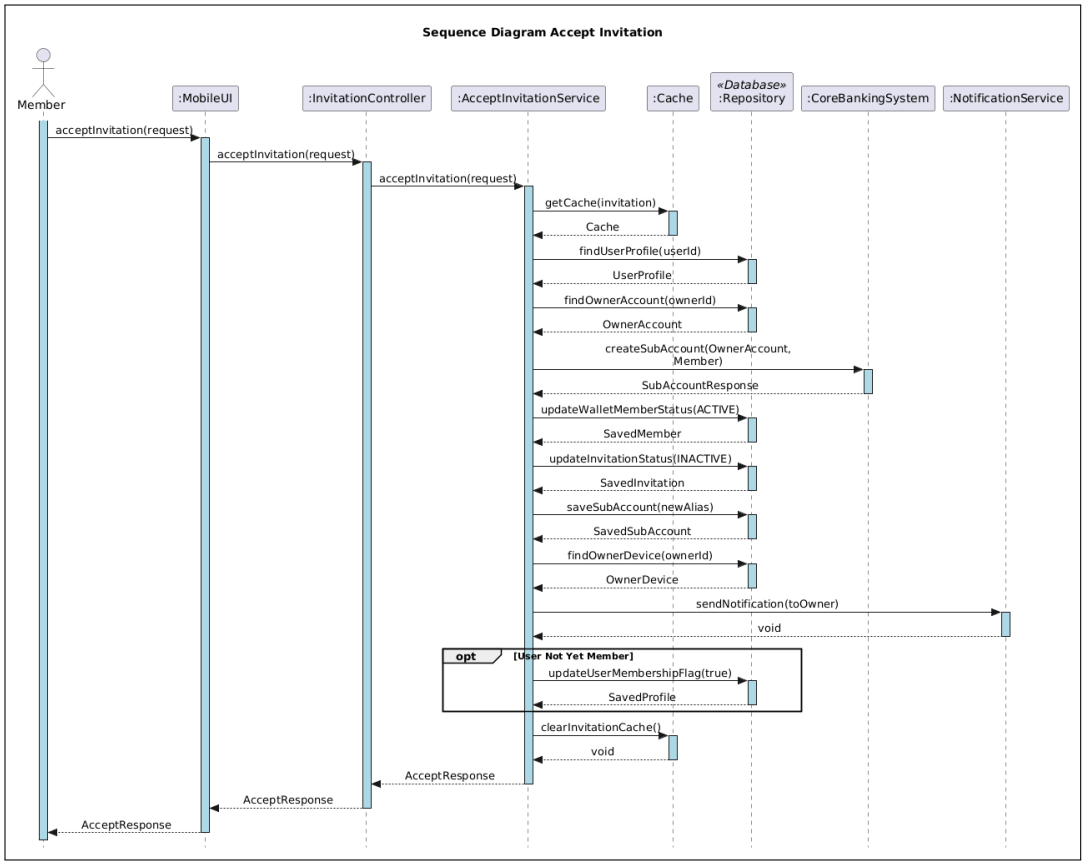
    Sistem menyusun riwayat transaksi dengan mengidentifikasi akun yang relevan berdasarkan konteks akses pengguna. Data transaksi diambil sesuai periode yang diminta, kemudian diproses dan diformat agar dapat disajikan secara terstruktur pada antarmuka aplikasi, seperti digambarkan pada Gambar 3.25.



Gambar 3.25. *Sequence Diagram View Transaction History*
Sumber [11]

14. *Accept Invitation*

    Penerimaan undangan divalidasi dengan memeriksa batas kepemilikan *shared wallet* dan verifikasi parameter keamanan terkait rekening. Apabila seluruh syarat terpenuhi, sistem membentuk akun turunan yang terhubung ke rekening utama, memperbarui status keanggotaan menjadi aktif, dan menotifikasi pemilik *shared wallet* mengenai keberhasilan proses tersebut, sebagaimana ditunjukkan pada Gambar 3.26.

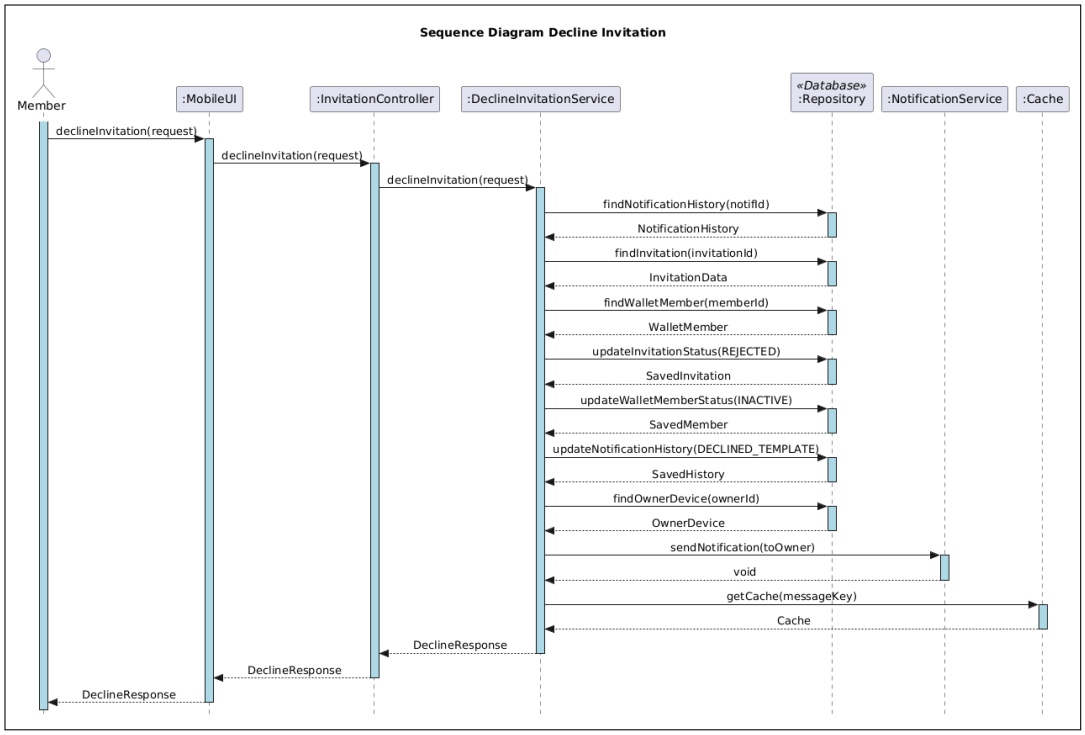


Gambar 3.26. *Sequence Diagram Accept Invitation*

Sumber [11]

15. *Decline Invitation*

    Penolakan undangan diproses dengan memverifikasi data undangan masuk. Sistem mengubah status undangan menjadi ditolak, menonaktifkan data keanggotaan terkait untuk membatalkan akses, serta mengirimkan pemberitahuan status penolakan kepada pemilik *shared wallet*, seperti terlihat pada Gambar 3.27.



Gambar 3.27. *Sequence Diagram Decline Invitation*
Sumber [11]

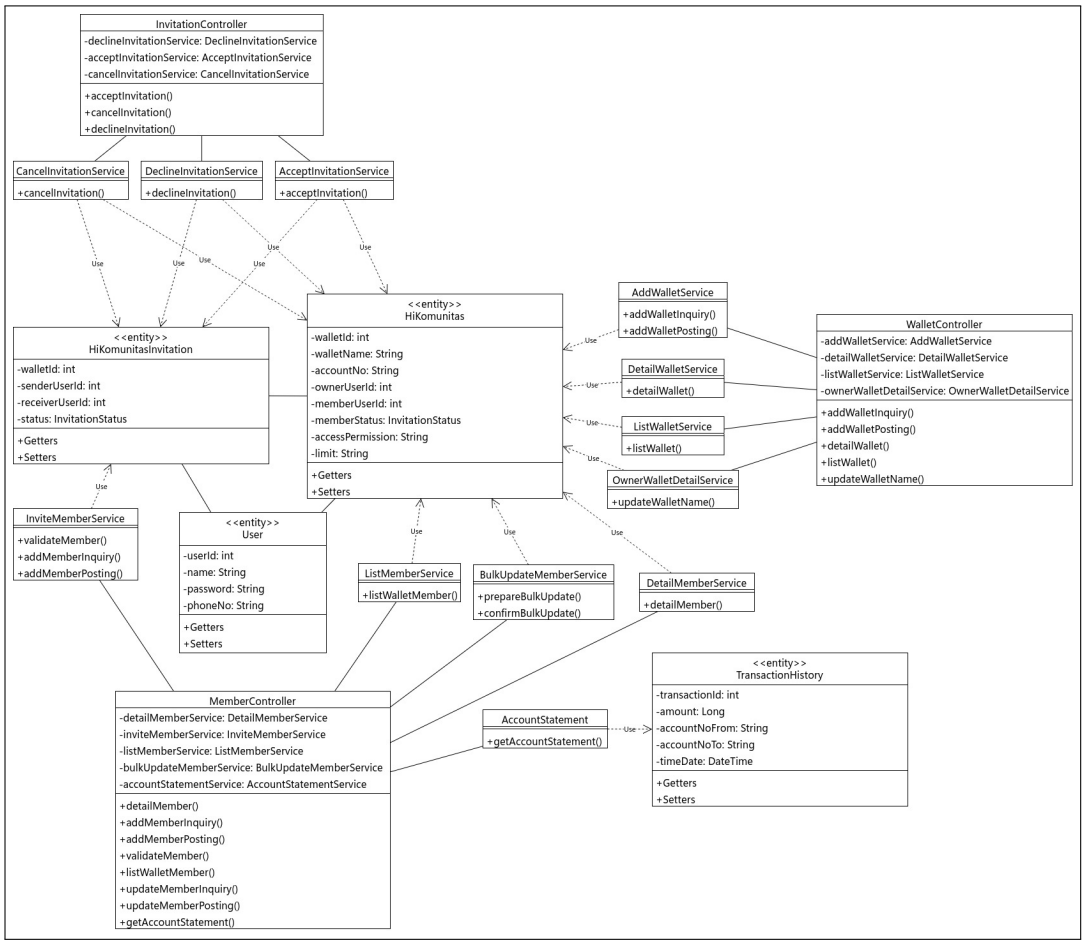
C.4 *Class Diagram*

Pemodelan struktur statis sistem divisualisasikan melalui *Class Diagram* guna menggambarkan arsitektur *backend* yang mengadopsi pola desain berlapis (*layered architecture*). Diagram ini menerapkan prinsip *Separation of Concerns* dengan mendistribusikan tanggung jawab ke dalam tiga komponen utama, yaitu *Controller* sebagai penangan permintaan, *Service* sebagai eksekutor logika bisnis, dan *Entity* sebagai representasi objek data persisten. Berdasarkan diagram tersebut, arsitektur sistem terbagi menjadi tiga domain fungsional utama yang dikelola oleh *Controller* spesifik.

1. *Invitation Management* dikelola oleh *InvitationController* yang memfasilitasi respons pengguna terhadap undangan. Kontroler ini menangani logika bisnis ke layanan terpisah untuk penerimaan (*Accept*), pembatalan (*Cancel*), dan penolakan (*Decline*) undangan, yang kemudian memperbarui status pada entitas *HiKomunitasInvitation* dan *HiKomunitas*.

2. *Wallet Management* dikelola oleh *WalletController* yang bertugas menangani siklus hidup *shared wallet*. Fitur penciptaan *shared wallet* (*AddWallet*), pemantauan daftar (*ListWallet*), serta akses detail informasi ditangani oleh layanan spesifik yang berinteraksi langsung dengan entitas utama *HiKomunitas*.

3. *Member & Transaction Management* dikelola oleh *MemberController* yang memiliki kompleksitas fungsional tertinggi. Komponen ini menangani manajemen keanggotaan mulai dari proses undangan (*InviteMember*), pengaturan hak akses massal (*BulkUpdate*), hingga penyajian data historis. Kontroler ini berinteraksi dengan entitas *HiKomunitas* untuk data profil dan terhubung dengan entitas *TransactionHistory* melalui komponen *AccountStatement* untuk penyajian riwayat transaksi keuangan.



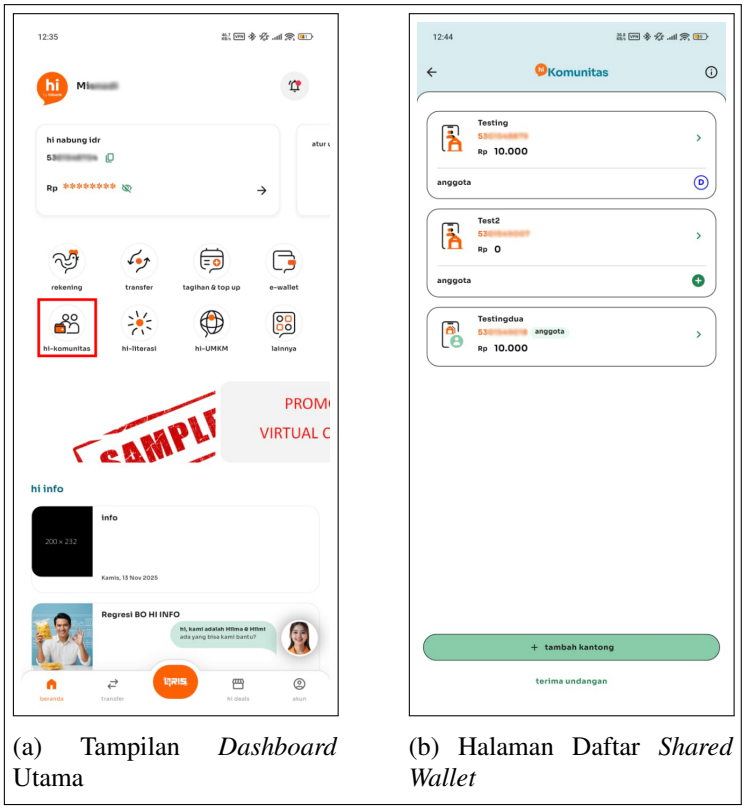
Gambar 3.28. *Class Diagram* Layanan Hi-Komunitas
Sumber [11]

D *Implementation*

Bagian ini mendokumentasikan hasil implementasi layanan *backend* dari seluruh fungsionalitas Hi-Komunitas. Antarmuka aplikasi ditampilkan sebagai representasi visual dari eksekusi logika bisnis yang telah disusun berdasarkan alur prosedur utama.

D.1 *View Shared Wallet List*

Implementasi navigasi utama dan halaman daftar *shared wallet* secara keseluruhan disajikan pada Gambar 3.29. Proses dimulai dari *dashboard* aplikasi Hibank yang telah terotentikasi seperti pada Gambar 3.29a, di mana pengguna dapat memilih menu Hi-Komunitas. Sistem kemudian mengarahkan pengguna ke halaman daftar *Shared Wallet List* sebagaimana diperlihatkan pada Gambar 3.29b. Halaman ini menampilkan seluruh *shared wallet* yang terasosiasi dengan akun pengguna, serta menyediakan tombol ”+ Tambah *shared wallet*” dan ”Terima Undangan”.



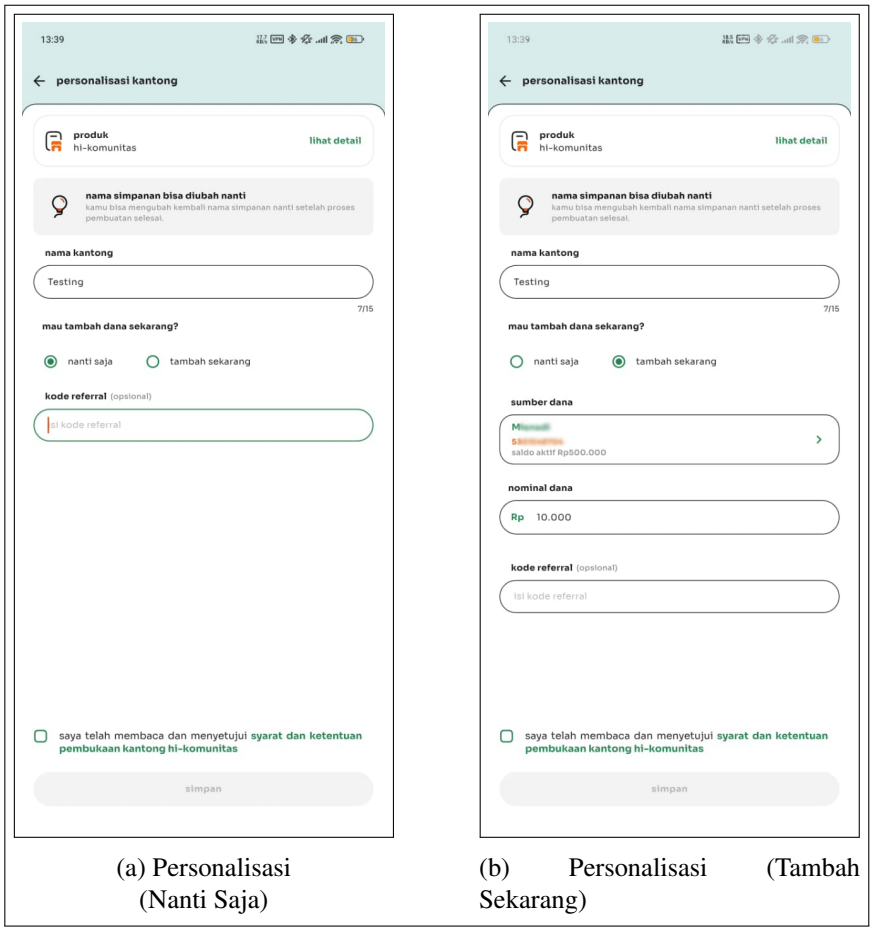
(a) Tampilan *Dashboard* Utama

(b) Halaman Daftar *Shared Wallet*

Gambar 3.29. Implementasi Navigasi Utama dan Daftar *Shared Wallet*
Sumber Aplikasi *System Integrated Testing* Hibank

D.2 *Create Shared Wallet*

Proses pembuatan *shared wallet* dimulai dari tombol tambah pada halaman daftar *shared wallet*. Halaman personalisasi yang muncul menyediakan dua skenario pengisian data sebagaimana dirangkum pada Gambar 3.30. Skenario pertama adalah opsi ”nanti saja” seperti pada Gambar 3.30a yang hanya meminta nama *shared wallet*, sedangkan skenario kedua adalah ”tambah sekarang” sebagaimana Gambar 3.30b yang memunculkan *field* tambahan untuk setoran awal.

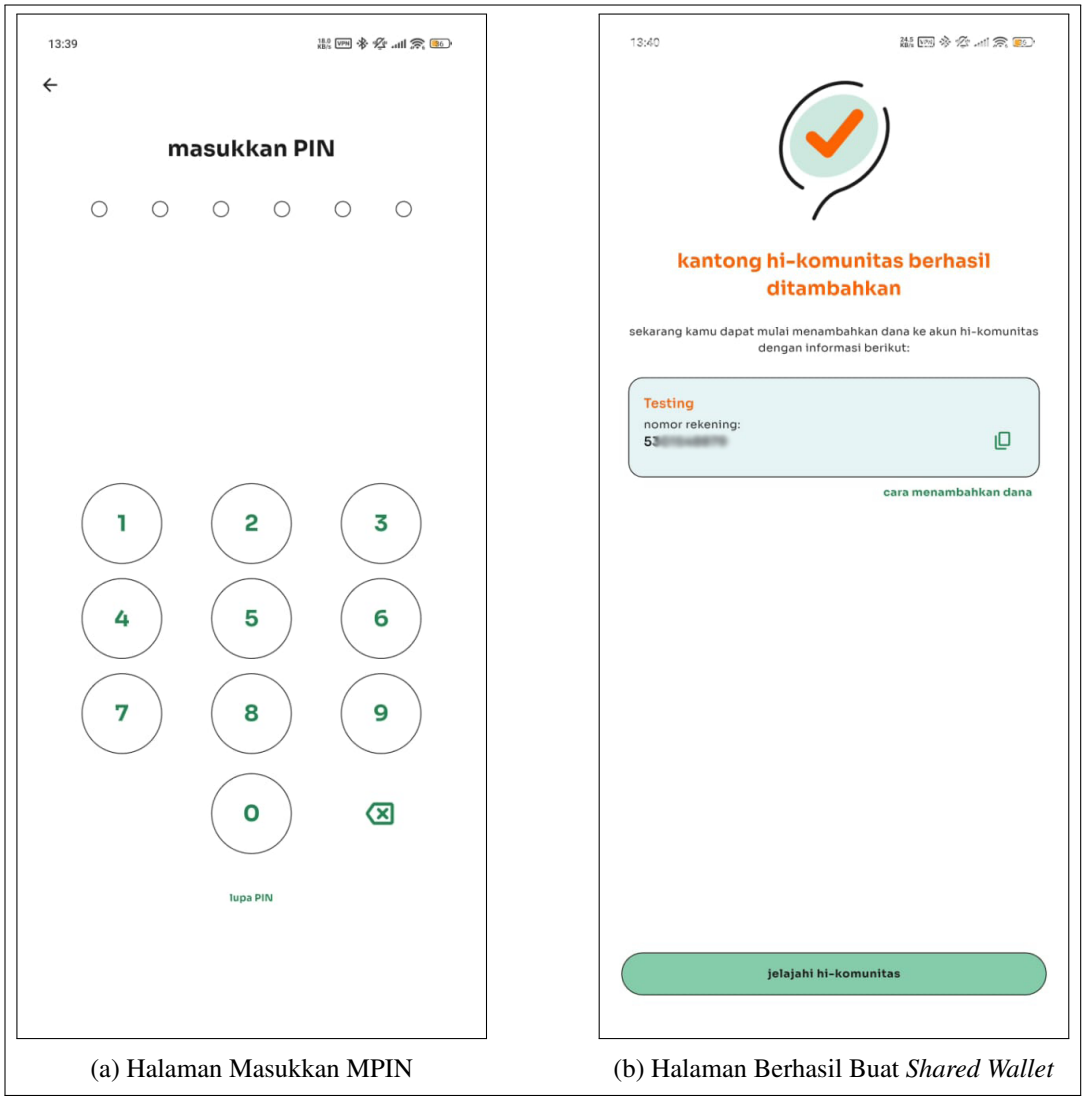


(a) Personalisasi (Nanti Saja)

(b) Personalisasi (Tambah Sekarang)

Gambar 3.30. Implementasi Halaman Personalisasi *Shared Wallet*
Sumber Aplikasi *System Integrated Testing* Hibank

Tahap finalisasi pembuatan *shared wallet*, mulai dari otorisasi hingga konfirmasi sukses, divisualisasikan pada Gambar 3.31. Setelah pengguna menyimpan data, sistem meminta otorisasi keamanan melalui *input* MPIN seperti terlihat pada Gambar 3.31a. Jika validasi berhasil, sistem menampilkan halaman sukses yang memuat nomor rekening baru sebagaimana Gambar 3.31b.



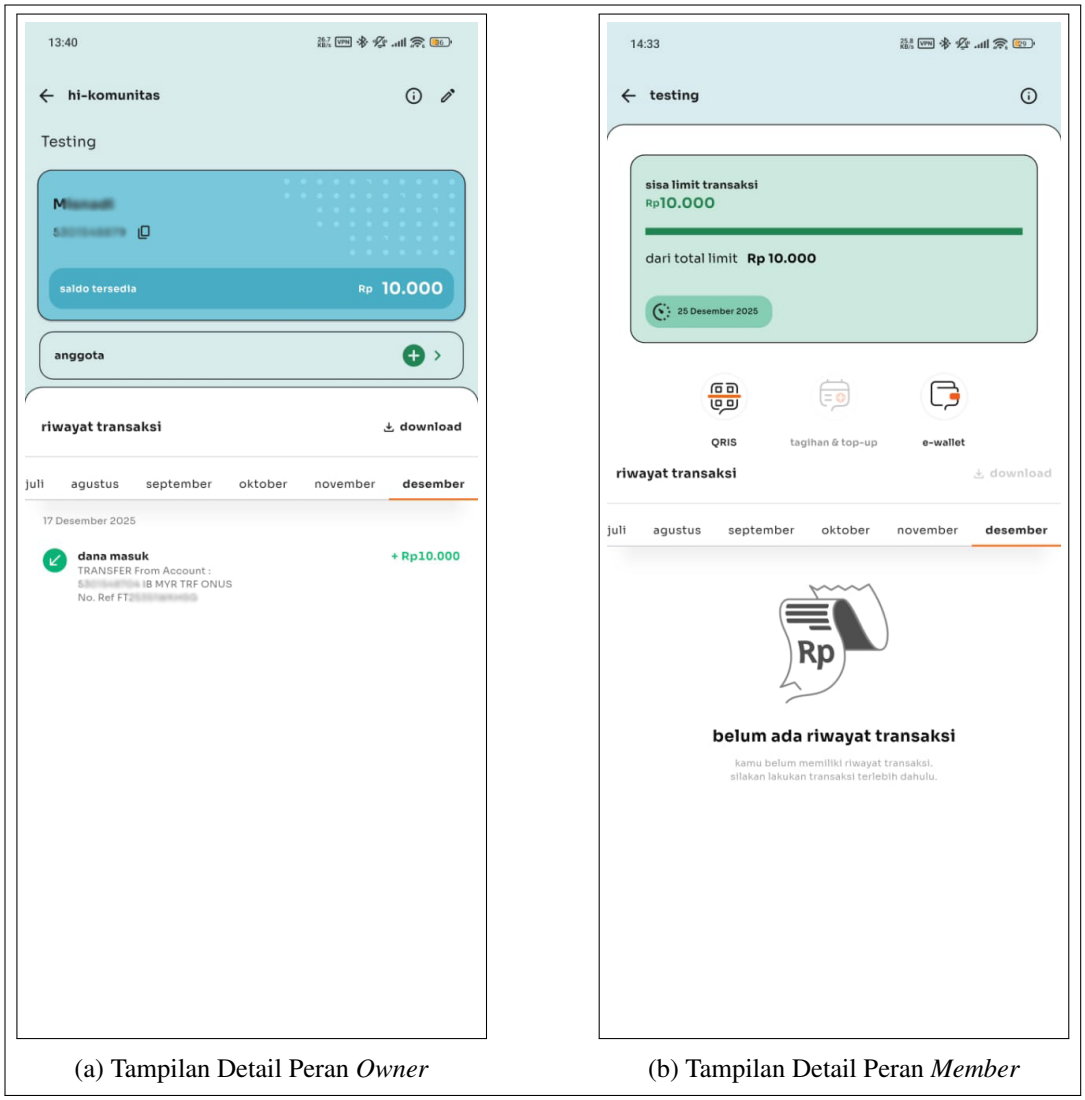
(a) Halaman Masukkan MPIN

(b) Halaman Berhasil Buat *Shared Wallet*

Gambar 3.31. Tahap Finalisasi Pembuatan *Shared Wallet*
Sumber Aplikasi *System Integrated Testing* Hibank

D.3 *View Shared Wallet Detail*

Sistem membedakan tampilan detail *shared wallet* berdasarkan peran pengguna sebagaimana diperlihatkan secara komparatif pada Gambar 3.32. Pemilik (*Owner*) mendapatkan akses penuh ke informasi saldo dan manajemen anggota seperti pada Gambar 3.32a, sedangkan anggota (*Member*) hanya melihat sisa *limit* transaksi dan fitur pembayaran cepat sebagaimana Gambar 3.32b.



(a) Tampilan Detail Peran *Owner* (b) Tampilan Detail Peran *Member*

Gambar 3.32. Perbandingan Implementasi Detail *Shared Wallet* Berdasarkan Peran Pengguna

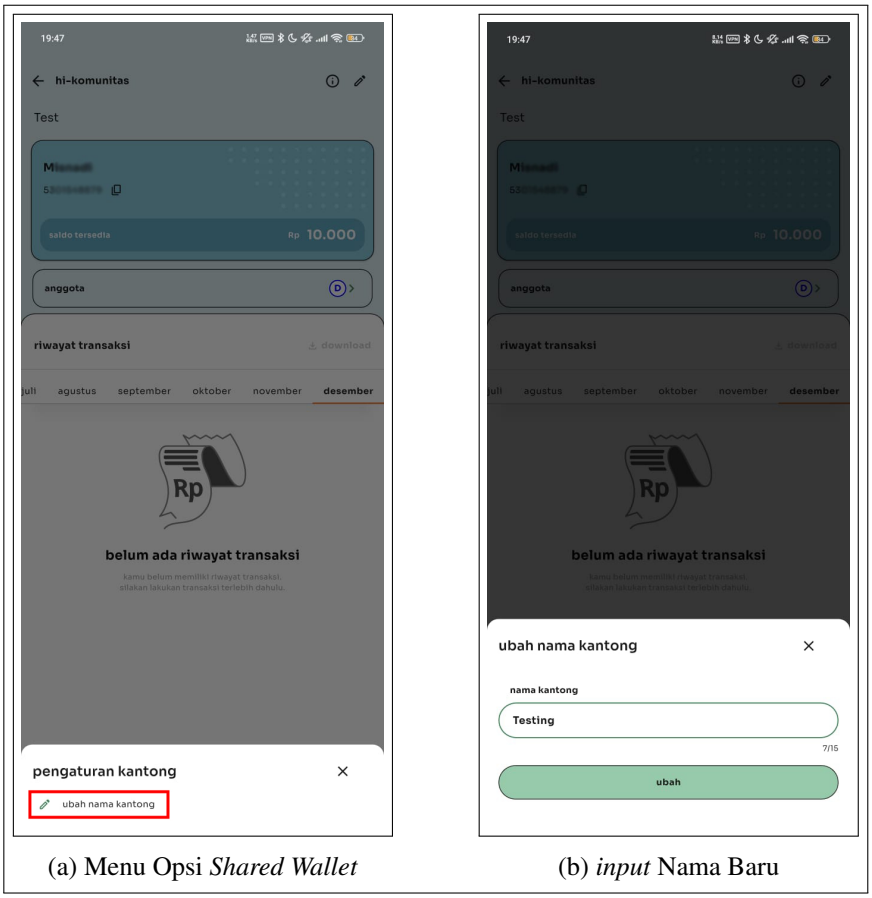
Sumber Aplikasi *System Integrated Testing* Hibank

D.4 *View Transaction History*

Fitur riwayat transaksi juga menerapkan kontrol akses berbasis peran yang tercermin pada Gambar 3.32 di atas. Pada tampilan pemilik seperti yang ditunjukkan oleh Gambar 3.32a, sistem menyajikan transparansi penuh atas seluruh mutasi dana. Sebaliknya, pada tampilan anggota seperti Gambar 3.32b, sistem membatasi informasi hanya pada transaksi yang dilakukan oleh anggota tersebut demi menjaga privasi.

D.5 *Update Shared Wallet Name*

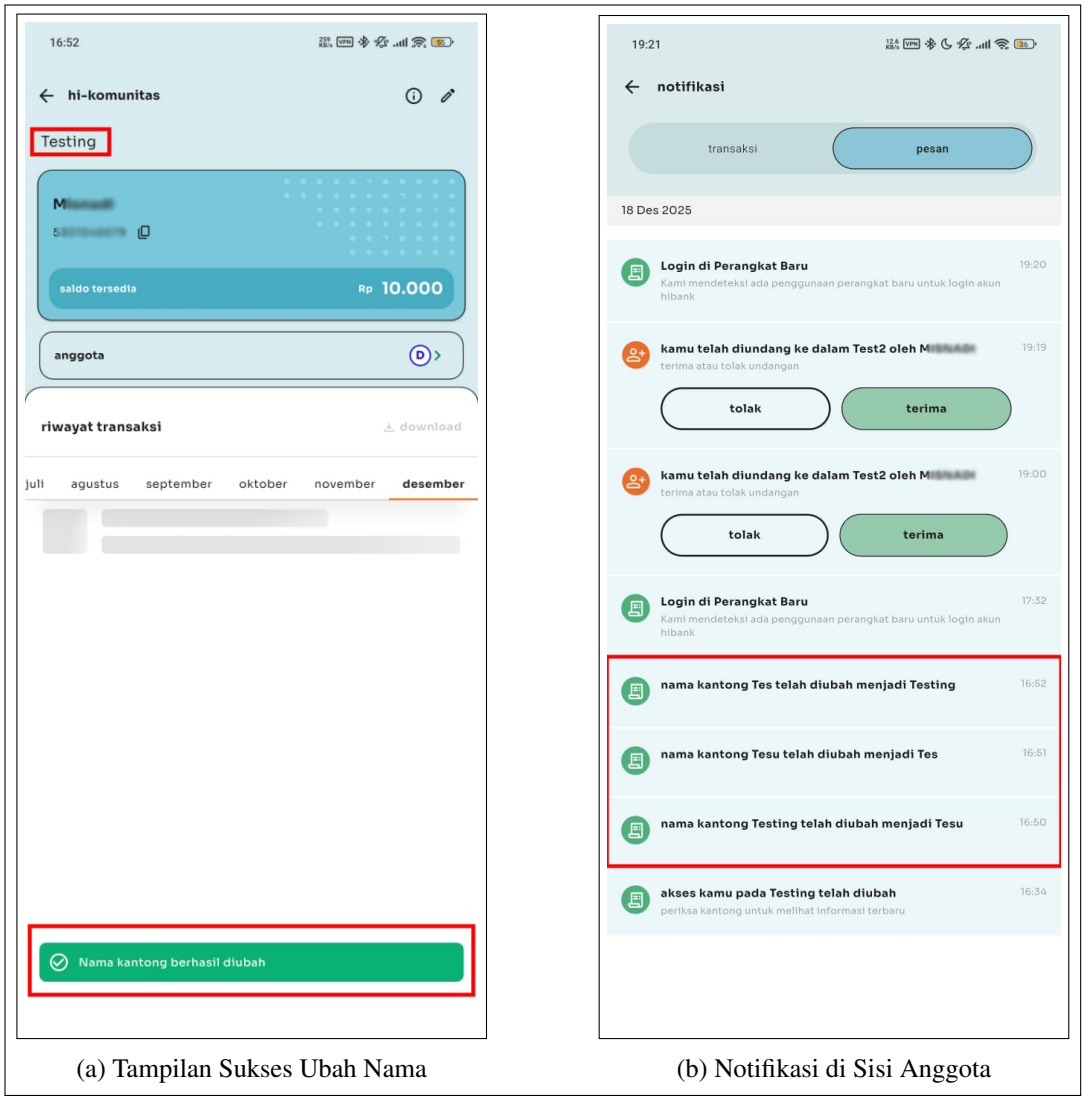
Proses inisiasi pengubahan nama *shared wallet*, sebagaimana disajikan secara lengkap pada Gambar 3.33, diawali dengan menekan ikon pensil yang terletak di sudut kanan atas halaman detail *shared wallet*. Interaksi ini akan memunculkan menu opsi pengelolaan seperti pada Gambar 3.33a, di mana pemilik dapat memilih fitur ”Ubah Nama Kantong” untuk menampilkan formulir *input* nama baru sebagaimana Gambar 3.33b.



(a) Menu Opsi *Shared Wallet* (b) *input* Nama Baru

Gambar 3.33. Tahap Inisiasi Pengubahan Nama *Shared Wallet*
Sumber Aplikasi *System Integrated Testing* Hibank

Hasil dari proses perubahan nama tersebut diperlihatkan pada Gambar 3.34. Sistem memberikan umpan balik sukses kepada pemilik seperti terlihat pada Gambar 3.34a dan secara paralel mengirimkan notifikasi kepada seluruh anggota terkait sebagaimana Gambar 3.34b.



(a) Tampilan Sukses Ubah Nama

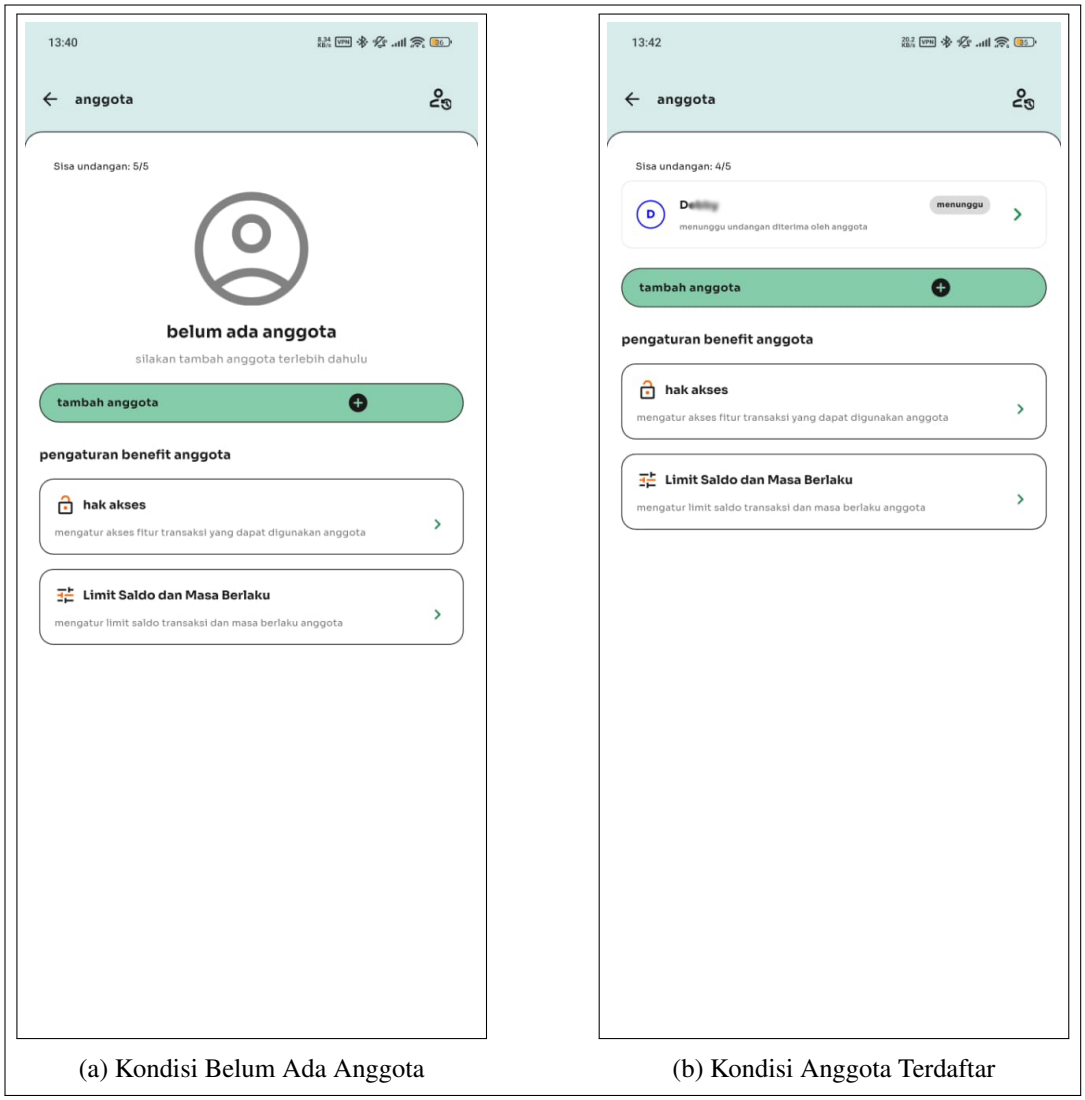
(b) Notifikasi di Sisi Anggota

Gambar 3.34. Hasil Pengubahan Nama dan Notifikasi Anggota

Sumber Aplikasi *System Integrated Testing* Hibank

D.6 *View Member List*

Halaman daftar anggota menampilkan seluruh pengguna yang terasosiasi dengan *shared wallet*, dengan variasi tampilan kondisi data yang disajikan pada Gambar 3.35. Terdapat dua kondisi utama, yaitu kondisi awal saat *shared wallet* baru dibuat dan belum memiliki anggota tambahan seperti pada Gambar 3.35a, serta kondisi saat anggota telah terdaftar sebagaimana Gambar 3.35b.

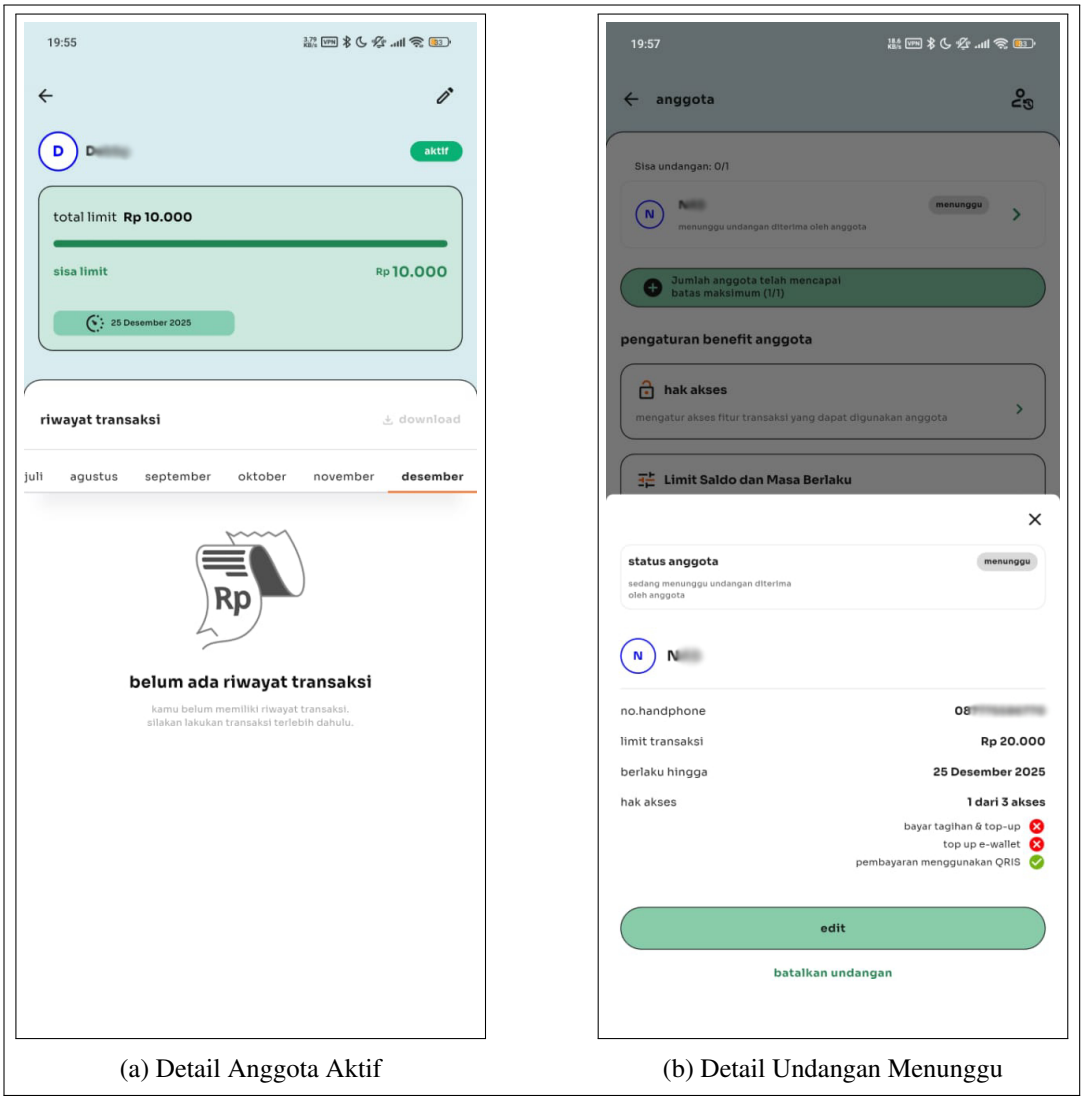


(a) Kondisi Belum Ada Anggota (b) Kondisi Anggota Terdaftar

Gambar 3.35. Implementasi Daftar Anggota *Shared Wallet* Dalam Berbagai Kondisi Data Sumber Aplikasi *System Integrated Testing* Hibank

D.7 *View Member Detail*

Rincian informasi anggota ditampilkan dengan penyesuaian berdasarkan status keanggotaan, sebagaimana diperlihatkan pada Gambar 3.36. Untuk anggota aktif seperti pada Gambar 3.36a, sistem menampilkan sisa *limit* dan riwayat transaksi penuh. Sedangkan untuk calon anggota dengan status menunggu sebagaimana Gambar 3.36b, sistem menampilkan ringkasan data undangan dalam format *bottom sheet*.

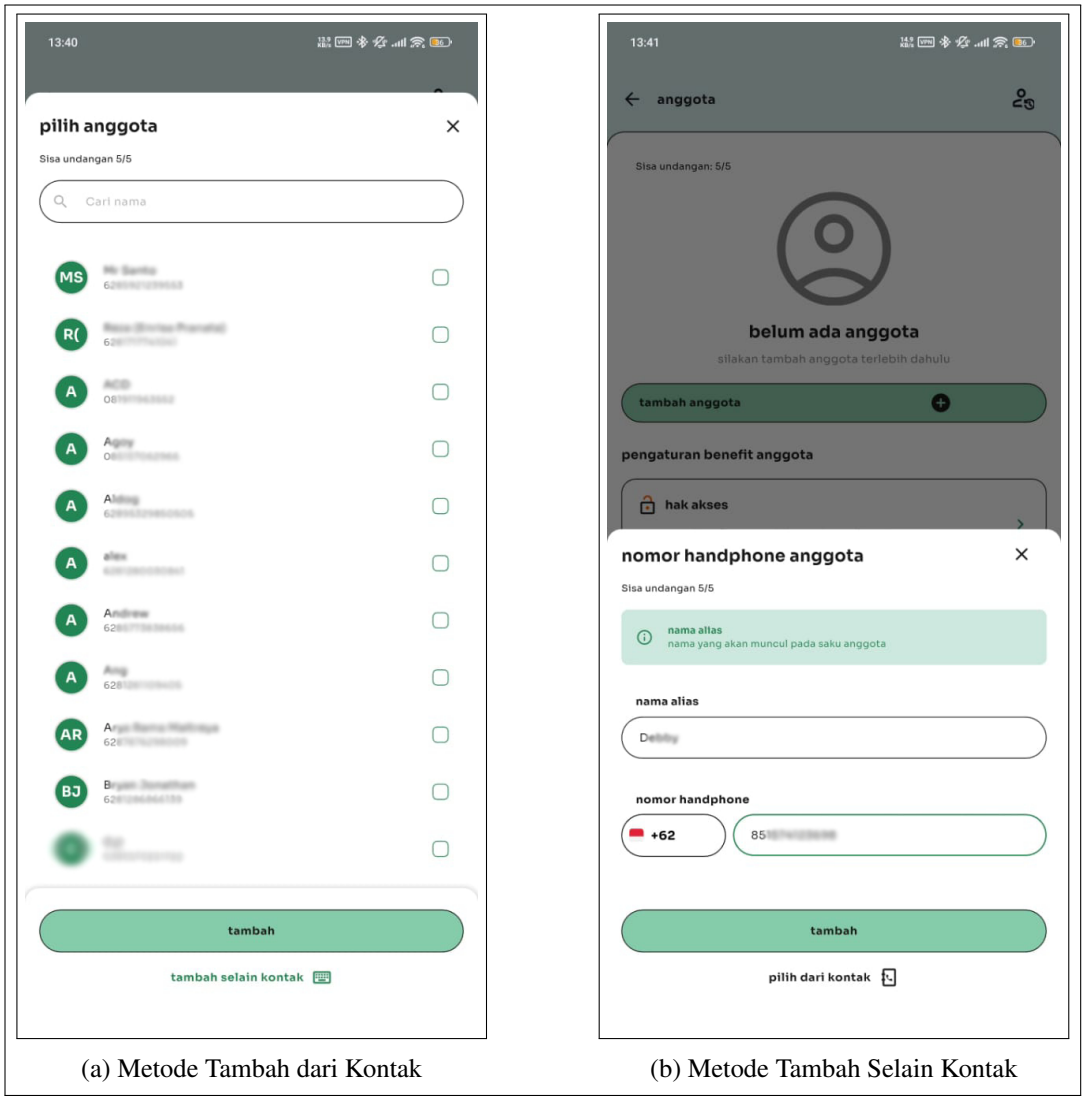


(a) Detail Anggota Aktif (b) Detail Undangan Menunggu

Gambar 3.36. Implementasi Detail Anggota Berdasarkan Status Keanggotaan
Sumber Aplikasi *System Integrated Testing* Hibank

D.8 *Invite Member*

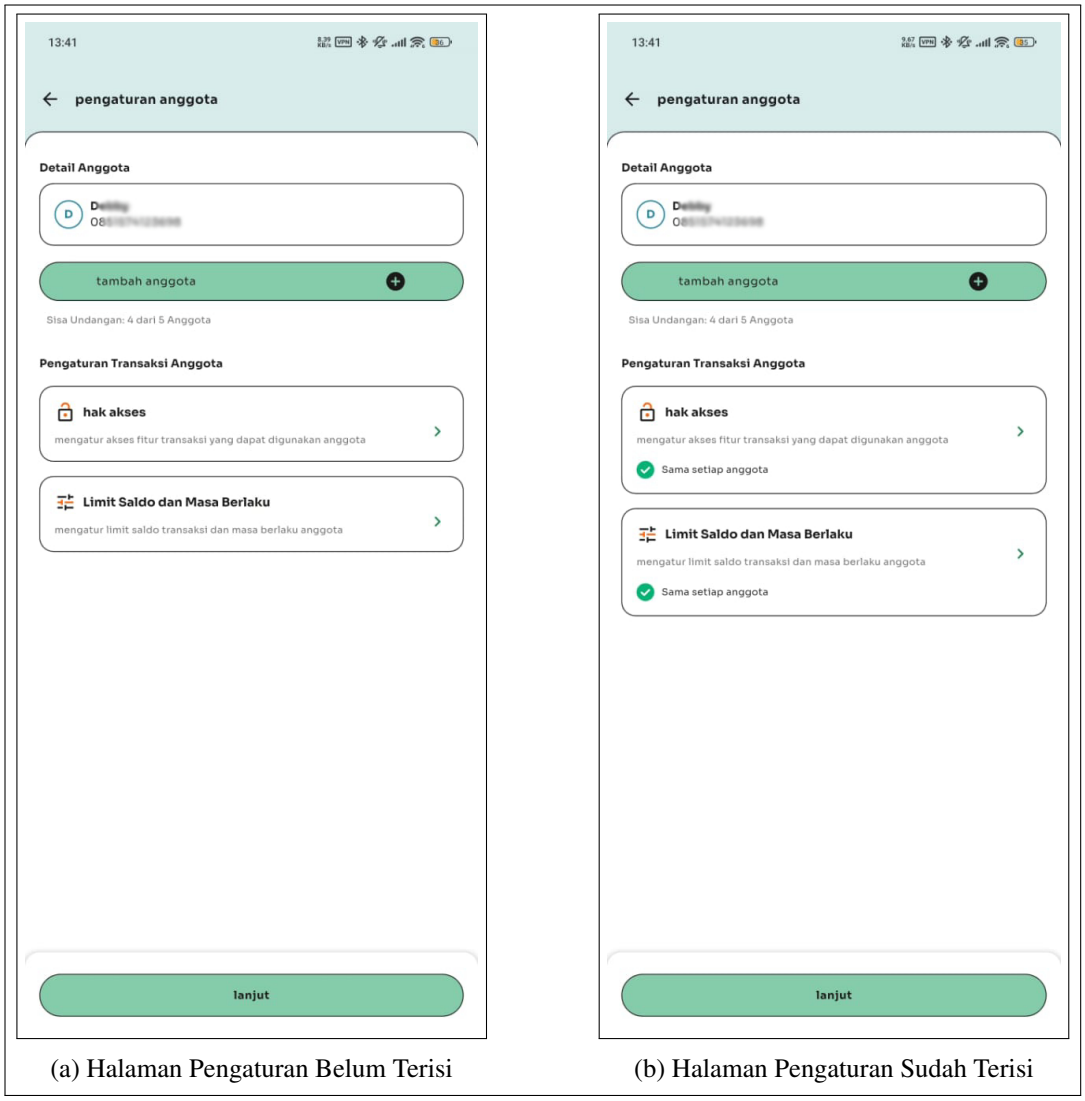
Sistem menyediakan dua metode untuk menambahkan anggota yang visualisasinya disajikan pada Gambar 3.37. Pemilik dapat memilih metode ”tambah dari kontak” seperti pada Gambar 3.37a untuk integrasi langsung dengan buku telepon, atau ”tambah manual” sebagaimana Gambar 3.37b dengan melakukan *input* data alias dan nomor telepon.



(a) Metode Tambah dari Kontak (b) Metode Tambah Selain Kontak

Gambar 3.37. Implementasi Pilihan Metode Undangan Anggota
Sumber Aplikasi *System Integrated Testing* Hibank

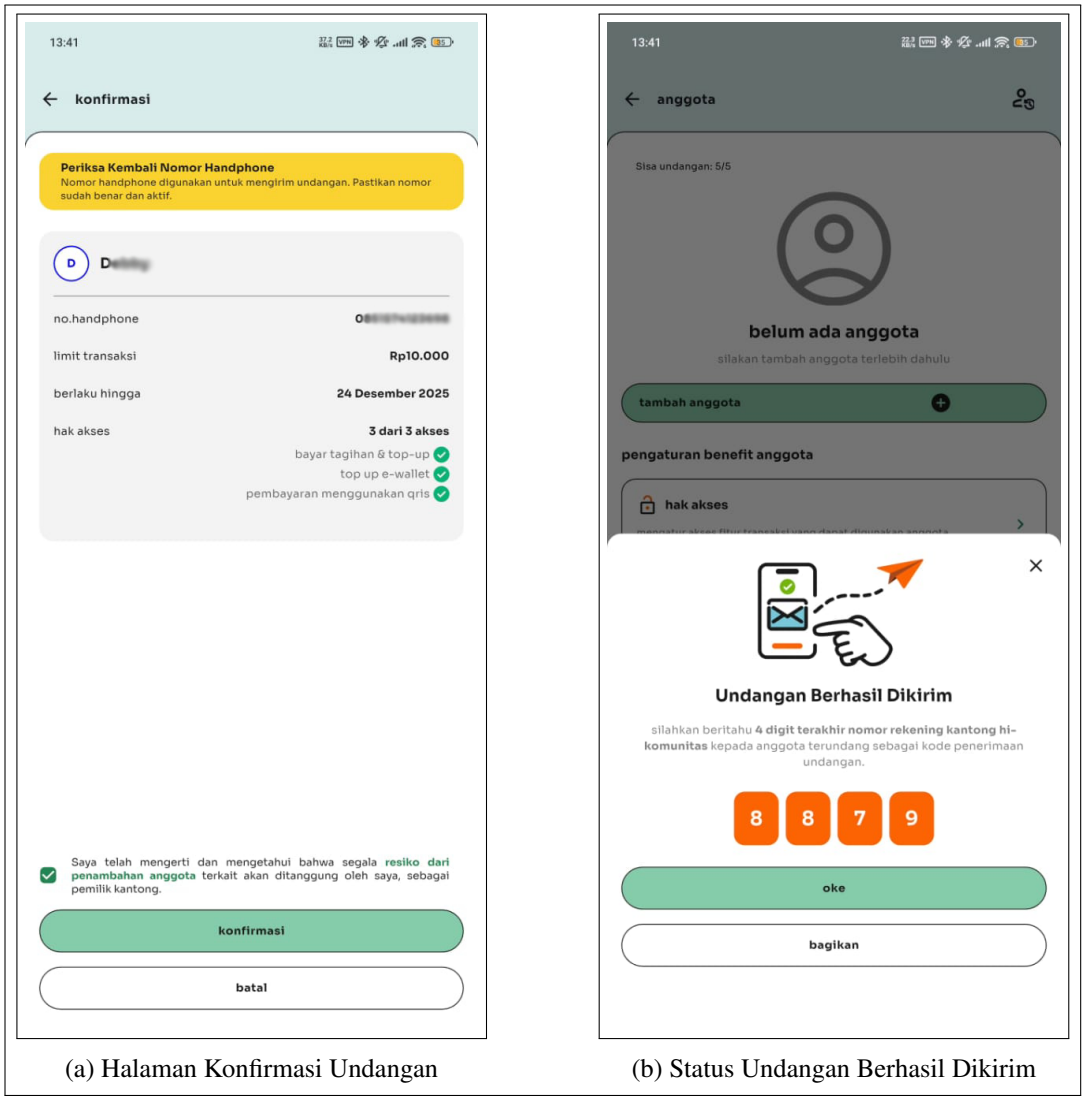
Setelah memilih anggota, proses berlanjut ke konfigurasi izin yang status kelengkapannya diperlihatkan pada Gambar 3.38. Tombol ”Lanjut” hanya akan aktif seperti terlihat pada Gambar 3.38b setelah pengguna melengkapi seluruh parameter yang sebelumnya kosong sebagaimana Gambar 3.38a.



(a) Halaman Pengaturan Belum Terisi (b) Halaman Pengaturan Sudah Terisi

Gambar 3.38. Implementasi Perubahan Status Kelengkapan Pengaturan Anggota
Sumber Aplikasi *System Integrated Testing* Hibank

Tahap akhir pengiriman undangan, yang mencakup konfirmasi dan notifikasi sukses, dirangkum pada Gambar 3.39. Pengguna memverifikasi data pada halaman konfirmasi seperti pada Gambar 3.39a, melakukan otorisasi MPIN, dan kemudian menerima umpan balik berupa *bottom sheet* sukses sebagaimana Gambar 3.39b.

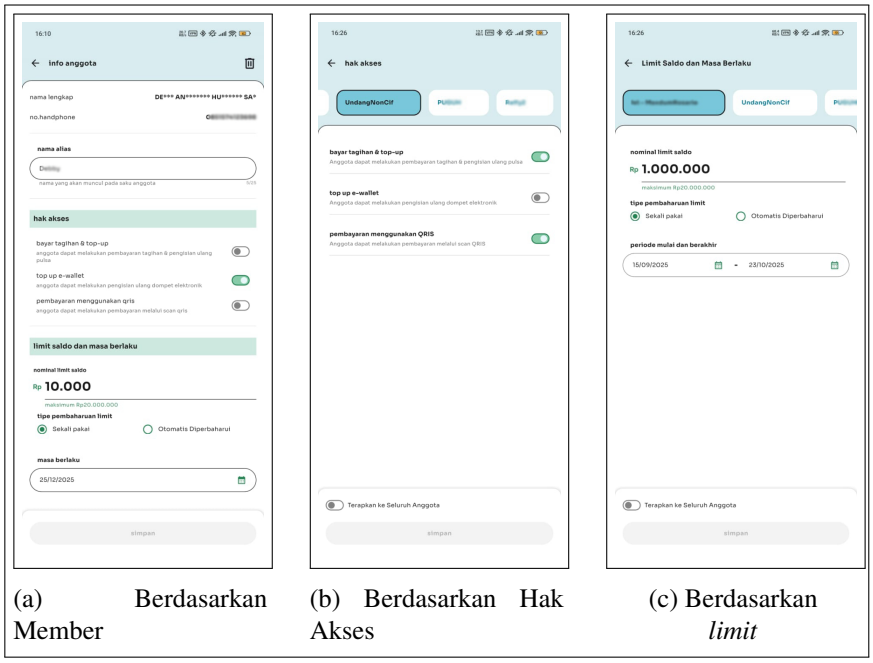


(a) Halaman Konfirmasi Undangan (b) Status Undangan Berhasil Dikirim

Gambar 3.39. Implementasi Konfirmasi dan Status Akhir Undangan Anggota
Sumber Aplikasi *System Integrated Testing* Hibank

D.9 *Manage Member Permission*

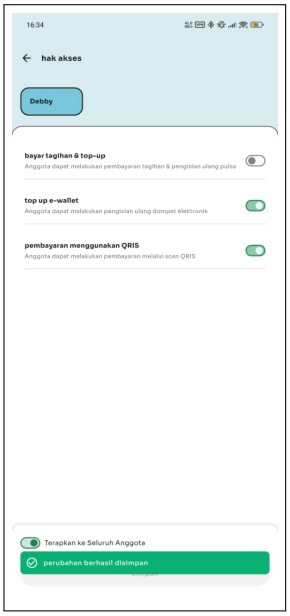
Manajemen izin anggota menawarkan fleksibilitas konfigurasi yang visualisasinya disajikan pada Gambar 3.40. Pemilik dapat mengatur berdasarkan individu anggota seperti pada Gambar 3.40a, berdasarkan kategori hak akses sebagaimana Gambar 3.40b, atau berdasarkan kategori *limit* saldo seperti Gambar 3.40c.



(a) Berdasarkan Member (b) Berdasarkan Hak Akses (c) Berdasarkan *limit*

Gambar 3.40. Implementasi Manajemen Izin Berdasarkan Perspektif Anggota dan Kategori
Sumber Aplikasi *System Integrated Testing* Hibank

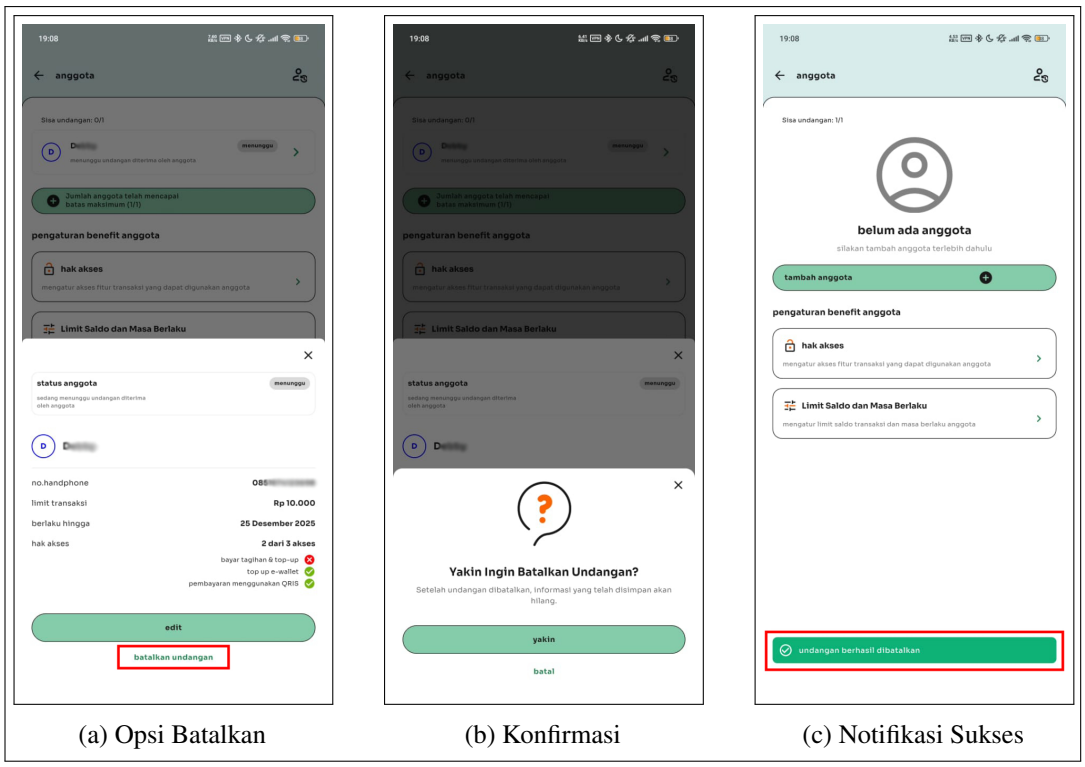
Setelah konfigurasi disimpan dan divalidasi dengan MPIN, sistem menampilkan indikator keberhasilan berupa notifikasi panel hijau, sebagaimana diperlihatkan pada Gambar 3.41.



Gambar 3.41. Indikator Keberhasilan Penyimpanan Pengaturan
Sumber Aplikasi *System Integrated Testing* Hibank

D.10 *Cancel Invitation*

Alur pembatalan undangan oleh pemilik divisualisasikan secara lengkap pada Gambar 3.42. Proses ini dimulai dari opsi batalkan pada detail anggota seperti pada Gambar 3.42a, dilanjutkan dengan jendela konfirmasi sebagaimana Gambar 3.42b, dan diakhiri dengan notifikasi sukses seperti terlihat pada Gambar 3.42c.

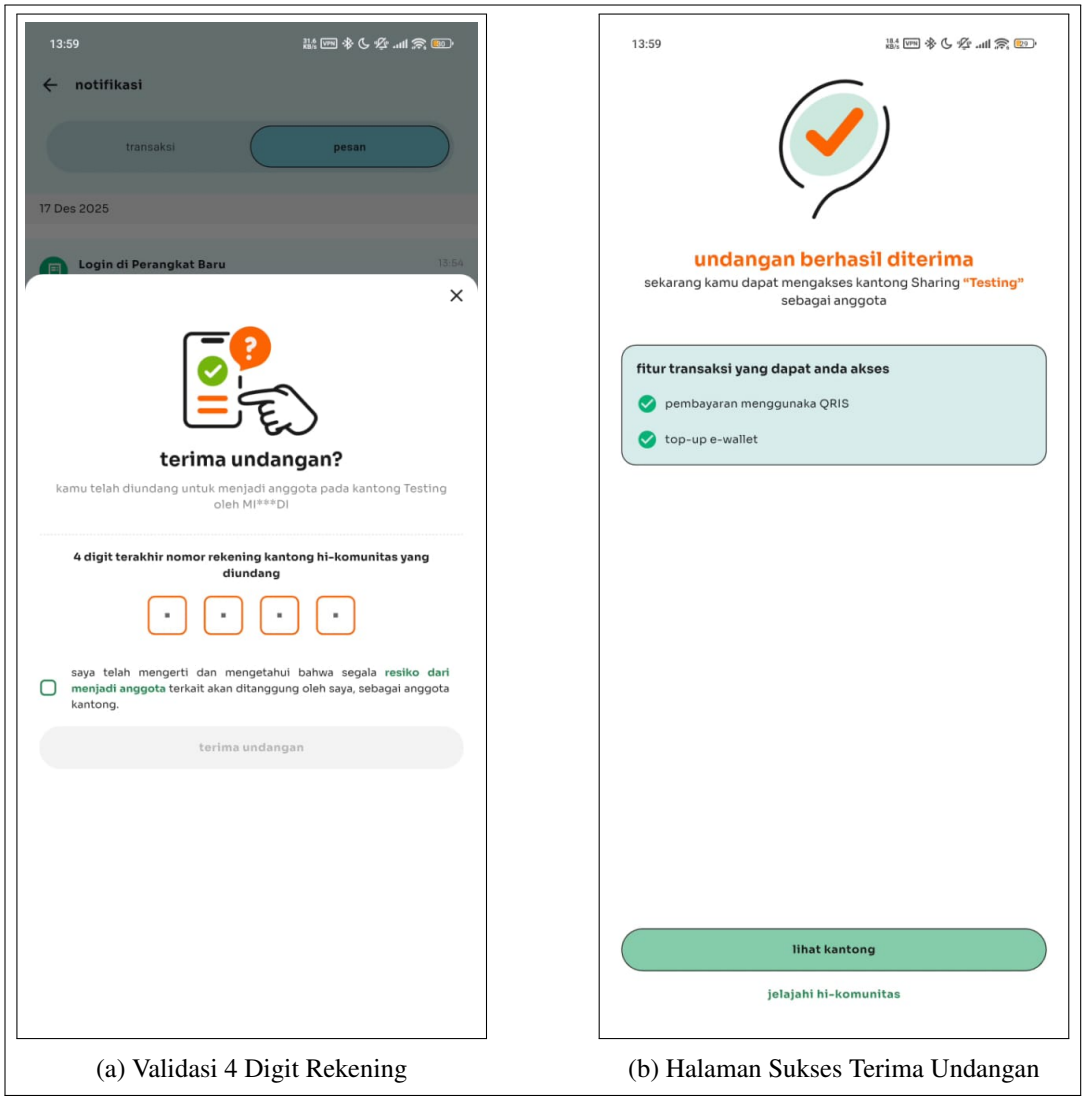


(a) Opsi Batalkan (b) Konfirmasi (c) Notifikasi Sukses

Gambar 3.42. Alur Pembatalan Undangan oleh Pemilik
Sumber Aplikasi *System Integrated Testing* Hibank

D.11 *Accept Invitation*

Rangkaian proses penerimaan undangan dan validasi keamanannya diilustrasikan pada Gambar 3.43. Pengguna diwajibkan memasukkan 4 digit terakhir nomor rekening seperti pada Gambar 3.43a sebelum sistem menampilkan status sukses dan memberikan akses ke *shared wallet* sebagaimana Gambar 3.43b.

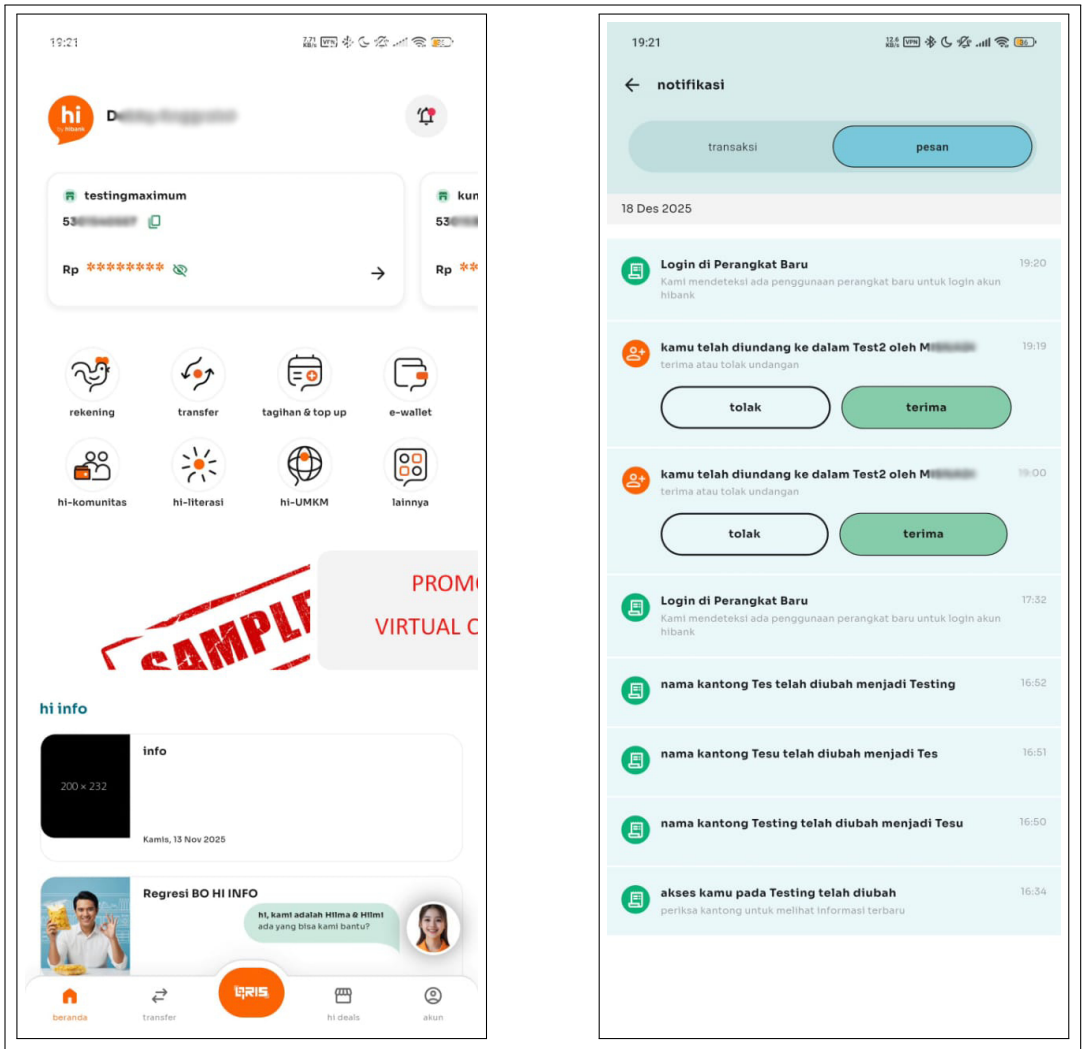


(a) Validasi 4 Digit Rekening (b) Halaman Sukses Terima Undangan

Gambar 3.43. Alur Penerimaan Undangan dan Validasi Keamanan
Sumber Aplikasi *System Integrated Testing* Hibank

D.12 *Decline Invitation*

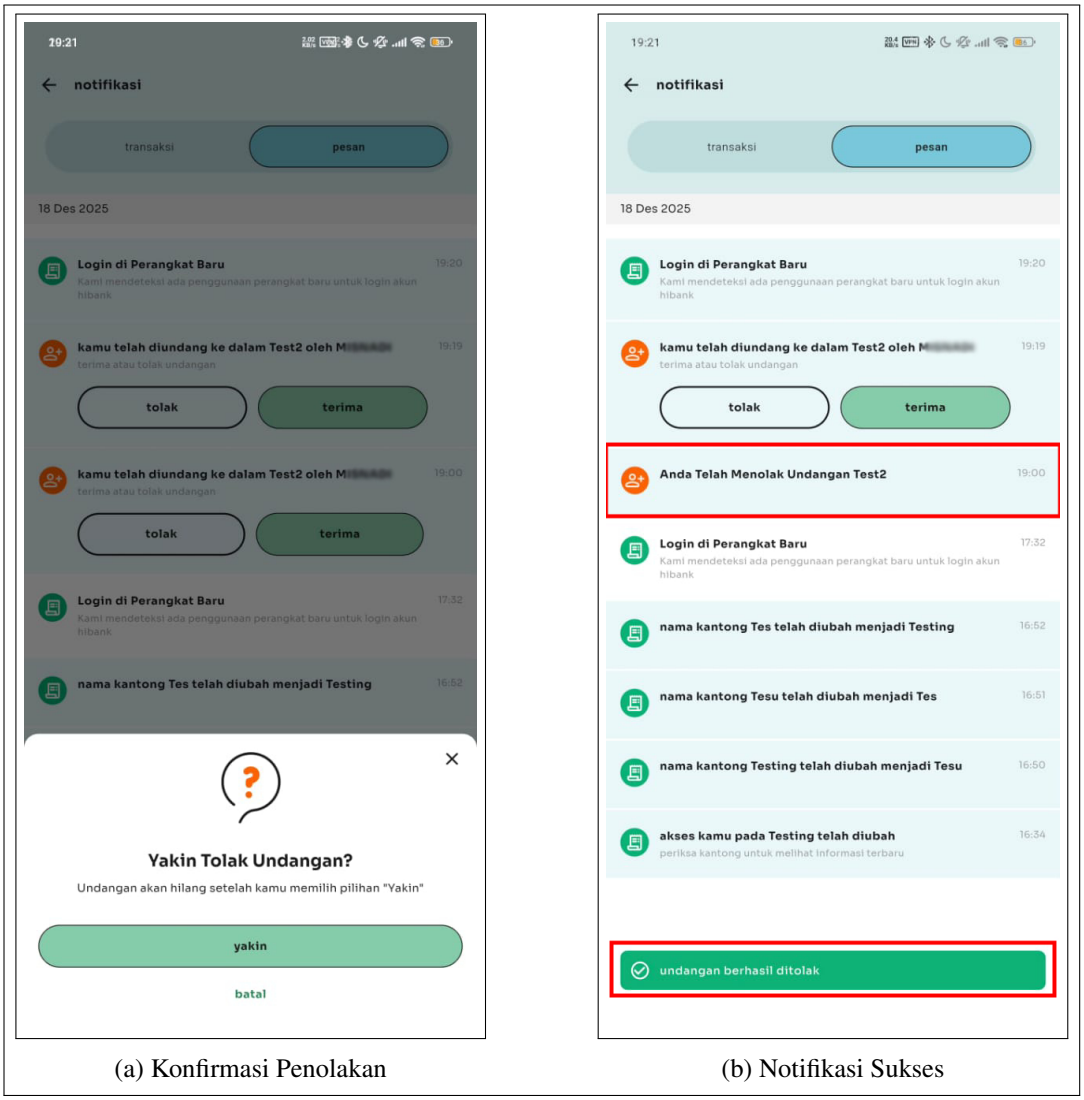
Tahap akses awal untuk menolak undangan disajikan pada Gambar 3.44, di mana pengguna mengakses notifikasi dari *dashboard* seperti pada Gambar 3.44a dan memilih pesan undangan terkait sebagaimana Gambar 3.44b.



(a) Akses Notifikasi dari *Dashboard* (b) Daftar Pesan Undangan

Gambar 3.44. Tahap Akses Notifikasi Undangan pada Aplikasi Hibank
Sumber Aplikasi *System Integrated Testing* Hibank

Eksekusi penolakan tersebut diakhiri dengan konfirmasi dan umpan balik sistem sebagaimana diperlihatkan pada Gambar 3.45. Pengguna menyetujui penolakan pada jendela konfirmasi seperti pada Gambar 3.45a dan menerima notifikasi sukses sebagaimana Gambar 3.45b.



(a) Konfirmasi Penolakan (b) Notifikasi Sukses

Gambar 3.45. Tahap Konfirmasi dan Status Akhir Penolakan
Sumber Aplikasi *System Integrated Testing* Hibank

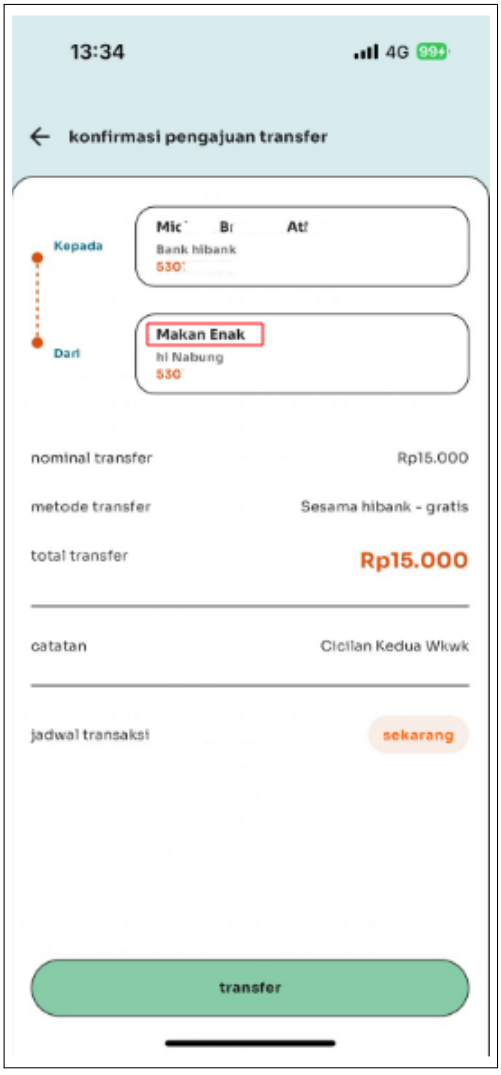
3.3.2 Perbaikan *Bug* pada Layanan Eksisting

Dalam siklus pengembangan perangkat lunak, pemeliharaan stabilitas sistem merupakan aspek krusial yang berjalan paralel dengan implementasi fitur baru. Bagian ini menguraikan rangkaian aktivitas perbaikan (*bug fixing*) yang dilakukan sebagai respons terhadap anomali teknis yang ditemukan pada layanan eksisting selama fase pengujian (*System Integrated Testing*). Fokus utama dari perbaikan ini mencakup analisis akar masalah (*root cause analysis*), penyesuaian logika bisnis pada sisi *backend*, serta validasi ulang guna memastikan integritas data dan kinerja layanan tetap optimal sebelum dirilis ke lingkungan produksi.

A Perbaikan *Mapping* Nama Rekening Hi-Nabung

A.1 Deskripsi Masalah

Pada fitur transfer, ditemukan inkonsistensi data di mana kolom pengirim menampilkan *shared wallet name* dan bukan nama legal nasabah. Hal ini menyebabkan ambiguitas identitas pada bukti transaksi penerima sebagaimana diperlihatkan pada Gambar 3.46.



Gambar 3.46. Tampilan Bukti Kesalahan Nama *Shared Wallet* Pada Halaman Transfer Sumber Asana

A.2 Analisis Akar Masalah (*Root Cause*)

Kesalahan terletak pada logika inisiasi data (*data initialization*) di level *backend*. Sistem memetakan *input user* secara literal tanpa membedakan antara label personal (nama *shared wallet*) dan identitas legal (nama rekening).

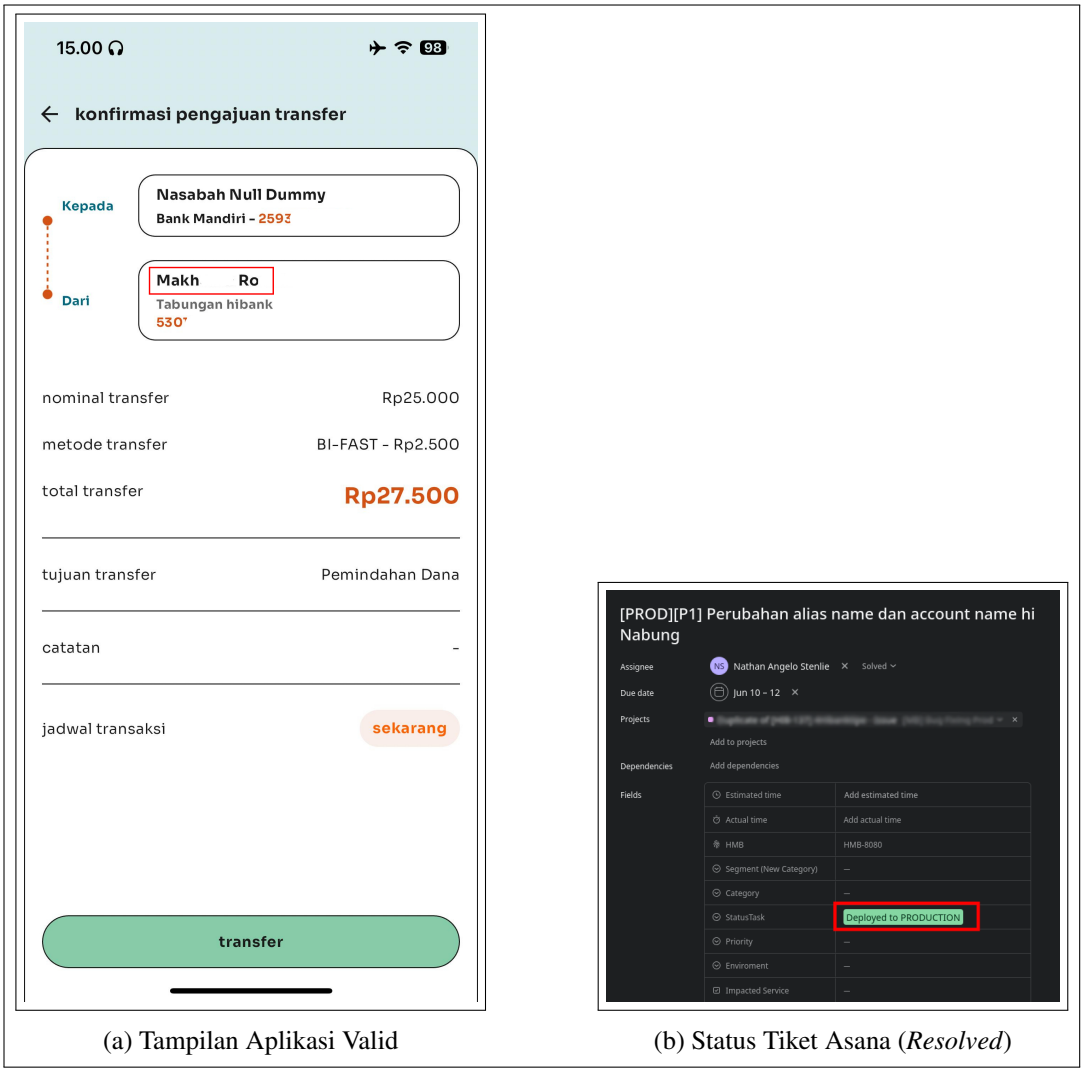
A.3 Langkah Perbaikan

Perbaikan dilakukan dengan merekonstruksi alur pemetaan data sebagai berikut.

1. Atribut *Account Name* diubah sumber datanya menjadi *Customer Information File* (CIF) untuk menjamin validitas hukum identitas pengirim.
2. *input* nama *shared wallet* dipindahkan ke atribut *Account Alias*, sehingga tetap tersimpan sebagai personalisasi tanpa mengganggu data transaksi.
3. Menambahkan validasi untuk memastikan atribut *Account Name* tidak boleh bernilai *null* atau string kosong.

A.4 Hasil

Setelah dilakukan *hotfix* dan pengujian ulang, bukti transfer kini menampilkan nama nasabah yang valid sesuai KTP sebagaimana diperlihatkan pada Gambar 3.47a. Selain itu, isu ambiguitas data dinyatakan selesai (*Resolved*) sesuai status tiket pada Gambar 3.47b.



(a) Tampilan Aplikasi Valid

(b) Status Tiket Asana (*Resolved*)

Gambar 3.47. Hasil Perbaikan *Bug* Mapping Nama dan Bukti Tiket Selesai Sumber Asana

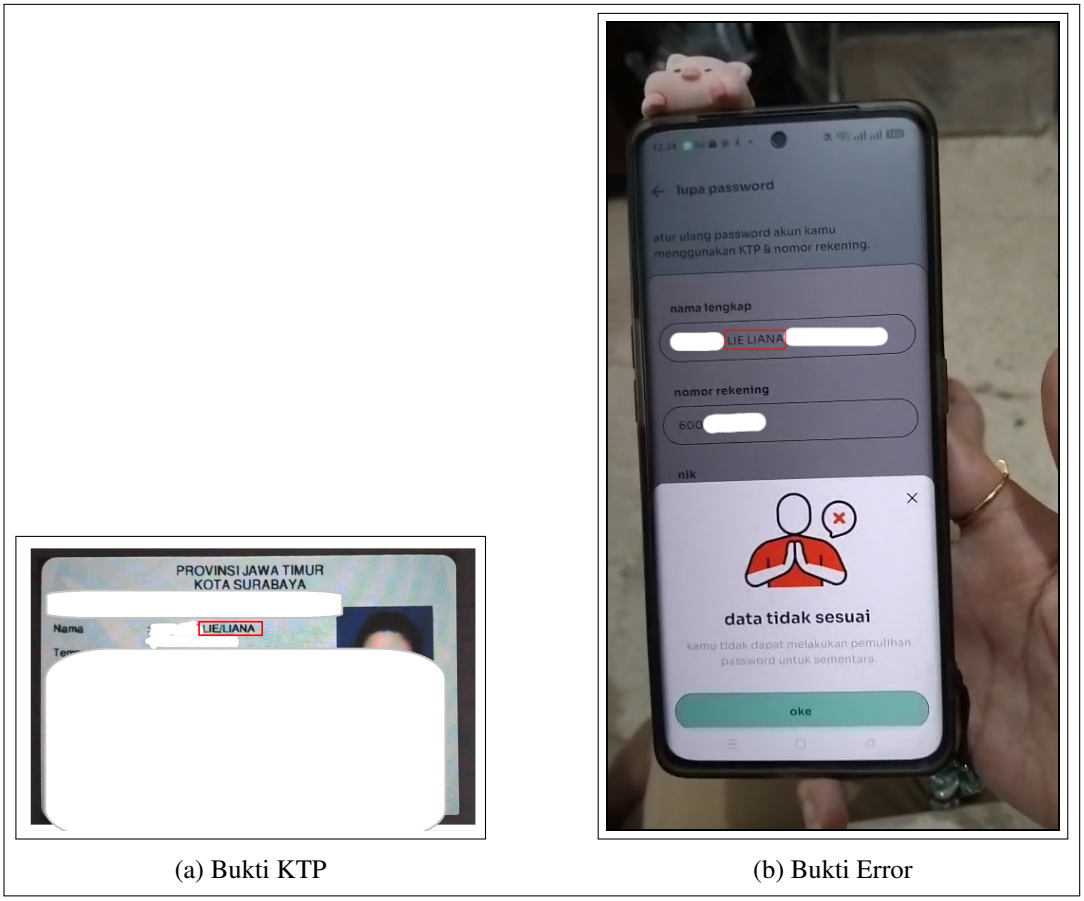
B Penyesuaian Validasi Karakter Spesial pada Layanan Registrasi

B.1 Deskripsi Masalah

Pada lingkungan produksi (*Production*), ditemukan kendala di mana sistem gagal memproses *input* data nasabah yang memiliki karakter spesial, khususnya tanda garis miring (*slash*) '/' pada kolom Nama Lengkap dan Nama Gadis Ibu Kandung.

Permasalahan ini menghambat proses *onboarding* karena terdapat nasabah yang secara legal memiliki karakter tersebut dalam dokumen identitas kependudukan (KTP) sebagaimana diperlihatkan pada Gambar 3.48a. Namun,

ketika nasabah mencoba memasukkan data sesuai KTP pada alur Registrasi (*NTB/ETB*) maupun *Forgot Password*, sistem menolak *input* tersebut dan menampilkan pesan kesalahan validasi seperti terlihat pada Gambar 3.48b.



(a) Bukti KTP (b) Bukti Error

Gambar 3.48. Bukti Kendala *input* Karakter Spesial
Sumber Asana

B.2 Analisis Akar Masalah (*Root Cause*)

Investigasi teknis menunjukkan bahwa penolakan ini disebabkan oleh konfigurasi *Regular Expression* (*Regex*) pada lapisan keamanan aplikasi yang bersifat terlalu restriktif. Pola *regex* yang diterapkan sebelumnya menggunakan mekanisme *allow-list* yang ketat namun belum mencakup karakter garis miring '/' sebagai karakter yang diizinkan, sehingga *input* tersebut dikategorikan sebagai karakter ilegal.

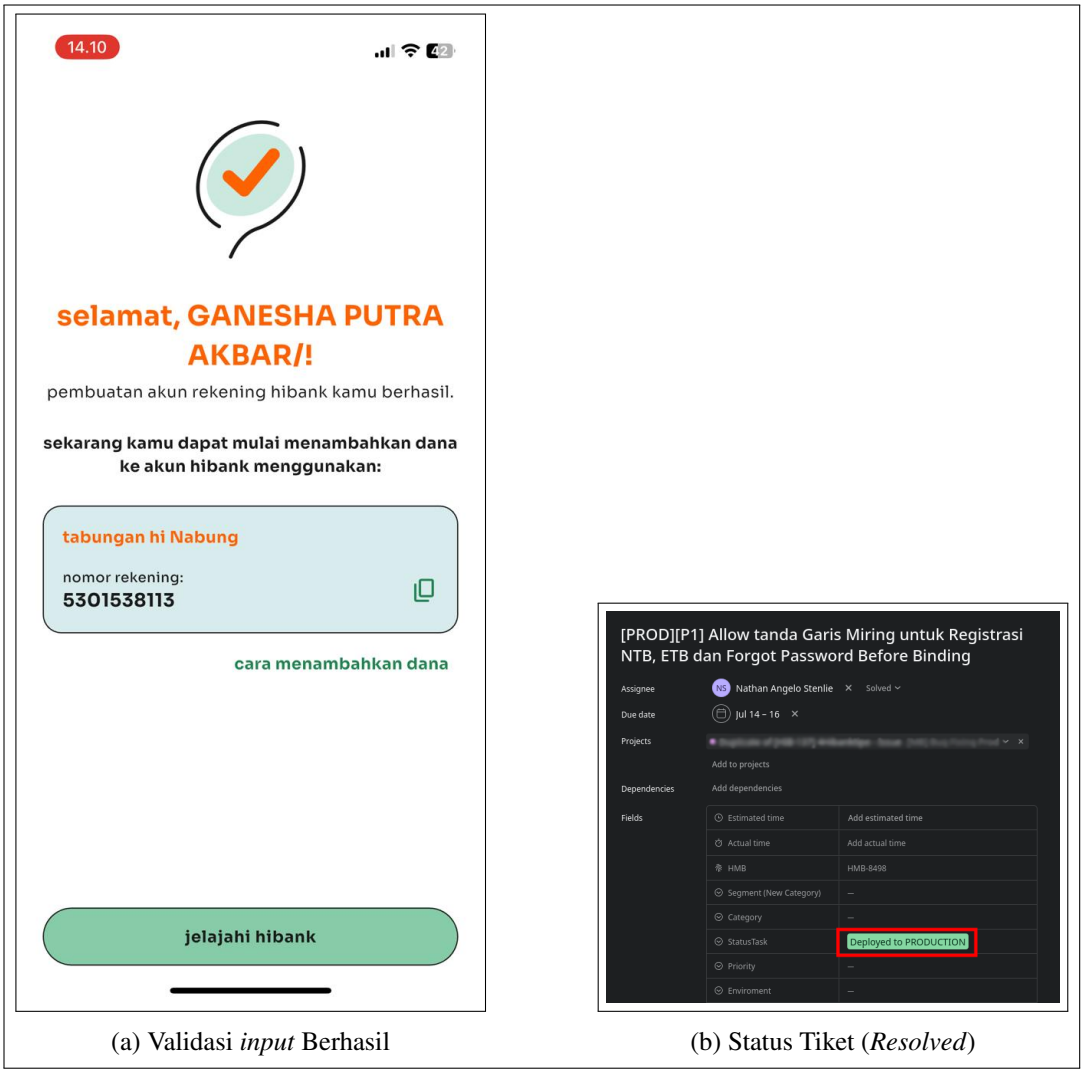
B.3 Langkah Perbaikan

Perbaikan dilakukan dengan memodifikasi logika validasi *input* (*Input Validation Logic*) melalui langkah-langkah berikut.

1. Memperbarui pola *Regex* pada *validator* untuk mengakomodasi karakter garis miring '/' sebagai karakter valid.
2. Menerapkan standarisasi aturan validasi baru ini secara konsisten pada seluruh *endpoint* API yang menangani modul Registrasi Nasabah Baru, Nasabah Eksisting, dan inisiasi *Forgot Password*.

B.4 Hasil

Pasca implementasi *patch* perbaikan, hasil pengujian verifikasi dirangkum pada Gambar 3.49. Sistem kini mampu memproses *input* nama yang mengandung karakter garis miring dengan benar seperti pada Gambar 3.49a. Hal ini memastikan data sesuai dokumen legal tanpa error, dan status perbaikan telah divalidasi selesai pada sistem manajemen proyek sebagaimana Gambar 3.49b.



(a) Validasi *input* Berhasil (b) Status Tiket (*Resolved*)

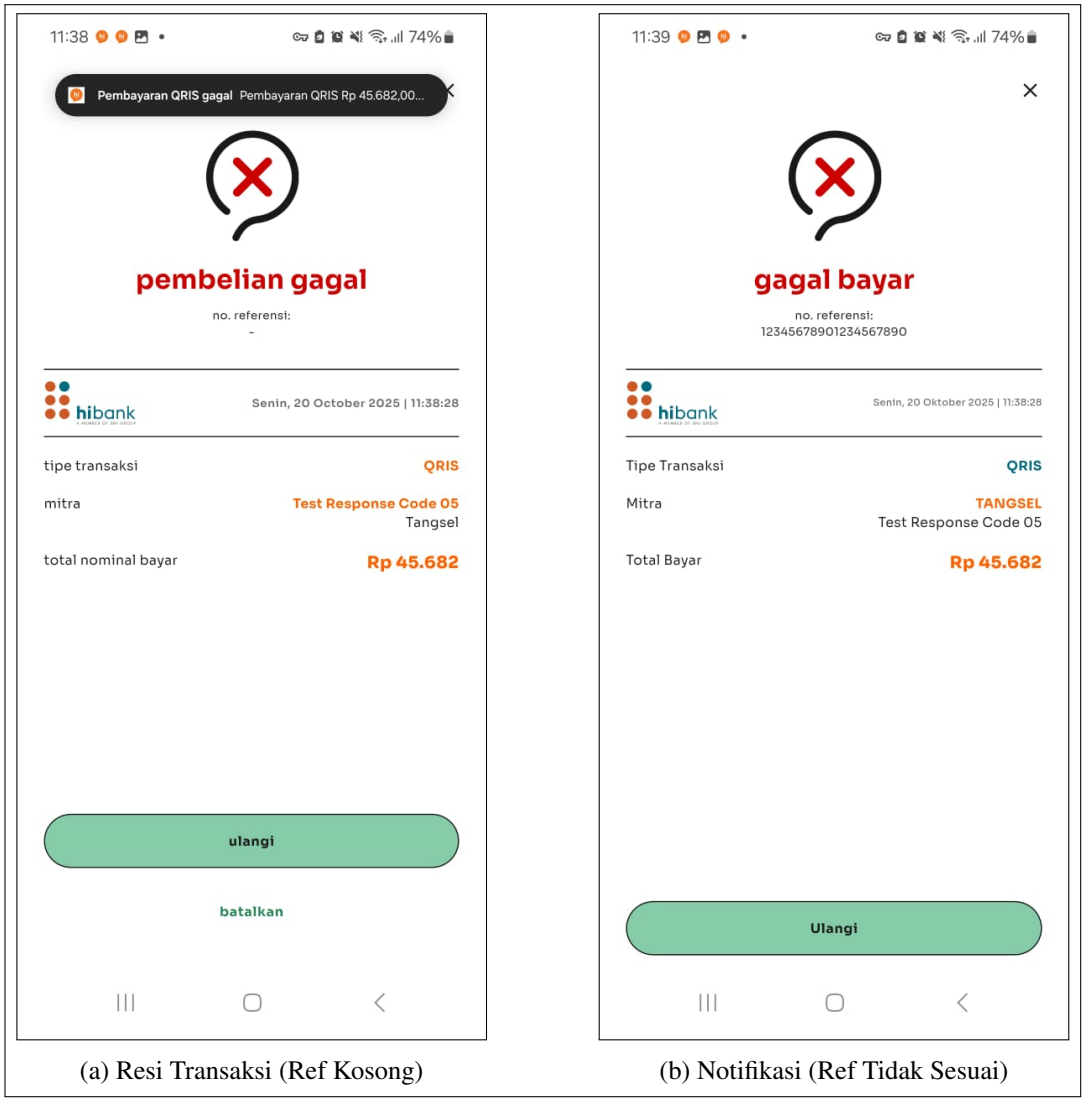
Gambar 3.49. Hasil Perbaikan *Bug* Karakter Spesial dan Bukti Tiket Selesai Sumber Asana

C Perbaikan Anomali Data Nomor Referensi pada Transaksi QRIS

C.1 Deskripsi Masalah

Ditemukan inkonsistensi data transaksi pada fitur pembayaran QRIS yang menggunakan Sumber Dana Hi-Komunitas sebagaimana diperlihatkan pada Gambar 3.50. Anomali ini terlihat pada dua titik antarmuka pengguna. Pertama, pada halaman akhir resi transaksi seperti pada Gambar 3.50a, atribut *Reference Number* (No. Reff) tidak tampil atau bernilai kosong (*null*). Kedua, ketidaksesuaian data juga ditemukan pada pusat notifikasi di mana detail notifikasi menampilkan data referensi yang tidak valid atau tidak sinkron dengan data transaksi yang

sebenarnya, sebagaimana terlihat pada Gambar 3.50b.



(a) Resi Transaksi (Ref Kosong) (b) Notifikasi (Ref Tidak Sesuai)

Gambar 3.50. Bukti Kendala Data Referensi pada Resi dan Notifikasi
Sumber Asana

C.2 Analisis Akar Masalah (*Root Cause*)

Permasalahan ini berakar pada dua isu teknis utama.

1. Layanan *backend* gagal memetakan (*populate*) atribut nomor referensi ke dalam objek respons API (*response body*) yang digunakan khusus untuk tipe transaksi Hi-Komunitas.

2. Terjadi kegagalan sinkronisasi data (*data race condition*) antara *Transaction*

Service dan *Notification Service*, menyebabkan notifikasi mengambil ID internal yang salah alih-alih nomor referensi eksternal yang valid.

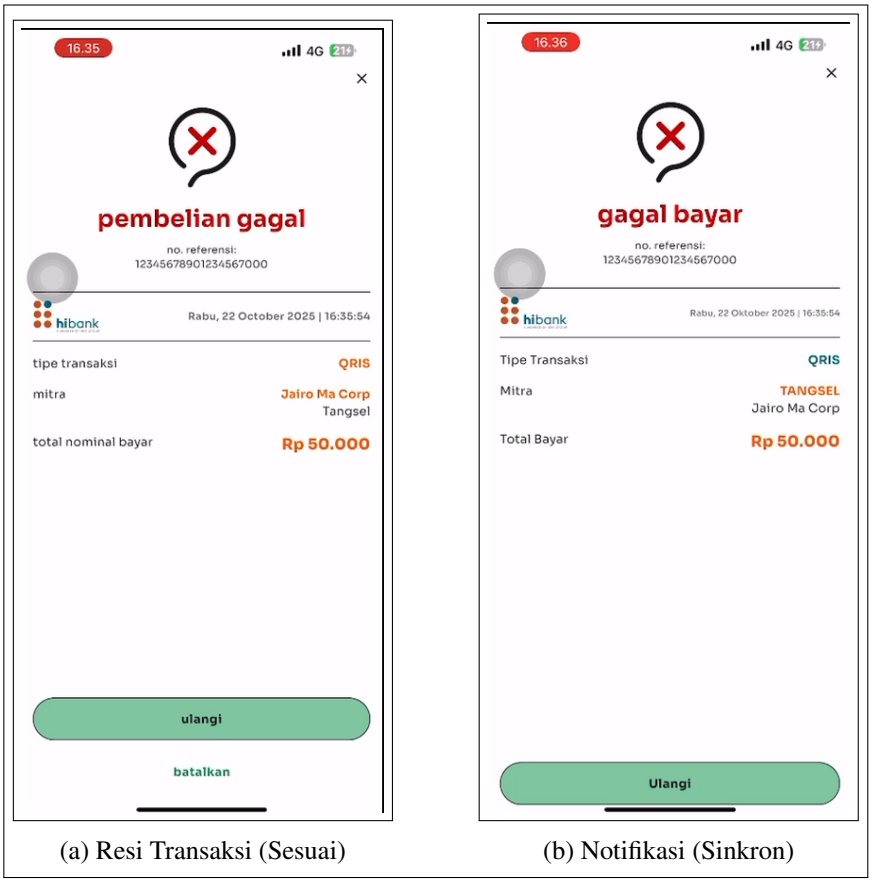
C.3 Langkah Perbaikan

Perbaikan dilakukan dengan menyelaraskan alur distribusi data antar-servis melalui langkah berikut.

1. Memperbaiki logika *Data Mapper* pada modul QRIS untuk menjamin *Reference Number* selalu terisi pada setiap respons sukses, terlepas dari jenis sumber dana yang digunakan.
2. Mengimplementasikan validasi silang untuk memastikan nilai *Reference Number* yang dikirimkan ke layanan notifikasi (*Push Notification*) identik dengan data yang tercatat di basis data transaksi utama.

C.4 Hasil

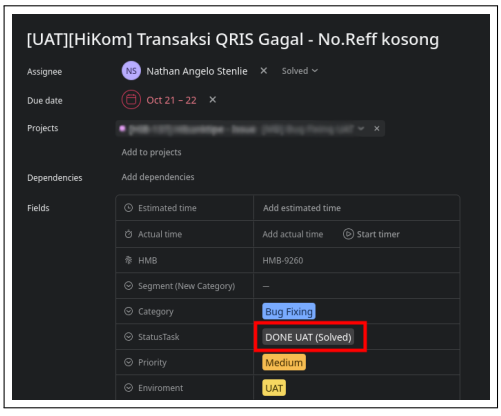
Pasca implementasi perbaikan, verifikasi dilakukan pada kedua halaman terkait. Gambar 3.51a menunjukkan bahwa Halaman Resi Transaksi kini telah menampilkan Nomor Referensi secara lengkap. Selain itu, sinkronisasi data juga telah berhasil diperbaiki sebagaimana terlihat pada Gambar 3.51b, di mana Halaman Detail Notifikasi kini menampilkan informasi referensi yang konsisten dan identik dengan data pada resi. Status perbaikan ini juga telah dikonfirmasi selesai melalui tiket manajemen proyek pada Gambar 3.52.



(a) Resi Transaksi (Sesuai) (b) Notifikasi (Sinkron)

Gambar 3.51. Tampilan Resi dan Notifikasi Pasca Perbaikan

Sumber Asana



Gambar 3.52. Bukti Tiket Selesai (*Resolved*)

Sumber Asana

3.3.3 Pengembangan Kode *Unit Test*

Pengujian unit (*Unit Testing*) dikembangkan menggunakan kerangka kerja *JUnit* dan *Mockito*. Fokus utama pengujian adalah menjamin kestabilan sistem melalui dua pendekatan skenario pengujian.

1. Skenario positif atau *positive case* bertujuan menguji alur proses utama menggunakan *input* data yang valid untuk memastikan sistem menghasilkan status sukses sesuai logika bisnis, contohnya saat pengguna berhasil melakukan transfer dana dengan saldo yang mencukupi.

2. Skenario negatif atau *negative case* bertujuan menguji ketahanan sistem terhadap *input* tidak valid atau kondisi kegagalan eksternal untuk memastikan penanganan *error* berjalan tepat, contohnya saat sistem menangani data bernilai *null* yang dikembalikan oleh *repository* tanpa menyebabkan aplikasi berhenti bekerja.

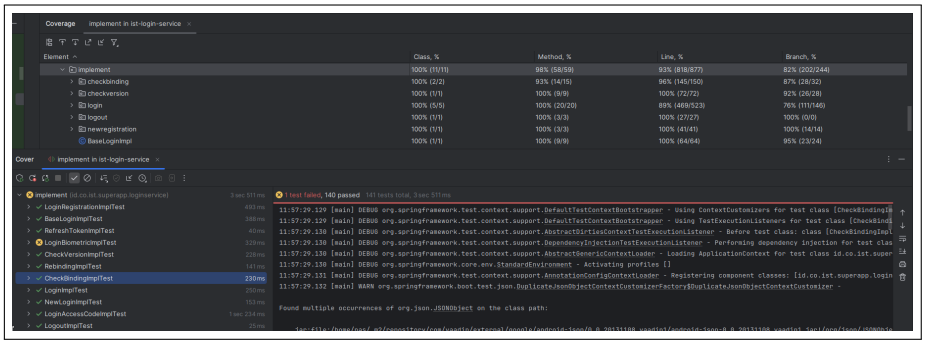
A Capaian *Code Coverage*

Berdasarkan hasil eksekusi pengujian, implementasi *unit test* berhasil mencakup rata-rata lebih dari 90% logika bisnis pada modul-modul krusial. Angka ini memastikan bahwa mayoritas alur kode, termasuk skenario positif dan negatif (*edge cases*), telah terverifikasi secara otomatis.

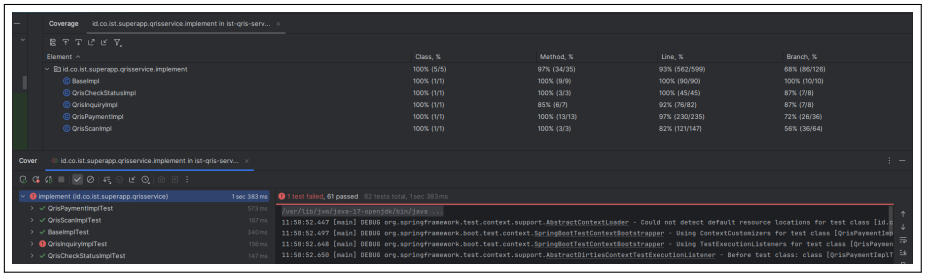
Rincian capaian coverage per modul dirangkum pada Tabel 3.2, sedangkan bukti visual hasil eksekusi pengujian dapat dilihat secara rinci pada Gambar 3.53 dan Gambar 3.54.

Tabel 3.2. Rekapitulasi *Code Coverage* per Modul *Service*

Nama *Service*	***Class Coverage***	***Method Coverage***	***Line Coverage***
Login Service	100%	98%	93%
QRIS Service	100%	97%	93%



Gambar 3.53. Hasil Eksekusi *Unit Test* dan *Coverage* pada *Login Service*
Sumber Intelij



Gambar 3.54. Hasil Eksekusi *Unit Test* dan *Coverage* pada *QRIS Service*
Sumber Intelij

B Efektivitas Deteksi *Bug*

Selain mencapai coverage tinggi, rangkaian *unit test* ini terbukti efektif dalam menjaga kualitas kode dengan mendeteksi anomali logika pada kode warisan (*existing code*) sebelum dirilis.

Dua temuan krusial yang berhasil diidentifikasi dan diperbaiki meliputi.

1. *Unit test* mendeteksi potensi `ArithmeticException (/ by zero)` pada perhitungan biaya admin. Isu ini disebabkan oleh inisiasi variabel pembagi bernilai 0 pada kode sumber.

2. *Unit test* mengidentifikasi kesalahan urutan parameter (*parameter swapping*) pada respon *error*, yang berpotensi menyebabkan pesan kesalahan tidak terbaca oleh *front-end*.

3.3.4 Kendala dan Solusi

Selama pelaksanaan aktivitas magang di PT Infosys Solusi Terpadu, khususnya dalam lingkup pengembangan layanan *backend* Hibank, ditemukan beberapa tantangan teknis dan dinamika proyek yang memengaruhi ritme pengembangan. Berikut adalah rincian kendala yang dihadapi beserta strategi penyelesaian yang diterapkan.

1. Luasnya cakupan fungsionalitas pengembangan fitur pada modul Hi-Komunitas, yang memerlukan pemetaan logika mendalam untuk memastikan keterkaitan antar fitur berjalan sesuai spesifikasi.
2. Ketergantungan sistem pada layanan *middleware Service Orchestration API* (SOA) milik pihak klien yang sering mengalami gangguan operasional (*downtime*), sehingga menghambat proses validasi fitur (*SIT/UAT*) dan verifikasi perbaikan *bug*.
3. Adanya perbedaan antara rancangan desain awal pada Figma dengan kebutuhan implementasi teknis akibat perubahan dinamis dari sisi bisnis klien atau penyesuaian efisiensi sistem secara mendadak.

Guna menjaga stabilitas *timeline* proyek dan memastikan kualitas luaran tetap optimal di tengah tantangan tersebut, diterapkan langkah-langkah mitigasi strategis sebagai berikut.

1. Memanfaatkan *Daily Standup Meeting* untuk melakukan konfirmasi kebutuhan secara berkala serta meminta arahan teknis kepada *supervisor*, guna meminimalisir kesalahan interpretasi logika.
2. Melakukan eskalasi isu terkait layanan SOA kepada pihak terkait dan mengalihkan fokus kerja sementara pada pengembangan komponen independen (seperti *Unit Testing* dan dokumentasi teknis) untuk menjaga produktivitas selama masa pemulihan sistem.
3. Menerapkan manajemen waktu yang adaptif dengan menyusun ulang prioritas pengerjaan fitur. Hal ini dilakukan untuk mengakomodasi perubahan permintaan klien tanpa mengorbankan kualitas kode maupun tenggat waktu penyelesaian fitur utama.