Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Perpustakaan Univeristas Multimedia Nusantara

BAB 3

PELAKSANAAN KERJA MAGANG

3.1. Kedudukan dan Organisasi

Kerja magang dilakukan pada divisi *frontend engineer* yang langsung disupervisi oleh Albert S. Darmali yang berlaku sebagai Project Manager. Dalam *meeting*, Project Manager akan memberi tahu tim tentang modul yang akan dibuat serta membuat *timeline* yang sesuai dengan modul tersebut. Setelah *timeline* sudah dibuat, tim UI/UX akan membuat tampilan yang sesuai dengan permintaan dan divisi *backend engineer* akan mulai untuk membuat API. Setelah desain sudah ada, divisi *frontend* akan mengimplementasi tampilan tersebut, dan jika divisi *backend* sudah selesai membuat API maka divisi *frontend* akan mengimplementasi API. Setelah modul selesai, maka aplikasi diberikan ke QA untuk dilakukan uji coba. Jika aplikasi memiliki *bug*, maka akan dilakukan perbaikan. Proses ini akan beriterasi sampai tidak ketemunya *bug* lain sebelum di-*update* di Playstore.

3.2. Tugas yang Dilakukan

Selama periode magang, tugas yang diberikan sebagai *frontend engineer* lebihnya adalah kustomisasi pada aplikasi LINIPOIN. LINIPOIN adalah aplikasi di mana *user* dapat mengumpulkan poin untuk mendapatkan berbagai hadiah dan bertransaksi menggunakan poin tersebut. Kustomisasi terhadap aplikasi LINIPOIN dibagi menjadi tiga bagian dalam periode magang ini yaitu:

- Perubahan LINIBILLS
- Perbaikan *penetration testing*

- Penambahan fitur baru SIIIP

LINIBILLS adalah fitur di LINIPOIN di mana *user* dapat bertransaksi seperti membayar listrik, beli pulsa dan beli *voucher game*. Perubahan LINIBILLS yang dilakukan adalah menambah *field* baru untuk ditampilkan serta *error handling* untuk pembelian token listrik prabayar dan pascabayar.

Penetration testing adalah pengujian untuk sebuah sistem komputer untuk mencari kerentanan keamanan yang dapat diancam oleh *attacker.* Perbaikan *penetration testing* yang dilakukan adalah lanjutan dari *penetration testing* yang sudah dilakukan oleh *frontend engineer* serta *backened engineer* lainnya sebelum periode magang. Dikarenakan perbaikan ini hanya merupakan lanjutan, di bagian magang ini dilakukan *refractoring* dan *cleansing code*. Proses ini melibatkan perubahan API serta *merging code* dari *branch penetration testing* ke aplikasi yang telah di-*upload* di playstore.

SIIIP adalah pelayanan asuransi di Indonesia yang melakukan penjualan secara *online* melalui perangkat lunak seperti LINIPOIN. Penambahan fitur SIIIP melibatkan perubahan di halaman *home* serta halaman SIIIP. Fitur tersebut telah dikembangkan oleh tim berbeda sehingga tim LINIPOIN hanya menampilkan menu di halaman *home* dan menampilkan *website* SIIIP menggunakan browser dalam aplikasi atau menggunakan iframe tergantung *routing* dari *backend*.

Kerja magang dilaksanakan selama 3 (tiga) bulan dengan *timeline* kerja sebagai berikut.

Tabel 3.1 Jadwal kerja magang

Nama Kegiatan	Minggu													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Pengenalan *framework* dan alur kerja														
Update LINIBILLS														
Testing dan Bug Fixing														
Refactoring/Cleansing Code														
Testing dan Bug Fixing														
Fitur Baru: LINIGUARD/SIIIP														
Testing dan Bug Fixing														
Penetration test														
Testing dan Bug Fixing														
Merging														

Dalam Tabel 3.1, *highlight* dengan warna kuning adalah indikator pengerjaan tim lainnya yang berada di Codify. Untuk setiap modul ada *testing* oleh QA dan *bug fixing* oleh tim *frontend* dan *backend* LINIPOIN.

3.3. Uraian Pelaksanaan Magang

Untuk setiap bagian kerja magang, dilakukan rincian rencana target pengerjaan yang dijadikan sebagai *progress tracker* kerja harian. Berikut adalah *framework* yang digunakan Codify untuk membuat sebuah proyek:

a. UI Design: Zeplin

b. Frontend: Ionic dikombinasi dengan AngularJS

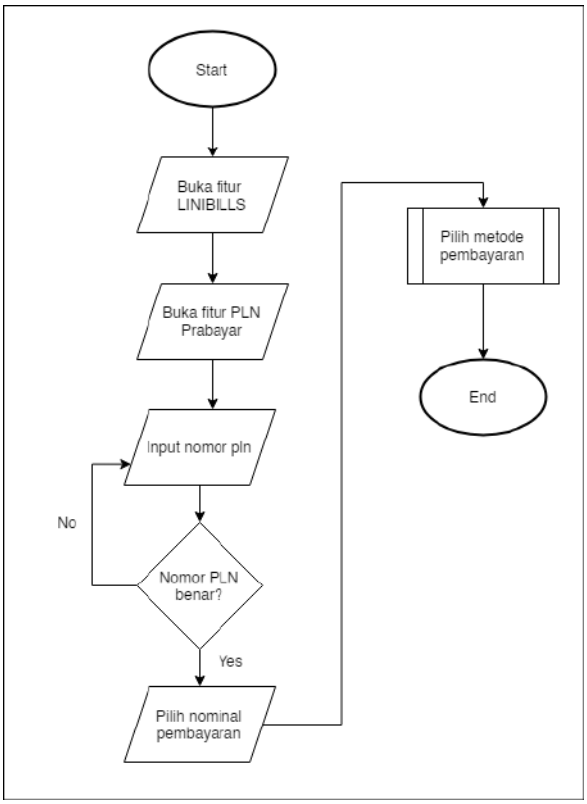
c. Backend: ExpressJS dan PostgreSQL

3.3.1. LINIBILLS

LINIBILLS adalah fitur dari aplikasi LINIPOIN untuk bertransaksi seperti bayar listrik, beli pulsa, dan beli *voucher game*. Atas permintaan *third-party* LINIPOIN yaitu BILLFAZZ untuk menambahkan beberapa hal dalam pembayaran listrik, pekerjaan magang ini fokus pada bagian bayar listrik (PLN prabayar dan pascabayar).

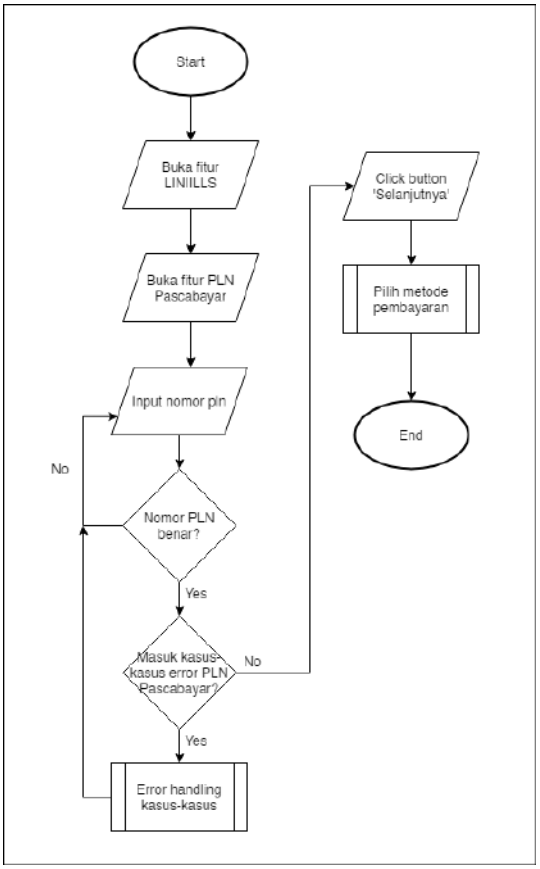
A. Flowchart

Sebagai acuan, Gambar 3.1 dan Gambar 3.2 adalah cara untuk membeli PLN Prabayar serta PLN Pascabayar LINIBILLS:



Gambar 3.1 Alur fitur PLN prabayar
(Sumber: Draf kasar meeting)

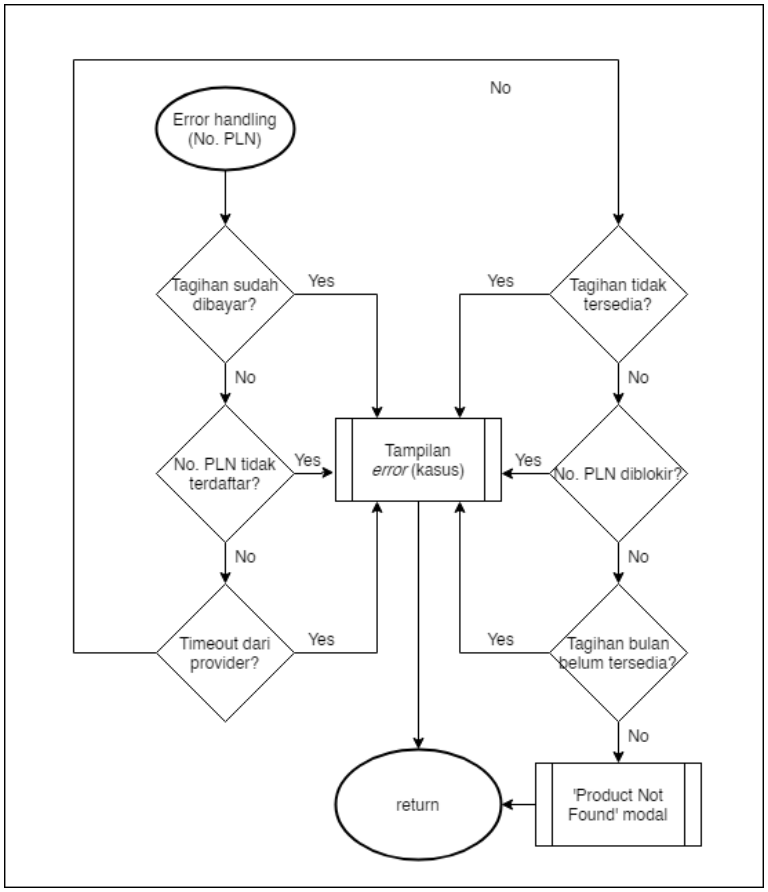
Gambar 3.1 adalah alur fitur PLN prabayar di mana setelah *user* telah buka LINIBILLS dan memilih fitur PLN Prabayar, maka *user* akan diminta *input* nomor pelanggan PLN. Jika nomor pelanggan salah, akan ada *error message* sehingga tidak bisa melanjutkan ke pembayaran. Jika nomor pelanggan benar, maka *user* dapat pilih nominal pembayaran. Setelah klik nominal, maka *user* akan dialihkan ke halaman metode pembayaran di mana *user* dapat memilih untuk menggunakan poin yang telah dikumpulkan di aplikasi, OVO, Gopay, LINIPAY, atau *split payment* untuk melakukan pembayaran. *Split payment* adalah metode pembayaran jika poin *user* tidak mencukupi total nominal, sehingga harus menggunakan uang digital untuk melakukan pembayaran. Setelah memilih metode pembayaran maka *user* akan mendapatkan resi dengan status sukses, gagal, atau *pending*.



Gambar 3.2 Alur fitur PLN pascabayar
(Sumber: Draf kasar meeting)

Gambar 3.2 menjelaskan alur fitur PLN Pascabayar di mana setelah *user* membuka LINIBILLS dan memilih fitur PLN Pascabayar, *user* akan diminta input nomor pelanggan PLN. Jika nomor pelanggan salah, maka akan ditampilkan *error message* sehingga *user* harus memasuki ulang nomor pelanggannya. Jika nomor pelanggan benar tetapi ada masalah dengan kasus-kasus yang diberi oleh BILLFAZZ, maka proses akan masuk ke fungsi *error handling*. Kasus-kasus tersebut akan dijelaskan di Gambar 3.3. Jika nomor pelanggan benar dan tidak ada masalah, maka akan memasuki ke halaman metode pembayaran dengan 3 kasus yang menunjukkan tagihan untuk 3 bulan, 4 bulan, atau 5 bulan. Proses metode pembayaran sama dengan PLN Prabayar dan jika sudah memilih metode maka *user* akan mendapatkan resi dengan status sukses, gagal, atau *pending*.

Dapat dilihat di Gambar 3.2 bahwa PLN Pascabayar mempunyai *error handling* untuk kasus-kasus tertentu. Alur *error handling* ini dapat dilihat di Gambar 3.3. Kasus tersebut diberi oleh BILLFAZZ dan terdapat 6 kondisi gagal. Berikut adalah *flowchart* untuk penjelasan *error handling* serta kasus-kasus gagal PLN Pascabayar.

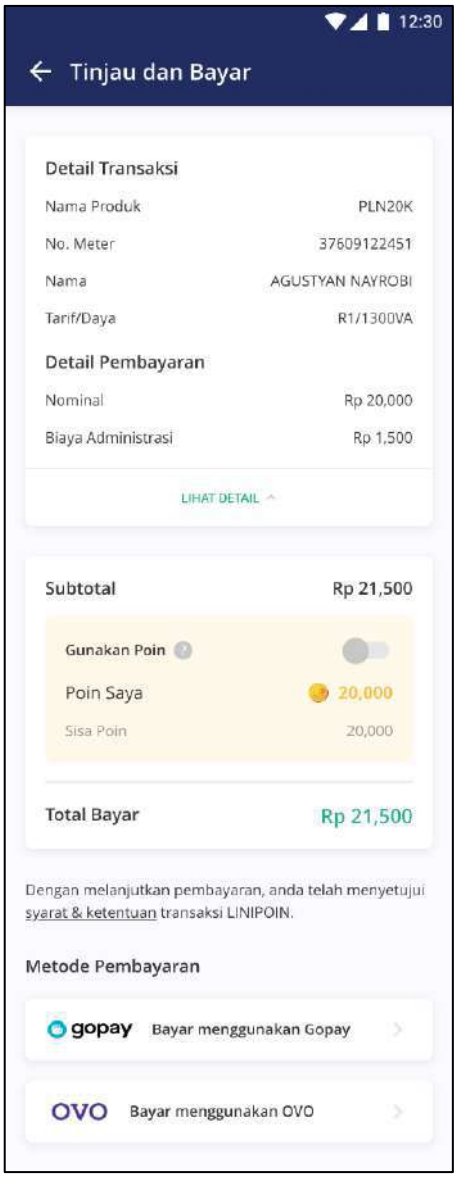


Gambar 3.3 Alur kerja *error handling* PLN pascabayar
(Sumber: Draf kasar meeting)

Dapat dilihat dari Gambar 3.3 bahwa ada 6 kasus yang diberi oleh BILLFAZZ yaitu apakah tagihan sudah dibayar, nomor pelanggan PLN tidak terdaftar, tagihan tidak tersedia, nomor pelanggan PLN terblokir, tagihan bulan belum tersedia atau adanya *timeout* dari pihak BILLFAZZ. Setiap kasusnya memiliki tampilan dengan *error message* yang diberikan oleh BILLFAZZ untuk masing-masing kasus. Jika *error* yang ditemukan tidak termasuk dari kasus-kasus yang diberi oleh BILLFAZZ, maka akan ditampilkan modal ‘Product Not Found’.

B. Desain

Berikut adalah desain UI/UX yang diberikan kepada bagian *frontend engineer* sebagai panduan tampilan PLN prabayar.



Gambar 3.4 Rancangan halaman pembayaran PLN prabayar LINIBILLS
(Sumber: Dokumentasi internal)

Gambar 3.4 merupakan tampilan setelah user telah memasuki nomor pelanggan PLN. Dalam pekerjaan magang ini diminta untuk mengganti field Tarif/Daya serta Biaya Administrasi sehingga *value* yang didapatkan adalah *value* dari *backend*.

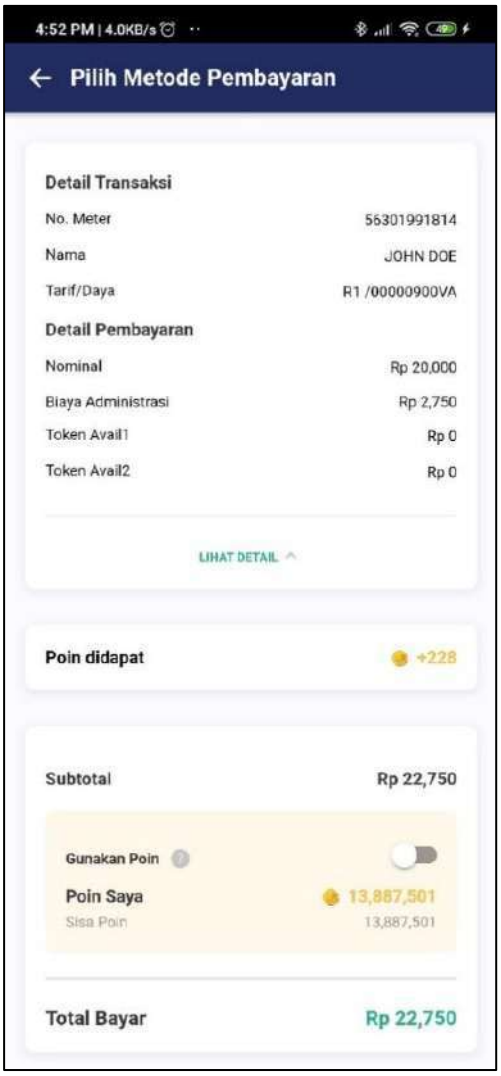
Gambar 3.5 Rancangan halaman resi PLN LINIBILLS
(Sumber: Dokumentasi Internal)

Gambar 3.5 merupakan tampilan setelah berhasil membeli token PLN prabayar maupun pascabayar. Status transaksi dapat berhasil, gagal, atau *pending*. Dalam pekerjaan magang ini diminta untuk menambah field Meterai, PPn, PPJ, Angsuran, serta Admin Bank.

C. Implementasi

Ada dua bagian besar yang dilakukan dalam PLN LINIBILLS yaitu menambahkan *field* baru dalam halaman pembayaran dan resi, serta menambahkan tampilan yang sesuai berdasarkan *error code* yang ditemukan.

Tampilan halaman pembayaran dan resi yang baru memiliki sedikit perbedaan dari desain yang diberikan dikarenakan adanya permintaan dari *product manager* dan BILLFAZZ. Berikut adalah tampilan untuk halaman pembayaran dan resi PLN prabayar LINIBLLS.



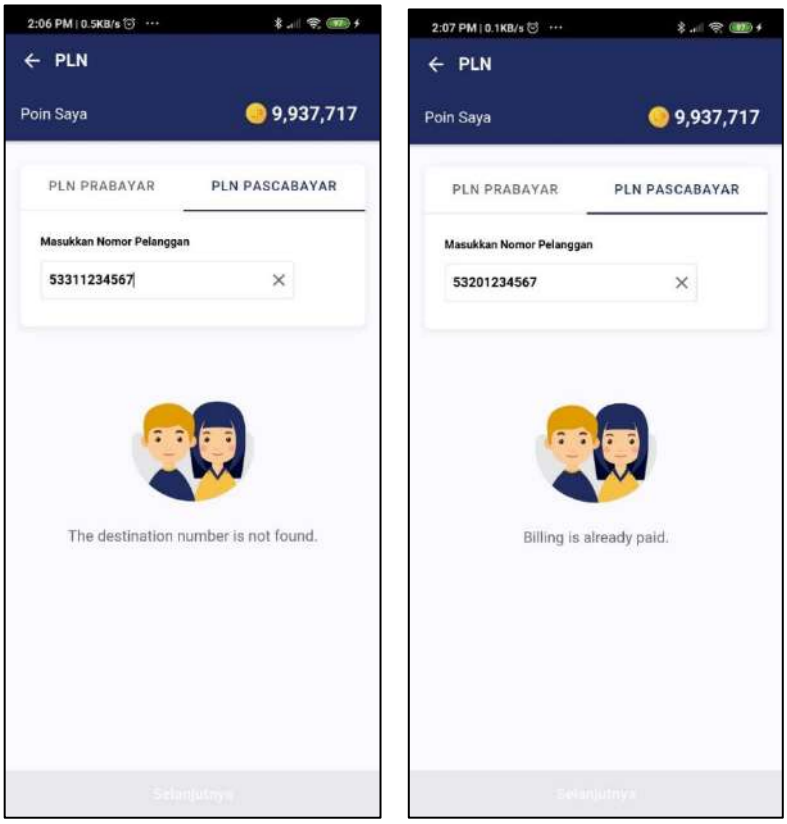
Gambar 3.6 Halaman pembayaran PLN prabayar LINIBILLS
(Sumber: Aplikasi LINIPOIN)

Gambar 3.6 adalah hasil tampilan halaman pembayaran PLN prabayar yang telah berhasil diimplementasikan sesuai desain. Hal yang ditambah sesuai permintaan Product Manager dan BILLFAZZ selain yang dijabarkan di bagian desain adalah Token Avail1 dan Token Avail2. Ada perubahan juga di kalkulasi bagian sub total di mana biaya admin sekarang diambil dari *backend* sehingga biaya admin tersebut bersifat dinamis.

Gambar 3.7 Halaman resi PLN prabayar LINIBILLS
(Sumber: Aplikasi LINIPOIN)

Gambar 3.7 adalah hasil tampilan jika transaksi PLN prabayar sukses. Tampilan juga telah sukes diimplementasi dari desain tetapi jika dibandingkan Gambar 3.5 dengan Gambar 3.7, *field* Meterai diganti dengan Materai dan Stroom/Token diganti untuk cara menampilkannya. Adanya juga perubahan dengan API sehingga Biaya Administrasi dan Rp Bayar diambil nominalnya dari *backend*.

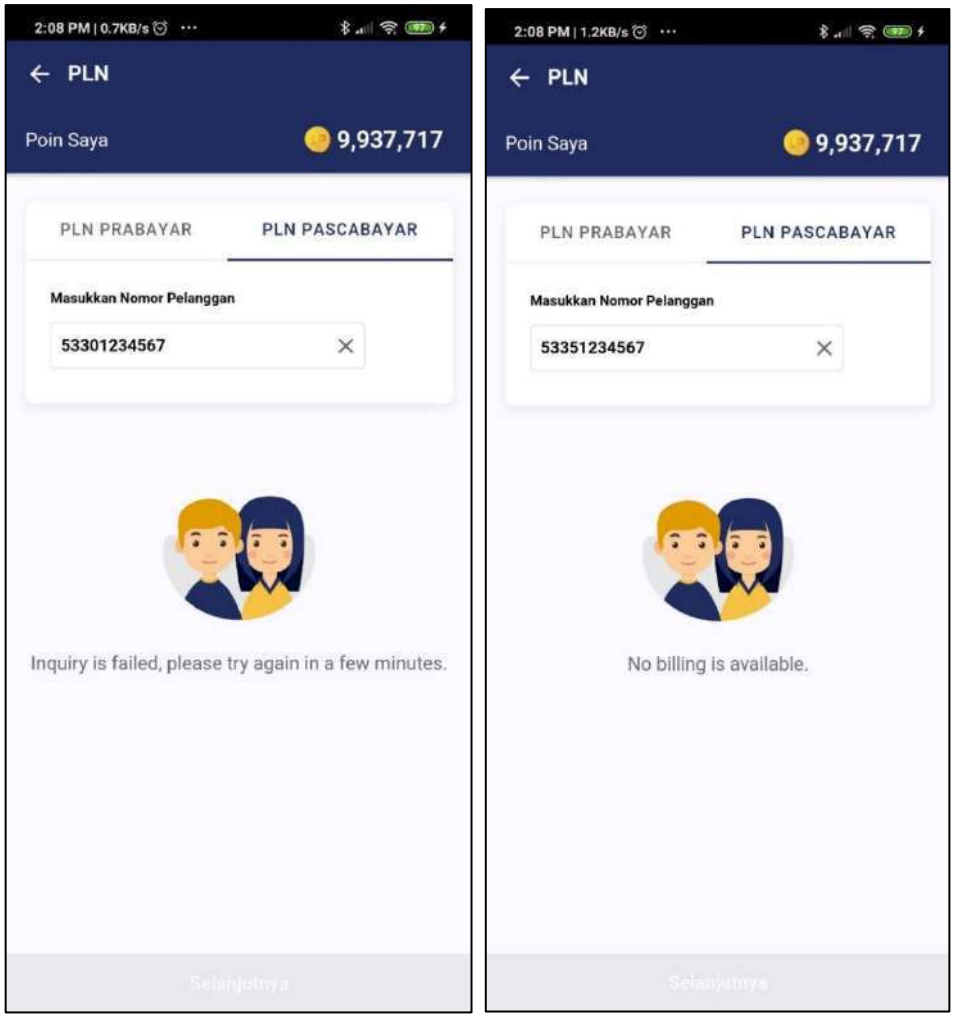
Berikut adalah tampilan untuk kasus-kasus *error* PLN pascabayar LINIBILLS yang telah dijabarkan. Nomor pelanggan yang dimasukkan merupakan nomor dari *development* sehingga dapat kasus-kasus tersebut. Teks untuk seluruh *error code* diambil dari API sehingga ada perbedaan dalam bahasa.



(a) (b)

Gambar 3.8 Tampilan *error* jika nomor tujuan tidak ditemukan dan tagihan telah dibayar
(Sumber: Aplikasi LINIPOIN)

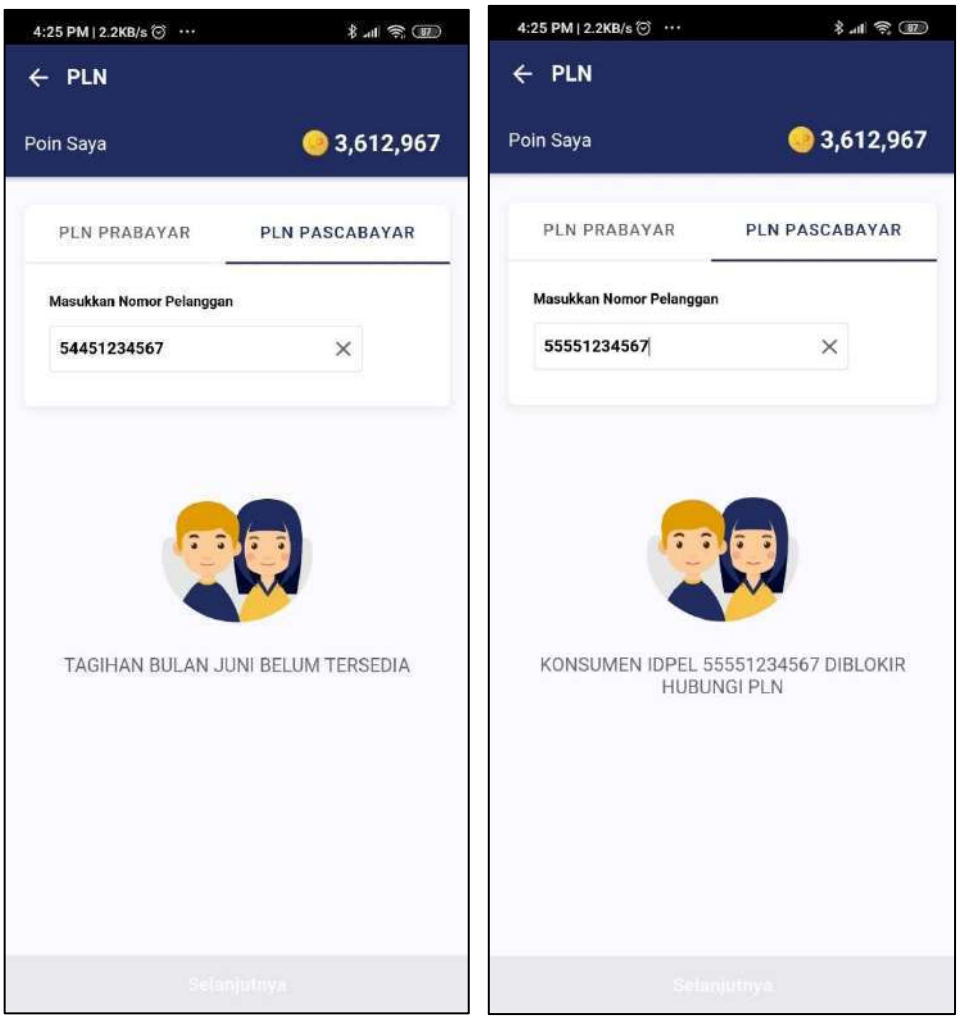
Gambar 3.8 (a) adalah tampilan *error* jika nomor pelanggan tidak ditemukan dan Gambar 3.8 adalah tampilan *error* jika tagihan untuk nomor pelanggan telah dibayar.



(a) (b)

Gambar 3.9 Tampilan *error* jika permintaan gagal dan tidak ada tagihan tersedia (Sumber: Aplikasi LINIPOIN)

Gambar 3.9 (a) adalah tampilan *error* jika mendapat *timeout* dari pihak BILLFAZZ dan Gambar 3.9 (b) adalah tampilan *error* jika tagihan tidak tersedia.



(a) (b)

Gambar 3.10 Tampilan *error* jika tagihan bulan belum tersedia dan nomor pelanggan telah diblokir
(Sumber: Aplikasi LINIPOIN)

Gambar 3.10 (a) adalah tampilan *error* jika tagihan untuk bulan belum tersedia dan Gambar 3.10 (b) adalah tampilan *error* jika nomor pelanggan telah diblokir.

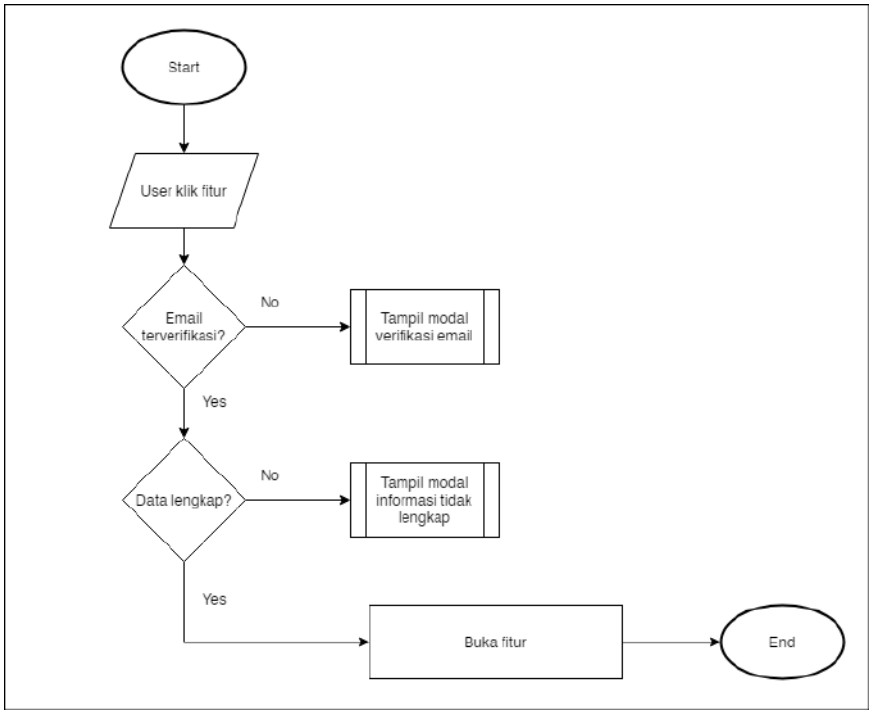
3.3.2. Penetration Testing

Perbaikan *penetration test* selama pekerjaan magang lebih banyak ke arah *backend*, sehingga *frontend* melakukan *refactoring* serta *cleansing code*. Pada pengerjaan tersebut ada perubahan besar dalam cara memanggil API *service*, sehingga membutuhkan waktu yang lama dan harus menyesuaikan cara pemanggilan API *service* tersebut. Selain itu ditambahkan juga *counter ban* untuk *rate limitiation* pada *login* dan *sign up.* Setelah memperbaiki *penetration test* yang baru, dilakukan *testing* oleh pihak Veda Praxis selama beberapa hari dan dari hasil yang diterima dilakukan *bug fixing*.

A. Flowchart

Salah satu *refactoring* yang dilakukan adalah terhadap modal konfirmasi, di mana modal tersebut akan muncul jika *user* klik sebuah fitur tetapi *user* tersebut masih belum memverifikasi e-mail atau melengkapi data akun. Modal tersebut adalah *global modal*, sehingga dalam satu komponen dapat digunakan untuk beberapa modal.

Untuk membuka fitur-fitur tertentu, diperlukan *user* untuk verifikasi e-mail serta melengkapi data pribadi. Gambar 3.4 adalah kasus untuk menampilkan modal jika *user* belum verifikasi email atau belum melengkapi data.

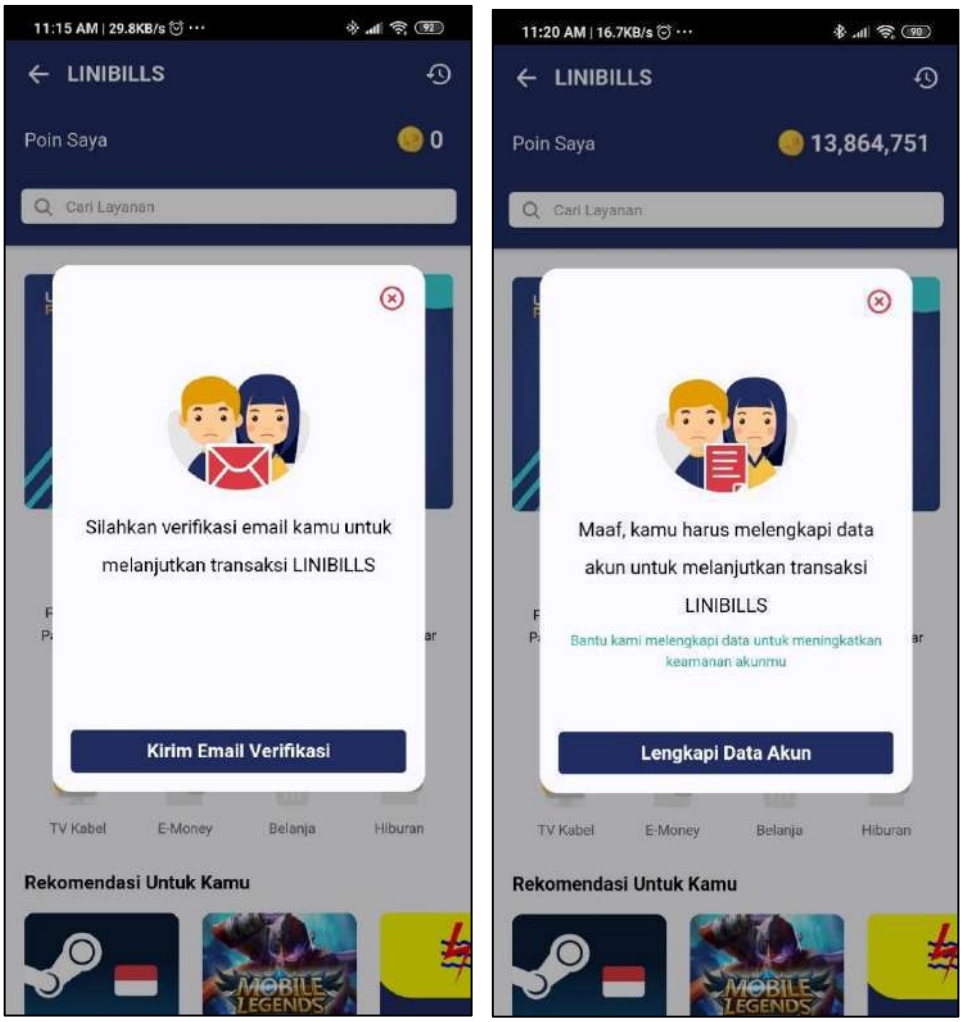


Gambar 3.11 Alur konfirmasi modal e-mail verifikasi dan lengkapi data akun (Sumber: Draf kasar meeting)

Dari Gambar 3.11, setelah *user* telah klik fitur yang memerlukan pengecekan email verifikasi dan data akun yang lengkap, yang akan dicek terlebih dahulu adalah verifikasi e-mail. Jika e-mail telah diverifikasi, program akan lanjut untuk mengecek jika data akun telah lengkap. Masing-masing kasus ini memiliki modal sendiri. Jika *user* telah verifikasi e-mail dan melengkapi data akun, maka fitur dapat dipakai atau dibuka.

B. Implementasi

Berikut adalah tampilan untuk modal konfirmasi verifikasi e-mail dan melengkapi data. Fitur yang dipilih untuk membuka modal tersebut adalah LINIBILLS.



(a) (b)

Gambar 3.12 Tampilan konfirmasi modal untuk verifikasi e-mail dan lengkapi data akun
(Sumber: Aplikasi LINIPOIN)

Gambar 3.12 (a) adalah tampilan modal jika *user* belum memverifikasi e-mail. Gambar 3.12 (b) adalah tampilan jika *user* belum melengkapi data akun Untuk kasus ini, setelah user memilih fitur LINIBILLS dan memilih satu fitur, aplikasi akan cek jika user telah melakukan email verifikasi dan melengkapi data akun seperti yang telah dijelaskan di Gambar 3.11. Jika klik *button* 'Kirim Email Verifikasi' maka *user* akan mendapat e-mail untuk mengkonfirmasi e-mail. Jika

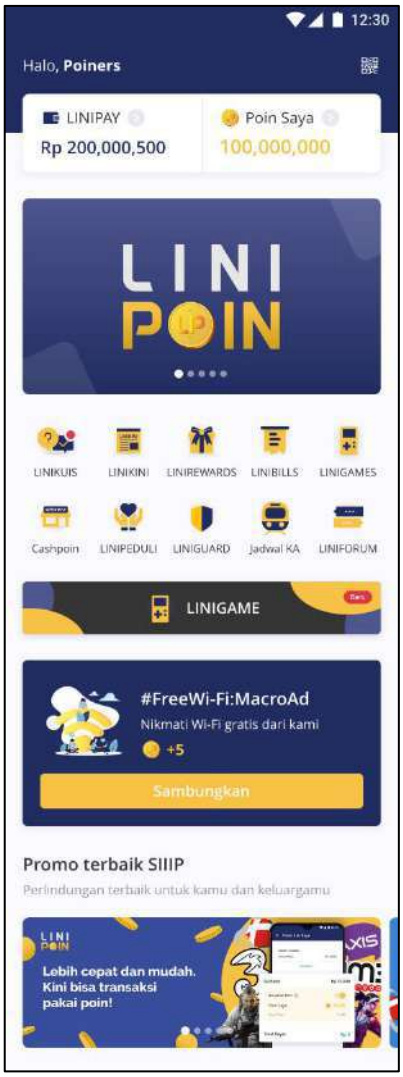
klik ‘Lengkapi Data Akun’ maka *user* akan dialihkan ke halaman profil di mana *user* harus melengkapi data akun.

3.3.3. SIIIP

SIIIP adalah pelayanan asuransi di Indonesia yang melakukan penjualan secara *online* melalui perangkat lunak seperti LINIPOIN. Dalam pekerjaan magang, dilakukan penambahan tampilan di halaman *home* serta *routing* untuk menampilkan *website* SIIIP.

A. Desain

Berikut adalah desain UI/UX yang diberikan kepada bagian *frontend engineer* sebagai panduan tampilan untuk bagian SIIIP.

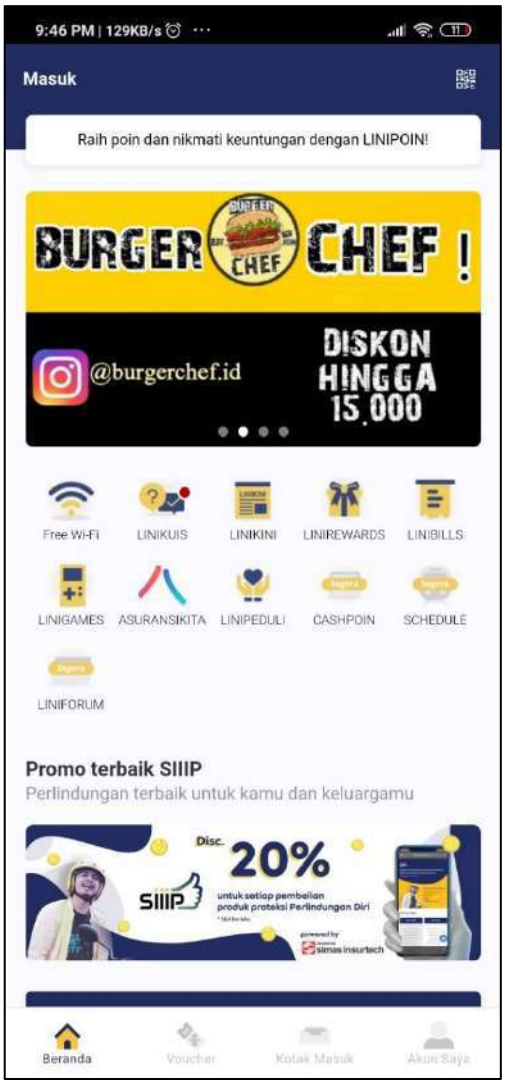


Gambar 3.13 Rancangan halaman *home* dengan fitur baru SIIIP
(Sumber: Dokumentasi Internal)

Gambar 3.13 adalah hasil tampilan halaman *home* dengan bagian SIIIP yang paling baru. Tampilan tersebut menggunakan *slider banner* di mana *banner* tersebut mempunyai *routing* dari *backend* yang akan membuka browser di dalam aplikasi atau akan dialihkan ke halaman SIIIP yang menggunakan iframe. Halaman SIIIP dapat tampilan *custom header text* dan *header color* yang dapat diatur oleh admin, tetapi halaman tersebut tidak ada desain dikarenakan hanya mempunyai *header* serta iframe.

B. Implementasi

Berikut adalah hasil implementasi untuk tampilan *home* dengan fitur SIIIP serta halaman SIIIP dengan *header* yang diatur oleh *backend*.



Gambar 3.14 Halaman *home* dengan bagian SIIIP baru
(Sumber: Aplikasi LINIPOIN)

Dari Gambar 3.14, jika user klik *banner* yang berada di bagian 'Promo terbaik SIIIP', maka *user* akan dialihkan ke browser dalam aplikasi atau ke halaman SIIIP

dengan iframe tergantung *routing* yang diberi oleh *backend*. Judul dan sub judul juga bersifat dinamis sehingga bisa diatur dari *backend*.

Gambar 3.15 *Custom* teks *header* dan warna *header* SIIIP
(Sumber: Aplikasi LINIPOIN)

Gambar 3.15 adalah tampilan jika *banner* mengalihkan ke halaman SIIIP menggunakan iframe. Selain bagian *header,* tampilan diambil dari https://siiip.linipoin.com menggunakan iframe. Fitur SIIIP ini merupakan fitur yang dinamis sehingga nama fitur dapat berganti (dalam kasus ini menjadi AsuransiKita) dan warna *header* juga dapat berganti (dalam kasus ini mengikuti warna *header* di *website* SIIIP).

3.4. Kendala dan Solusi yang Ditemukan

Selama proses magang, tidak terlalu ada kesulitan, tetapi sering kali dapat kendala seperti berikut:

- Informasi yang berganti dengan mendadak sehingga pekerja harus melakukan lebih banyak pekerjaan.
- Komunikasi pada bagian *backend* terkadang ke-*double* karena tim *frontend* sering kali mengirim pesan pribadi kepada dia. Informasi yang dikirim oleh *backend* mungkin sampai ke saya tetapi tidak ke anggota *frontend* yang lainnya.

Upaya yang telah dilakukan untuk mengatasi kendala yang telah dijabarkan adalah sebagai berikut:

- Mengkonfirmasi ulang apa saja pekerjaan yang harus dilakukan sehingga tidak ada kesalahan dalam komunikasi.
- Tim *frontend* dan *backend* membuat *group chat* di Slack.

3.5. Finalisasi Proyek

Setiap hari magang diperlukan untuk mengisi *progress tracker* yang disediakan oleh Bapak Albert S. Darmali. Media yang digunakan untuk mengisi *progress tracker* tersebut adalah sebagai berikut:

- Google Spreadsheets, yaitu tabel laporan yang dibuat untuk seluruh bawahan Bapak Albert sehingga dapat mengerti apa saja yang dilakukan setiap orang di tim tertentu.

- Zeplin, Trello, dan Slack adalah media yang digunakan untuk berkomunikasi dengan satu sama lain serta menginformasikan apa yang seharusnya, sedang, dan telah dilakukan untuk sebuah proyek.