BIDANG ILMU: 538/REKAYASA PERANGKAT LUNAK

LAPORAN AKHIR

JUDUL:
IMPLEMENTASI ALGORITMA TIME-BASED ONE TIME PASSWORD DALAM OTENTIKASI TOKEN INTERNET BANKING

TIM PENGUSUL :
Uung Ungkawa, M.T (NIDN: 0411105902)
Kurnia Ramadhan Putra. (NRP: 15-2009-070)

Dibiayai oleh **Pribadi**

INSTITUT TEKNOLOGI NASIONAL
Januari 2017

Halaman Pengesahan

Judul Penelitian	:	IMPLEMENTASI ALGORITMA TIME-BASED ONE TIME PASSWORD DALAM OTENTIKASI TOKEN INTERNET BANKING
Kode/ Nama Rumpun Ilmu	:	538/ Rekayasa Perangkat Lunak
Ketua Peneliti	:	
a. Nama Lengkap	:	Ir Uung Ungkawa, M.T.
b. NIDN	:	0411105902
c. Jabatan Fungsional	:	Lektor
d. Program Studi	:	Teknik Informatika
e. Nomor HP	:	08121443095
f. Alamat surel (e-mail)	:	uung@itenas.ac.id
Anggota Peneliti (1)		
a. Nama Lengkap	:	Kurnia Ramadhan Putra
b. NRP	:	15-2009-070
c. Instansi	:	Mahasiswa Institut Teknologi Nasional (ITENAS) Bandung
Anggota Peneliti (2)		
a. Nama Lengkap	:	
b. NIDN	:	
c. Perguruan Tinggi	:	
Lama Penelitian Keseluruhan	:	6 bulan
Penelitian Tahun ke	:	1 (Satu)
Biaya Penelitian Keseluruhan	:	Rp. 8.000.000,-
aya Tahun Berjalan	:	- diusulkan ke DIKTI Rp. 0,- - dana internal PT Rp.0 - dana institusi lain Rp.0 - inkind sebutkan

Bandung, Januari 2017

Menyetuji
Ketua LPPM Itenas

Dr. Tarsisius Kristyadi, S.T., M.T
NIP/NIK

Ir Uung Ungkawa, M.T
NIP/NIK. 12007120 2017

IMPLEMENTASI ALGORITMA TIME-BASED ONE TIME PASSWORD DALAM OTENTIKASI TOKEN INTERNET BANKING

Kurnia Ramadhan Putra, Uung Ungkawa, Irma Amelia Dewi

Jurusan Teknik Informatika

Fakultas Teknologi Industri

Institut Teknologi Nasional

INTISARI

Algoritma *Time-Based One Time Password* (TOTP) adalah salah satu algoritma yang memiliki kemampuan untuk menghasilkan *password* sekali pemakaian. *Password* yang dihasilkan oleh algoritma TOTP memiliki masa berlaku yang terbatas dan selalu berubah dalam periode tertentu. Cara kerja algoritma TOTP yaitu menggabungkan antara *secret key* dengan *current time* kemudian dilakukan *hashing* menggunakan algoritma enkripsi *SHA256*. Algoritma TOTP merupakan implementasi dari *Two Factor Authentication* (2FA) yaitu penggabungan dua metode otentikasi berbeda secara bersamaan. Otentikasi yang digabungkan adalah *password* statis sebagai akses login dan token *device* yang dibangun secara *virtual* sehingga dapat diaplikasikan pada *smartphone android*. Pada penelitian ini 2FA diaplikasikan pada sistem *internet banking* dimana antara token *virtual* dan *server* dipasang algoritma TOTP untuk menghasilkan *password* sebagai otentikasi tambahan dalam proses transaksi finansial seperti pengiriman uang, pembayaran tagihan dan pembayaran tiket. Sebelum transaksi di proses, *server* meminta *user* untuk memasukkan *password* OTP yang dihasilkan oleh token *virtual* kemudian *server* melakukan validasi terhadap *password* OTP tesebut. Dari hasil pengujian yang dilakukan bahwa *password* OTP tidak muncul secara berulang dan *secret key* yang dihasilkan secara acak juga tidak muncul secara berulang tetapi mempunyai presentase kemiripan tertinggi sebesar 0,03 %.

Kata kunci: *Time-Based One Time Password* (TOTP)*, SHA256*, *Two Factor Authentication* (2FA), *internet banking, token virtual*.

TIME-BASED ONE TIME PASSWORD ALGORITHM IMPLEMENTATION IN THE AUTHENTICATION TOKEN INTERNET BANKING

Kurnia Ramadhan Putra, Uung Ungkawa, Irma Amelia Dewi

Department of Informatics Engineering
Faculty of Industrial Technology
National Institute of Technology

ABSTRACT

Time-Based One Time Password (TOTP) algorithm is one of the algorithm that has ability to generate one time password. Generated password by TOTP algorithm has limited time span and always changing in a certain period. How it works of TOTP algorithm is combining between secret key and time then hashing them using encryption algorithm SHA256. TOTP algorithm is the implementation of Two Factor Authentication (2FA) is combination two different authentication methods simultaneously. The security methods combined are static password as login access and virtual token applied on android smartphone. In this research 2FA applied on internet banking system which between the virtual token and internet banking server planted Time-Based One Time Password algorithm to generate password as an additional authentication in financial transaction process such as remittance, bill payment and ticket payment. Before the transaction being processed, the server asks the user to enter OTP password was generated by virtual token then validate the OTP password. From the test result that OTP password does not appear repeatedly and the secret key was generated randomly so does not appear repeatedly but has the highest similarity percentage of 0,03 %.

Keywords *: Time-Based One Time Password* (TOTP)*, SHA256*, *Two Factor Authentication* (2FA), *internet banking, virtual token*.

DAFTAR ISI

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada tahun 2000, beberapa bank di Indonesia mulai menerapkan sistem *E-Banking* mulai dari ATM, *phone banking* dan *internet banking*. *Internet banking* digunakan untuk melakukan transaksi pembayaran, cek saldo, pengiriman uang dan transaksi lainnya melalui *internet* pada *website* yang dimiliki oleh bank yang bersangkutan. Pemanfaatan *internet banking* akan meningkatkan efisiensi, efektivitas dan produktivitas sekaligus meningkatkan penjualan yang lebih daripada bank konvensional[1].

Permasalahan yang sering terjadi dalam penggunaan *internet banking*[2] adalah sering terjadinya pembobolan rekening tabungan yang mana dana nasabah tidak dapat dikembalikan dan pihak bank menganggap bahwa nasabah lalai dalam menjaga kerahasiaan PIN mereka. Pembobolan rekening nasabah tersebut dilakukan dengan cara mencuri *username* dan *password* yang digunakan sebagai akses *login* pada *website internet banking*, kemudian melakukan transaksi untuk memindahkan uang dari rekening nasabah ke suatu rekening tertentu.

Saat ini metode keamanan yang diterapkan pada *internet banking* adalah *Two Factor Authentication* (2FA) yaitu penggabungan dua metode otentikasi berbeda secara bersamaan yaitu *password* statis dan alat otentikasi tambahan berupa token *hardware*, yaitu sebuah alat dengan bentuk fisik seperti kalkulator yang mampu menghasilkan kode yang berubah-ubah secara periodik karena di dalamnya dipasang sebuah algoritma *Time-Based One Time Password* (TOTP) yang mampu menghasilkan *password* sekali pemakaian yang mana *password* akan berubah dalam rentang waktu tertentu.

Kelemahan dalam penggunaan token *hardware* adalah pengguna harus mengikuti beberapa prosedur yang panjang untuk mendapatkan *hardware* tersebut pada bank, membayar denda kepada pihak bank jika terjadi kerusakan maupun kehilangan dan sering lupa membawanya ketika bepergian.

Pemanfaatan teknologi yang ada pada token *hardware* dapat dijadikan sebagai solusi untuk mengatasi permasalahan tersebut yaitu membangun aplikasi token *virtual* yang memiliki kemampuan seperti token *hardware* dengan tingkat keamanan yang sama dan dapat dipasang pada sebuah *smartphone android.*

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang masalah yang dikemukakan, maka peneliti merumuskan beberapa masalah sebagai berikut :

1. Bagaimana menerapkan algoritma *Time-Based One Time Password* (TOTP) pada sisi *server* dan *client internet banking*.
2. Bagaimana membangun aplikasi *client* token *internet banking* secara *virtual* berbasis *Android*.
3. Bagaimana membangun aplikasi *server* simulasi *internet banking* berbasis *web*.

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah menerapkan algoritma *Time-Based One Time Password* (TOTP) dalam otentikasi token *internet banking* berbasis *android*.

1.4 Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Metode otentikasi yang digabungkan yaitu *Something You Know* (SYK) dengan *Something You Have* (SYH).
2. Mode yang digunakan dalam menghasilkan *One Time Password*, menggunakan *Challenge*/ *Response* (C/R).
3. Algoritma enkripsi yang digunakan adalah *Secure Hash Algorithm* 256 (SHA-256).
4. Token *virtual* dibangun menggunakan *Android* versi 4.3 (*Jelly Bean*) dan sisi *server* yaitu menggunakan bahasa pemograman *Scala*.

5. Transaksi transfer uang antar bank belum diterapkan pada *internet banking*.

1.5 Metodologi Penelitian

Adapun metodologi penelitian yang dilakukan, meliputi :

1. Metode Pengumpulan Data

   Pengumpulan data dilakukan dengan studi kepustakaan yaitu dengan mengumpulkan data dan informasi tentang algoritma *Time-Base One Time Password* dan token *internet banking* melalui literatur media cetak seperti buku, skripsi, jurnal dan media *online* yaitu artikel-artikel yang didapat dari kegiatan *browsing*.

2. Studi Pengembangan Sistem

   Metodologi pengembangan sistem pada penelitian ini menggunakan Metodologi *Iconix Process*, seperti yang dijelaskan pada `Gambar 1` yang meliputi :

   1) *Functional requirements*, tahap ini menjelaskan apa yang dapat dilakukan oleh sistem yang melibatkan *business analyst*, *customer*, *end user*, *programmer* dan *stakeholder* lainnya yang dipaparkan pada **sub bab 3.3**.
   2) *Domain model*, tahap ini untuk menyamakan persepsi dalam penamaan istilah yang akan dipakai dalam aplikasi, implementasinya yaitu *class diagram* sederhana yang nantinya dikembangkan menjadi lebih rinci yand dipaparkan pada **sub bab 3.3 poin 3.3.1.1**.
   3) *Use case*, tahap ini mendefesnisikan *behaviour requirement* berdasarkan *functional requirements* dengan menspesifikasikan seperti apa cara kerja sistem termsuk responnya ke dalam tabel skenario yang dipaparkan pada **sub bab 3.3 poin 3.3.1.2**.
   4) *Requirement review*, tahap ini memastikan bahwa *use case* sudah sesuai dengan yang diharapkan yang dipaparkan pada **sub bab 3.3 poin 3.3.1.4**.

5) *Robustness analysis*, tahap ini untuk menjembatani antara analisis dan perancangan yang diterapkan pada *use case* yang ada.
6) *Preliminary Design Review*, tahap ini memastikan semua yang telah dibuat sesuai dengan kebutuhan.
7) *Technical architecture*, tahap ini menentukan *framework* apa saja yang digunakan yang dipaparkan pada **sub bab 3.3 poin 3.3.1.6**.
8) *Sequence diagram*, tahap ini menemukan operasi pada *domain model* karena pada *domain model* baru hanya berisi data yang dipaparkan pada **sub bab 3.3 poin 3.3.1.8**.
9) *Critical design review*, memastikan bahwa tidak ada kekurangan pada *sequence diagram* yaitu setiap *class* sudah memiliki atribut dan operasi.
10) *Coding*, tahap ini merubah hasil rancangan menjadi kode program yang dipaparkan pada **Lampiran E Algoritma TOTP**.

1.6 Tinjauan Pustaka

Penelitian tentang Implementasi Algoritma T*ime-Based One Time Password* Dalam Otentikasi Token *Internet Banking* Berbasis *Android* terinspirasi dari penelitian-penelitian yang telah dilakukan sebelumnya.

1. Made Bayu Wedagama, dkk dari Universitas Telkom, telah melakukan penelitian dengan judul "*Desain dan Implementasi Sistem Keamanan Algoritma Rindjael (AES) Dengan One Time Password Untuk Optimasi Layanan SMS Banking*" yang menjelaskan bahwa dengan menggabungkan teknik enkripsi/ dekripsi AES 128 bit (Algoritma Rindjael) dengan metode otentikasi *One Time Password* (OTP) berbasis *Challenge*/ *Response with Changeable-Rules* sebagai usaha untuk mengoptimasi terhadap layanan perbankan yang menggunakan *SMS Banking* (**Made Bayu Wedagama, Iwan Iwut Tritoasmoro dan Uke Kurniawan, 2009**).

2. Aisyah Dzulqaidah dari Institut Teknologi Bandung, telah melakukan penelitian dengan judul "*Pembangkitan Kombinasi Kode Pada Google Authenticator*" yang menjelaskan bahwa dengan *2-step verification* membuat akun menjadi lebih aman yaitu selain menggunakan *password* statis, *user* juga masukkan kode sebanyak 6 digit untuk verifikasi. Enam digit kode tersebut dibangkitkan dengan algoritma *Hmac-Based One Time Password* dan algoritma *Time-Based One Time Password* yang digunakan pada aplikasi *Google Authenticator* (**Aisyah Dzulqaidah, 2012**).

3. Obeth Pasanda dari Universitas Kristen Duta Wacana, telah melakukan penelitian dengan judul "*Pemanfaatan One Time Password dan Algoritma HMAC-SHA1 pada USB Device*" yang menjelaskan bahwa untuk menjaga keamanan dan kerahasiaan data pada sebuah *website* perlu dilakukan pergantian *password* secara berkala. Akan tetapi, dengan menggunakan *One Time Password* (OTP), *user* tidak perlu melakukan pergantian *password* secara berkala karena OTP merupakan *password* yang hanya digunakan sekali. Untuk membangkitkan OTP tersebut digunakan algoritma *HMAC-SHA1* yang di program kedalam sebuah bentuk *usb device* (**Obeth Pasanda, 2013**).

4. Berdasarkan penelitian yang dilakukan oleh **Made Bayu Wedagama, dkk** dalam tugas akhir mereka yang berjudul "*Desain dan Implementasi Sistem Keamanan Algoritma Rindjael (AES) Dengan One Time Password Untuk Optimasi Layanan SMS Banking*", **Aisyah Dzulqaidah** dalam tugas akhirnya yang berjudul "*Pembangkitan Kombinasi Kode Pada Google Authenticator*" dan **Obeth Pasanda** dalam tugas akhirnya yang berjudul "*Pemanfaatan One Time Password dan Algoritma HMAC-SHA1 pada USB Device*" sehingga penulis berkeinginan melakukan penelitian tentang Algoritma *Time-Based One Time Password* (TOTP) dalam tugas akhir yang berjudul "Implementasi Algoritma *Time-Based One Time Password* Dalam Otentikasi Token *Internet Banking* Berbasis *Android*".

Adapun ringkasan dari beberapa penelitian yang telah dibahas diatas dapat dilihat pada Tabel 1.

Tabel 1. Tabel Perbandingan Penelitian

Nama	Judul	Tahun	Algoritma	Tujuan
Made Bayu Wedagama, dkk	Desain dan Implementasi Sistem Keamanan Algoritma *Rindjael* (AES) dengan *One Time Password* Untuk Optimasi Layanan *SMS Banking*	2009	*Rindjael*	Langkah-langkah bertransaksi dengan layanan *SMS Banking* tidak saja aman tetapi juga *user-friendly*.
Aisyah Dzulqaidah	Pembangkitan Kombinasi Kode Pada *Google Authenticator*	2012	TOTP dan HOTP	Layanan *2-step verification* dapat mengamankan akun-akun yang diusung *google* tetapi juga untuk media sosial lainnya seperti *facebook*, *dropbox* dan lain-lain.
Obeth Pasanda	Pemanfaatan *One Time Password* dan Algoritma *HMAC-SHA1* Pada *USB*	2013	HMAC-SHA1	Algoritma *HMAC-SHA1* akan menghasilkan *One Time Password* (OTP) sebagai solusi dalam pergantian *password* statis secara berkala.
Kurnia Ramadhan Putra	Implementasi Algoritma *Time-Based One Time Password* Dalam Otentikasi Token *Internet Banking* Berbasis *Android*	2015	TOTP	Menerapkan algoritma *Time-Based One Time Password* (TOTP) pada sisi *server* dan *client internet banking*.

1.7 Sistematika Penulisan

Untuk lebih memberikan gambaran secara jelas mengenai tugas akhir ini, pembahasan garis besar akan dibangun menjadi lima bab dengan sistematika penulisan :

1. **BAB I PENDAHULUAN**

   Bab ini menjelaskan tentang informasi umum yang berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, tinjauan pustaka dan sistematika penulisan.

2. **BAB II LANDASAN TEORI**

   Bab ini berisi dasar-dasar teori yang mendasari penelitian tugas akhir yang diambil dari beberapa buku, skripsi, jurnal dan media *online* yaitu artikel-artikel yang didapat dari kegiatan *browsing*.

3. **BAB III ANALISIS DAN PERANCANGAN**

   Bab ini berisi tentang tahapan analisis kebutuhan sistem dan perancangan dari aplikasi.

4. **BAB IV IMPLEMENTASI DAN PENGUJIAN**

   Bab ini berisi tahapan dalam implementasi dari sistem yang dibangun.

5. **BAB V PENUTUP**

   Bab ini berisi kesimpulan yang didapatkan dari hasil uji coba dan analisis mengenai keterkaitan antara tujuan dengan implementasi sistem.

BAB II
LANDASAN TEORI

2.1 Metode Otentikasi[3]

Seth Rosenblat merupakan seorang penulis berita dari CNET-News, menjelaskan bahwa otentikasi bertujuan untuk membuktikan siapa Anda sebenarnya dan apakah Anda benar-benar pengguna yang sah (**Seth Rosenblatt, 2015**). Untuk membuktikan pengguna yang sah tersebut dapat dijelaskan dalam beberapa metode otentikasi sebagai berikut :

1. *Something You Know*

   Metode otentikasi ini adalah yang paling umum dengan mengandalkan kerahasiaan informasi, contohnya adalah *password* atau PIN dengan asumsi bahwa tidak ada seorangpun yang mengetahui rahasia tersebut kecuali dirinya sendiri.

2. *Something You Have*

   Metode otentikasi ini merupakan faktor tambahan untuk membuat otentikasi menjadi lebih aman dengan mengandalkan barang yang sifatnya unik seperti *smart card*, *hardware token*, *usb token* dan sebagainya, dengan asumsi bahwa tidak seorangpun memiliki *hardware* tersebut kecuali dirinya sendiri.

3. *Something You Are*

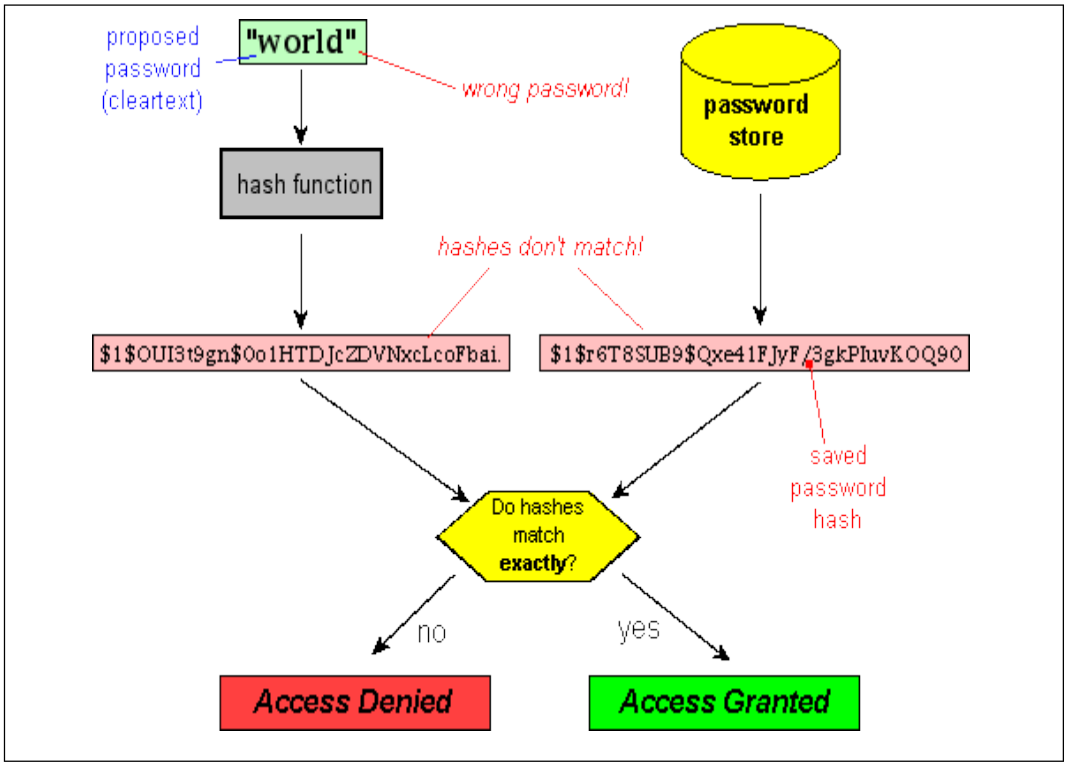
   Metode otentikasi ini jarang dipakai karena faktor teknologi dan manusia, yaitu dengan mengandalkan bagian-bagian tubuh yang menjadi ciri fisik manusia seperti sidik jari, sidik retina dan lain-lain.

4. *Something You Can*

   Metode otentikasi ini berasumsi bahwa tidak ada seorangpun di dunia ini yang mampu menirukannya, seperti tanda tangan yang dianggap sebagai bukti otentik dari diri pengguna.

2.2 Proses Otentikasi[4]

Seperti otentikasi pada umumnya, agar otentikasi berhasil maka *password yang di kirimkan client = password yang disimpan di server*. Untuk meningkatkan keamanan, *server* tidak menyimpan *password* dalam bentuk *plain-text* melainkan dalam bentuk *hash* sehingga tidak dapat di kembalikan lagi menjadi bentuk *plain-text* seperti pada Gambar 1.



Gambar 1. Proses Otentikasi

(Sumber : www.unixwiz.net/techtips/iguide-crypto-hashes.html diakses pada tanggal 5 September 2015)

Pada Gambar 1, user mencoba memasukkan *password* yang salah dengan kata "world" kemudian password tersebut dilakukan *hashing*. *Password* yang sudah di *hashing* di bandingkan dengan *password hashing* yang tersimpan di dalam *database* dan hasilnya tidak cocok sehingga akses *user* ditolak.

2.3 *Two Factor Authentication*[5]

Pada aplikasi kritis dan sensitif seperti transaksi keuangan, satu metode otentikasi saja tidak cukup maka muncul istilah 2FA (*Two Factor Authentication*) yang merupakan sistem otentikasi yang menggabungkan 2 metode otentikasi berbeda (**Seth Rosenblatt, 2015**). Empat dari metode otentikasi dapat digabungkan untuk meningkatkan keamanan contohnya *Something You Have* (kartu ATM) dapat digabungkan dengan *Something You Know* (PIN). Contoh lain yaitu ketika belanja di *Supermarket* menggunakan kartu kredit, ada beberapa motode otentikasi yang digabungkan yaitu *Something You Have* (kartu kredit), *Something You Know* (PIN) yang diinputkan ke mesin EDC (*Electronic Data Capture*) dan *Something You Can* (tanda tangan) ke nota pembayaran. Pada *internet banking* juga menggunakan 2FA dengan menggabungkan *Something You Know* (*password*) dengan *Something You Have* (*hardware token*).

2.4 Mode Pembangkit *Password*[6]

Rizki Wicaksono merupakan seorang penulis dari *www.ilmuhacking*.com, menjelaskan bahwa pada umumnya ada dua macam mode dalam menghasilkan password token *internet banking* (**Rizki Wicaksono, 2009**) :

1. Mode *Challenge and Response* (C/ R)

   Mode C/R sering dipakai ketika bertransaksi. Dalam mode ini *server* memberikan *challenge* berupa sederetan angka. Angka tersebut dimasukkan ke dalam mesing token untuk mendapatkan *response* (jawaban). Kemudian pengguna memasukkan angka yang muncul pada tokennya ke dalam form situs *internet banking*. Token akan mengeluarkan kode yang berbeda-beda walaupun dengan *challenge* yang sama dalam periode tertentu.

2. Mode *Self Generated* (SG)

   Pada mode SG, *server* tidak memberikan *challenge* atau tantangan apapun karena token dapat menghasilkan sendiri angka tanpa harus memasukkan *challenge*. Seperti mode C/R token juga dapat menghasilkan kode yang berbeda-beda secara periodik.

2.5 *Security Internet Banking*[7]

Budi Sulistyo merupakan seorang *Security Expert* dari Lembaga Riset Telematika Sharing Vision Bandung, menjelaskan bahwa sistem *internet banking* telah memiliki beberapa mekanisme pengamanan sebagai berikut (**Budi Sulistyo, 2015**) :

1. Otentikasi *Server Internet Banking*

   Pengguna melalui PC yang digunakannya harus dapat memastikan bahwa *server internet banking* yang dituju merupakan *server* yang benar-benar valid.

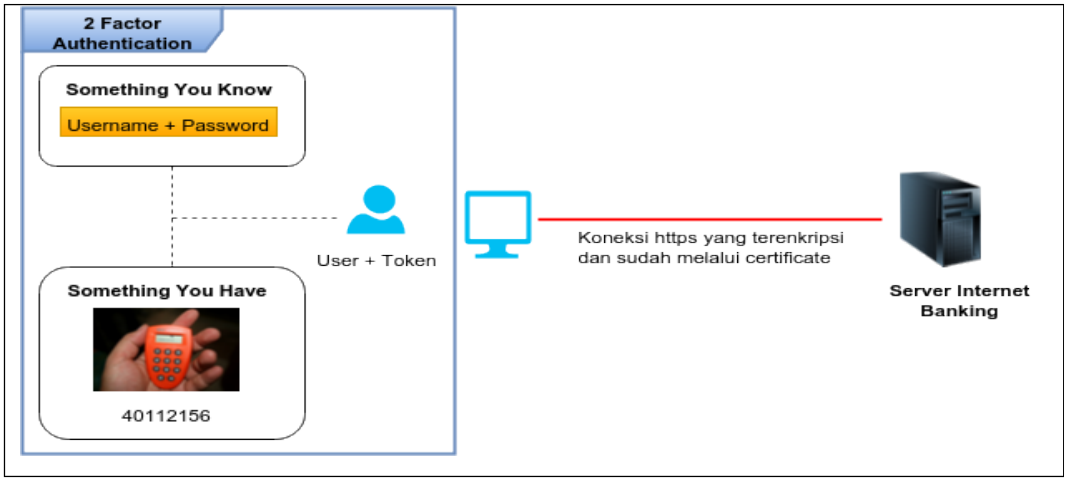
2. Enkripsi Komunikasi antara *PC* dan *Server*

   Enkripsi komunikasi antara *PC* pengguna dan *server internet banking* harus dijamim kerahasiaanya.

3. Otentikasi Pengguna

   Server internet banking harus dapat memastikan bahwa nasabah yang menggunakan layanan adalah benar-benar pengguna *internet banking* yang sah.

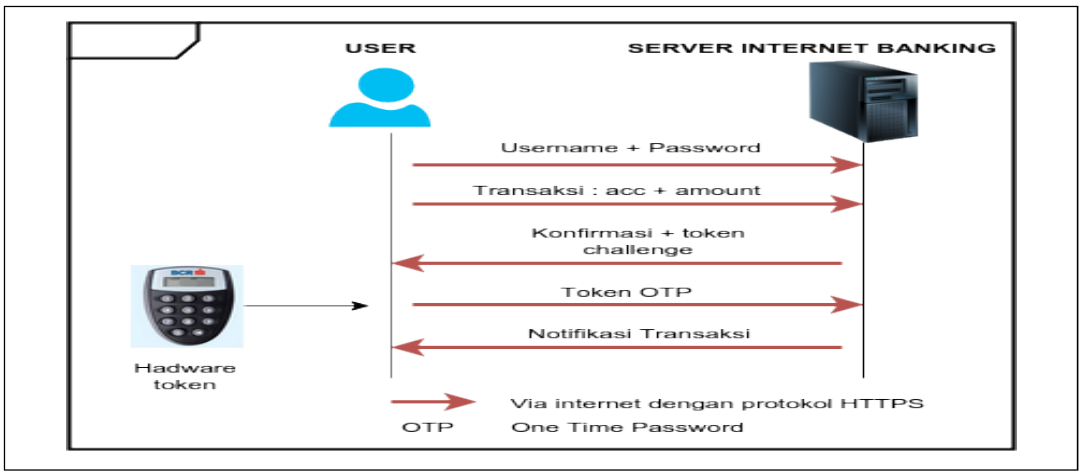
Kriteria keamanan (1) dan (2) dipenuhi dengan menerapkan protokol *HTTPS*. Dalam protokol *HTTP*, *PC* pengguna akan mengotentikasi *server internet banking* dengan cara memeriksa *digital certificate*-nya. Berikut *security internet banking* seperti pada Gambar 2.



Gambar 2. *Security internet banking*

(Sumber : https://niatnulis.wordpress.com/tag/internet-banking/ diakses pada tanggal 14 September 2015)

Kriteria keamanan (3) direalisasikan dengan menggunakan *2 Factor Authentication* yang terdiri dari *Something You Know* dan *Something You Have*. Pada *Something You Know*, pengguna harus mengetahui *username* dan *password* ke halaman *login internet banking*, sedangkan pada *Something You Have*, pengguna harus memasukkan *password* token sekali pakai (token *one time password*) yang diminta. Berikut protokol keamanan *internet banking* dengan menggunakan *hardware token* seperti pada Gambar 3.

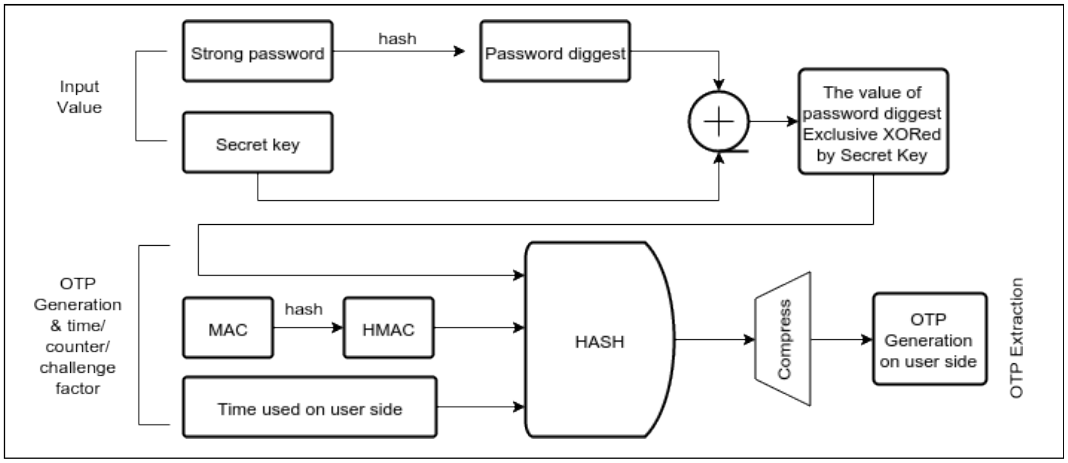


Gambar 3. Protokol keamanan *internet banking* menggunakan *hardware token*

(Sumber : https://niatnulis.wordpress.com/tag/internet-banking/ diakses pada tanggal 14 September 2015

2.6 Algoritma *Time-Based One Time Password* (TOTP)[8]

Dr. David M'Raihi merupakan seorang ahli di bidang *security* dan kriptografi menjelaskan bahwa *Time-Based One Time Password* (TOTP) adalah contoh dari HMAC (*Hashes Message Authentication Code*) yang menggabungkan antara *secret key* dengan *current timestamp* (waktu saat ini) dan menggunakan fungsi kriptografi *hash* untuk menghasilkan *One Time Password* (**M'Raihi, 2011**). Untuk memberikan penjelasan cara kerja algoritma TOTP dapat dilihat pada Gambar 4.

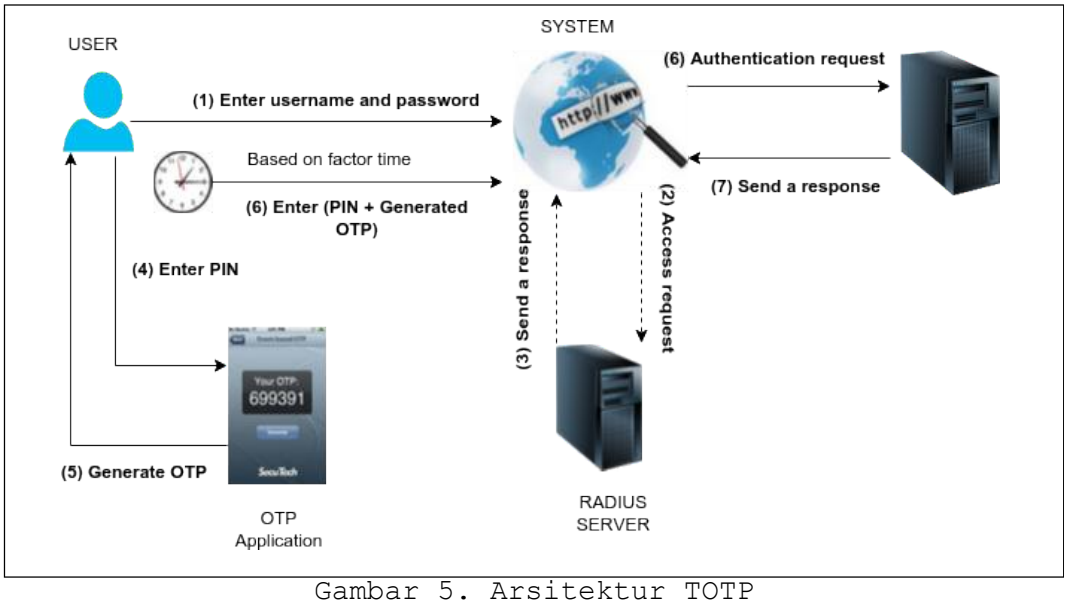


Gambar 4. Cara kerja algoritma *One Time Password*

(Sumber : http://library.iugaza.edu.ps/thesis/113489.pdf diakses pada tanggal 14 September 2015)

2.6.1 Arsitektur *Time-Based One Time Password (TOTP)*

Berikut arsitektur TOTP dengan menampilkan proses integral (**Amna S.M Abukeshipa, 2014**) seperti pada Gambar 5.



Gambar 5. Arsitektur TOTP

(Sumber : http://library.iugaza.edu.ps/thesis/113489.pdf diakses pada tanggal 14 September 2015)

Ilustrasi arsitektur TOTP dapat dilihat pada Tabel 2.

Tabel. 2 Ilustrasi arsitektur TOTP

Langkah	Deskripsi
1	User memasukkkan *username* dan *password* pada layar *login*, password harus berisi kurang lebih 6 karakter.
2	Sistem mengirimkan data *user* ke *Radius Server* untuk mengotentikasi *user*
3	*Radius Server* memverifikasi *username* dan *password* dan memberi respon kepada sistem bahwa *user* diterima atau ditolak.
4	User membuka aplikasi OTP dan memasukkan PIN
5	Aplikasi OTP menghasilkan password yang berlaku selama 3 menit.
6	User memasukkan PIN dan kode OTP ke dalam sistem
7	Sistem mengirim permintaan ke *Authentication Server* untuk diperiksa, PIN, OTP dan *secret key* dari *user*.
8	*Authentication Server* mengotentikasi PIN, OTP dan *Secret Key* kemudian memberikan respon bahwa akses diterima atau ditolak.

2.6.2 Rumus Perhitungan TOTP

Proses menghasilkan *One Time Password* menggunakan algoritma TOTP membutuhkan suatu perhitungan (**M'Raihi, 2011**) yang ditunjukkan pada persamaan (1).

TOTP = HMAC (Secret Key, TC) ..... **(1)**

Dimana :

TOTP : *Time-Based One Time Password*

HMAC : *Hash Message Authentication Code*

Secret Key : *seed* yang disimpan pada *server* dan *client* dengan panjang 64 byte.

TC : *time counter* yang digunakan dalam menghitung waktu untuk menghasilkan *one time password*.

Untuk menghitung *time counter* (TC) menggunakan persamaan (2).

TC = unixtime(now) – T0/ TS, ... **(2)**

Dimana :

TC : *Time Counter*

Unixtime(now) : waktu saat ini (*current time*)

T0 : di inisialisasi dengan nilai = 0.

TS : time step dengan inisialisasi nilai = 30 detik.

Proses menghitung algoritma *Hash Message Authentication Code* (HMAC) menggunakan persamaan (3).

HMAC = SHA256 (K + 0x5c5c)... | SHA256 (K + 0x5c5c)... | C ..... **(3)**

Dimana :

K : Secret Key

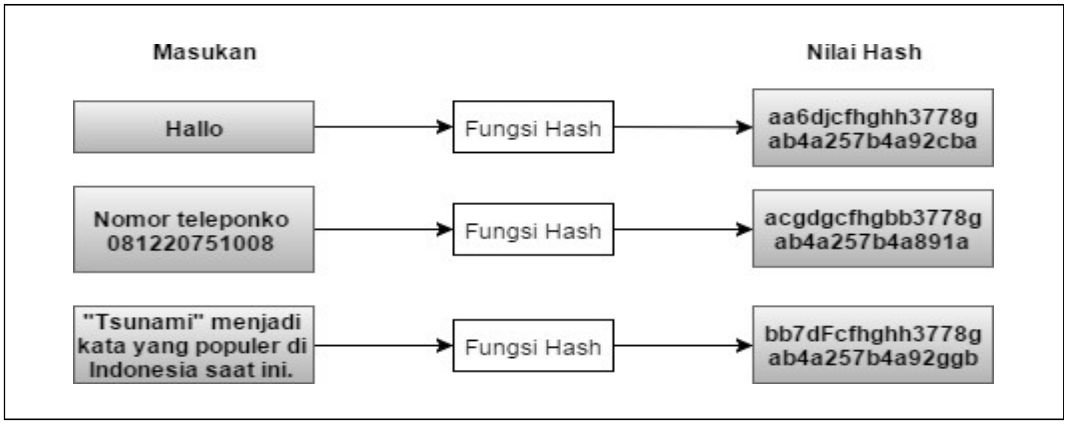
0x5c5c : nilai *hexadecimal* dari waktu.

2.7 *Hash*[9]

Hash adalah adalah bagian dari teknik enkripsi dengan cara melakukan perubahan *string* dari karakter menjadi sebuah nilai kunci berupa penggabungan antara angka, karakter dan simbol (**Margaret Rouse, 2005**). Beberapa fungsi *hash* adalah untuk otentikasi *password*, otentikasi *file*, tanda tangan digital dan sebagainya tetapi secara umum digunakan untuk otentikasi.

Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan mengkonversinya menjadi *string* keluaran yang panjangnya tetap atau *fixed*. Fungsi *hash* yang dihasilkan biasanya dituliskan dalam notasi persamaan $h = H(m)$ (**Christian Angga, 2011**) .

Pada persamaan tersebut,*h* merupakan nilai hash yang dihasilkan, sedangkan *H* adalah fungsi *hash*-nya itu sendiri dan *M* adalah *message* atau pesan yang akan dikonversikan menjadi nilai *hash* (*hash value*). Nilai *hash* yang dihasilkan biasa disebut juga dengan pesan ringkas atau *message diggest*. Berikut adalah penggunaan fungsi *hash* yang mengubah suatu *string* yang panjang menjadi *message diggest* dengan panjang tetap seperti pada Gambar 6.



Gambar 6. Contoh penggunaan fungsi Hash

(http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2010-2011/Makalah2/Makalah2-IF3058-Sem2-2010-2011-046.pdf diakses pada tanggal 19 Oktober 2015)

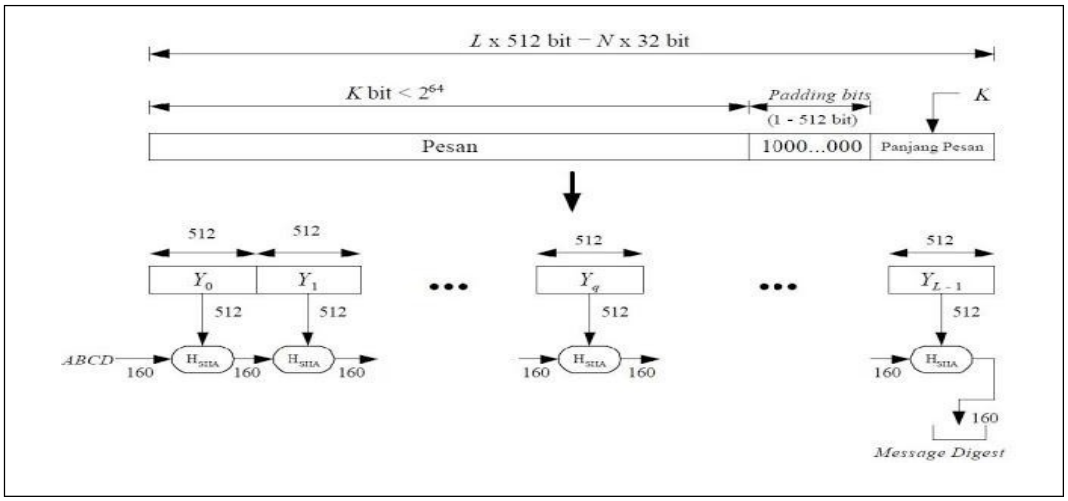
Fungsi *hash* pada dasarnya bekerja satu arah, berarti pesan asli atau pesan semula akan diubah menjadi sebuah *mesage diggest*, namun *message diggest* yang dihasilkan tidak dapat dikembalikan menjadi pesan asli atau pesan semula kembali.

2.8 *Secure Hash Algorithm* (SHA)[10]

Standar menetapkan *Secure Hash Algorithm* (SHA) yang diperlukan untuk menjamin keamaanan *Digital Signature Algorithm* (DSA). Ketika dengan pesan sepanjang 2^{64} bit dimasukkan, SHA menghasilkan 160 bit keluaran yang disebut dengan *message diggest* (MD) (**Yusuf Kurniawan, 2004**).

MD ini kemudian dimasukkan ke dalam DSA yang menghitung tanda tangan untuk pesan digital tersebut. Penandatangan MD meningkatkan efesiensi proses, karena MD jauh lebih kecil dibanding pesan aslinya. MD pesan yang sama seharusnya dapat diperiksa ketika menerima pesan dari pengirim dengan cara memasukkan pesan tersebut ke fungsi *hash* SHA. SHA dikatakan aman karena di desain secara matematis untuk tidak dapat menghasilkan pesan aslinya. SHA dibuat berdasarkan rancangan yang serupa dengan MD4 oleh Profesor Ronald L.Rivest dari MIT. SHA menghasilkan sidik jari 160 bit lebih panjang dari MD5.

Cara kerja algoritma SHA dapat dilihat pada Gambar 7.



Gambar 7. Cara kerja algoritma SHA

2.9 *Brute Force Attack*[11]

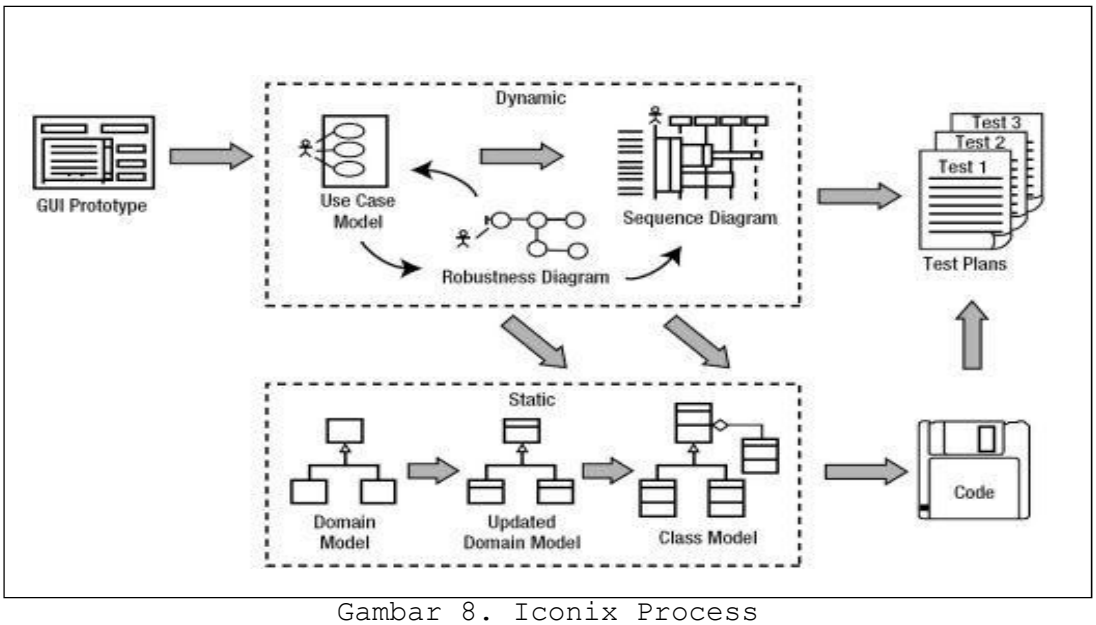
Brute force attack adalah algoritma yang digunakan untuk memecahkan masalah dengan sangat sederhana, langsung dan jelas. Penyelesaian permasalahan *password cracking* menggunakan algoritma *brute force* yaitu dengan mencari semua kemungkinan *password* dengan memasukkan karakter dan panjang password tertentu dan melakukan kombinasi sampai menemukan *password* yang cocok.

Proses pencocokan *String* pada *brute force* :

1. Menyediakan teks yang panjangnya n karakter.
2. Menentukan *pattern* (pola) yang menjadi target pencarian dengan panjang m karakter (m<n).
3. Melakukan pencarian pada baris pertama dari teks yang sesuai dengan *pattern*.
4. Proses pencarian bergerak dari kiri ke kanan dan membandingkan setiap karakter yang ada di dalam teks dengan *pattern* sampai :
   a. Semua karakter yang di bandingkan cocok maka pencarian berhasil.
   b. Menumkan ketidakcocokan dalam pencarian karakter maka pencarian gagal.

2.10 *Iconix Process*[12]

Iconix Process adalah salah satu metode pengembangan perangkat lunak yang menjembatani antara analisis dan perancagan. Langkah-langkah proses pada metode Iconix Process dapat dilihat pada Gambar 8.



Gambar 8. Iconix Process

(Sumber : http://www.simple-talk.com/iconix-process-diagram diakses pada 17 Januari 2016)

Proses yang terdapat pada Gambar 8 meliputi :

1) *Functional requirements* yaitu menentukan kebutuhan sistem.
2) *Domain model* yaitu penyamaan istilah yang dipakai di dalam sistem.
3) *Use case* yaitu mendefesnisikan cara kerja sistem.
4) *Requirement review* yaitu memastikan bahwa *use case* sudah benar.
5) *Robustness analysis* tahap ini untuk menjembatani antara analisis dan perancangan.
6) *Preliminary Design Review* memastikan sistem sudah sesuai kebutuhan.
7) *Technical architecture* yaitu menentukan *framework* yang digunakan.
8) *Sequence diagram* yaitu menemukan operasi pada *domain model*.
9) *Critical design review* yaitu memastikan bahwa tidak ada kekurangan pada *sequence diagram*.
10) *Coding* yaitu merubah hasil rancangan menjadi kode program.

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai analisis dan perancangan dari *aplikasi* token *internet banking* yang akan dibangun. Perancangan aplikasi ini menggunakan *iconix process* sebagai permodelan aplikasi seperti yang telah dipaparkan pada sub bab 1.5.

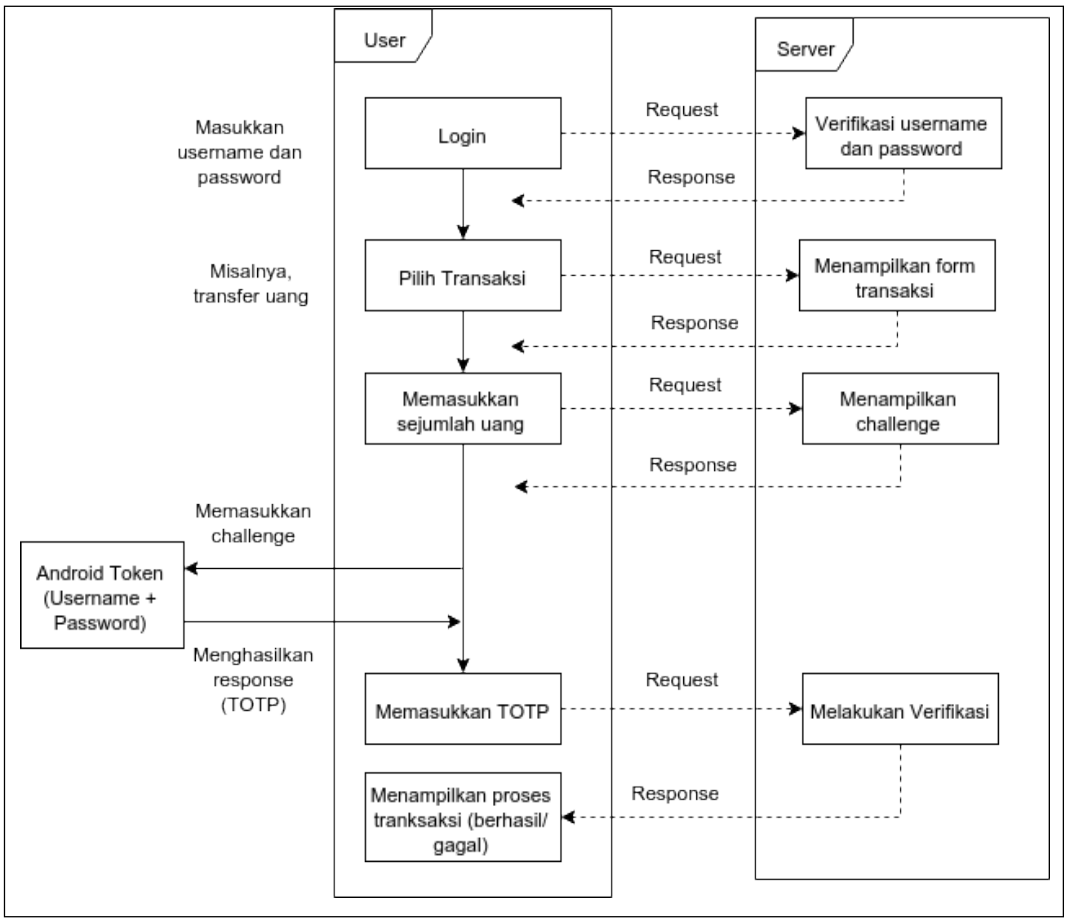
3.1 Analisis

Pada tahap analisis kebutuhan sistem, akan dibahas rancangan yang digunakan dan langkah-langkah yang dilakukan dalam penelitian untuk pembangunan aplikasi token *internet banking* berbasis *android*. Aplikasi yang dibangun merupakan implementasi dari *two factor authentication* yang merupakan teknik pengamanan sebuah sistem dengan cara menggabungkan dua buah metode otentikasi berbeda seperti *something you know* dan *something you have*. Metode otentikasi *something you know* diimplementasikan dengan cara memasukkan *password* statis yang digunakan oleh *user* untuk melakukan *login* ke sebuah sistem, sedangkan metode otentikasi *something you have* diimplementasikan dengan cara menggunakan alat otentikasi tambahan yaitu sebuah *smartphone android* yang dipasang aplikasi token untuk menghasilkan *password* sekali pemakaian. Untuk menghasilkan *password* sekali pemakaian ini atau yang lebih dikenal dengan *one time password* menggunakan algoritma *Time-Based One Time Password*.

Pada dasarnya, sebuah algoritma *Time-Based One Time Password* (TOTP) merupakan algoritma yang digunakan untuk menghasilkan beberapa digit angka yang dapat digunakan sebagai *password*. Dalam menghasilkan *password* tersebut adalah merupakan gabungan *secret key* ditambah dengan waktu saat ini (*current time*) kemudian dilakukan *hashing*. Hasil *hashing* ini berupa bilangan *hexadecimal*, sehingga harus di *convert* ke dalam bilangan *decimal* untuk diambil 8 digit angka sebagai *one time password*.

Algoritma *Time-Based One Time Password* (TOTP) dinyatakan ke dalam bentuk notasi *TOTP = HOTP<K, T>,* dimana T adalah sebuah nilai *integer* yang melambangkan jumlah perpindahan waktu antara inisial counter *T0* dengan *current time*. Untuk mencari nilai *T* maka dinyatakan ke dalam bentuk notasi *T = (current time - T0) / X*, dimana *T0* di inisialisasi dengan nilai nol dan *X* merupakan perpindahan waktu dalam detik yang di inisialisasi dengan 3 menit.

Gambaran secara umum dari sistem *internet banking* ini dapat dilihat pada Gambar 9.



Gambar 9. Blok diagram token internet banking berbasis Android

Tahapan analisis terhadap kebutuhan perancangan aplikasi token *internet banking* dengan tahapan proses sebagai berikut :

1. *User login* pada halaman *web internet banking* dengan memasukkan *username* dan *password*.
2. Sistem melakukan *request* ke *server* untuk melakukan verifikasi *username* dan *password* dari *user*.
3. *Server* memberikan respon dengan asusmsi bahwa *user* berhasil melakukan *login*.
4. *User* memilih menu transaksi pada sistem *internet banking*, misalnya transaksi untuk transfer uang.
5. Sistem memberikan respon dengan menampilkan *form* transaksi yang dipilih.
6. *User* memasukkan data-data dan nominal uang yang akan di transfer.
7. Sebelum transaksi di proses, *server* memberikan sederetan kode *challenge* yang ditampilkan pada sistem.
8. *User* memasukkan kode *challenge* dari sistem ke aplikasi token yang telah dipasang pada *smartphone android*.
9. Aplikasi token pada *smartphone android* menjalankan algoritma TOTP untuk menghasilkan kode *one time password*.
10. *User* memasukkan kode *One Time Password* (OTP) yang dihasilkan token *android* ke dalam sistem *internet banking*.
11. Sistem melakukan *request* ke *server* untuk melakukan verifikasi terhadap kode OTP.
12. *Server* memberikan respon dengan kondisi kode OTP valid dan transaksi berhasil dilakukan.

3.1.1 Analisis Keamanan Sistem *Internet Banking*

Analisis keamanan *internet banking* dapat dilihat dari sisi implementasi di lapangan secara umum dan implementasi pada penelitian yang dijelaskan lebih rinci pada poin 3.1.1.1 dan 3 1 1 2.

3.1.1.1 Mekanisme Keamanan *Internet Banking* Secara Umum

Sistem *internet banking* secara umum memiliki beberapa mekanisme pengamanan sebagai berikut :

1. Otentikasi Server *Internet Banking*

   Pengguna melalui PC yang digunakan untuk mengakses *web internet banking*, harus dapat memastikan bahwa *server* yang dituju adalah *server* yang benar-benar *valid*.

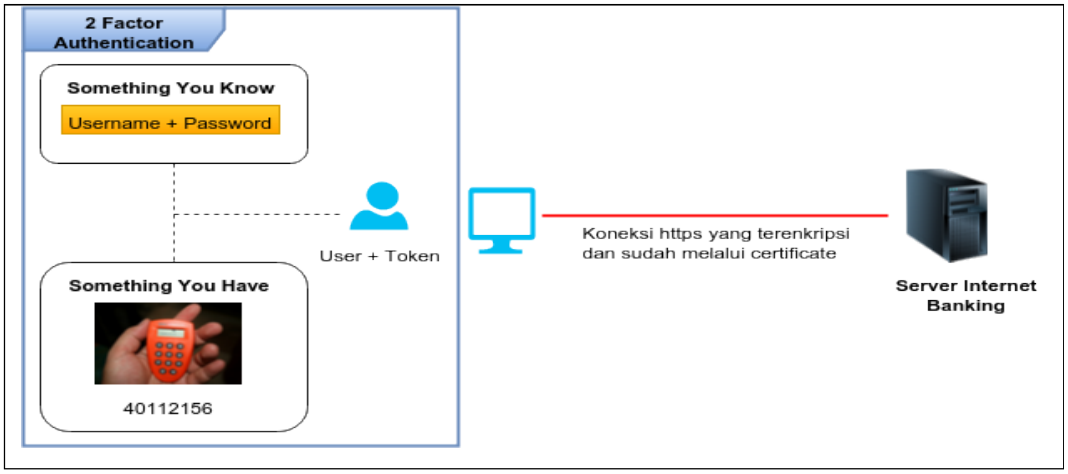
2. Enkripsi Komunikasi antara PC dan *Server*

   Enkripsi jalur komunikasi PC pengguna dengan *server internet banking* digunakan untuk menjamin kerahasiaan informasi selama proses transaksi pada *web internet banking*.

3. Otentikasi Pengguna

   Server internet banking harus dapat memastikan bahwa pengguna yang menggunakan layanan *internet banking* adalah pengguna yang benar-benar valid.

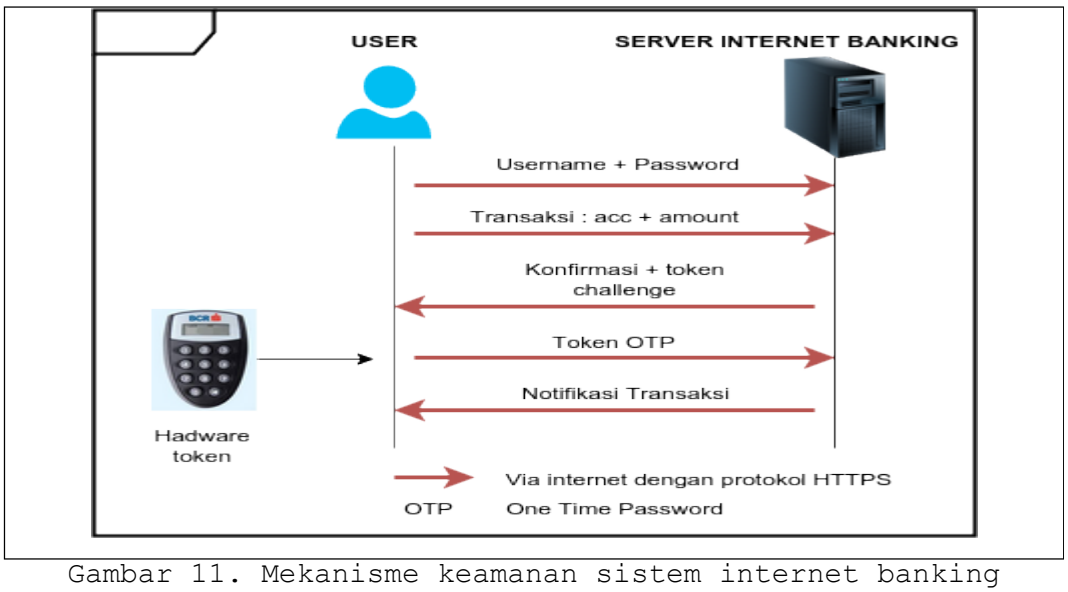
Kriteria kemanan (1) dan (2) diterapkan dengan menggunakan protokol keamanan HTTPS yang berada pada lapisan *Secure Socket Layer* (SSL) yang merupakan sebuah standar teknologi keamanan yang dapat melakukan enkripsi komunikasi data antara *web browser* dengan *web server*. Berikut mekanisme keamanan utama sistem *internet banking* secara umum pada Gambar 10.



Gambar 10. Mekanisme keamanan sistem internet banking secara umum

(Sumber : https://niatnulis.wordpress.com/tag/internet-banking/ 14 September 2015)

Kriteria keamanan (3) diterapkan dengan menggunakan *Two Factor Authentication* (2FA) yang terdiri dari *Something You Know* (SYK) dan *Something You Have* (SYH). Pada SYK pengguna harus mengetahui username dan password yang digunakan untuk login, sedangkan pada SYH pengguna harus memasukkan password OTP yang digunakan untuk transaksi. Berikut mekanisme keamanan internet banking menggunakan token device pada Gambar 11.



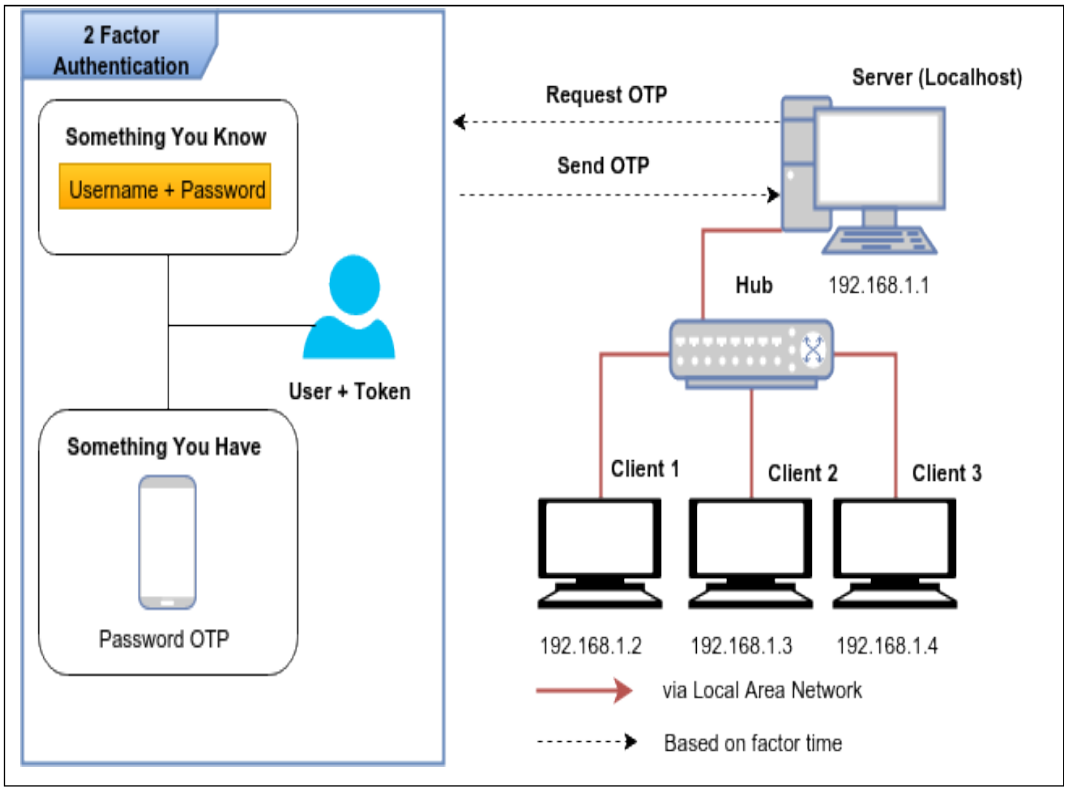
Gambar 11. Mekanisme keamanan sistem internet banking menggunakan token device

(Sumber : https://niatnulis.wordpress.com/tag/internet-banking/ 14 September 2015)

3.1.1.2 Mekanisme Keamanan *Internet Banking* Pada Penelitian

Pada penelitian ini jalur komunikasi antara *web browser* dengan *web server* tidak menggunakan protokol keamanan HTTPS karena penggunaan protokol keamanan HTTPS digunakan jika hanya *web internet banking* yang dibangun di rilis ke publik dan sertifikat digital protokol keamanan HTTPS hanya dikeluarkan oleh *Certificate Authorities* (CAs) yaitu sebuah organisasi yang dipercaya untuk melakukan verifikasi, identifikasi dan legitimasi terhadap sertifikat tersebut.

Untuk kebutuhan penelitian, *server internet banking* diimplementasikan pada komputer pribadi sehingga *web internet banking* diakses secara lokal dengan alasan bahwa protokol keamanan HTTPS memerlukan biaya yang besar untuk mendapatkan sertifikat digitalnya. Berikut mekanisme keamanan yang diterapkan pada penelitian pada Gambar 12.



Gambar 12. Mekanisme keamanan internet banking pada penelitian

Mekanisme keamanan sistem *internet banking* pada penelitian yang terdapat pada Gambar 12 diuraikan sebagai berikut:

1. Komunikasi antara *client* dengan *server* menggunakan jaringan *Local Area Network* (LAN) karena web server disimpan pada localhost dengan sistem keamanan menggunakan *firewall*.

2. Komunikasi antara token *virtual* dengan *server* tidak menggunakan jaringan sama sekali karena antar perangkat dipasang algoritma TOTP kemudian dilakukan sinkronisasi waktu agar menghasilkan password OTP yang sama.

3.1.2 Analisis Algoritma TOTP

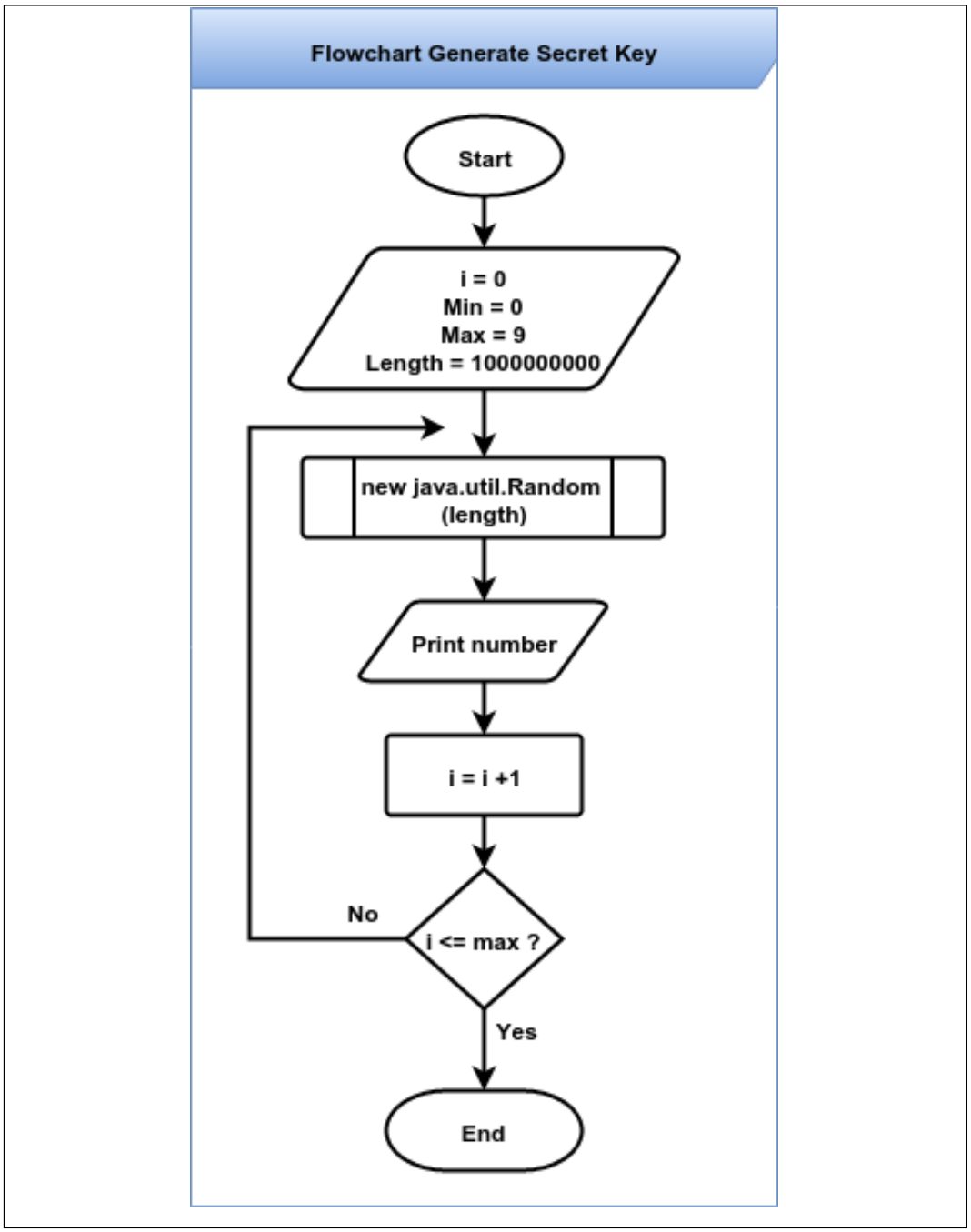
Untuk menguji keamanan algoritma TOTP maka dilakukan beberapa analisis terhadap beberapa parameter yang dimiliki oleh algoritma TOTP seperti *Secret Key*, *Times* dan *Epoch*.

3.1.2.1 Analisis *Secret Key*

Secret Key dihasilkan dari kombinasi angka secara acak menggunakan fungsi matematika yang terdapat pada *library java.util.Random* dalam bahasa pemograman *java*. *Secret key* yang dihasilkan oleh *library java* tersebut dapat dikatakan aman karena memenuhi syarat :

1. *Secret key* muncul secara acak dan tidak dapat di prediksi kemunculannya.
2. *Secret key* yang dihasilkan tidak dipengaruhi oleh kontrol manusia.
3. *Secret key* yang dihasilkan kemungkinan muncul secara berulang tetapi jika hal tersebut terjadi maka *secret key* yang baru dihasilkan akan ditolak karena sudah pernah digunakan sebelumnya.
4. *Secret key* yang muncul memungkinkan mempunyai kemiripan dengan *secret key* yang lain tetapi tidak terlalu signifikan, jika kemiripan terlalu mendekati maka *secret key* yang baru akan ditolak dan harus dihasilkan *secret key* yang baru.
5. Kemiripan angka *secret key* yang muncul tidak boleh melebihi dari 4 angka secara berurutan.

Berikut *flowchart* untuk menghasilkan nilai *secret key* acak pada Gambar 13.



Gambar 13. Flowchart Generate Random Secret Key

Fungsi matematika yang digunakan oleh *library java.util.Random* menggunakan rumus perhitungan pada persamaan (4).

$$Z_i = (aZ_{i-1} + c) \bmod m \quad \text{..... (4)}$$

Dimana :

Z_i = bilangan acak ke i dari deretnya

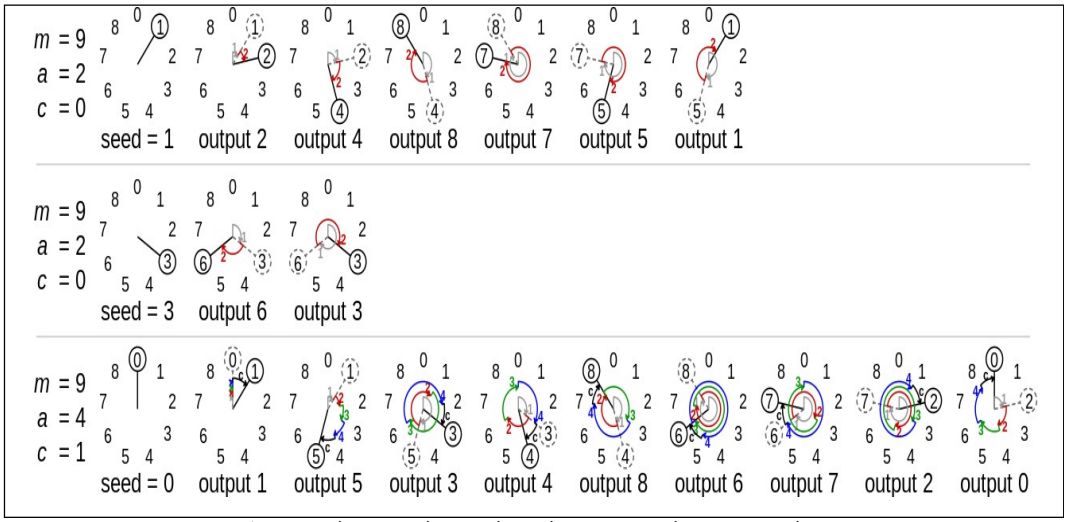
Z_{i-1} = bilangan acak sebelumnya

a = faktor pengali

C = *increment*

M = *modulus*

Ilustrasi dari *library java.util.Random* dalam menghasilkan angka secara acak dapat dilihat pada Gambar 14.



Gambar 14. Visualisasi library java.util.Random

(Sumber : http://utvisu.blogspot.co.id/2013/02/random-number-generator.html diakses pada 14 Januari 2016)

Untuk melihat apakah *secret key* muncul secara berulang atau mempunyai kemiripan, dilakukan pengujian untuk menghasilkan *secret key* sebanyak 10 kali.

Pengujian 1 dan pengujian 2 untuk menghasilkan *secret key* acak dapat dilihat pada Tabel 3.

Tabel 3. Hasil Pengujian Generate Secret Key ke 1 dan 2

Pengujian 1		Pengujian 2	
Array [ ]	**Secret Key**	**Array [ ]**	**Secret Key**
Array [0]	256444175	Array [0]	310567127
Array [1]	658685494	Array [1]	430925415
Array [2]	943771664	Array [2]	847863299
Array [3]	884122147	Array [3]	427483987
Array [4]	852750373	Array [4]	577780340
Array [5]	322369228	Array [5]	555000978
Array [6]	710641775	Array [6]	299313565
Array [7]	133123893	Array [7]	237221822
Array [8]	282004231	Array [8]	505912260
Array [9]	973575346	Array [9]	359013733

Pengujian 3 dan pengujian 4 untuk menghasilkan *random secret key* dapat dilihat pada Tabel 4.

Tabel 4. Hasil Pengujian Generate Secret Key ke 3 dan 4

Pengujian 3		Pengujian 4	
Array [ ]	**Secret Key**	**Array [ ]**	**Secret Key**
Array [0]	348770396	Array [0]	963542603
Array [1]	544603049	Array [1]	251196046
Array [2]	822440552	Array [2]	919319429
Array [3]	766932161	Array [3]	309030788
Array [4]	402017053	Array [4]	750450987
Array [5]	387567064	Array [5]	977418721
Array [6]	196452725	Array [6]	244125978
Array [7]	170502039	Array [7]	731695536
Array [8]	177745060	Array [8]	331546526
Array [9]	827905424	Array [9]	259386079

Pengujian 5 dan pengujian 6 untuk menghasilkan *random secret key* dapat dilihat pada Tabel 5.

Tabel 5. Hasil Pengujian Generate Secret Key ke 5 dan 6

Pengujian 5		Pengujian 6	
Array [ ]	**Secret Key**	**Array [ ]**	**Secret Key**
Array [0]	901510967	Array [0]	572738426
Array [1]	422455894	Array [1]	501897662
Array [2]	280831881	Array [2]	705083336
Array [3]	363503895	Array [3]	347877266
Array [4]	559377806	Array [4]	123965503
Array [5]	155235877	Array [5]	429129107
Array [6]	851361133	Array [6]	748873859
Array [7]	276830188	Array [7]	685583669
Array [8]	553403663	Array [8]	291516786
Array [9]	260312562	Array [9]	901735743

Pengujian 7 dan pengujian 8 untuk menghasilkan *random secret key* dapat dilihat pada Tabel 6.

Tabel 6. Hasil Pengujian Generate Secret Key ke 7 dan 8

Pengujian 7		Pengujian 8	
Array [ ]	**Secret Key**	**Array [ ]**	**Secret Key**
Array [0]	815536667	Array [0]	840151770
Array [1]	217237805	Array [1]	339282347
Array [2]	471637347	Array [2]	450303680
Array [3]	649598347	Array [3]	242237385
Array [4]	182233916	Array [4]	390880804
Array [5]	936085982	Array [5]	648143495
Array [6]	350376614	Array [6]	118103180
Array [7]	164333225	Array [7]	346543680
Array [8]	662099898	Array [8]	695399756
Array [9]	219748705	Array [9]	642148147

Pengujian 9 dan pengujian 10 untuk menghasilkan *random secret key* dapat dilihat pada Tabel 7.

Tabel 7. Hasil Pengujian Generate Secret Key ke 9 dan 10

Pengujian 9		Pengujian 10	
Array [ ]	**Secret Key**	**Array [ ]**	**Secret Key**
Array [0]	145276227	Array [0]	452266266
Array [1]	169321395	Array [1]	477997519
Array [2]	609158116	Array [2]	202958476
Array [3]	382443225	Array [3]	165402890
Array [4]	410744814	Array [4]	152817122
Array [5]	639846808	Array [5]	333723351
Array [6]	753402972	Array [6]	874949427
Array [7]	806443729	Array [7]	916857095
Array [8]	811702821	Array [8]	702717781
Array [9]	173301834	Array [9]	800369263

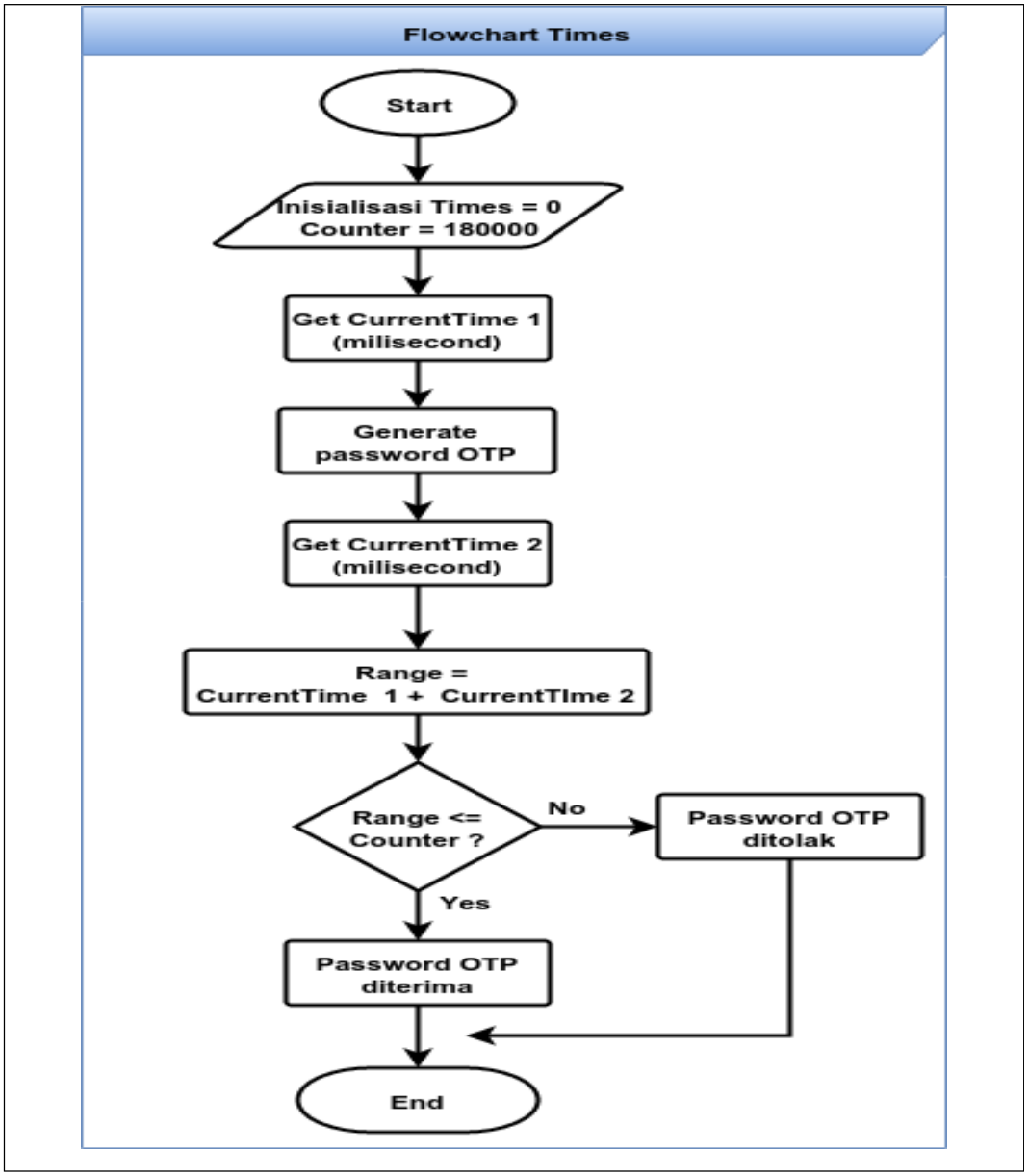
Dari hasil pengujian 1 sampai pengujian 10 tidak ditemukan *secret key* muncul berulang tetapi mempunyai kemiripan seperti yang dijabarkan pada Tabel 8.

Tabel 8. Hasil Kemunculan Kemiripan Secret Key

A	B	C	D	E
Angka Kemiripan	**Secret Key**	**Frekuensi**	**Total Pengujian**	**Presentase (%) = C/D**
777	577780340, 1777745060	2	100	0,02
901	901510967, 901735743	2	100	0,02
347	471637347, 649598347, 339282347	3	100	0,03
680	450303680, 346543680	2	100	0,02
3225	164333225, 382443225	2	100	0,02

3.1.2.2 Analisis *Times*

Times merupakan masa berlaku *password* OTP yang diambil dari standar perbankan di Indonesia yaitu sebesar 3 menit. Berikut *flowchart times* OTP pada Gambar 15.



Gambar 15. Flowchart masa berlaku password OTP

Untuk membuktikan bahwa waktu 3 menit tersebut dianggap cukup untuk setiap transaksi maka dilakukan analisis terhadap 10 kali transaksi yang dijabarkan pada Tabel 9.

Tabel 9. Hasil pengujian waktu transaksi

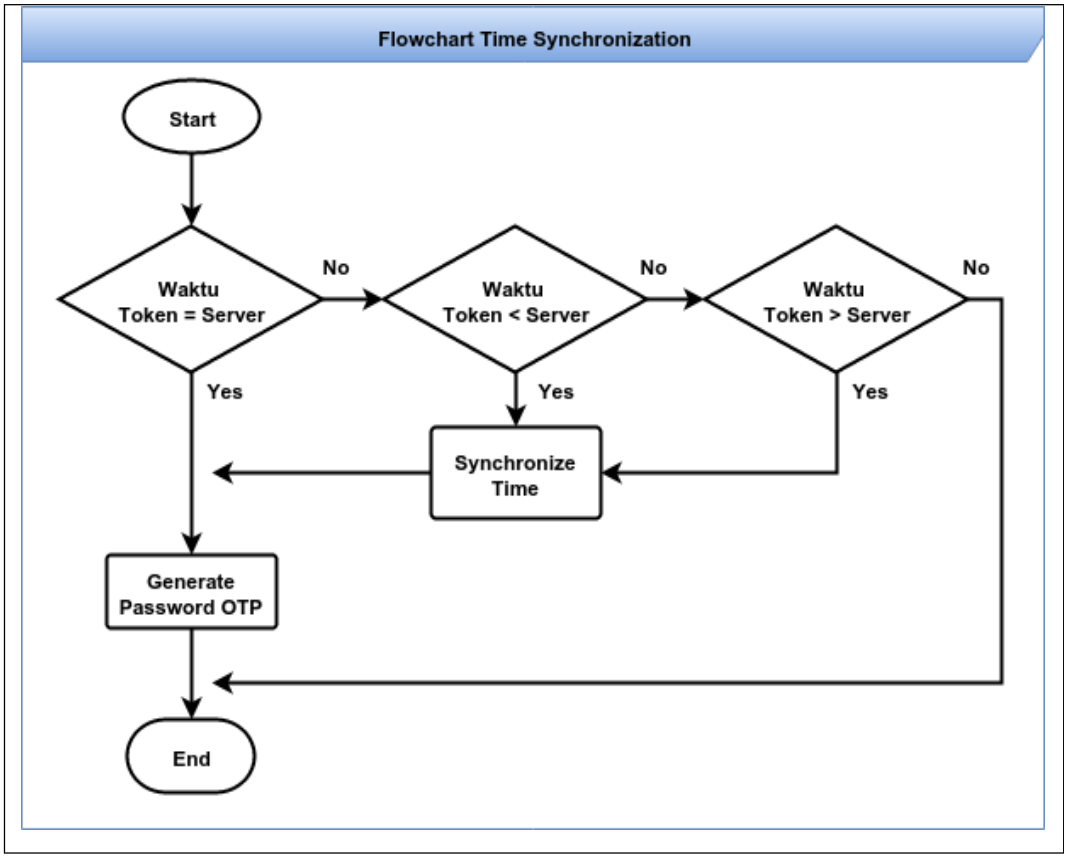
Transaksi	Waktu Server	Waktu Client	OTP Server	OTP Client	Range (detik)
Transaksi 1	10:51:25	10:53:42	31317385	31317385	137
Transaksi 2	11:2:7	11:4:16	48003485	48003485	129
Transaksi 3	11:8:47	11:11:58	26711874	78988445	191
Transaksi 4	11:15:30	11:18:37	14058783	04156698	187
Transaksi 5	11:20:41	11:22:41	28460988	28460988	180
Transaksi 6	11:24:12	11:27:1	38422088	38422088	169
Transaksi 7	11:28:29	11:30:59	28288636	28288636	150
Transaksi 8	11:32:2	11:35:1	21976698	21976698	179
Transaksi 9	11:37:20	11:39:50	30475534	30475534	150
Transaksi 10	11:42:23	11:45:57	28061985	03585516	204

Pada Tabel 9 dapat diperoleh kesimpulan bahwa :

1. Total waktu yang dibutuhkan untuk 10 kali transaksi
   = 137+129+191+187+180+169+150+179+150+204
   = 1676 detik
2. Rata-rata waktu yang dibutuhkan untuk melakukan transaksi
   = Total waktu / jumlah transaksi
   = 1676 / 10
   = 167.6 detik
   = 167.6 / 60 = 2 menit 47 detik.
3. Rentang waktu 3 menit cukup untuk melakukan satu transaksi.

3.1.2.3 Analisis *Epoch*

Epoch adalah jumlah detik dari waktu saat ini (*current time*) yang digunakan untuk sinkronisasi antara *server* dengan token *virtual*. Proses sinkronisasi waktu dapat dilihat pada *flowchart* pada Gambar 16.



Gambar 16. Flowchart Time Synchronization

Pada Gambar 16 dapat dijelaskan sebagai berikut :

1. Memeriksa apakah waktu token virtual sudah sama dengan waktu server.
2. Jika sudah maka dapat dilakukan proses *generate password* OTP.
3. Jika tidak maka harus dilakukan proses sinkronisasi waktu token virtual yang disamakan dengan waktu server.

Untuk membuktikan bahwa proses sinkronisasi waktu antara *server* dengan token *virtual* berpengaruh terhadap password OTP yang dihasilkan maka dilakukan pengujian pada Tabel 10.

Tabel 10. Hasil Sinkronisasi waktu token virtual dengan server

Keterangan	Inisialisasi Waktu Server	Waktu Client	OTP Server	OTP Client
Waktu client < server	6:39:0	6:38:0	26410380	20727102
Waktu client = server	6:39:0	6:39:0	**26410380**	**26410380**
Waktu client > server	6:39:0	6:42:1	26410380	97431993

Dari Tabel 10 dapat dilihat bahwa *password* OTP *valid* jika hanya waktu antara token *virtual* sama dengan waktu *server*.

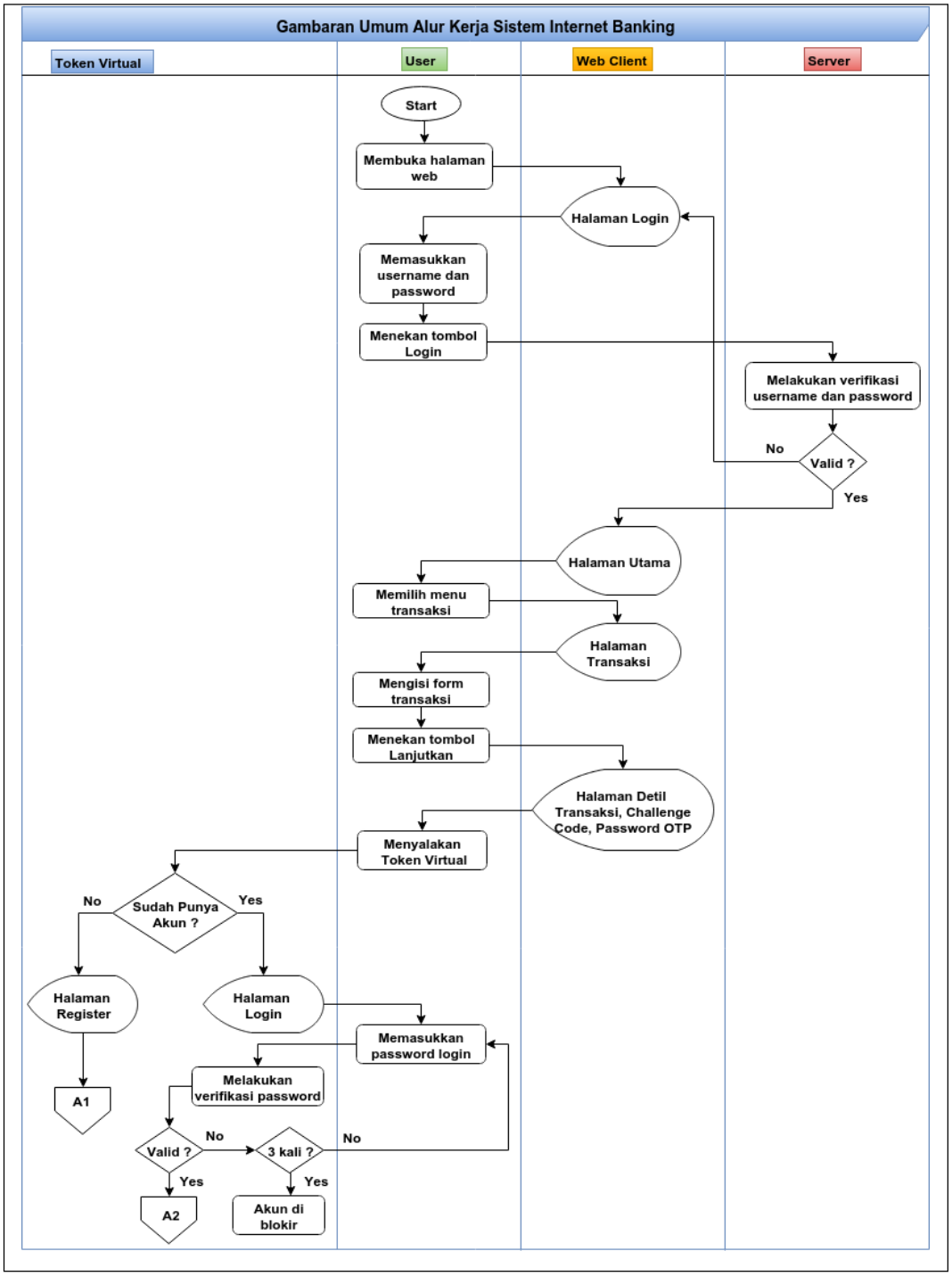
3.2 Alur Kerja Utama Sistem *Internet Banking*

Prosedur alur kerja utama sistem *internet banking* dijabarkan pada Tabel 11.

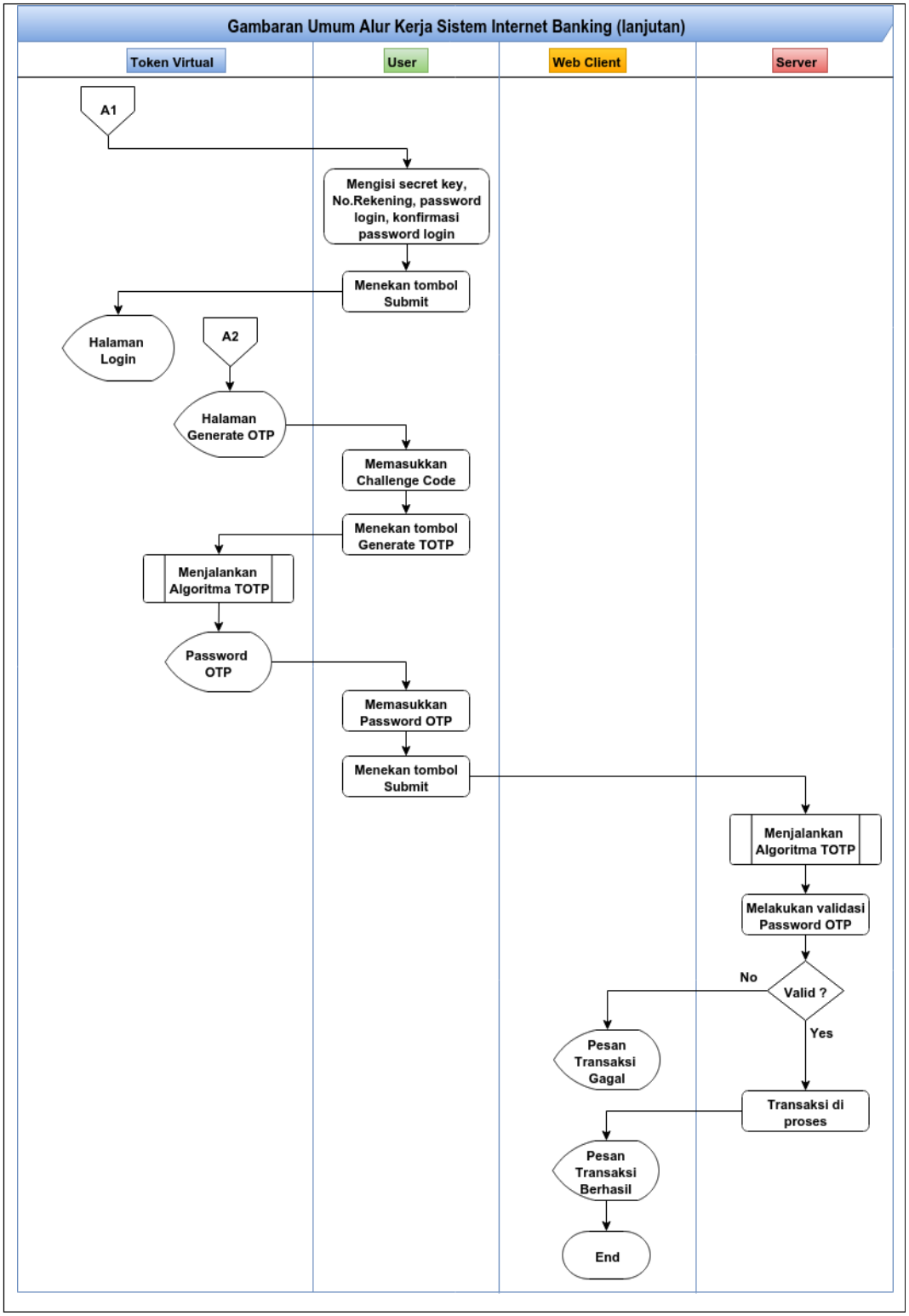
Tabel 11. Prosedur alur kerja utama sistem internet banking

Alur Kerja Utama Sistem Internet Banking		
Input	**Output**	**Stakeholder**
1. *Username* dan *password login* 2. Data transaksi 3. *Password* OTP	Terlaksananya proses transaksi finansial.	*Token Virtual, User, Web Client* dan *Web Server*
PROSEDUR : 1. User membuka halaman *web internet banking* : localhost:9000 2. Web client menampilkan halaman login. 3. User memasukkan username dan password. 4. User menekan tombol login. 5. Web server melakukan verifikasi username dan password. 6. Web client menampilkan halaman utama. 7. User memilih menu transaksi. 8. User mengisi form transaksi. 9. User menekan tombol lanjutkan. 10. Web client menampilkan halaman detil transaksi dan *challenge code* kemudian meminta *password* OTP. 11. User menyalakan token virtual. 12. Jika token virtual belum terdaftar maka harus dilakukan registrasi terlebih dahulu. 13. Jika sudah maka dapat melakukan login dengan memasukkan password login. 14. Token virtual melakukan verifikasi password login. 15. Token virtual menampilkan halaman Generate OTP. 16. User memasukkan challenge code. 17. Token virtual menjalankan algoritma TOTP. 18. Token virtual menampilkan password OTP. 19. User memasukkan password OTP pada Web Client. 20. User menekan tombol submit. 21. Server menjalankan algoritma TOTP. 22. Server melakukan validasi password OTP antara token virtual dengan server. 23. Jika valid maka web client menampilkan pesan bahwa transaksi berhasil. 24. Jika tidak maka web client menampilkan pesan bahwa password OTP tidak valid.		

Gambaran alur kerja utama sistem *internet banking* dapat di lihat pada Gambar 17 dan Gambar 18.



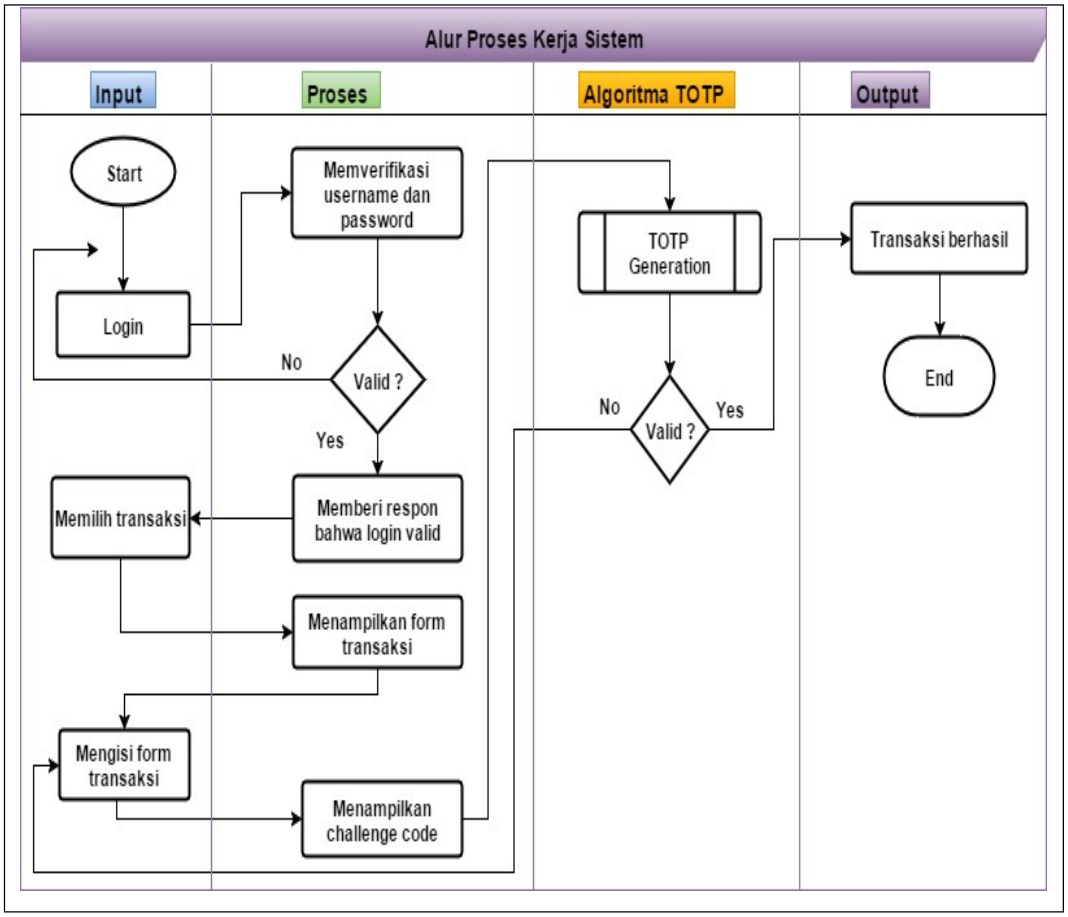
Gambar 17. Alur kerja utama sistem internet banking



Gambar 18. Alur kerja utama sistem internet banking (lanjutan)

3.2.1 Alur Kerja *Server Internet Banking*

Alur proses kerja server internet banking dapat dilihat pada Gambar 19.



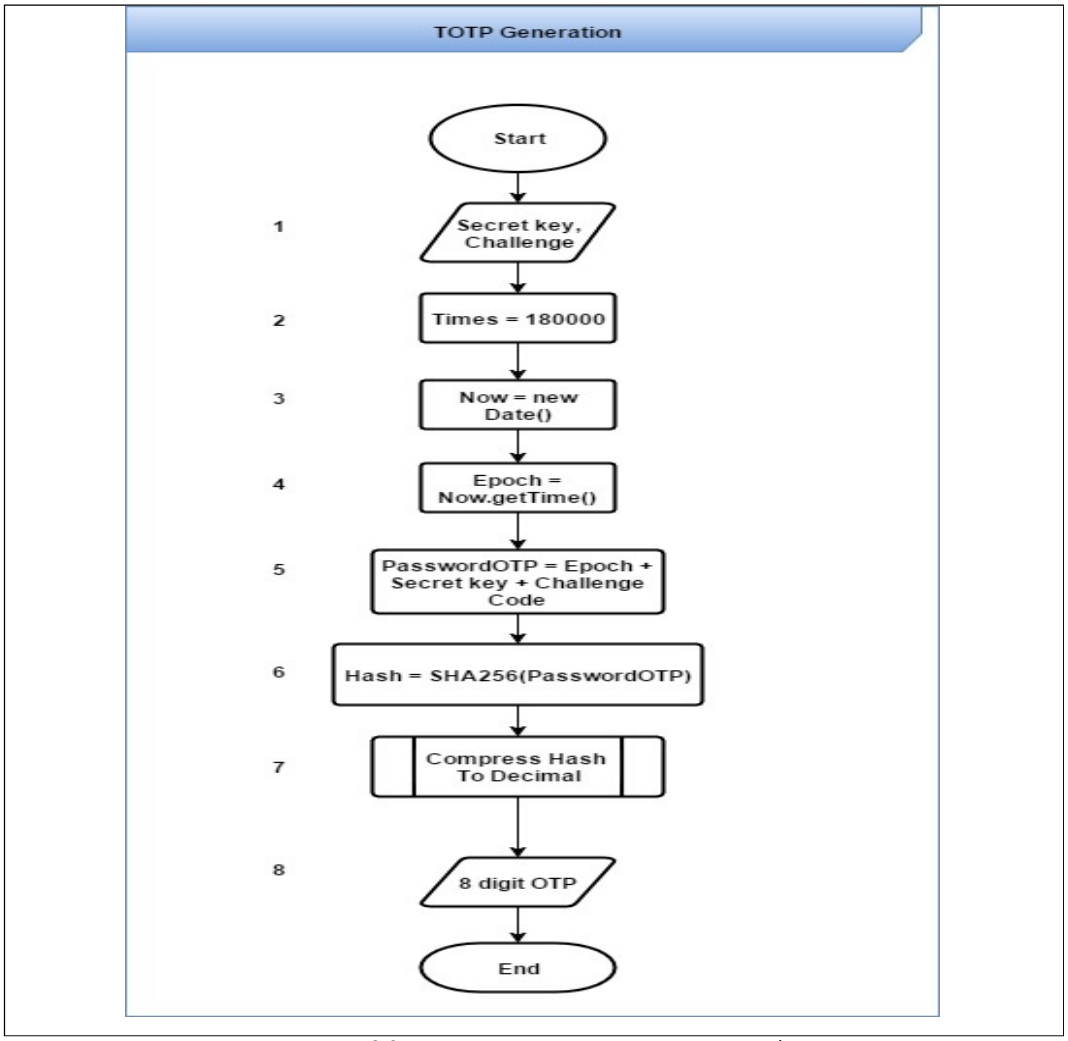
Gambar 19. Alur Proses Kerja Sistem

Proses yang terdapat pada Gambar 19 dijabarkan sebagai berikut :

1. Proses yang pertama adalah *login* dengan memasukkan *username* dan *password* yang dilakukan oleh *user* kemudian *server* melakukan verifikasi terhadap *username* dan *password user* tersebut apakah valid atau tidak.
2. Setelah *username* dan *password* dianggap valid oleh *server*, maka *user* dapat masuk ke dalam sistem untuk melakukan transaksi misalnya transaksi untuk pengiriman uang, tetapi jika tidak maka *user* harus memasukkan *username* dan *password* kembali.
3. Sebelum transaksi di proses oleh sistem, maka sistem memberikan sederetan angka yang disebut dengan *challenge code* sebagai otentikasi tambahan.

4. *User* kemudian memasukkan *challenge code* tersebut ke dalam *android* token, dan algoritma TOTP dijalankan.
5. Proses kerja algoritma TOTP akan menghasilkan 8 digit *password* OTP yang dijelaskan pada Gambar 20.
6. Jika *password* OTP valid maka transaksi akan berhasil, tetapi jika tidak maka akan kembali ke *form* transaksi untuk menghasilkan *challenge code* yang baru.

Proses *TOTP Generation* untuk menghasilkan 8 digit password OTP dapat dilihat pada Gambar 20.



Gambar 20. Proses TOTP Generation

Proses yang terdapat pada Gambar 20 dijabarkan sebagai berikut :

1. Inisialisasi *secret key* dan *challenge code.* ***Secret key*** dihasilkan dari kombinasi angka secara acak menggunakan *library java.util.random* dan bersifat unik. Cara menghasilkan *secret key* adalah :
   1) Inisialisasi angka minimal = 0
   2) Inisialisasi angka maksimal = 9
   3) Inisialisasi panjang data = 10 digit.
   4) Melakukan kombinasi angka antara 0 sampai 9 secara acak menggunakan *library java.util.random*.

   Berikut hasil kombinasi *secret key* yang dihasilkan secara acak pada Tabel 12.

Tabel 12. Hasil kombinasi secret key

Array [ ]	Secret Key
Array [1]	256444175
Array [2]	658685494
Array [3]	943771664
Array [4]	884122147
Array [5]	852750373

Challenge code berupa 8 digit angka unik yang merupakan kombinasi dari 2 digit pertama secara acak dan 6 digit dari 6 digit terakhir nomor rekening pada Tabel 13.

Tabel 13.Hasil kombinasi challenge code

User	No. Rekening	Transaksi	Challenge Code
Uung	0552276910	1	32276910
		2	21276910
		3	53276910
		4	82276910
		5	16276910
Putra	0241758694	1	54758694
		2	66758694
		3	21758694
		4	18758694
		5	61758694

2. *Times* adalah proses untuk menentukan masa berlaku *password* OTP selama 3 menit = 180000 milidetik.
3. Proses mengambil tanggal saat ini.
4. Proses *epoch* yaitu mengambil jumlah milidetik dari tanggal saat ini yang digunakan sebagai sinkronisasi waktu.

   Berikut hasil *epoch* pada Tabel 14.

Tabel 14. Hasil nilai epoch

Tanggal	Epoch
14-01-2016 00:00:00	1452877200000
15-01-2016 00:03:00	1452773800000
16-01-2016 00:06:00	1452877560000

Untuk menghitung nilai *time counter* menggunakan rumus persamaan (5).

TC = Epoch/ Times ……..…… **(5)**

Time counter dapat dilihat pada Tabel 15.

Tabel 15. Hasil nilai time counter

Epoch	Times	TC
1452877200000	180000	9072540
1452773800000	180000	9071541
1452877560000	180000	9071542

5. Proses penggabungan *secret key*, *time counter* dan *challenge code* dapat dilihat pada Tabel 16.

Tabel 16. Hasil penggabungan secret key, challenge code dan time counter

Secret Key	Challenge Code	TC	Password OTP
256444175	54758694	9072540	25644417554758694907254 0
256444175	54758694	9071541	25644417554758694907154 1
256444175	54758694	9071542	25644417554758694907154 2

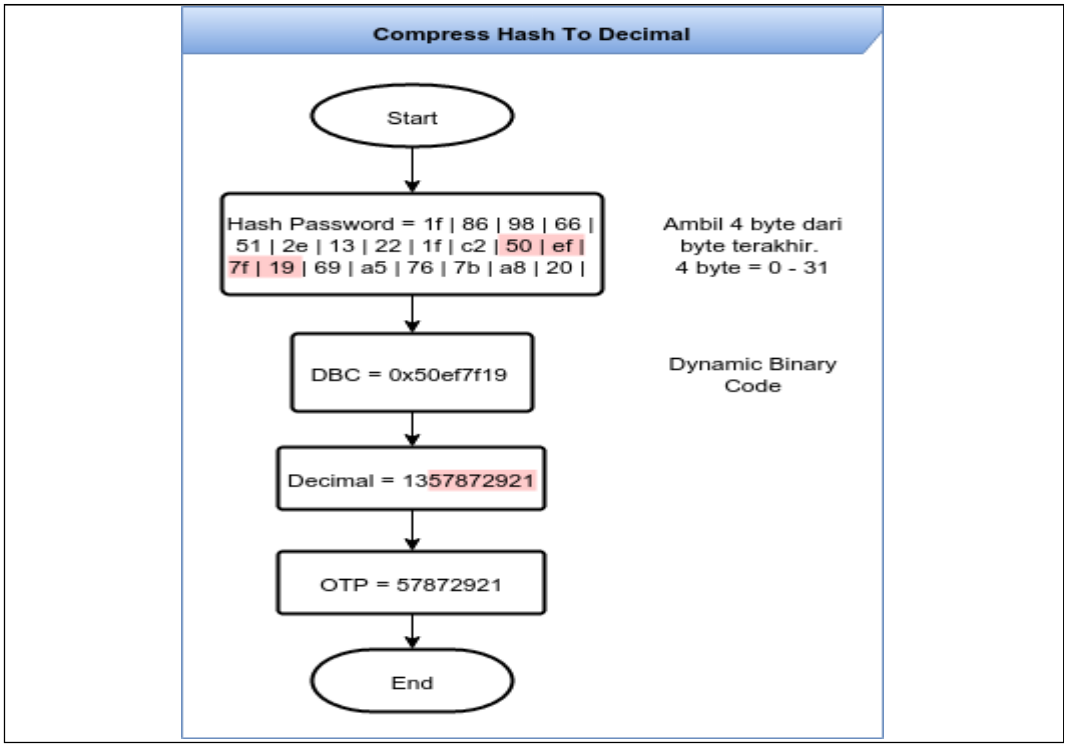
6. Proses *hashing* dari *password* OTP menggunakan algoritrma enkripsi *SHA256* dapat dilihat pada Tabel 17.

Tabel 17. Proses hashing password OTP menggunakan SHA256

Password OTP	SHA256
25644417554758694907254 0	264EED90935E78
25644417554758694907154 1	2BA78A56D14B4C
25644417554758694907154 2	9C9DE8622A3B88

7. Proses kompresi dari hasil *hashing* menjadi bilangan decimal dijabarkan sebagai berikut :
   a. *Hash password* merupakan hasil enkripsi *password* OTP menggunakan algoritma *SHA256*.
   b. Melakukan *truncate* atau pemotongan 4 *byte* terakhir dari *hash password* yaitu 32 bit dari 0 - 31.
   c. Hasil *truncate* dari *hash password* di konversi menjadi bilangan *decimal* menggunakan operasi matematika.
   d. *OTP password* diambil dari 8 bit dari hasil bilangan *decimal*.

Flowchart proses kompresi hasil *hashing* menjadi bilangan decimal dapat dilihat pada Gambar 21.



Gambar 21. Proses kompresi password hash menjadi decimal

Hasil kompresi *hash password* menjadi bilangan decimal dapat dilihat pada Tabel 18.

Tabel 18. Hasil kompresi hash password

SHA256	Decimal
264EED90935E78	10782831354797687
2BA78A56D14B4C	12287636602440524
9C9DE8622A3B88	44083717771770759

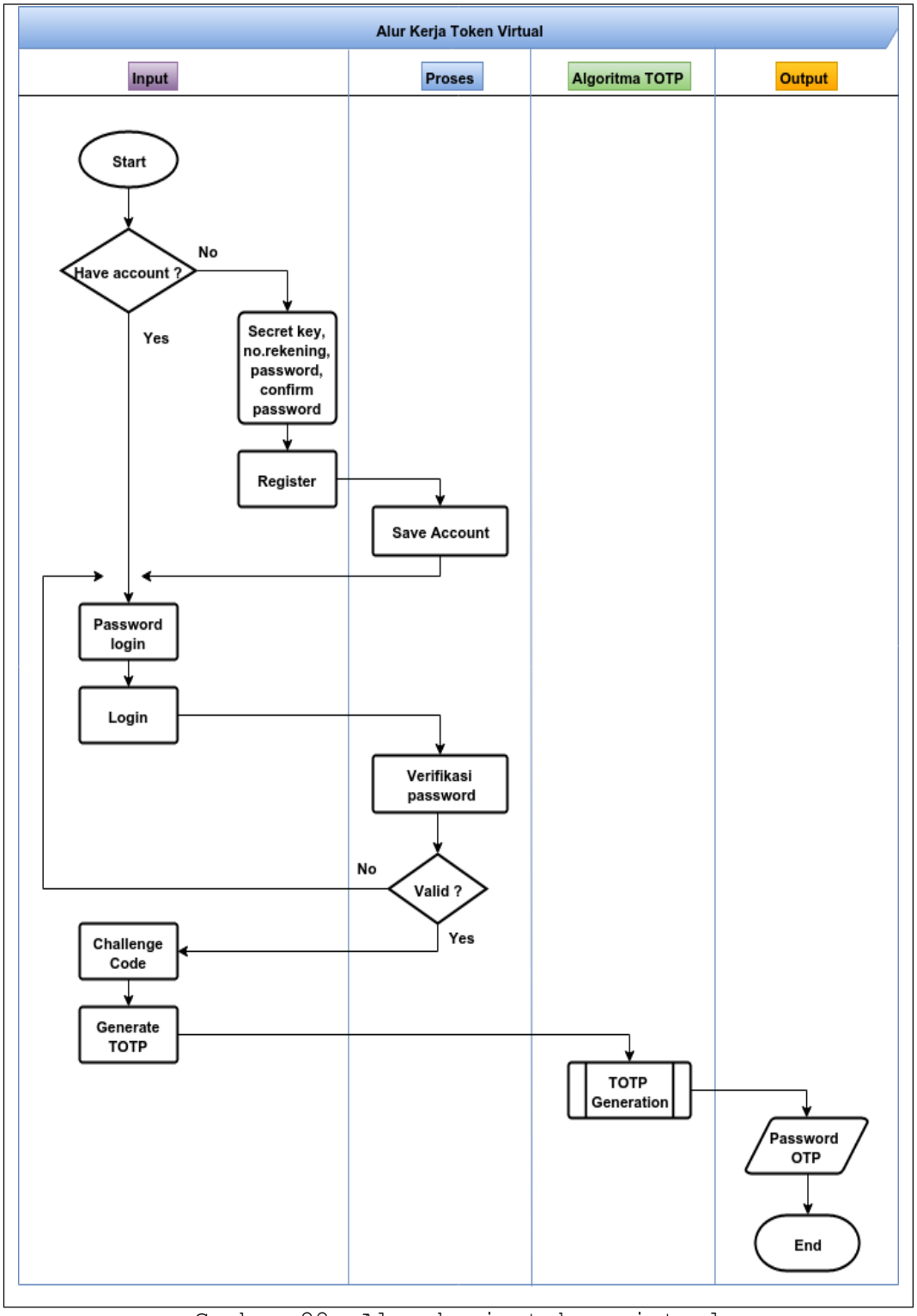
8. Melakukan *truncate* terhadap bilangan decimal sebanyak 8 digit yang dijadikan OTP dapat dilihat pada Tabel 19.

Tabel 19. Hasil password OTP

Decimal	8 digit OTP
10782831354797687	54797687
12287636602440524	02440524
44083717771770759	71770759

3.2.2 Alur Kerja Token *Virtual*

Alur proses kerja token virtual dapat dilihat pada Gambar 22.



Gambar 22. Alur kerja token virtual

Proses yang terdapat pada Gambar 22 dijabarkan sebagai berikut :

1. Token virtual memeriksa apakah akun sudah diregistrasi.
2. Jika tidak, user harus melakukan register dengan memasukkan secret key, nomor rekening, password dan confirm password.
3. Jika sudah diregistrasi *user* dapat melakukan *login* dengan memasukkan password *login*.
4. Token virtual memeriksa apakah *passsword* valid atau tidak
5. Jika tidak, sebanyak 3 kali login maka akun di blok.
6. Jika valid maka muncul menu generate password OTP.
7. User memasukkan challenge code dan menekan tombol *Generate* TOTP.
8. Token *virtual* menampilkan *password* OTP.

3.3 Perancangan

Pada tahap ini dilakukan perancangan berdasarkan *functional requirement*. Berikut adalah *functional requirement* dari aplikasi yang dibangun :

1. *User* harus melakukan *login* untuk dapat mengakses *web internet banking*.
2. *User* dapat memilih aktifitas favorit pada *web internet banking*.
3. *User* dapat mencari data saldo rekening dengan kategori pencarian berdasarkan nama atau jenis mata uang.
4. *User* dapat mencetak informasi saldo rekening ke dalam bentuk *file* pdf.
5. *User* dapat melihat histori transaksi dari rekening yang dipilih.
6. *User* dapat melakukan transfer uang ke rekening sesama atau antar rekening.
7. *User* dapat melihat detil informasi rekening berdasarkan nomor rekening yang dipilih.
8. *User* dapat menambah rekening tujuan pada *web internet banking*.
9. *User* dapat mencari kode bank lainnya jika ingin melakukan transfer antar bank.
10. *User* dapat melakukan *logout* untuk keluar dari *web internet banking*.
11. Sistem dapat melakukan validasi terhadap *username* dan *password* dari *user*.

12. Sistem dapat menampilkan aktifitas favorit dari *internet banking*.
13. Sistem dapat menampilkan informasi saldo rekening yang di cari oleh *user*.
14. Sistem dapat menampilkan histori transaksi yang dilakukan oleh *user*.
15. Sistem dapat menampilkan detil rekening yang di pilih oleh *user*.
16. Sistem harus dapat menampilkan daftar kode bank.
17. Sistem harus dapat menampilkan detil transaksi.
18. Sistem harus dapat menghasilkan *challenge code*.

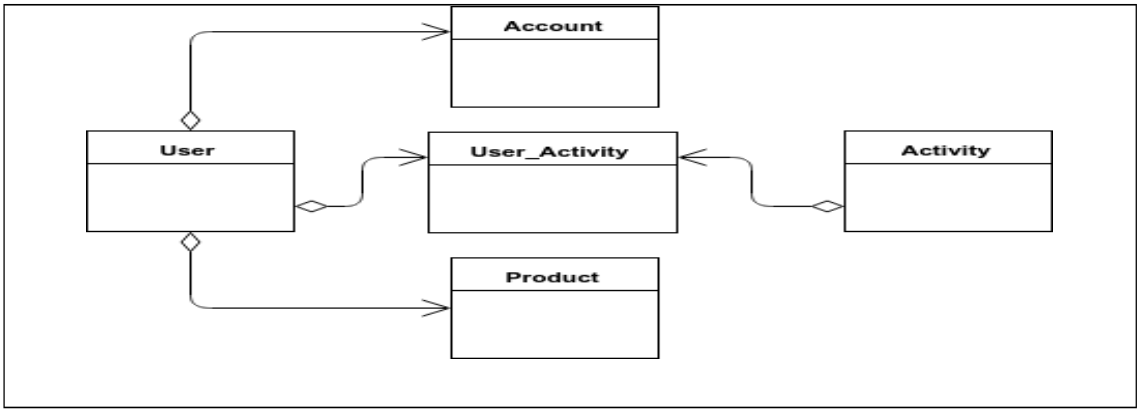
3.3.1 Perancangan Sistem

Perancangan sistem yang dilakukan pada penelitian ini menggunakan metode permodelan *Iconix Process* untuk merancang suatu perangkat lunak dan sebagai standar penulisan dengan memaparkan beberapa informasi dalam proses implementasi perangkat lunak tersebut.

Pada perancangan sistem aplikasi token *internet banking* mengunakan *functional requirement*, *domain model*, *use case diagram*, skenario *use case*, *class diagram*, *activity diagram* dan *sequence diagram* untuk menggambarkan dan menjelaskan alur aplikasi.

3.3.1.1 Domain Model

Domain model berfungsi untuk menyamakan istilah yang akan di pakai pada sistem. Pada *domain model* menggunakan relasi pewarisan (**is a**) dan agregrasi (**has a**) dan belum memiliki atribut dan operasi. Berikut *domain model* sistem *internet banking* berdasarkan perancangan pada Gambar 23.



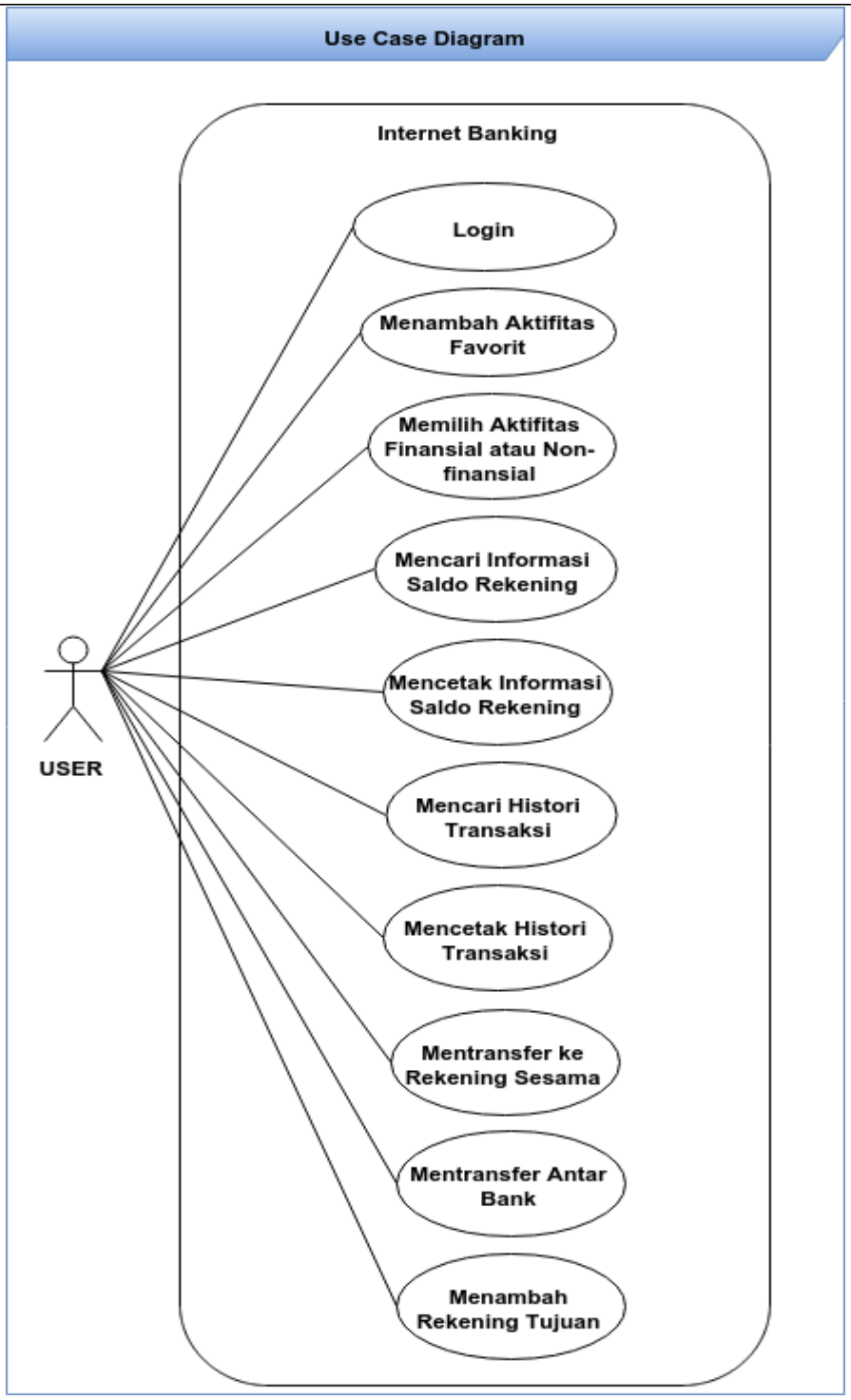
Gambar 23. Domain model sistem internet banking

Pada *domain model* pada Gambar 23 terdapat beberapa *model* yang saling berhubungan di dalam sistem. Berikut deskripsi dari masing-masing *model* :

1. **Model User**, model ini berisi informasi *user* yang dapat mengakses sistem.
2. **Model Account**, model ini berisi informasi rekening yang dimiliki oleh oleh *user*.
3. **Model Product,** *model* ini berisi informasi produk yang dimiliki oleh bank, contohnya : KTM, Tabungan Haji dan lain-lain.
4. **Model Activity**, *model* ini berisi informasi aktifitas-aktifitas yang terdapat pada *internet banking* seperti aktifitas finansial dan aktifitas non-finansial.
5. **Model User_Activity**, *model* ini muncul karena adanya relasi banyak ke banyak antara *model user* dengan *model activity*.

3.3.1.2 Use Case Diagram

Use Case Diagram digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. *Use Case Diagram* terdiri atas diagram untuk *use case* dan *actor*. Pada bagian ini *use case* meliputi *use case login*, menambah aktifitas favorit, memilih aktifitas finansial dan non-finansial, mencari informasi saldo rekening, mencetak informasi saldo rekening, mencari histori transaksi, mencetak histori transaksi, mentransfer ke rekening sesama, mentransfer antar bank dan menambah rekening tujuan. *Use Case Diagram* dapat dilihat pada Gambar 24.



Gambar 24.Use Case Diagram sistem internet banking

Pada *use case* yang terdapat pada Gambar 24, hanya terdapat satu *actor* yaitu *user*, serta beberapa fungsionalitas dari *user* dan sistem. Berikut adalah deskripsi dari masing-masing fungsionalitas yang terdapat pada Gambar 24.

1. **Fungsionalitas *Login***, pada fungsionalitas ini *user* pertama kali memasukkan *username* dan *password* dan mengklik tombol login yang berfungsi sebagai *trigger* untuk masuk ke dalam sistem.
2. **Fungsionalitas Menambah Aktifitas Favorit**, pada fungsionalitas ini *user* dapat menambah aktifitas favorit yang bertujuan untuk menentukan kegiatan yang sering dilakukan ketika mengakses *internet banking*.
3. **Fungsionalitas Memilih Aktifitas Finansial atau Non-finansial**, pada fungsionalitas ini sebagai akses cepat bagi *user* untuk masuk ke menu aktifitas yang sifatnya finansial atau non-finansial.
4. **Fungsionalitas Mencari Informasi Saldo Rekening**, pada fungsionalitas ini *user* dapat mencari informasi beberapa saldo rekening yang terdaftar pada internet banking.
5. **Fungsionalitas Mencetak Informasi Saldo Rekening**, pada fungsionalitas ini *user* dapat mencetak informasi saldo rekening yang telah di cari ke dalam bentuk file pdf.
6. **Fungsionalitas Mencari Histori Transaksi**, pada fungsionalitas ini *user* dapat mencari histori transaksi baik berupa pengiriman uang atau penarikan uang.
7. **Fungsionalitas Mencetak Histori Transaksi**, pada fungsionalitas ini *user* dapat mencetak histori transaksi yang telah di cari ke dalam bentuk *file* pdf.
8. **Fungsionalitas Mentransfer ke Rekening Sesama**, pada fungsionalitas ini *user* mengisi *form transfer* dan sistem menjalakan algoritma TOTP.
9. **Fungsionalitas Mentransfer Antar bank**, pada fungsionalitas ini *user* mengisi *form transfer* dan menambahkan kode rekening bank tujuan serta sistem menjalankan algoritma TOTP.

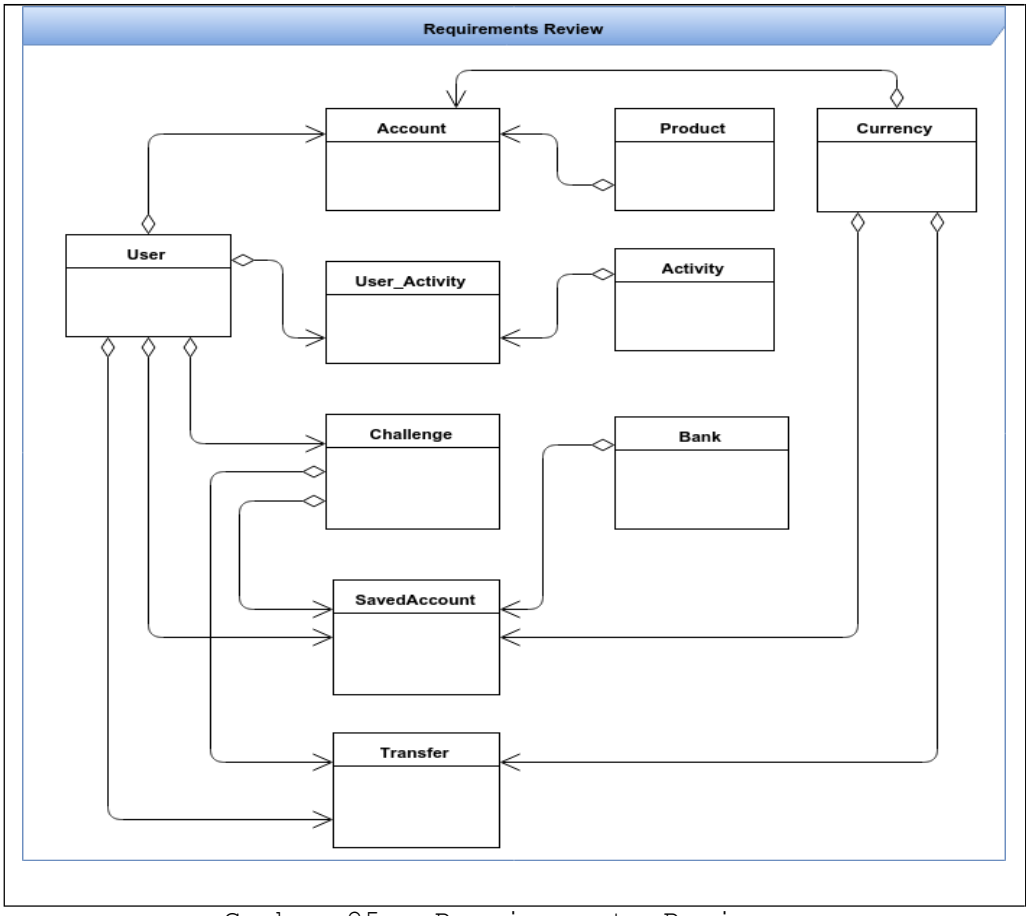
10. **Fungsionalitas Menambah rekening tujuan**, pada fungsionalitas ini *user* dapat menambahkan data rekening tujuan pada sistem agar saat melakukan transfer, secara otomatis bisa dipilih pada *dropdown*.

3.3.1.3 Skenario *Use Case*

Untuk memaparkan kejelasan dalam proses *logic* dalam melaksanakan fungsionalitas-fungsionalitas yang diterapkan oleh sistem, maka perlu dijabarkan skenario sistem sesuai dengan *use case* dari setiap kasus yang ada. Berikut adalah tabel skenario dari dari masing-masing kasus yang ditunjukkan pada Lampiran A Skenario *Use Case*.

3.3.1.4 Requirements Review

Berdasarkan metode *Iconix Process*, *requirements review* bertujuan untuk memastikan bahwa *domain model* dan *use case diagram* telah dibuat dengan baik. Setelah dilakukan peninjauan, ternyata pada *domain model* masih memiliki kekurangan sehingga harus diperbaiki relasi antar *model* dan juga perlu ditambahkan beberapa *model* lainnya seperti *model* Bank, *model* Currency, model Challenge, *model* SavedAccount dan *model* Transfer yang dijelaskan pada Gambar 25.



Gambar 25. Requirements Review

Berikut deskripsi dari beberapa *model* yang ditambahkan tersebut :

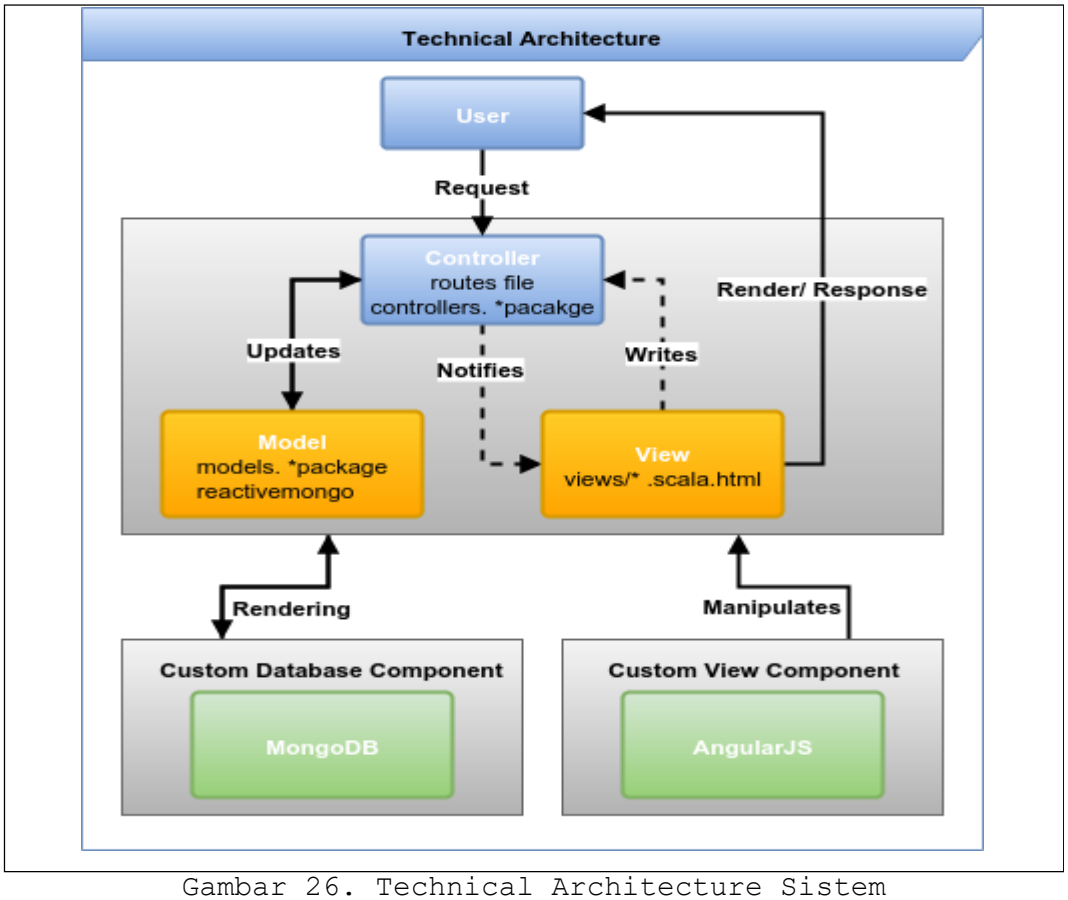
1. **Model Bank**, *model* ini berisi informasi tentang bank-bank tujuan pengiriman uang.
2. **Model Currency**, *model* ini berisi informasi tentang mata uang yang digunakan.
3. **Model Challenge**, *model* ini untuk menyimpan *challenge code* yang yang muncul saat melakukan transaksi.
4. **Model SavedAccount**, model ini untuk menyimpan rekening baru saat *user* melakukan penambahan rekening tujuan.
5. **Model Transfer**, model ini untuk menyimpan data transaksi pengiriman uang yang dilakukan oleh *user*.

3.3.1.5 Class Diagram

Class Diagram adalah diagram *Unified Modeling Language* (UML) yang terdapat pada metode *Iconix Process* yang merupakan pengembangan dari *domain model* yaitu dengan menambahkan atribut dan operasinya. Berdasarkan tabel skenario yang dijelaskan pada tabel Lampiran A Skenario *Use Case* nomor 1 sampai 13 maka diperoleh *Class Diagram* yang ditunjukan pada Lampiran B *Class Diagram*.

3.3.1.6 Technical Architecture

Technical Architecture adalah diagram UML yang terdapat pada metode *Iconix Process* yang bertujuan untuk merancang arsitektur dari sistem seperti penggunaan *framework*, penggunaan *database* dan teknologi yang dipakai untuk melakukan desain tampilan sistem. Untuk lebih memahami gambaran tentang *Technical Architecture* dari sistem, maka dapat dilihat pada `Gambar 26`.



Gambar 26. Technical Architecture Sistem

Aplikasi token *internet banking* mengikuti pola arsitektur *Model View Controller* (MVC). Pola ini memisahkan aplikasi ke dalam beberapa *layer* yaitu *Presentation Layer* dan *Model Layer*. *Presentation Layer* dibagi menjadi 2 yaitu *View* dan *Controller Layer*.

1. ***Model***, yaitu sebuah *domain* yang merepresentasikan informasi dari operasi yang akan di proses. Contohnya seperti penghitungan total, presentase dan lain-lain. *Model* juga melakukan enkapsulasi data yang dapat digunakan sebagai *rendering* ke database menjadi sebuah tabel.
2. ***View***, yaitu melakukan *rendering* terhadap *model* menjadi sebuah *form* sebagai *user interface*. Dalam aplikasi *web*, *view* biasanya melakukan *rendering* terhadap *file* dengan format *.HTML*, *XML* dan *JSON*.
3. ***Controller***, yaitu merespon *event* yang dilakukan oleh *user* dan memprosesnya dan juga melakukan perubahan terhadap *model*. Dalam aplikasi *web*, *event-event* tersebut berupa *HTTP request* kemudian *controller* membaca *request* tersebut dan memprosesnya.
4. ***Custom Database Component*****,** yaitu mesin *database* yang digunakan adalah *MongoDB* yaitu sebuah salah satu *database NoSQL* karena data-data yang cukup besar pada aplikasi lebih baik melakukan *query* ke sebuah dokumen daripada melakukan join terhadap beberapa tabel di dalam *database* yang membuat proses semakin lama.
5. ***Custom View Component***, yaitu *framework* dari *front-end* yang digunakan adalah *AngularJS* yang sangat cocok untuk aplikasi yang kompleks karena fitur-fiturnya yang sangat mendukung seperti melakukan *load* data, *rendering template HTML* dan *request*/ *response* terhadap suatu *API*.

3.3.1.7 Activity Diagram

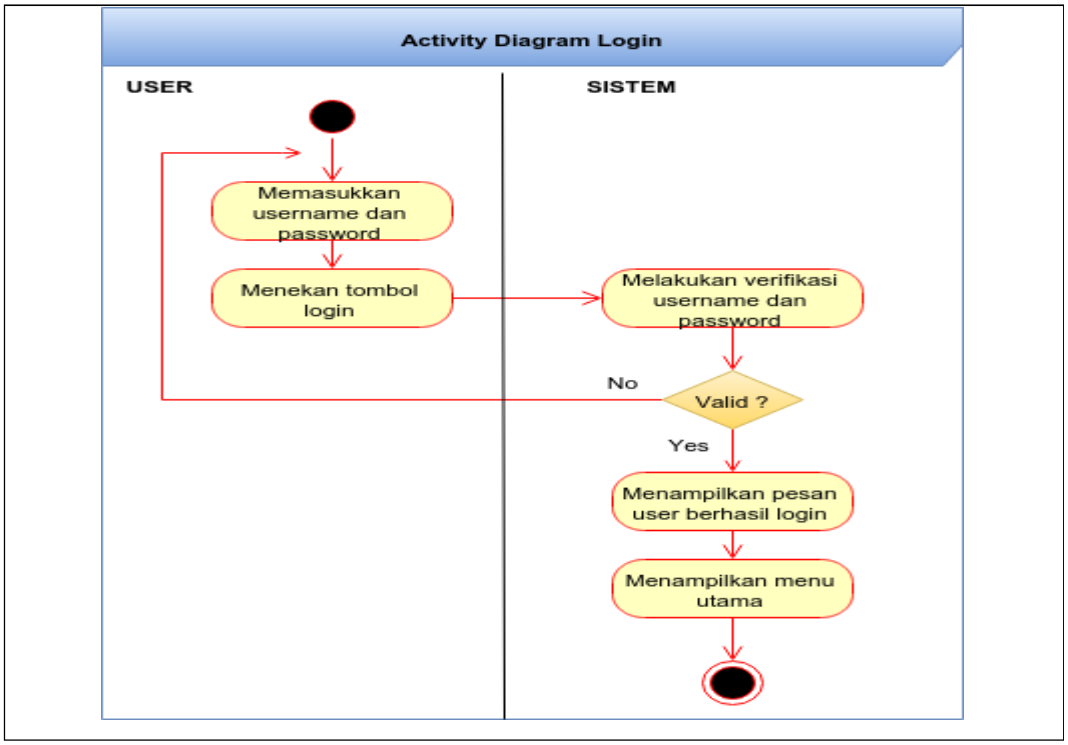
Activity Diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana awal dari masing-masing alur, keputusan yang mungkin terjadi dan bagaimana alur tersebut berakhir. *Activity Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi method. *Activity Diagram* merupakan *state* diagram khusus, diamana

sebagian besar *state* adalah *action* dan sebagian besar adalah transisi yang di-*trigger* oleh *internal processing*. Oleh karena itu, *Activity Diagram* tidak menggambarkan *behaviour internal* sebuah sistem, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Berdasarkan tabel skenario yang dijelaskan pada Lampiran A Skenario Use Case maka dapat diperoleh Activity Diagram. Berikut adalah Activity Diagram dari masing-masing Use Case yang telah dibuat.

3.3.1.7.1 Activity Diagram Login

Activity Diagram berdasarkan pada Lampiran A Skenario *Use Case* yaitu Tabel Lampiran 1 yang dijabarkan melalui gambar kegiatan yang *user* lakukan ketika melakukan *Login*. Fungsi *Login* tersebut dijabarkan pada Gambar 27.

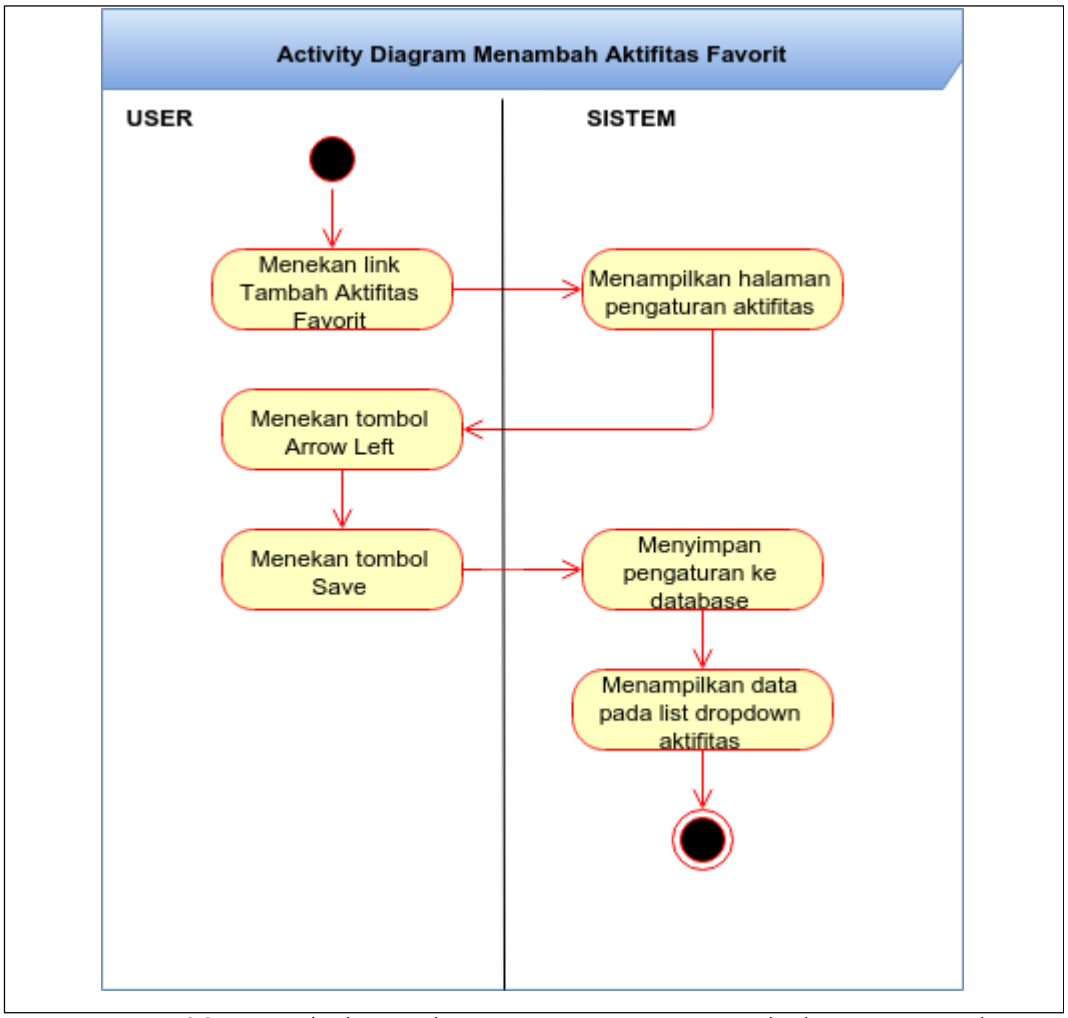


Gambar 27. Activity Diagram Login

3.3.1.7.2 *Activity Diagram* Menambahkan Aktifitas Favorit

Activity Diagram berdasarkan pada Lampiran A Skenario *Use Case* yaitu Tabel Lampiran 2 yang dijabarkan melalui gambar kegiatan yang *user*

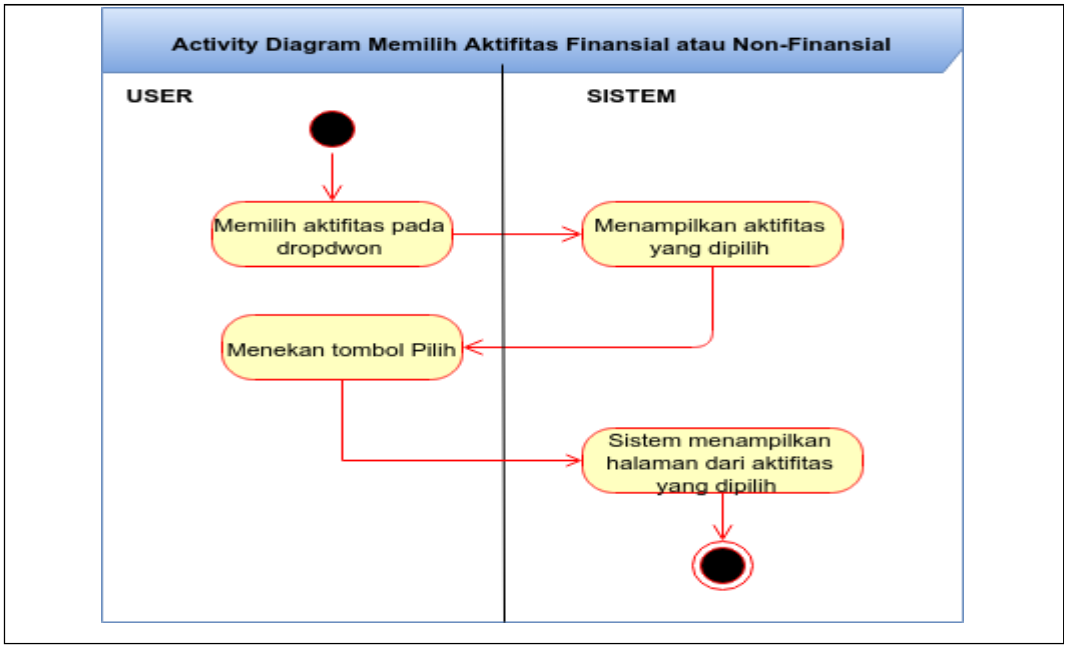
lakukan ketika menambah aktifitas yang sering dilakukan *user* dalam mengakses *web internet banking* menjadi aktivitas favorit. Fungsi menambahkan aktivitas favorit tesebut dijabarkan pada Gambar 28.



Gambar 28. Activity Diagram Menambah Aktifitas Favorit

3.3.1.7.3 *Activity Diagram* Memilih Aktifitas Finansial atau Non-Finansial

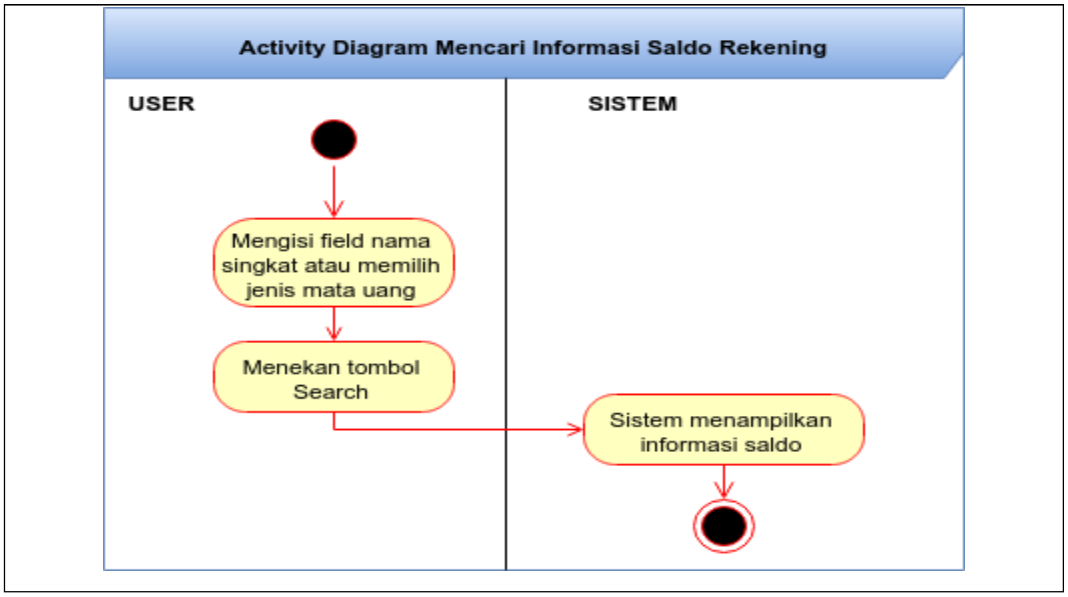
Activity Diagram berdasarkan pada Lampiran A Skenario *Use Case* yaitu Tabel Lampiran 3 yang dijabarkan melalui gambar kegiatan yang *user* lakukan ketika memilih aktifitas Finansial atau Non-Finansial pada *dropdown* sebagai akses cepat untuk menuju halaman yang diinginkan. Fungsi memilih aktivitas Finansial atau Non-Finansial dijabarkan pada Gambar 29.



Gambar 29. Activity Diagram Memilih Aktifitas Finansial atau Non-Finansial

3.3.1.7.4 *Activity Diagram* Mencari Informasi Saldo Rekening

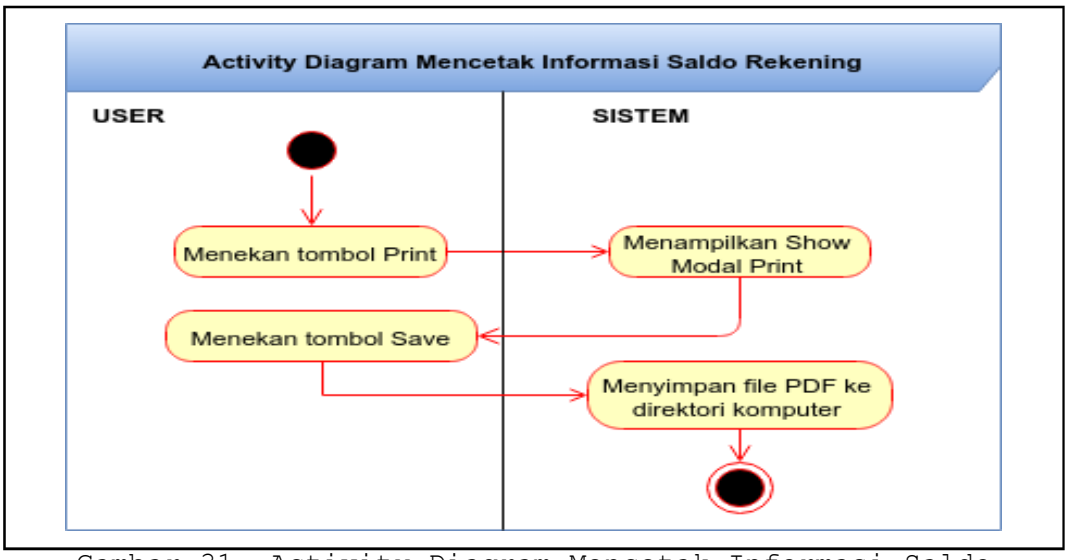
Pada tahap ini, *user* dapat mencari detil informasi rekening dari *user* yang sedang melakukan *login* seperti informasi saldo, nama pemilik rekening, nomor cif, tanggal pembukaan rekening dan lain-lain. Fungsi untuk mencari informasi saldo rekening dapat dilihat pada Gambar 30.



Gambar 30. Activity Diagram Mencari Informasi Saldo Rekening

3.3.1.7.5. *Activity Diagram* Mencetak Informasi Saldo Rekening

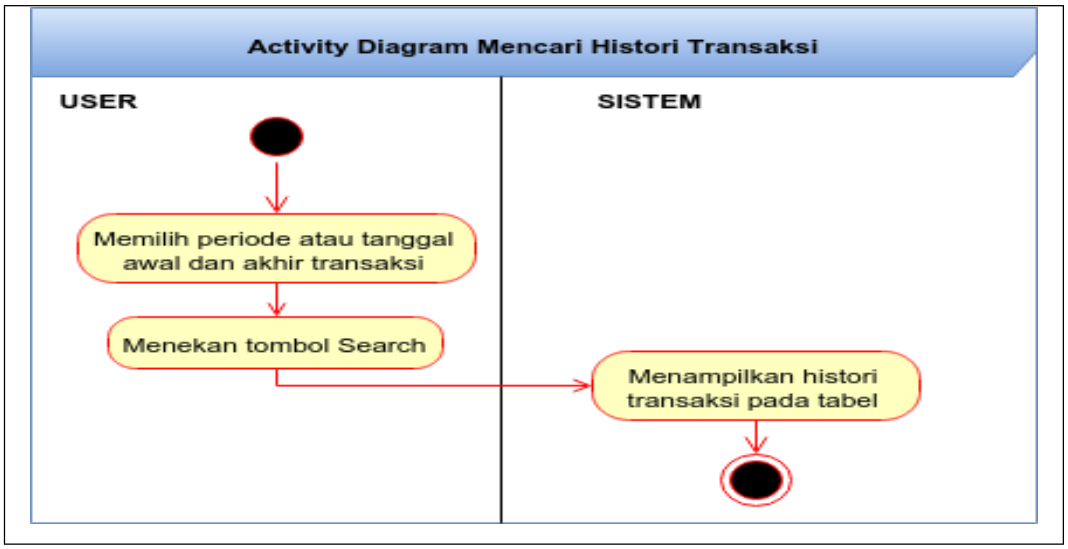
Pada tahap ini merupakan proses dimana *user* dapat mencetak informasi saldo rekening yang ada pada tabel ke dalam *file* pdf. Fungsi untuk mencetak informasi saldo rekening dapat dilihat pada Gambar 31.



Gambar 31. Activity Diagram Mencetak Informasi Saldo Rekening

3.3.1.7.6 *Activity Diagram* Mencari Histori Transaksi

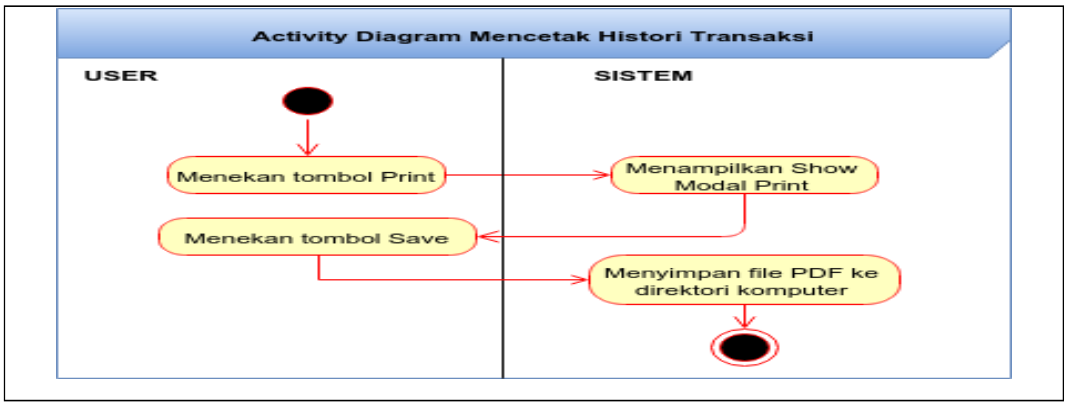
Pada tahap ini, *user* dapat mencari histori transaksi yang dilakukan dalam *internet banking* seperti penarikan atau pengiriman uang. Fungsi untuk mencari histori transaksi dapat dilihat pada Gambar 32.



Gambar 32. Activity Diagram Mencari Histori Transaksi

3.3.1.7.7 *Activity Diagram* Mencetak Histori Transaksi

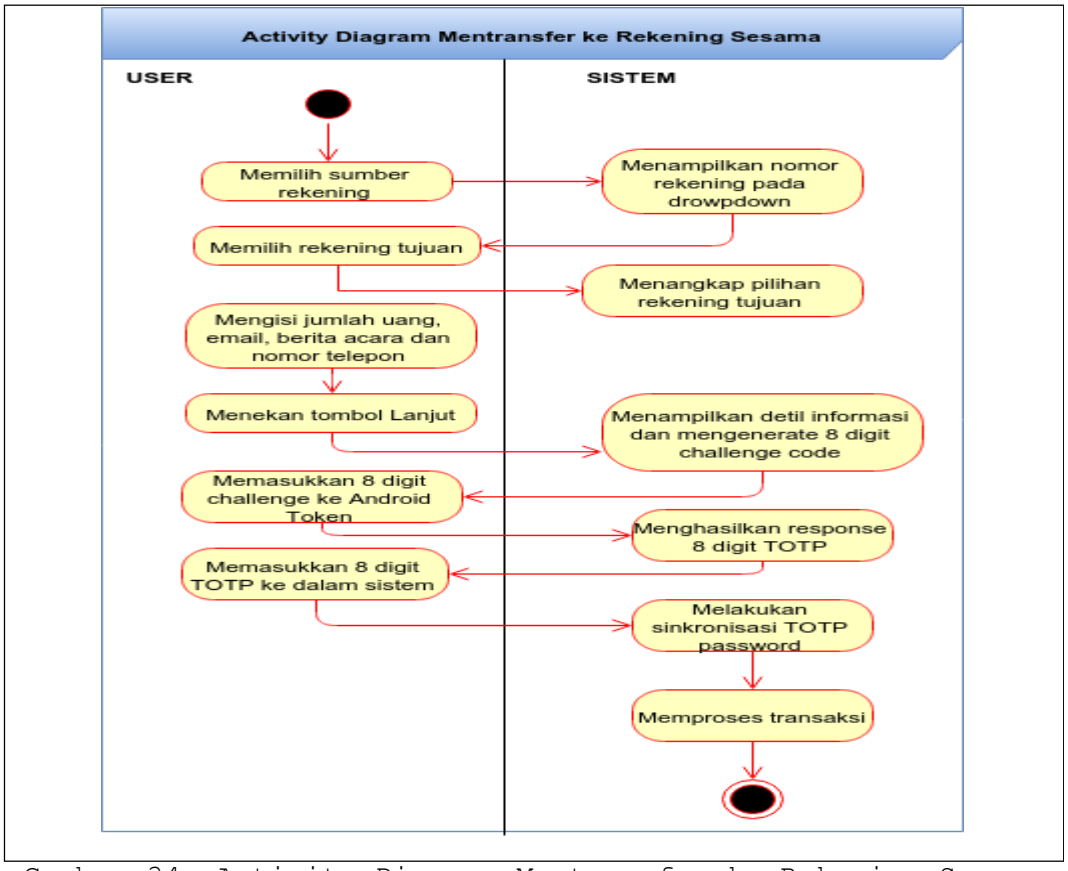
Pada tahap ini merupakan aktifitas mencetak histori tansaksi yang ada pada tabel ke dalam *file* pdf seperti pada Gambar 33.



Gambar 33. Activity Diagram Mencetak Histori Transaksi

3.3.1.7.8 *Activity Diagram* Mentransfer ke Rekening Sesama

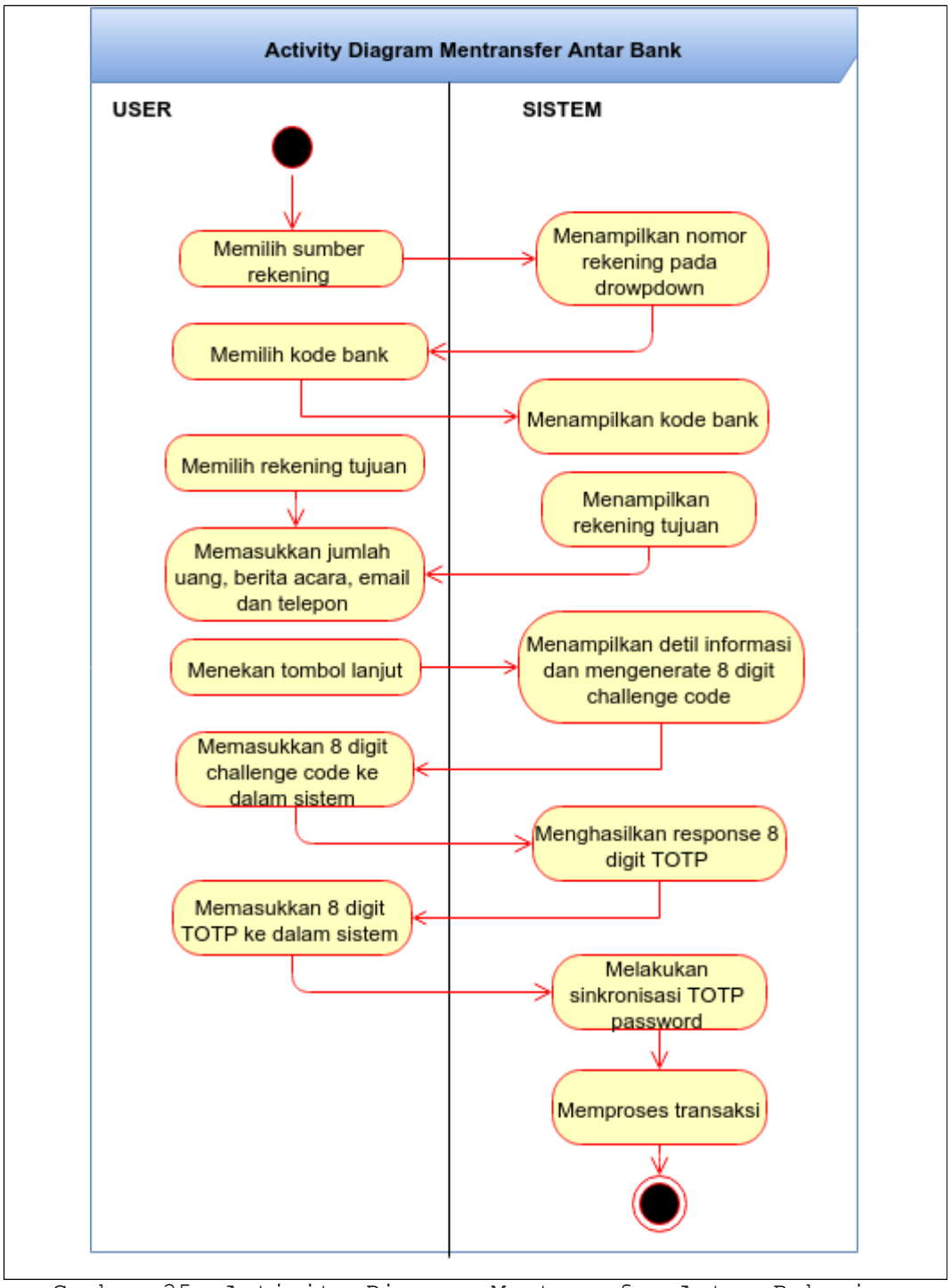
Pada tahap ini merupakan aktifitas mentransfer uang ke rekening sesame seperti pada Gambar 34.



Gambar 34. Activity Diagram Mentransfer ke Rekening Sesama

3.3.1.7.9 *Activity Diagram* Mentransfer Antar Bank

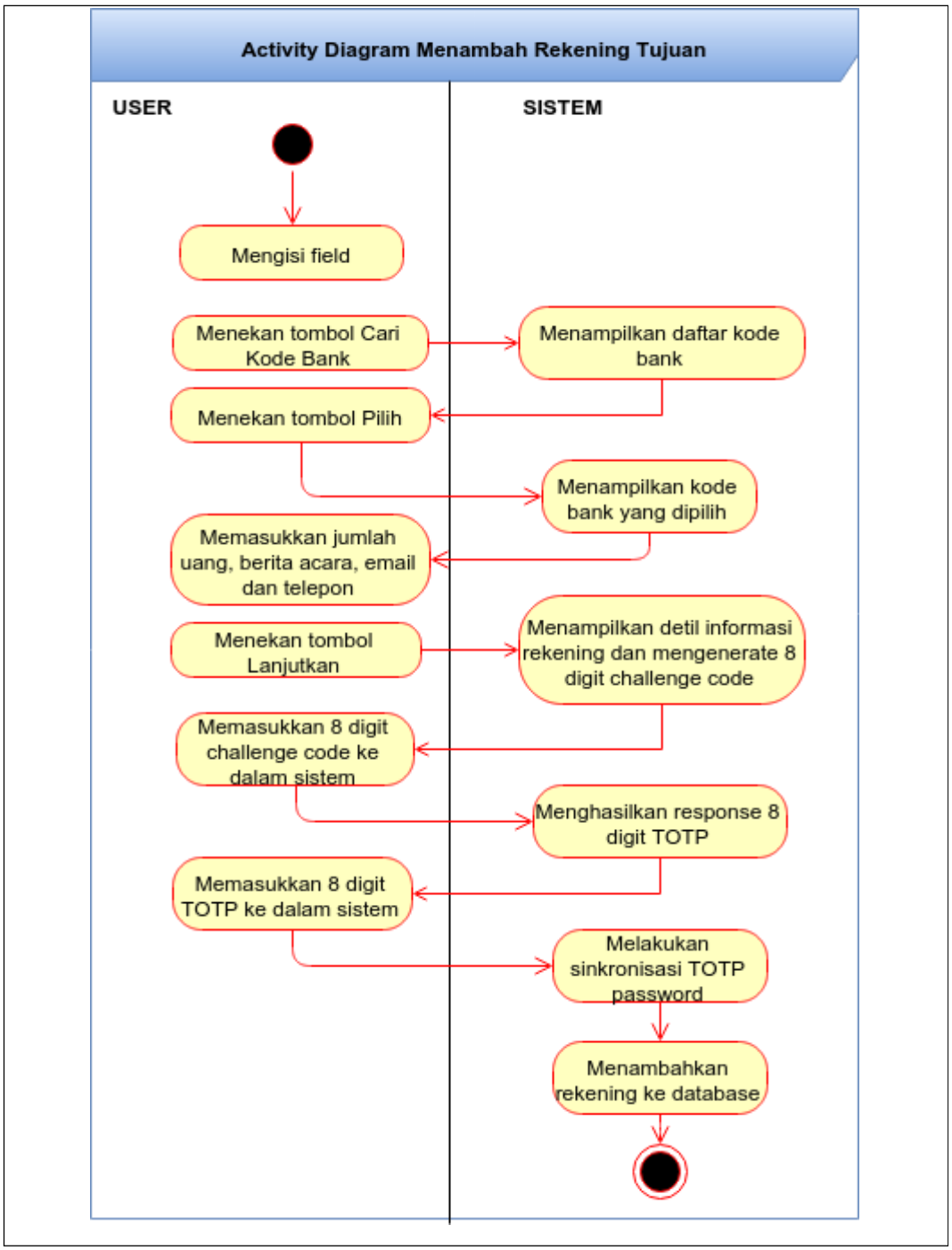
Pada tahap ini merupakan proses dimana *user* melakukan transfer uang ke sebuah rekening dengan bank yang berbeda. Fungsi untuk melakukan transfer antar bank dapat dilihat pada Gambar 35.



Gambar 35. Activity Diagram Mentransfer Antar Rekening

3.3.1.7.10 *Activity Diagram* Menambah Rekening Tujuan

Pada tahap ini merupakan proses dimana *user* menambahkan rekening tujuan baru ke dalam sistem. Fungsi untuk melakukan transfer antar bank dapat dilihat pada Gambar 36.



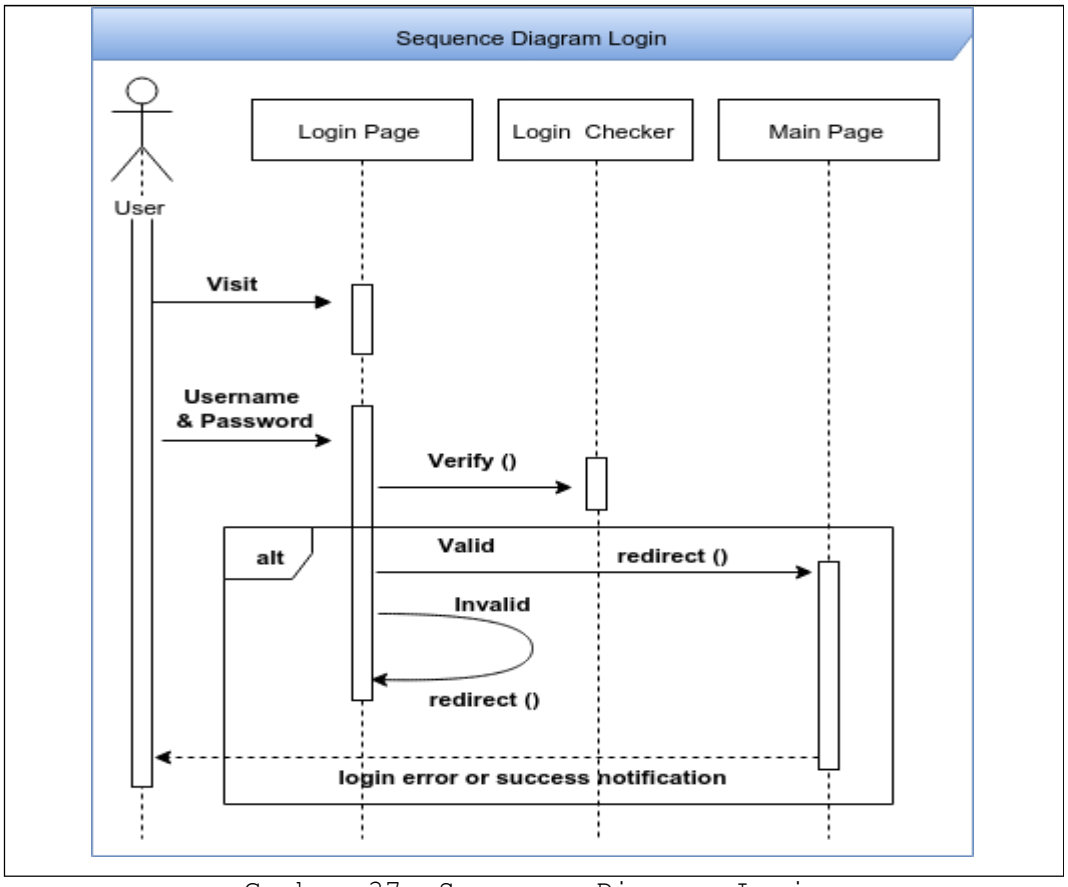
Gambar 36. Activity Diagram Menambah Rekening Tujuan

3.3.1.8 Sequence Diagram

Sequence Diagram digunakan untuk mengambarkan arus pekerjaan, pesan yang disampaikan dan bagaimana elemen-elemen didalamnya bekerja dari waktu ke waktu untuk mencapai hasil dari setiap *use case* yang sudah ada pada **sub bab 3.1.3.7**. Berikut adalah *sequence diagram* dari masing-masing *use case*.

3.3.1.8.1 Sequence Diagram Login

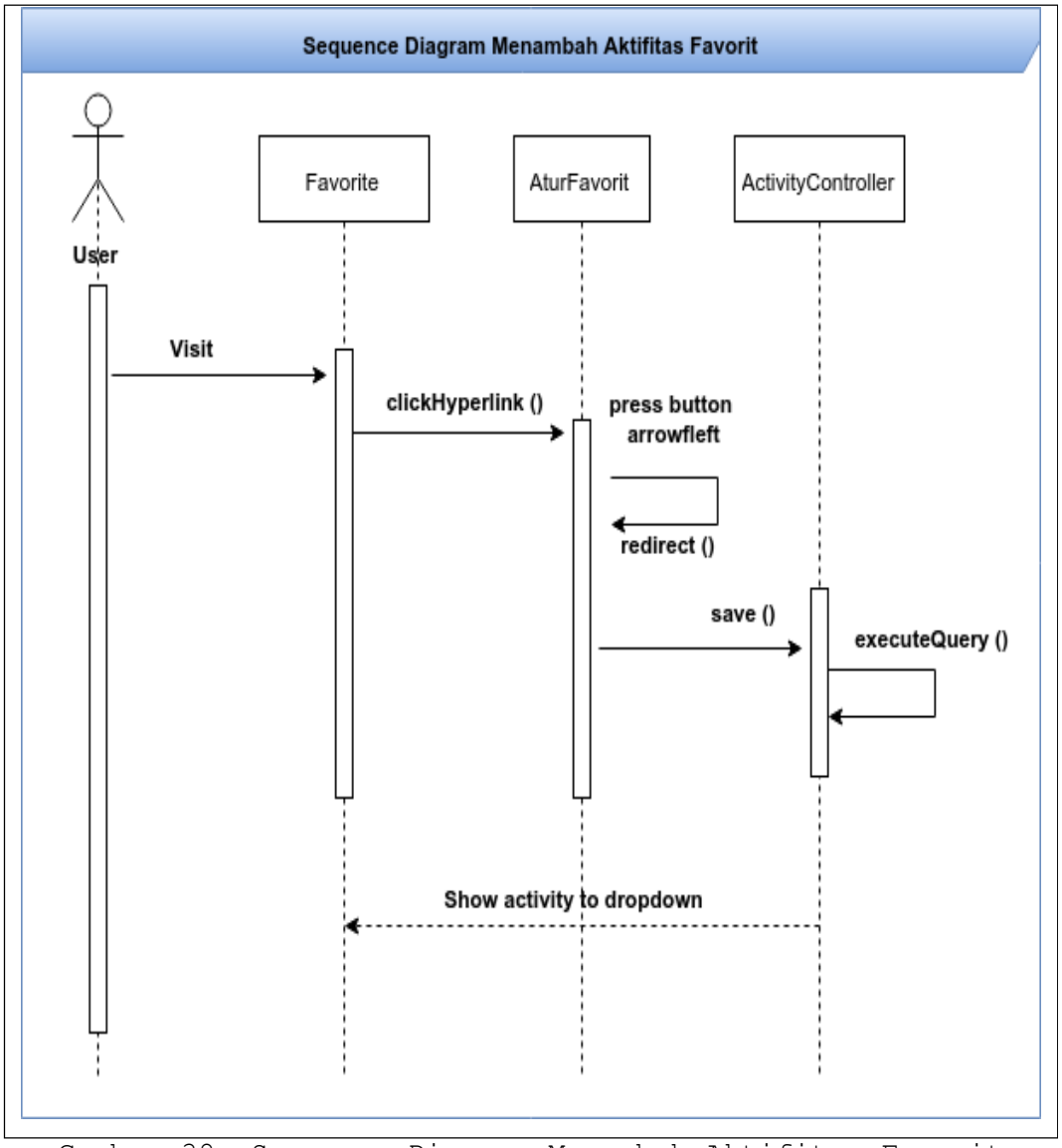
Sequence Diagram Login merupakan implementasi dari SUC-01 yaitu Skenario *Use Case* ke 1 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* melakukan login untuk dapat mengakses menu utama pada sistem. Gambaran tentang bagaimana proses tersebut berlangsung dapat dilihat pada Gambar 37.



Gambar 37. Sequence Diagram Login

3.3.1.8.2 *Sequence Diagram* Menambah Aktifitas Favorit

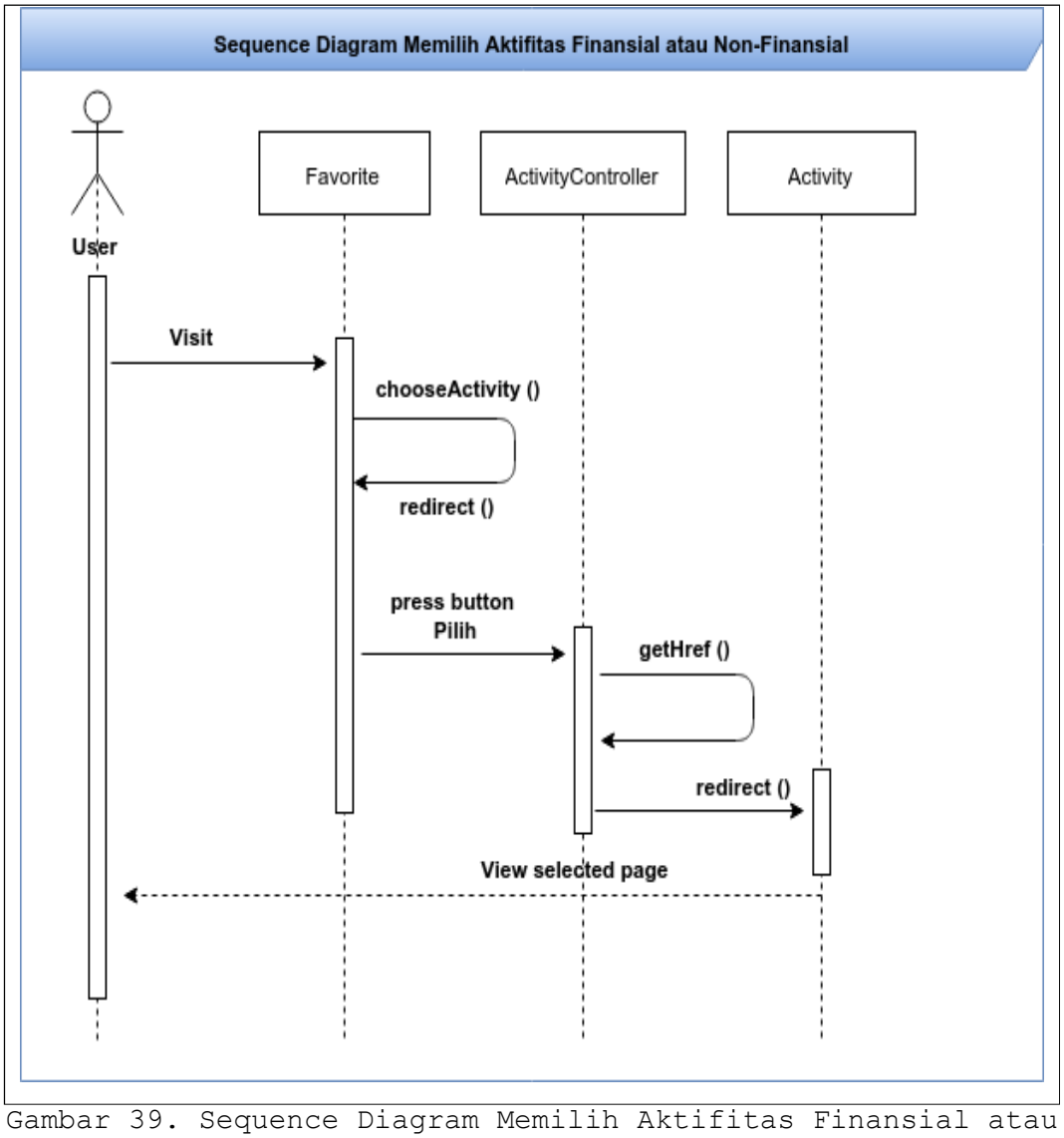
Sequence Diagram Menambah Aktifitas Favorit merupakan implementasi dari SUC-02 yaitu Skenario *Use Case* ke 2 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* menambahkan aktifitas yang sering dilakukan dalam mengakses *web internet banking* menjadi aktifitas favorit. Gambaran tentang bagaimana proses tersebut berlangsung dapat dilihat pada Gambar 38.



Gambar 38. Sequence Diagram Menambah Aktifitas Favorit

3.3.1.8.3 *Sequence Diagram* Memilih Aktifitas Finansial atau Non-Finansial

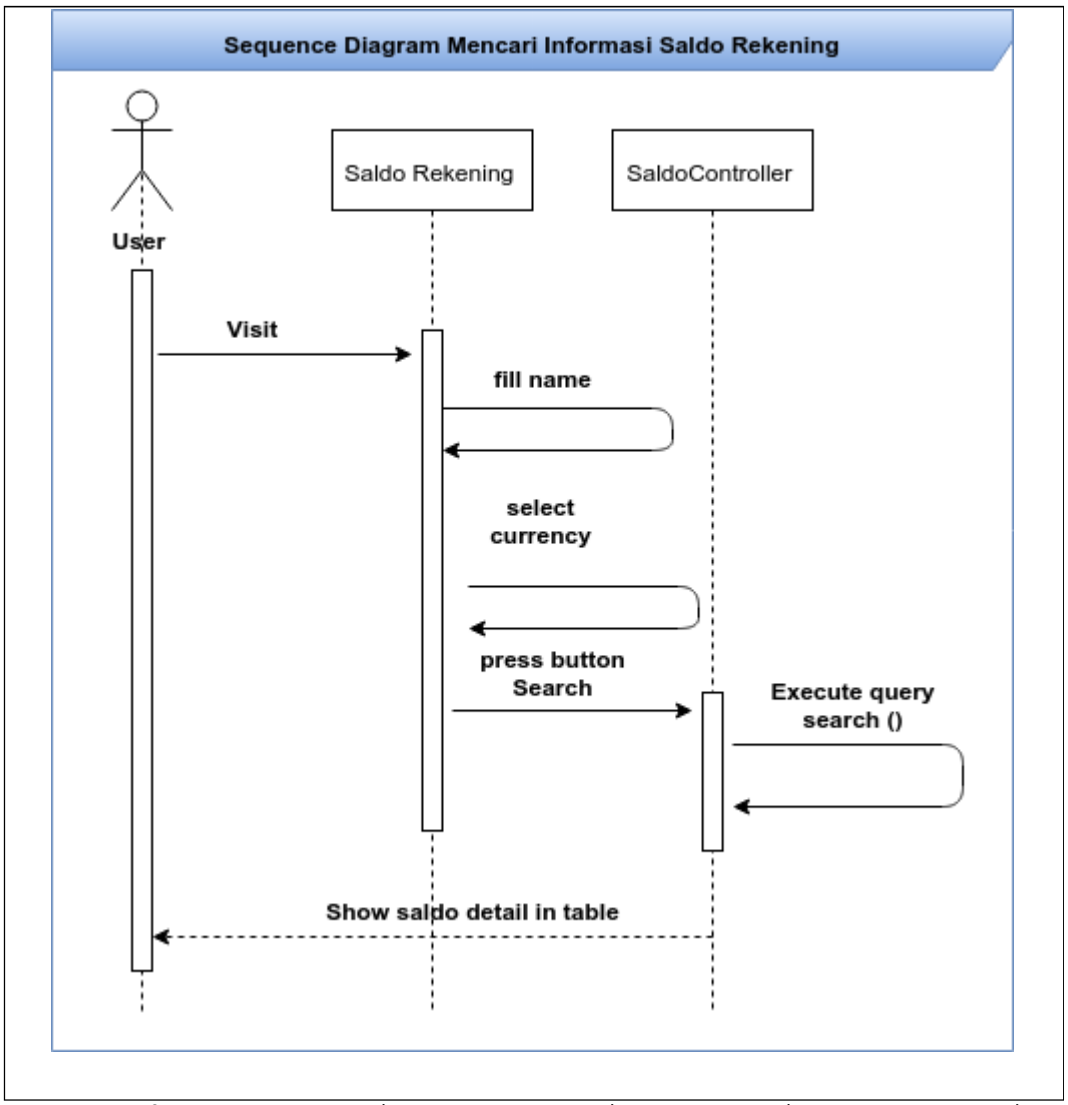
Sequence Diagram Memilih Aktifitas Finansial atau Non-Finansial merupakan implementasi dari SUC-03 yaitu Skenario *Use Case* ke 3 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat mengakses dengan cepat halaman transaksi finansial maupun non-finansial. Gambaran tentang bagaimana proses tersebut berlangsung dapat dilihat pada Gambar 39.



Gambar 39. Sequence Diagram Memilih Aktifitas Finansial atau Non-Finansial

3.3.1.8.4 *Sequence Diagram* Mencari Informasi Saldo Rekening

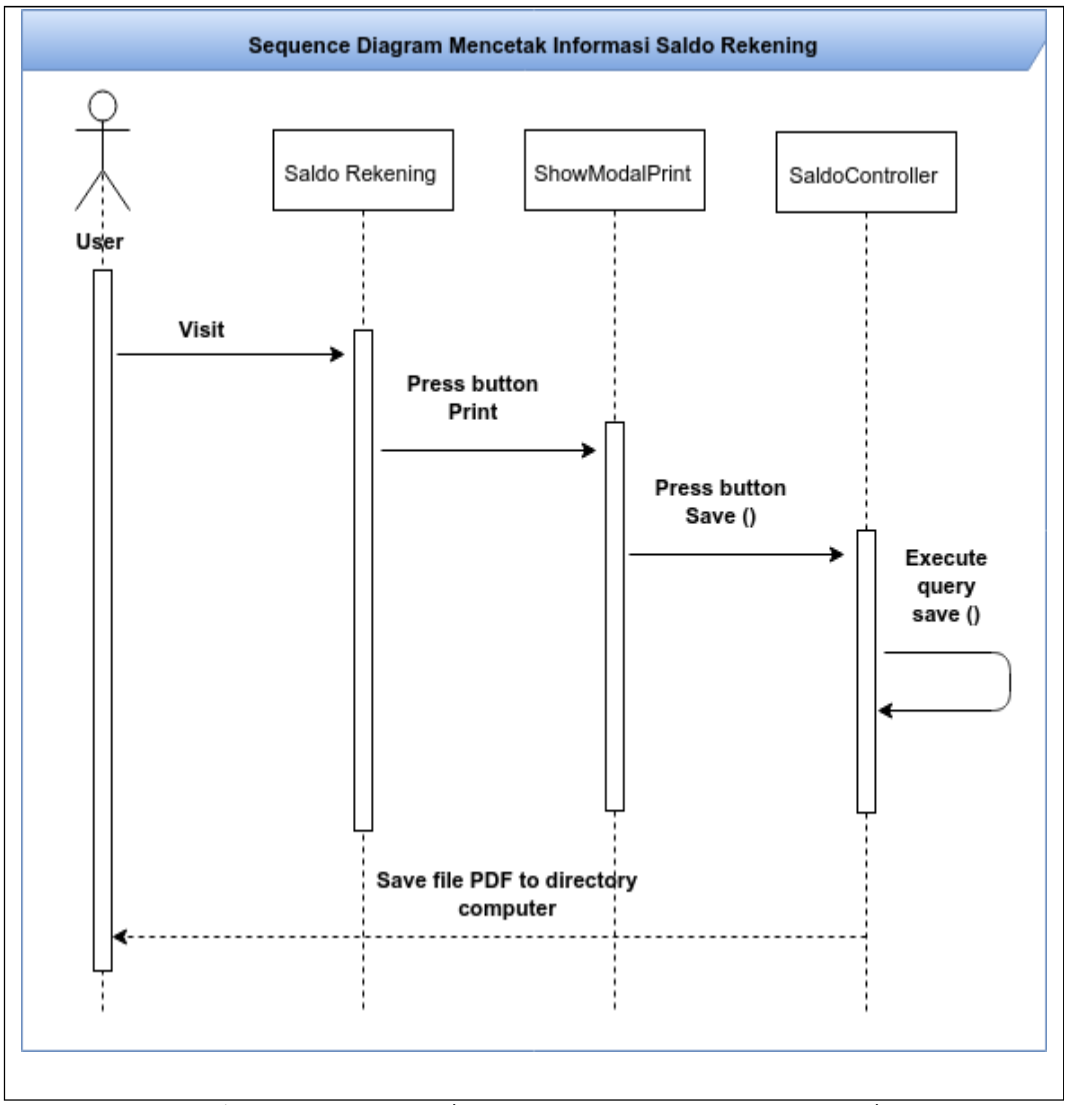
Sequence Diagram Mencari Informasi Saldo Rekening merupakan implementasi dari SUC-04 yaitu Skenario *Use Case* ke 4 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat mengetahui informasi saldo rekening secara detil. Gambaran tentang bagaimana proses tersebut berlangsung dapat dilihat pada Gambar 40.



Gambar 40. Sequence Diagram Mencari Informasi Saldo Rekening

3.3.1.8.5 *Sequence Diagram* Mencetak Informasi Saldo Rekening

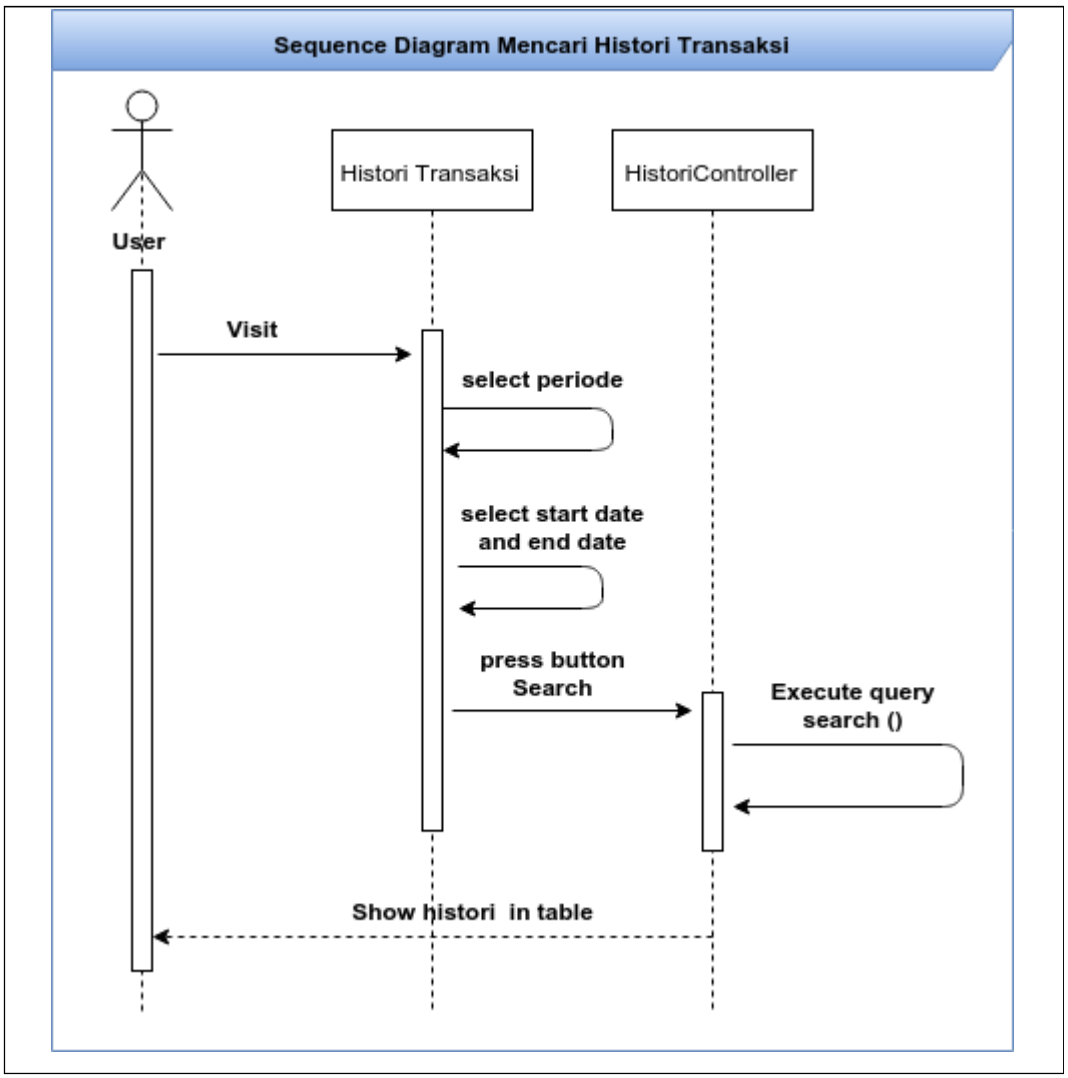
Sequence Diagram Mencari Informasi Saldo Rekening merupakan implementasi dari SUC-05 yaitu Skenario *Use Case* ke 5 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat mencetak informasi saldo rekening ke dalam *file* pdf. Gambaran proses tersebut dapat dilihat pada Gambar 41.



Gambar 41. Sequence Diagram Mencetak Informasi Saldo Rekening

3.3.1.8.6 *Sequence Diagram* Mencari Histori Transaksi

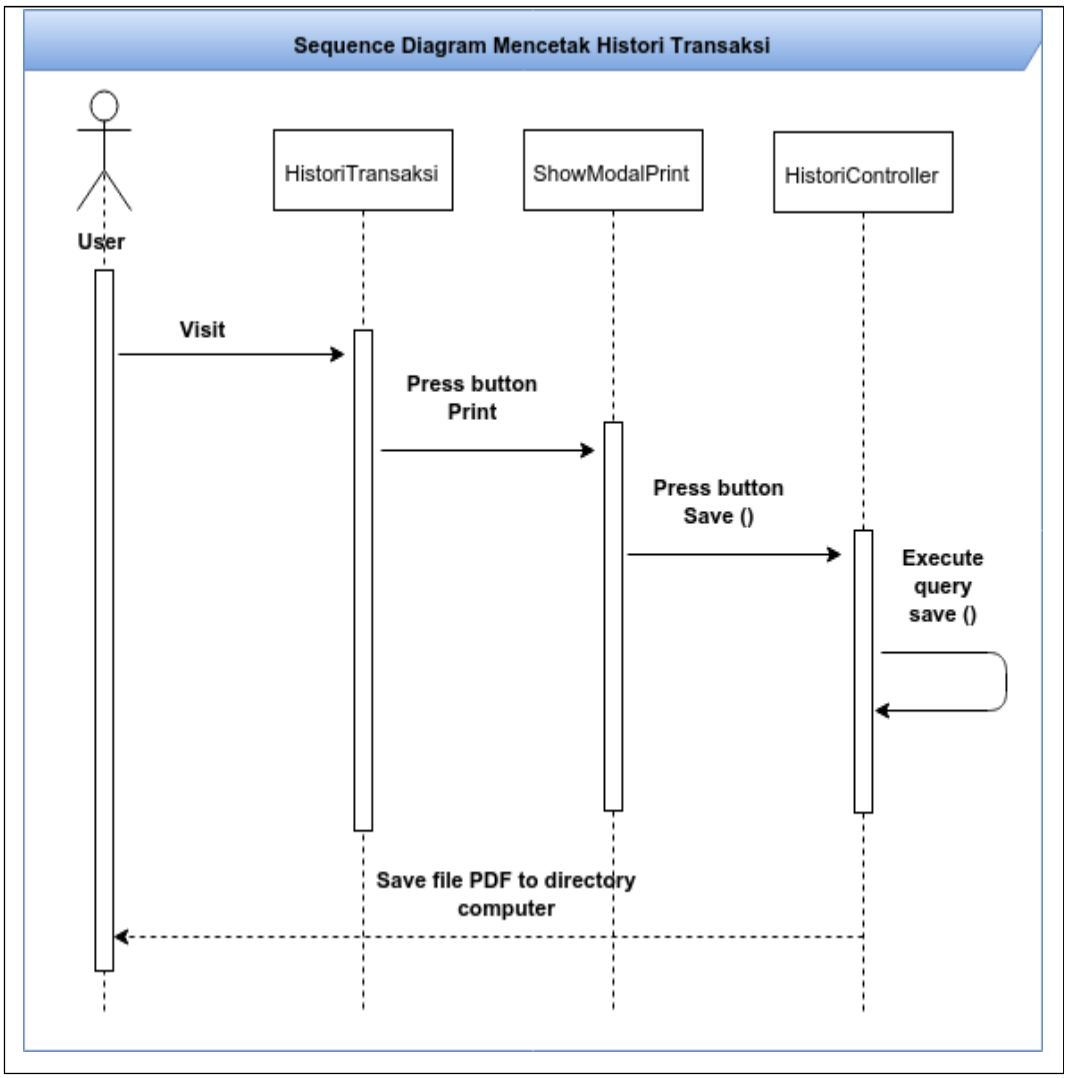
Sequence Diagram Mencari Histori Transaksi merupakan implementasi dari SUC-06 yaitu Skenario *Use Case* ke 6 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat mencari histori tansaksi pada *internet banking* seperti penarikan dan pengiriman uang. Gambaran proses tersebut dapat dilihat pada Gambar 42.



Gambar 42. Sequence Diagram Mencari Histori Transaksi

3.3.1.8.7 *Sequence Diagram* Mencetak Histori Transaksi

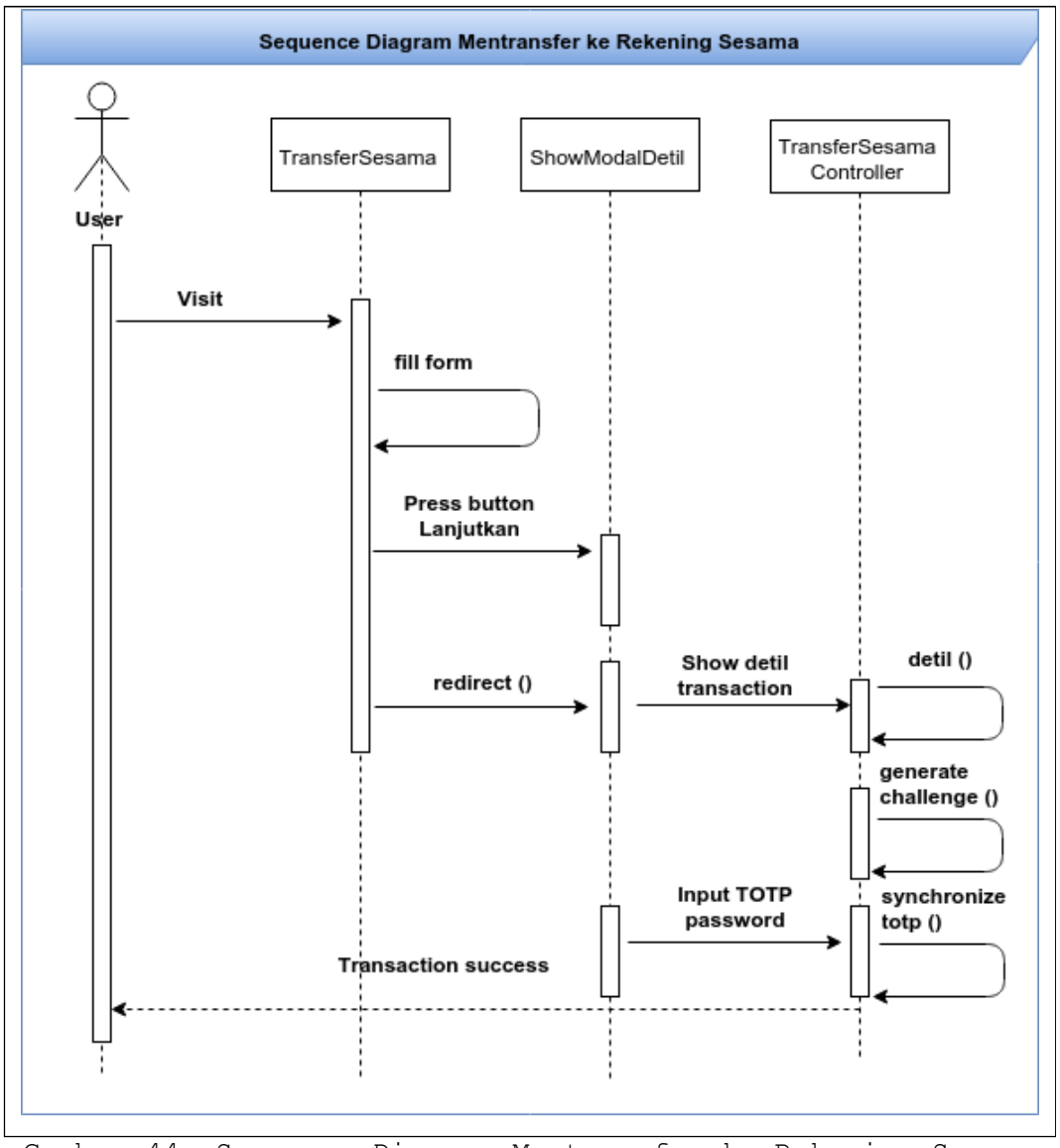
Sequence Diagram Mencetak Histori Transaksi merupakan implementasi dari SUC-07 yaitu Skenario *Use Case* ke 7 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat mencetak histori tansaksi ke dalam *file* pdf. Gambaran proses tersebut dapat dilihat pada Gambar 43.



Gambar 43. Sequence Diagram Mencetak Histori Transaksi

3.3.1.8.8 *Sequence Diagram* Mentransfer ke Rekening Sesama

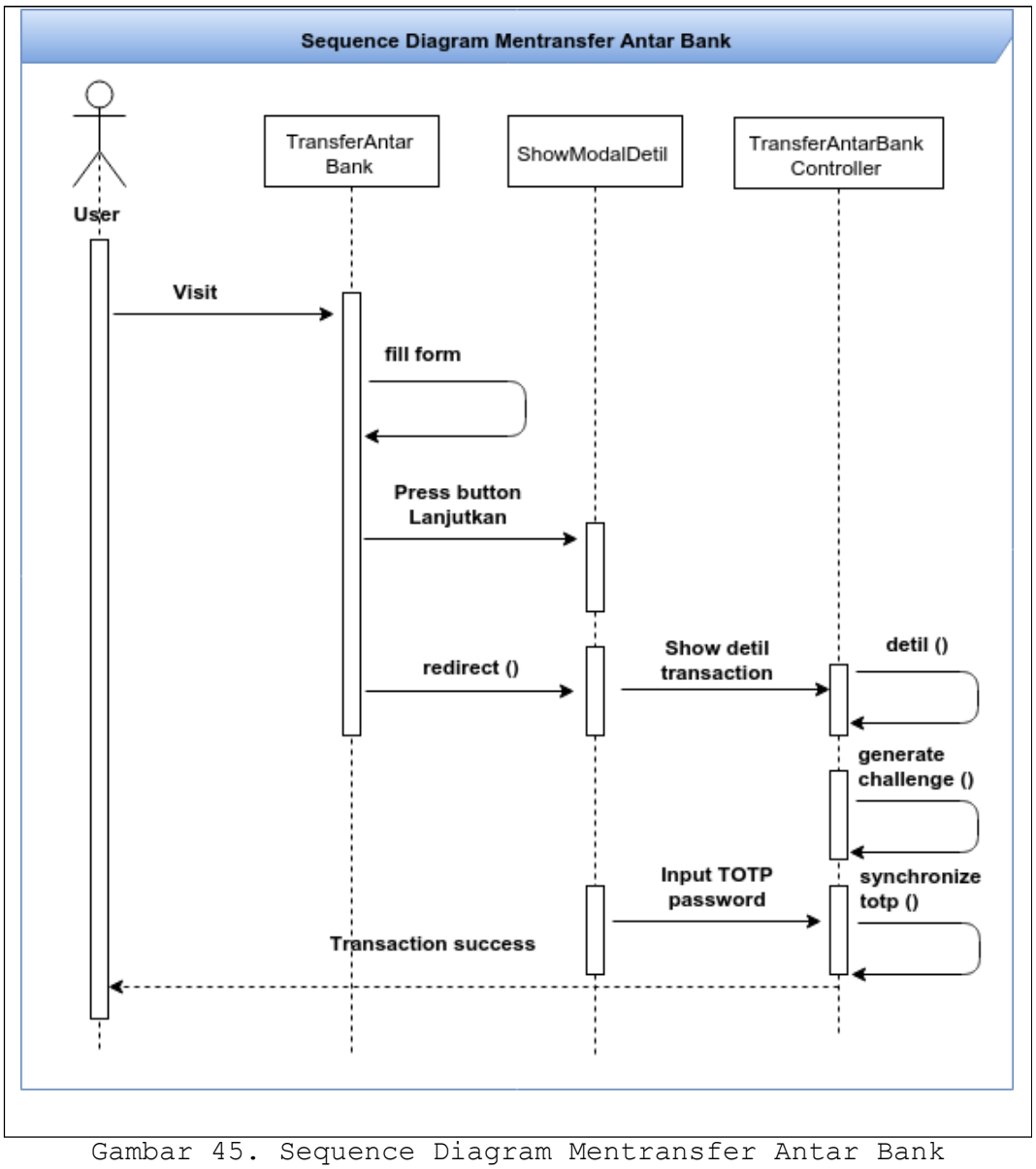
Sequence Diagram Mentransfer ke Rekening Sesama merupakan implementasi dari SUC-08 yaitu Skenario *Use Case* ke 8 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat melakukan transfer uang ke suatu rekening dengan bank yang sama. Gambaran proses tersebut dapat dilihat pada Gambar 44.



Gambar 44. Sequence Diagram Mentransfer ke Rekening Sesama

3.3.1.8.9 *Sequence Diagram* Mentransfer Antar Bank

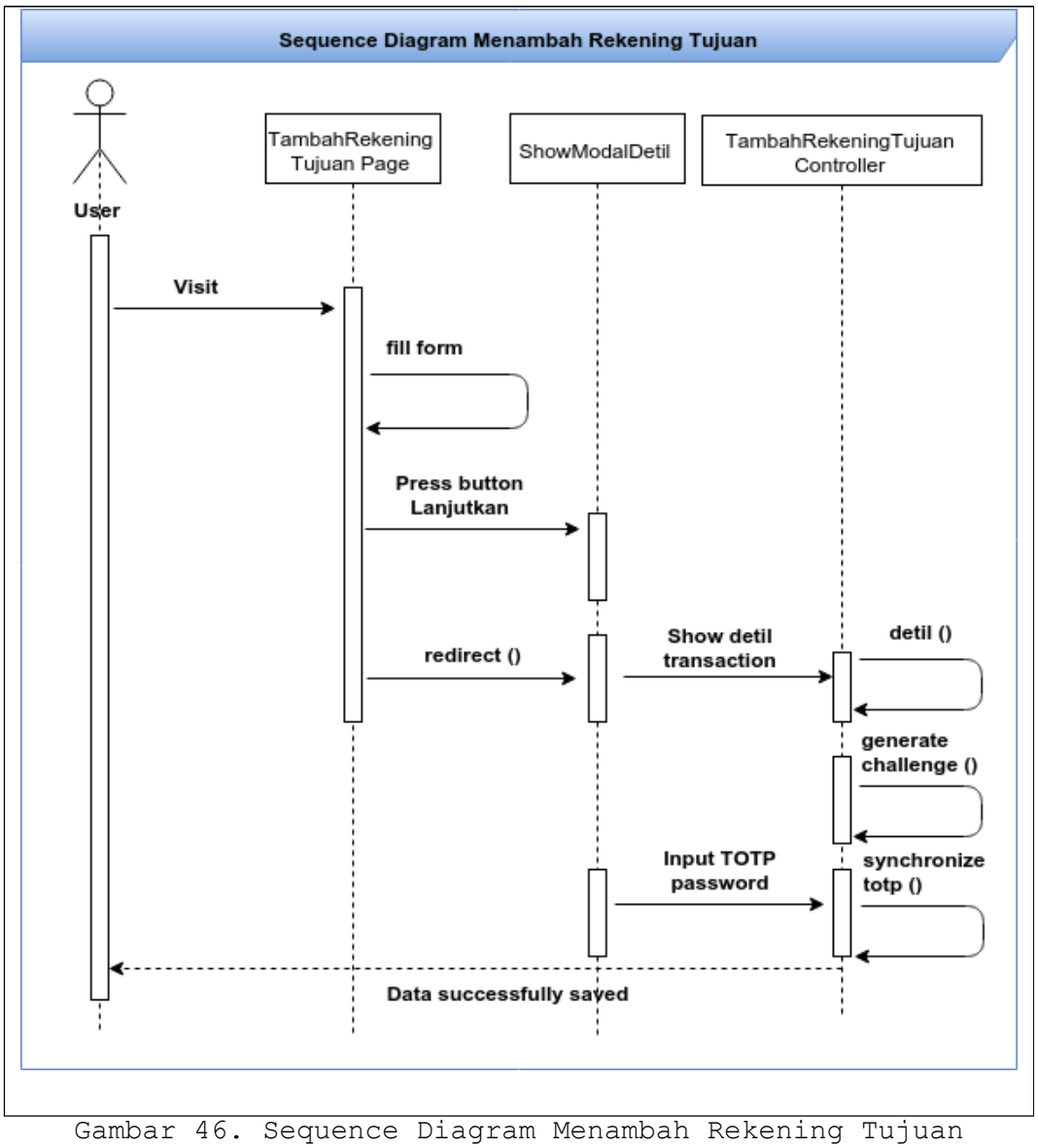
Sequence Diagram Mentransfer Antar Bank merupakan implementasi dari SUC-09 yaitu Skenario *Use Case* ke 9 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat melakukan transfer uang ke suatu rekening dengan bank yang berbeda. Gambaran proses tersebut dapat dilihat pada Gambar 45.



Gambar 45. Sequence Diagram Mentransfer Antar Bank

3.3.1.8.10 *Sequence Diagram* Menambah Rekening Tujuan

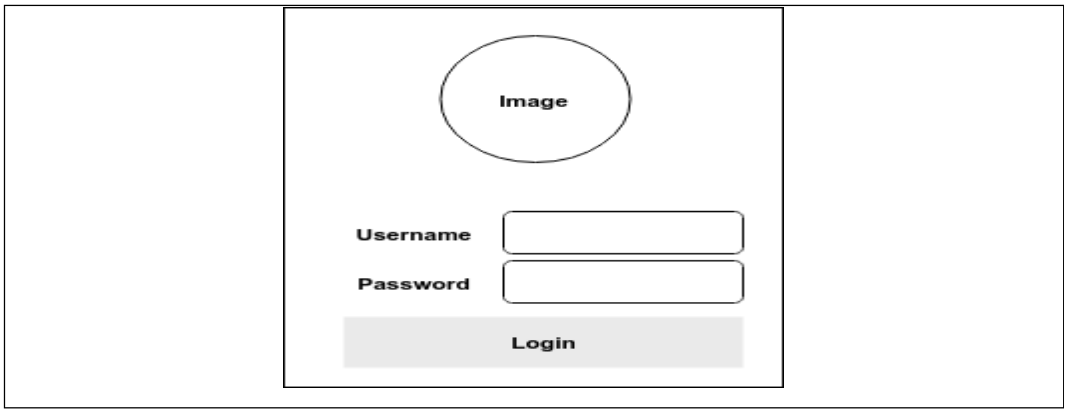
Sequence Diagram Menambah Rekening Tujuan merupakan implementasi dari SUC-10 yaitu Skenario *Use Case* ke 10 yang telah dipaparkan pada bab Lampiran Skenario *Use Case*. Pada tahap ini merupakan proses dimana *user* dapat menambahkan rekening tujuan baru ke dalam sistem.. Gambaran proses tersebut dapat dilihat pada Gambar 46.



Gambar 46. Sequence Diagram Menambah Rekening Tujuan

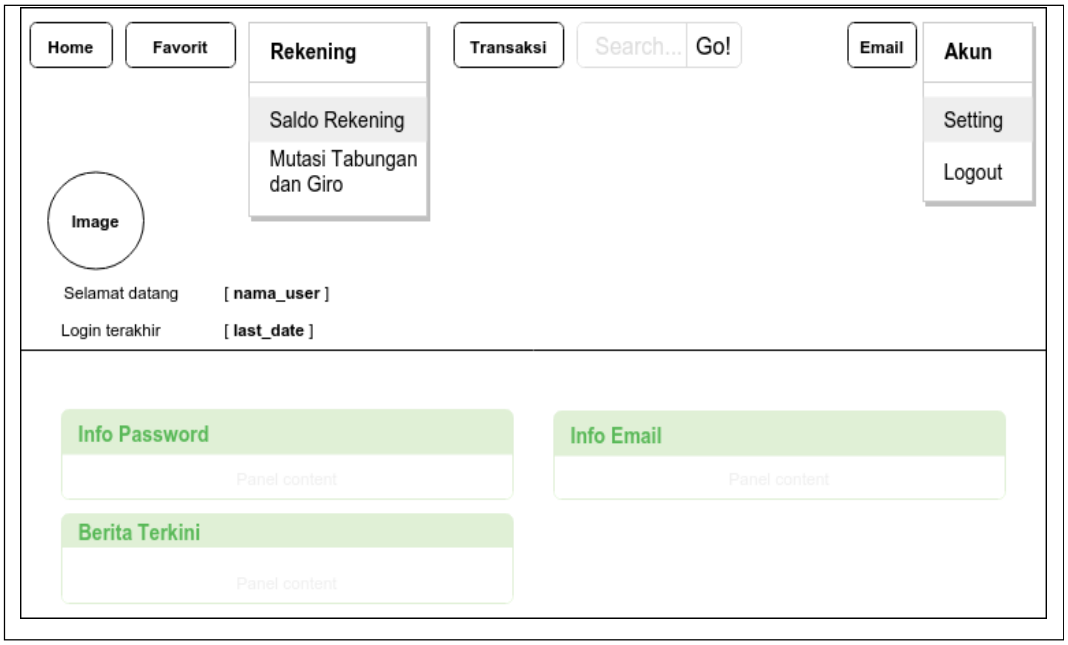
3.3.1.9 Design Interface

Perancangan *Design Interface* aplikasi sistem *internet banking* adalah sebagai berikut :



Gambar 47. Halaman awal login sistem

Pada Gambar 47 merupakan halaman awal memulai sistem yaitu halaman *login*. Halaman *login* disediakan untuk otentikasi level pertama di dalam sistem. Sistem memeriksa apakah *user* yang melakukan *login* terdaftar pada *database* atau tidak. Setelah *user* melakukan *login* dan data tersebut valid maka sistem akan menampilkan halaman utama.

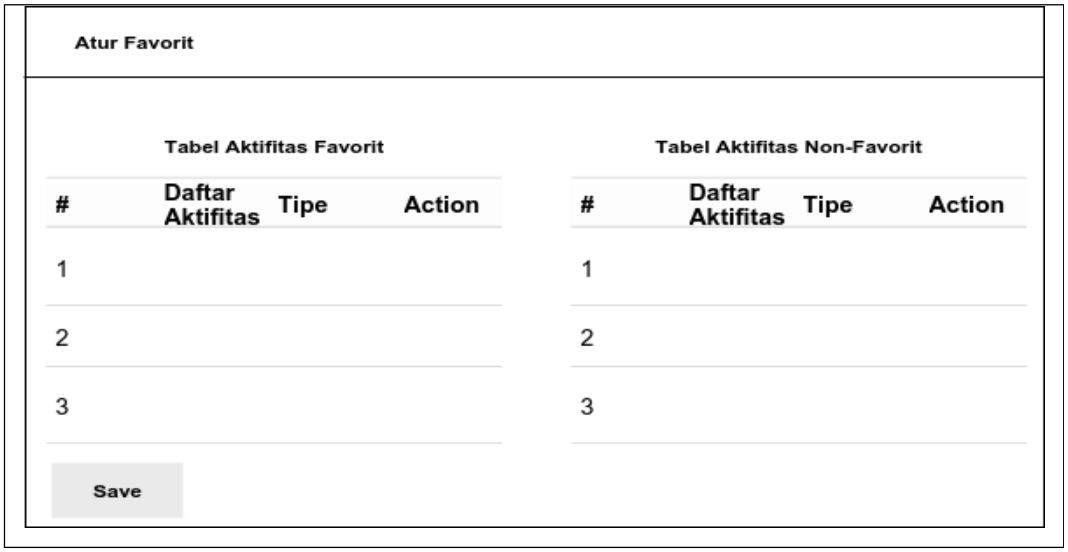


Gambar 48. Halaman utama sistem

Pada Gambar 48 merupakan halaman utama dari sistem yang menampilkan beberapa menu utama seperti *Home*, *Favorit*, *Rekening*, *Transaksi*, *Email* dan *Akun*. Pada halaman utama juga menampilkan nama *user* yang sedang melakukan *login* serta tanggal terakhir *user* tersebut melakukan *login*.

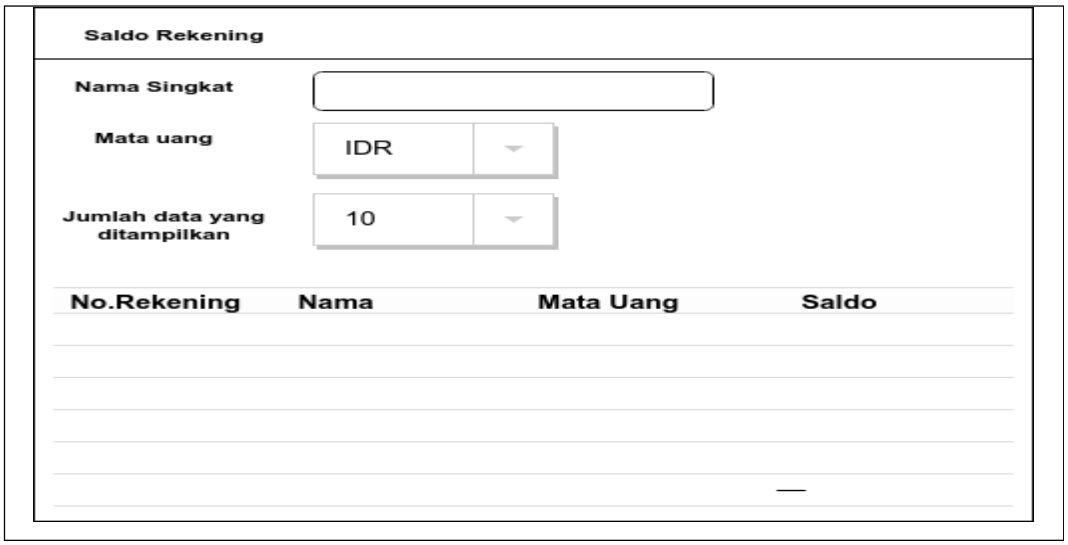
Gambar 49. Halaman Menu Favorit

Pada Gambar 49 merupakan halaman menu favorit yang digunakan untuk akses cepat menuju halaman transaksi yang diinginkan, dengan cara memilih aktifitas yang ada pada *dropdown* Aktifitas Finansial atau Non-Finansial. Jika ingin mendaftarkan aktifitas favorit, *user* dapat mengklik *link* "Klik disini untuk mendaftarkan aktifitas favorit Anda".



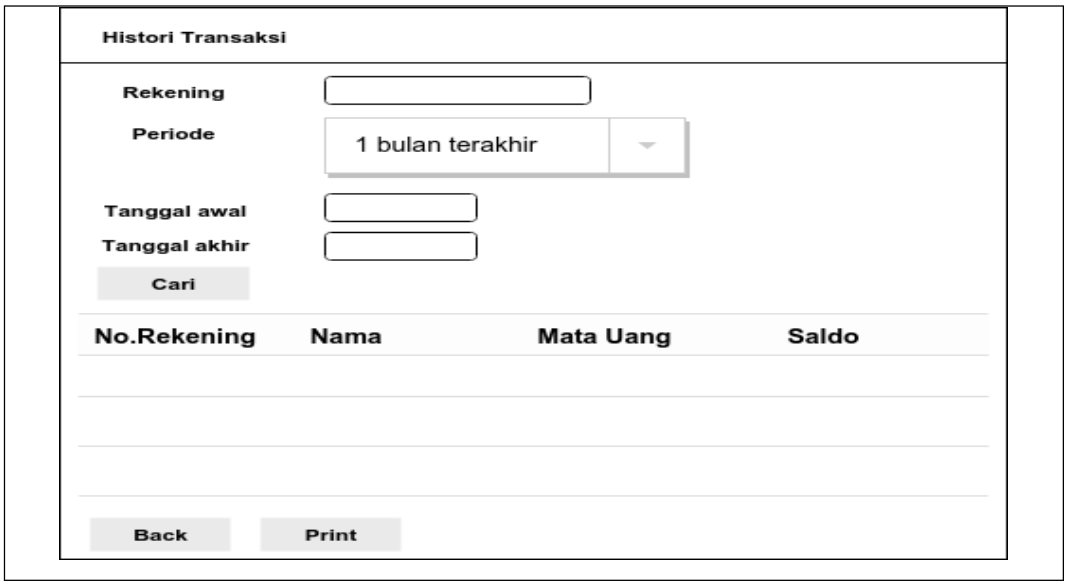
Gambar 50. Halaman Menu Atur Favorit

Pada Gambar 50 merupakan halaman menu atur favorit yang digunakan untuk memindahkan data-data yang ada pada tabel aktifitas non-favorit menjadi aktifitas favorit dengan tujuan *user* dapat mengakses menu yang sering diakses pada *web internet banking* dengan cepat.



Gambar 51. Halaman Menu Informasi Saldo Rekening

Pada Gambar 51 merupakan halaman menu informasi saldo rekening melihat detil rekening dari *user* yang sedang *login*.



Gambar 52. Halaman Menu Histori Transaksi

BAB IV
IMPLEMENTASI DAN PENGUJIAN

Pada bab ini menjelaskan tentang hasil implementasi dari perancangan dan pengujian terhadap aplikasi. Kedua tahap implementasi dan perancangan dilakukan setelah penerapan dari perancangan selesai. Selanjutnya proses implementasi dilanjutkan menggunakan bahasa pemograman yaitu *Scala*. Setelah implementasi, maka dilakukan pengujian dimana akan dilihat implementasi *class-class* yang dikembangkan dan hasil dari masing-masing pengujian yang dilakukan. Melalui pengujian ini dapat diketahui hasil dari perancangan yang telah dilakukan.

4.1 Lingkungan Pengembangan

Dalam lingkungan pengembangan, dijelaskan tentang perangkat yang digunakan dalam pengembangan aplikasi token *internet banking* dengan algoritma *Time-Based One Time Password*. Pada tahap ini akan dijabarkan spesifikasi lingkungan pengembangan yang digunakan oleh *developer* dalam pembangunan aplikasi.

4.1.1 Perangkat Keras

Spesifikasi komputer yang digunakan dalam pembangunan aplikasi token *internet banking* dengan algoritma *Time-Based One Time Password* adalah sebagai berikut :

1. *Processor* : Intel Celeron CPU 1017U 1.60 GHz
2. RAM : Intel Corporation 3rd Gen Core processor DRAM Controller 4 GB
3. VGA : Intel Corporation 3rd Gen Core processor Graphics Controller 1 GHz
4. *Monitor* : Dell Inspiron 1024x768.
5. *Communication* : Intel Corporation 7 Series/C210 Series Chipset Family MEI Controller

6. *Audio* : Corporation 7 Series/C210 Series Chipset Family High Definition Audio Controller.
7. *Network* : Broadcom Corporation BCM43142 802.11b/g/n
8. *Harddisk* : Serial SATA 500 GB

4.1.2 Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pengembangan aplikasi token *internet banking* ini adalah sebagai berikut :

1. Sistem operasi : Linux Ubuntu 12.04 LTS
2. *Compiler* : Java SDK 1.8 (*Java Standard Development Kit*)
3. Bahasa pemograman : Scala 2.11.7
4. *Database* : MongoDB
5. *Backend Framework* : Play Framework
6. *Frontend Framework* : AngularJS
7. GUI *Framework* : Bootstrap Material Design
8. IDE : Jetbrains Intellij IDEA 14

4.2 Implementasi

Implementasi dilakukan dalam dua tahap, yaitu implementasi *client internet banking* berbasis *android* dan implementasi *server* token *internet banking* berbasis *web* sesuai dengan perancangan yang telah dilakukan pada bab 3. Dengan kedua tahapan implementasi ini, diharapkan antara perancangan dan hasil pengujian dapat selaras dan sesuai dengan target yang dituju. Pada tahap implementasi juga akan terlihat kesimpulan dari beberapa pengujian yang dilakukan yaitu berapa banyak pengujian yang berhasil dan pengujian yang gagal.

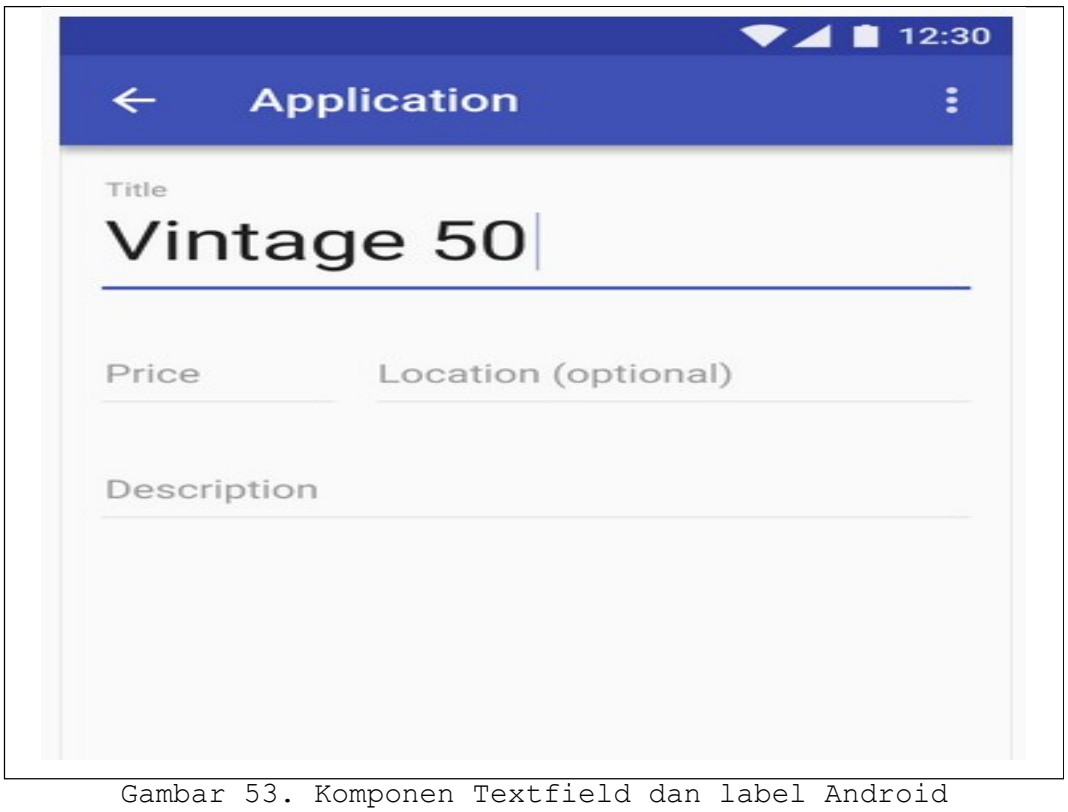
4.2.1 Implementasi *Client* Token *Internet Banking*

Implementasi terhadap perancangan yang telah dilakukan terhadap aplikasi token i*nternet banking* dari sisi *client* terbagi 2 tahap yaitu implementasi *Graphical User Interface* (GUI) dan implementasi modul.

4.2.1.1 Implementasi *Graphical User Interface* (GUI) *Client*

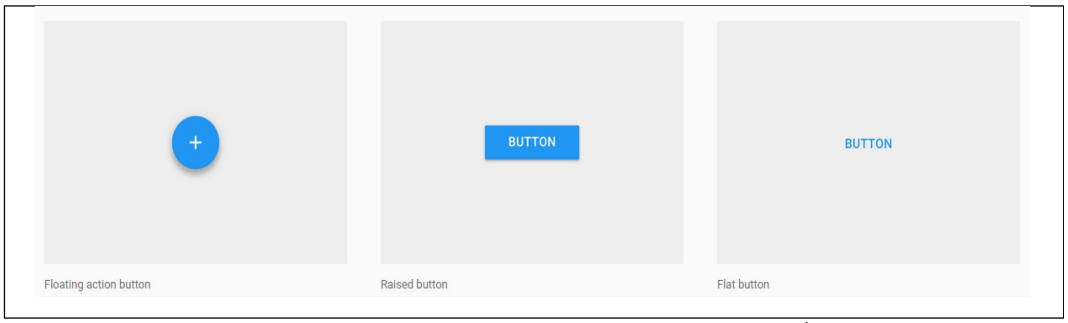
Implementasi *Graphical User Interface* (GUI) pada *client* berbasis *android* menggunakan *framework Google Material Design*. Berikut komponen-komponen yang digunakan pada aplikasi android yaitu :

1. *Textfield* dan *label*, merupakan komponen yang digunakan untuk meng-inputkan teks pada android. Berikut contoh komponen *Textfield* dan *label* pada Gambar 53.



Gambar 53. Komponen Textfield dan label Android

2. *Button*, komponen yang digunakan sebagai *trigger* (pemicu) untuk melakukan aksi tertentu seperti *login* dan *register*. Berikut contoh komponen *Button* pada Gambar 54.



Gambar 54. Komponen Button Android

4.2.1.2 Implementasi Modul *Client* (*Android*)

Modul *client* yang diimplementasi mencakup modul dari *activity*, *helper*, *service*, dan *layout*. Berikut penjelasan dari masing-masing modul :

1. Modul *Activity*

Modul *Activity* pada aplikasi *android* berfungsi untuk menampilkan dan mengelola halaman aplikasi sebagai tempat interaksi antara pengguna dengan aplikasi. *Class-class* yang terdapat pada modul activity dapat dilihat pada Tabel 20.

Tabel 20. Class-class pada modul Activity

No	Nama Class	Fungsi
1	LoginActivity.java	Mengatur segala aktifitas pada halaman activity_login.xml seperti menyimpan hasil inputan user pada Textfield dan menjalankan method submit ().
2	RegisterActivity.java	Mengatur segala aktifitas pada halaman activity_register.xml dan menjalankan method register ().
3	TOTPActivity.java	Mengatur segala aktifitas pada halaman activity_totp.xml seperti menyimpan challenge code dan menghasilkan totp password dengan menjalankan method generateTotp ().

2. Modul *Helper*

Modul *Helper* pada aplikasi *android* berfungsi untuk menangani *session login* saat pengguna masuk ke dalam aplikasi dan menangangi *query* ke *database*. *Class-class* yang terdapat pada modul helper dapat dilihat pada Tabel 21.

Tabel 21. Class-class pada modul Helper

No	Nama Class	Fungsi
1	SessionManager.java	Menangani session pada saat user melakukan login ke dalam aplikasi
2	SQLiteHandler.java	Menangani query ke database SQLite seperti insert, update, delete dan select.

3. Modul *Service*

Modul *Service* pada aplikasi *android* berfungsi sebagai penyedia layanan yang dibutuhkan oleh *class activity*. *Class-class* yang terdapat pada modul *service* dapat dilihat pada Tabel 22.

Tabel 22. Class-class pada modul Service

No	Nama Class	Fungsi
1	AccountService.java	Menyediakan layanan untuk mencocokkan kode rahasia yang ada pada bank dengan kode rahasia yang dimasukkan saat melakukan register.
2	TOTPService.java	Menyediakan layanan untuk menghasilkan totp password menggunakan algoritma Time-Based One Time Password

4. Modul *Layout*

Modul *Layout* pada aplikasi *android* berfungsi untuk menangani tampilan aplikasi seperti *textfield*, *label* dan *button*. *Class-class* yang terdapat pada modul *layout* dapat dilihat pada Tabel 23.

Tabel 23. Class pada modul Layout

No	Nama Class	Fungsi
1	activity_login.xml	Class yang berfungsi untuk mengatur desain tampilan halaman login
2	activity_register.xml	Class yang berfungsi untuk mengatur desain tampilan halaman register
3	activity_totp.xml	Class yang berfungsi untuk mengatur desain tampilan halaman untuk menghasilkan totp password

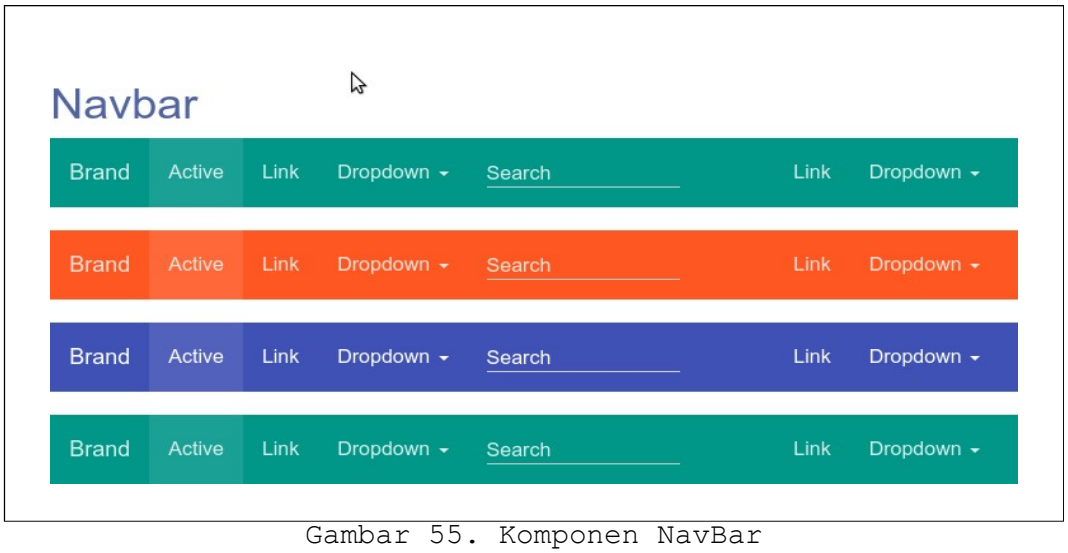
4.2.2 Implementasi *Server Token Internet Banking*

Implementasi terhadap perancangan yang telah dilakukan terhadap aplikasi token i*nternet banking* dari sisi *server* terbagi 2 tahap yaitu implementasi *Graphical User Interface* (GUI) dan implementasi modul.

4.2.2.1 Implementasi *Graphical User Interface* (GUI) *Server*

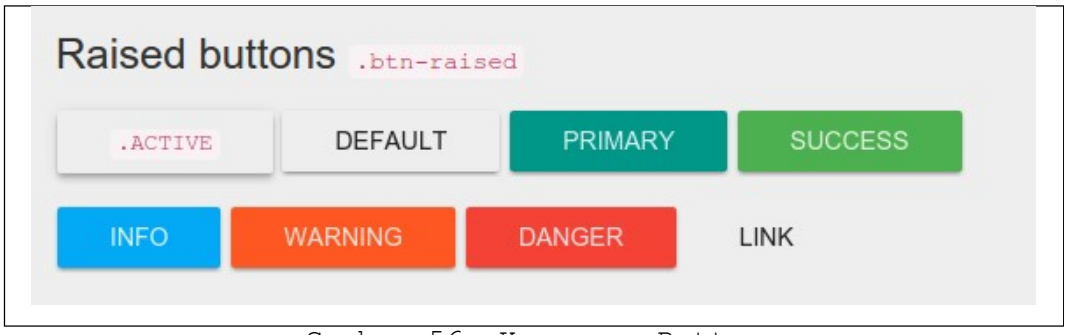
Implementasi *Graphical User Interface* (GUI) pada *server* menggunakan *framework Bootstrap Material Design*. Berikut komponen-komponen yang digunakan dalam implementasi antarmuka yaitu :

1. *NavBar*, merupakan komponen navigasi yang berbentuk *frame* berisi beberapa beberapa menu dengan fungsinya masing-masing. Berikut contoh komponen *NavBar* pada Gambar 55.



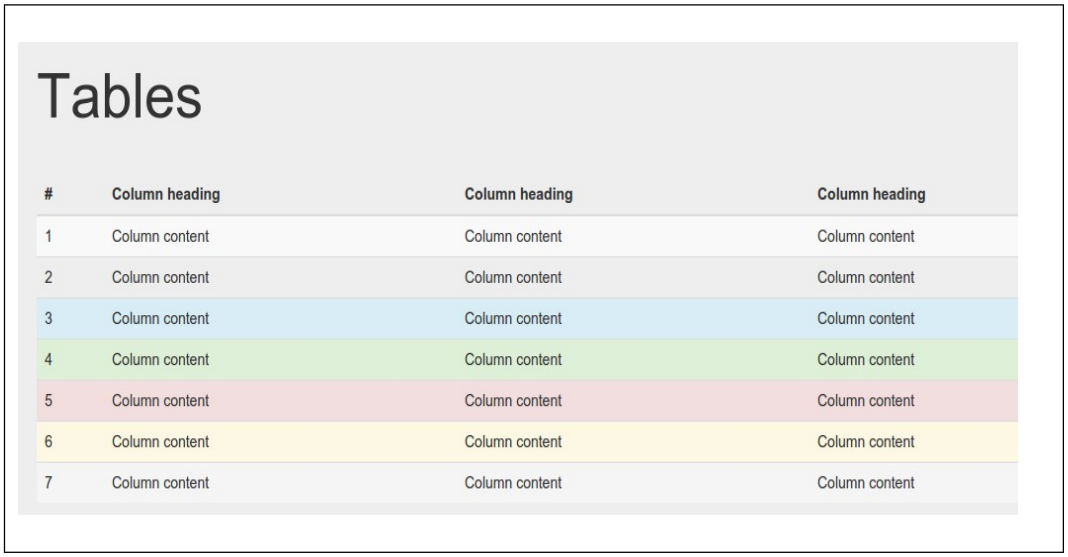
Gambar 55. Komponen NavBar

2. *Button*, merupakan komponen yang digunakan sebagai *trigger* (pemicu) untuk melakukan aksi tertentu seperti menyimpan, menampilkan dan lain-lain. Berikut contoh komponen *Button* pada Gambar 56.



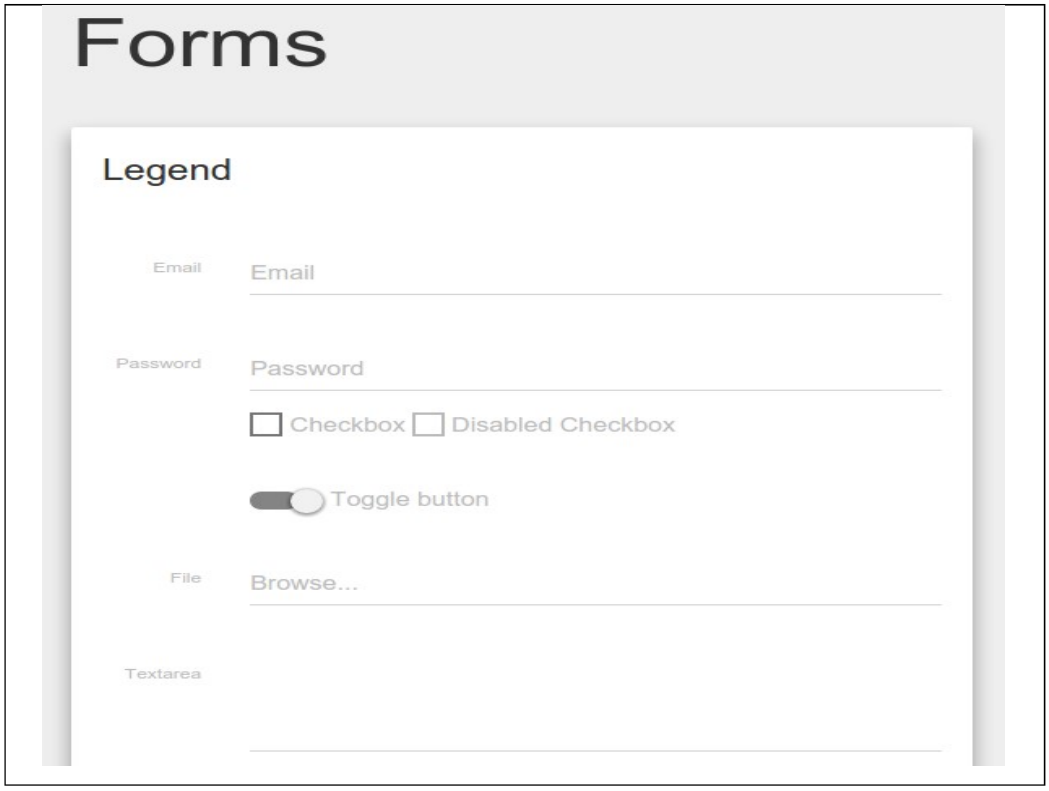
Gambar 56. Komponen Button

3. *Table*, merupakan komponen yang digunakan untuk menampilkan data dari *database*. Berikut contoh komponen *Table* pada Gambar 47.



Gambar 57. Komponen Table

4. *Form*, merupakan area yang berisi elemen-elemen *label*, *input*, *checkbox*, *radio button*, *dropdown* dan *button* yang dapat diisi oleh *user*. Berikut contoh *Form* pada Gambar 58.

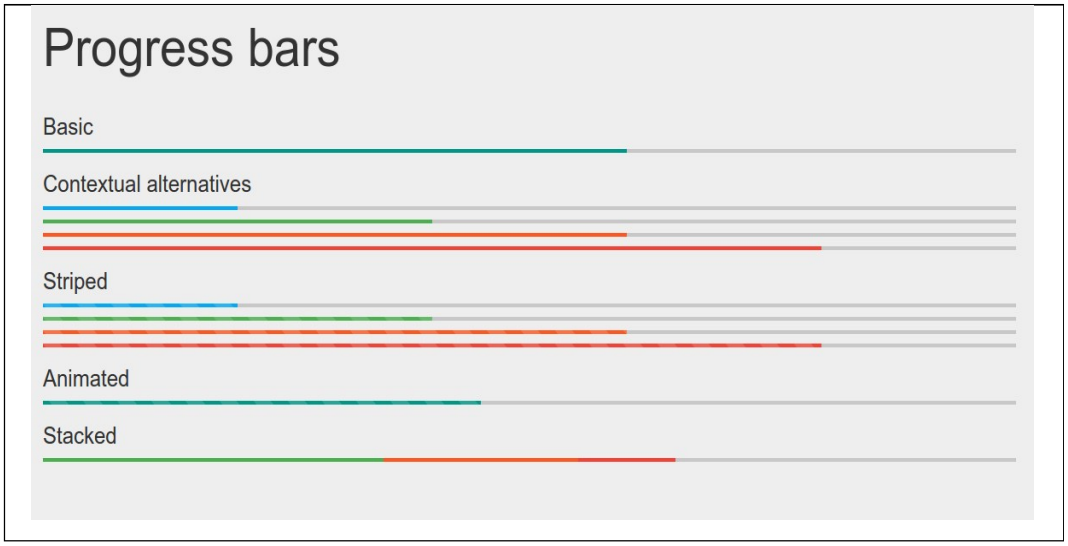


Gambar 58. Komponen Form

5. *Breadcrumbs*, merupakan komponen navigasi yang digunakan sebagai informasi kepada *user* untuk memberi tahu dimana posisi halaman *web* saat ini. Berikut contoh komponen *Breadcrumbs* pada Gambar 49.

Gambar 59. Komponen Breadcrumbs

6. *Progress Bar*, merupakan komponen yang dipakai untuk melihat proses yang sedang berlangsung. Berikut contoh komponen *Progress Bar* pada Gambar 60.

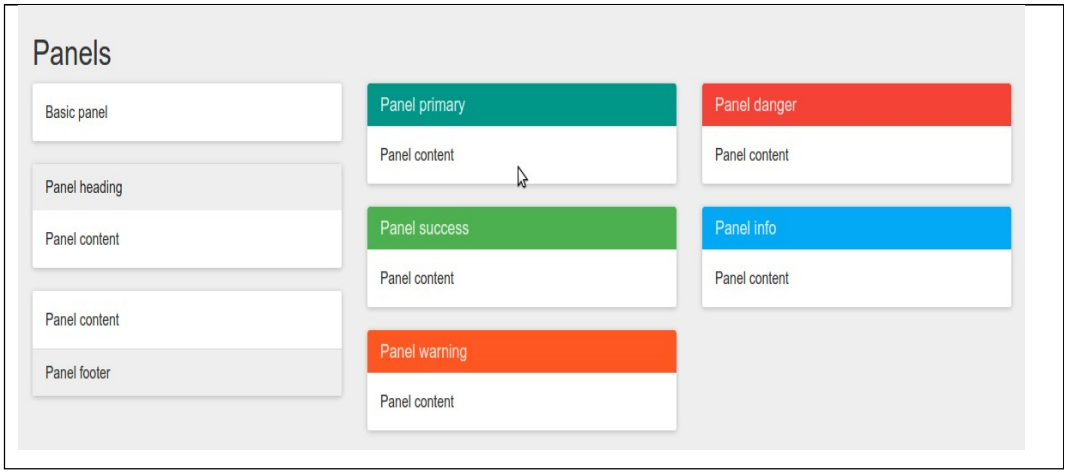


Gambar 60. Komponen Progress Bar

7. *Alert*, merupakan komponen yang digunakan untuk menampilkan pesan kepada user berupa *dialog box* sebagai peringatan sebelum aplikasi dilanjutkan. Berikut contoh komponen *Alert* pada Gambar 61.

Gambar 61. Komponen Alert

8. *Panel*, merupakan komponen yang digunakan sebagai area untuk menampung elemen-elemen pada *web*. Berikut contoh komponen Panel pada Gambar 62.



Gambar 62. Komponen Panel

4.2.2.2 Implementasi Modul *Server* (*Web*)

Modul *server* yang diimplementasi mencakup modul dari *model*, *view*, *controller*, *service*, *security* dan *util*. Berikut penjelasan dari masing-masing modul :

1. Modul *Model*, berfungsi sebagai enkapsulasi data yang dapat digunakan sebagai *rendering* ke *database* menjadi sebuah tabel. *Class-class* yang terdapat pada modul *model* dapat dilihat pada Tabel Lampiran 16, Lampiran B.1 *Class Diagram Model*.
2. Modul *View*, berfungsi sebagai *rendering* terhadap *model* menjadi sebuah *form* sebagai *user interface*. *Class-class* yang terdapat pada modul *view* dapat dilihat pada Tabel Lampiran 27, Lampiran B.3 *Modul View*.
3. Modul *Controller*, berfungsi untuk merespon *event* yang dilakukan oleh *user*, memprosesnya dan juga melakukan perubahan terhadap *model*. *Class-class* yang terdapat pada modul *controller* dapat dilihat pada Tabel Lampiran 17 sampai Tabel Lampiran 26, Lampiran B.2 *Class Diagram Controller*.
4. Modul *Service*, berfungsi sebagai pernyedia layanan yang dibutuhkan oleh *controller*. *Class-class* yang terdapat pada modul *service* dapat dilihat pada Tabel Lampiran 28, Lampiran B.4 *Modul Service.*

5. Modul *Util*, berfungsi sebagai *mapping* struktur data ke *database* dan *file* JSON. Class-class yang terdapat pada modul util dapat dilihat pada `Tabel Lampiran 29`, Lampiran B.5 *Modul Util*.

4.3 Pengujian Hasil Impelementasi Perangkat Lunak

Pada aplikasi token *internet banking*, dilakukan pengujian *blackbox testing* yang berfungsi untuk menguji setiap fungsionalitas aplikasi yang telah dibuat. Tujuan dari proses pengujian ini dilakukan untuk memeriksa apakah aplikasi ini telah menjalankan fungsionalitas sesuai dengan awal perancangan.

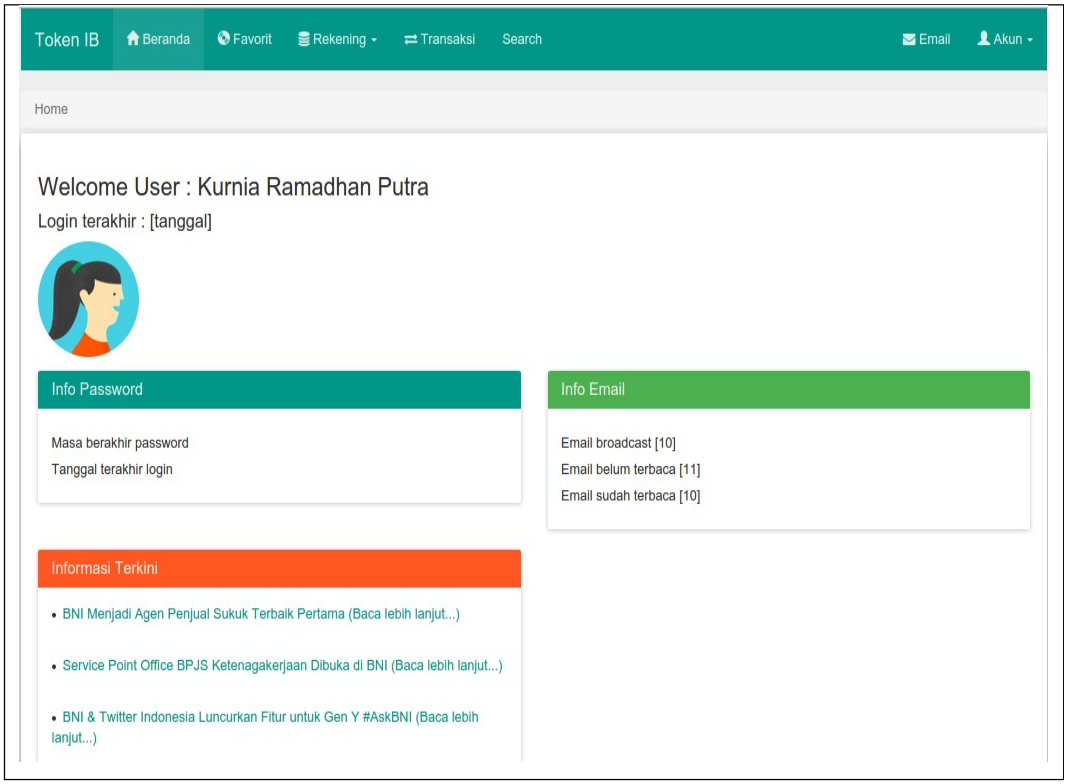
Proses pengujian dilakukan berdasarkan tabel skenario pengujian yang dijelaskan pada sub bab 4.3.1 sampai dengan 4.3.10. Sesuai dengan gambaran yang ada pada Lampiran A Skenario *Use Case*. Berikut adalah pengujian aplikasi dari masing-masing kasus yang ditunjukan pada Lampiran C Skenario Pengujian Aplikasi.

4.3.1 Pengujian Fungsionalitas *Login*

Pengujian fungsionalitas *Login* berfungsi untuk melihat apakah proses dalam melakukan *login* berfungsi dengan baik atau tidak. Pengujian dijabarkan pada `Tabel Lampiran 29. Pengujian Fungsi Login`. Berdasarkan hasil pengujian yang dilakukan pengguna terhadap fungsionalitas *Login* maka diperoleh hasil bahwa login berhasil. Pengujian fungsionalitas *Login* ditunjukkan pada `Gambar 63.`

`Gambar 63. Proses pengujian fungsionalitas Login`

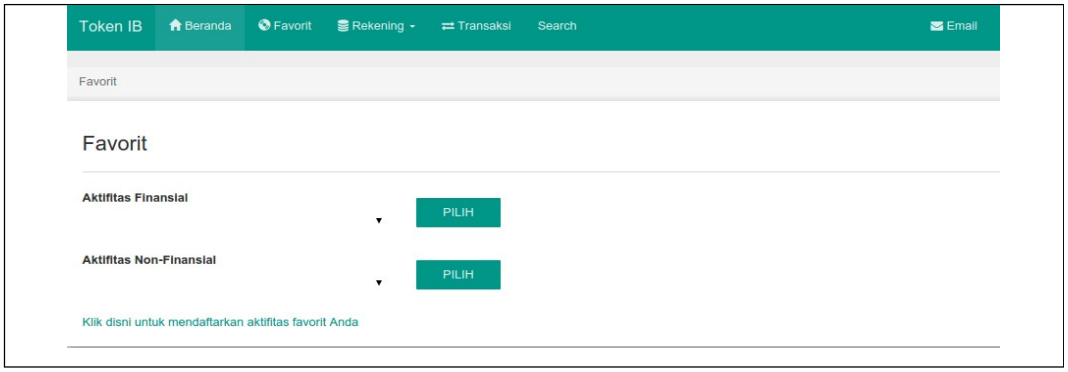
Proses login berhasil dan sistem menampilkan nama pemilik rekening yang melakukan login pada menu utama seperti yang terlihat pada Gambar 64.



Gambar 64. Menu utama aplikasi

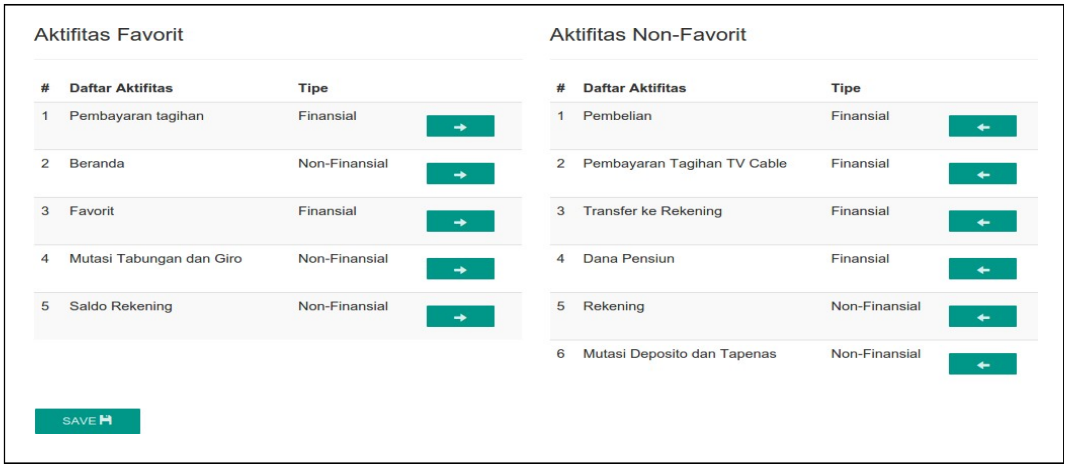
4.3.2 Pengujian Fungsionalitas Menambah Aktifitas Favorit

Pengujian fungsionalitas Menambah Aktifitas Favorit berfungsi untuk melihat apakah proses dalam menambahkan aktifitas yang sering diakses pada *web internet banking* menjadi aktifitas favorit berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 30 Pengujian Fungsi Menambah Aktifitas Favorit. Berdasarkan hasil pengujian yang dilakukan pengguna terhadap fungsionalitas Menambah Aktifitas Favorit maka diperoleh hasil pengujian berhasil. Pengujian fungsionalitas Menambah Aktifitas Favorit ditunjukkan pada Gambar 65.



Gambar 65. Proses menekan hyperlink penambahan aktifitas

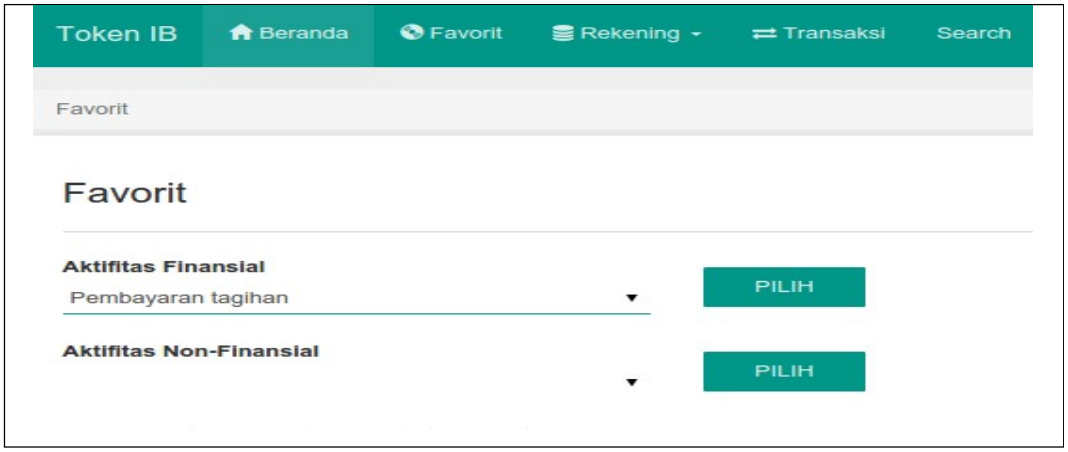
User menekan *hyperlink*, kemudian sistem menampilkan halaman pengaturan aktifitas dilanjutkan dengan menekan tombol panah yang ada pada tabel aktifitas non-favorit dan kemudian menekan tombol *Save* yang ditunjukkan oleh Gambar 56.



Gambar 66. Proses pengujian Menambah Aktifitas Favorit

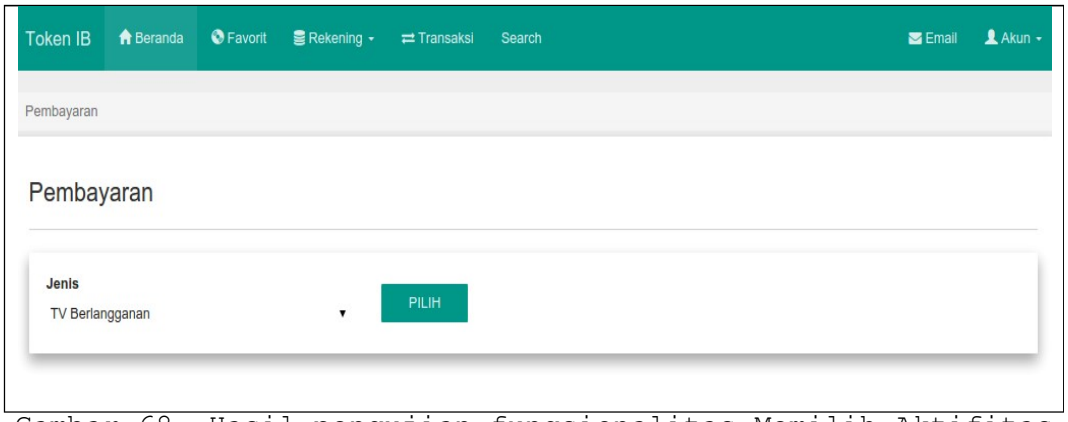
4.3.3 Pengujian Fungsionalitas Memilih Aktifitas Finansial atau Non-Finansial

Pengujian fungsionalitas Memilih Aktifitas Finansial atau Non-Finansial berfungsi untuk melihat apakah proses dalam memilih aktifitas sebagai akses cepat menuju halaman yang diinginkan *user* berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 31 Pengujian Fungsi Memilih Aktifitas Finansial atau Non-Finansial. Pengujian fungsionalitas Menambah Aktifitas Finansial atau Non-Finansial ditunjukkan pada Gambar 67.



Gambar 67. Proses pengujian fungsionalitas Memilih Aktifitas Finansial atau Non-Finansial

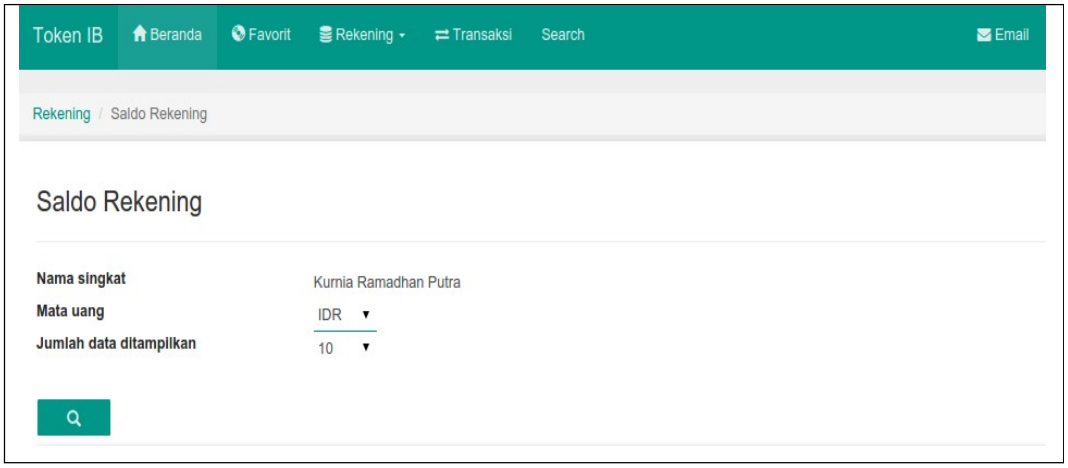
User menekan tombol Pilih, kemudian sistem akan menampilkan halaman yang dipilih, misalnya halaman pembayaran tagihan seperti pada Gambar 68.



Gambar 68. Hasil pengujian fungsionalitas Memilih Aktifitas Finansial atau Non-Finansial

4.3.4 Pengujian Fungsionalitas Mencari Informasi Saldo Rekening

Pengujian fungsionalitas Mencari Informasi Saldo Rekening berfungsi untuk melihat informasi rekening *user* apakah berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 32 Pengujian Fungsi Mencari Informasi Saldo Rekening. Pengujian fungsionalitas Mencari Informasi Saldo Rekening ditunjukkan pada Gambar 69.



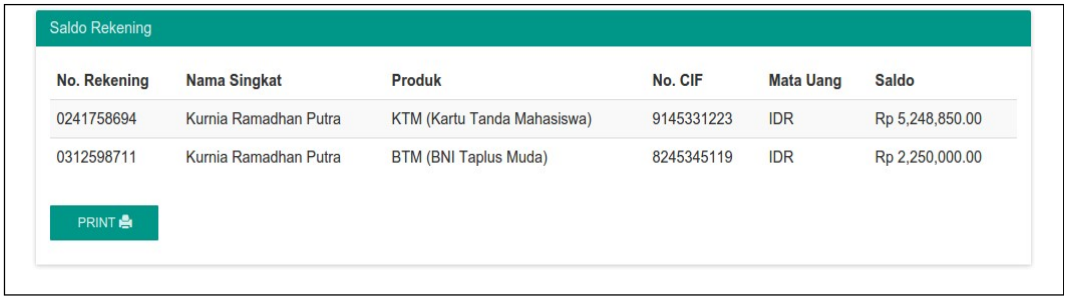
Gambar 69. Proses pengujian fungsionalitas Mencari Informasi Saldo Rekening

User menekan tombol *Search*, maka data akan ditampilkan pada tabel seperti pada Gambar 70.

Gambar 70. Hasil pengujian fungsionalitas Mencari Informasi Saldo Rekening

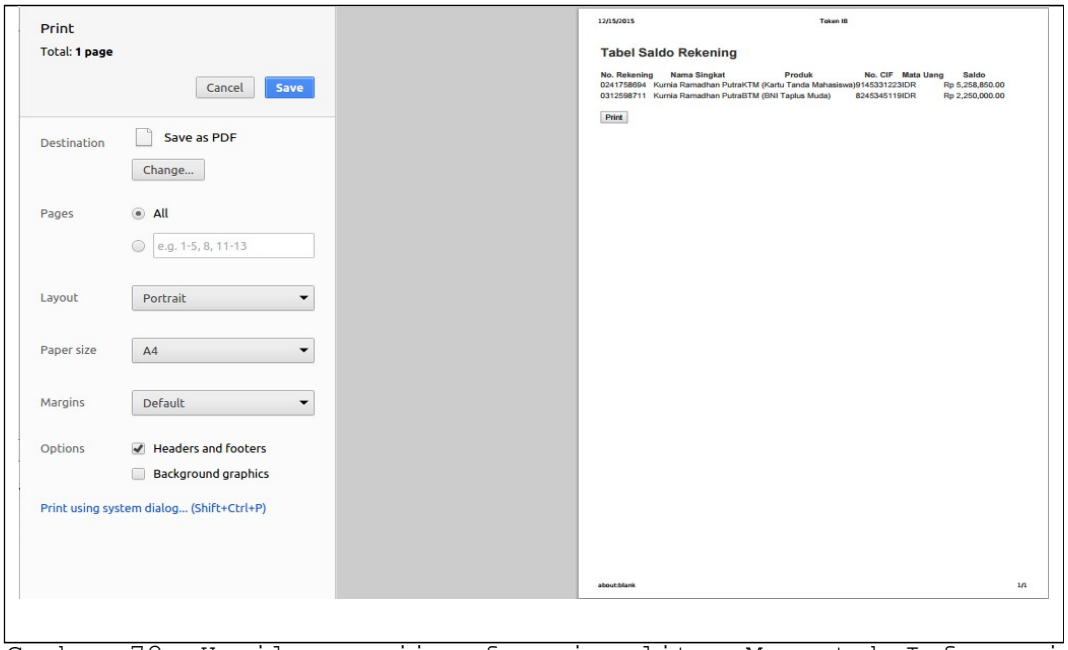
4.3.5 Pengujian Fungsionalitas Mencetak Informasi Saldo Rekening

Pengujian fungsionalitas Mencetak Informasi Saldo Rekening berfungsi untuk mencetak informasi saldo yang ditampilkan pada tabel ke dalam *file* .pdf apakah berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 33 Pengujian Fungsi Mencetak Informasi Saldo Rekening. Pengujian fungsionalitas Mencetak Informasi Saldo Rekening ditunjukkan pada Gambar 71.



Gambar 71. Pengujian fungsionalitas Mencetak Informasi Saldo Rekening

User menekan tombol *Print*, kemudian akan muncul *pop up* menyimpan ke dalam *file* .pdf seperti pada Gambar 72.



Gambar 72. Hasil pengujian fungsionalitas Mencetak Informasi Saldo Rekening

4.3.6 Pengujian Fungsionalitas Mencari Histori Transaksi

Pengujian fungsionalitas Mencari Histori Transaksi berfungsi untuk melihat histori transaksi *user* pada *web internet banking* apakah berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 34 Pengujian Mencari Histori Transaksi. Pengujian fungsionalitas Mencari Histori Transaksi ditunjukkan pada Gambar 73.

Gambar 73. Proses pengujian fungsionalitas Mencari Histori Transaksi

User menekan tombol *Search* dan data akan ditampilkan pada tabel seperti pada Gambar 74.

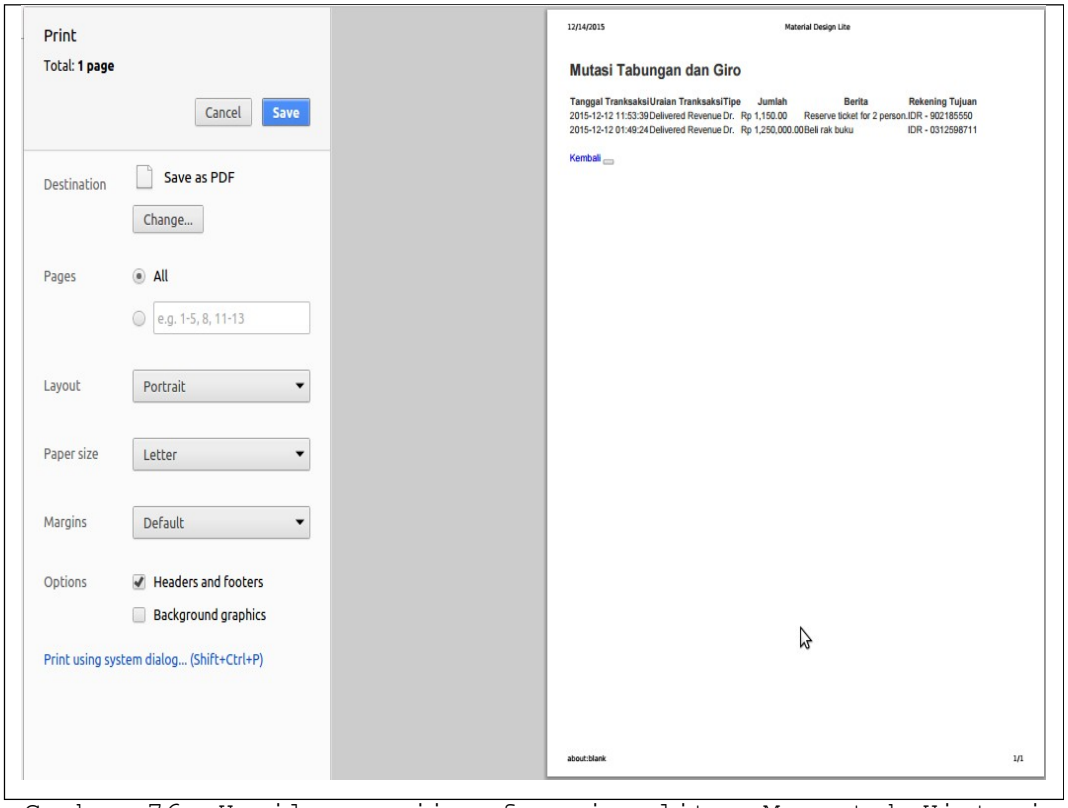
Gambar 74. Hasil pengujian fungsionalitas Mencari Histori Transaksi

4.3.7 Pengujian Fungsionalitas Mencetak Histori Transaksi

Pengujian fungsionalitas Mencetak Histori Transaksi berfungsi untuk mencetak histori transaksi yang ditampilkan pada tabel ke dalam *file* .pdf apakah berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 35 Pengujian Fungsi Mencetak Histori Transaksi. Pengujian fungsionalitas Mencetak Histori Transaksi ditunjukkan pada Gambar 75.

Gambar 75. Pengujian fungsionalitas Mencetak Histori Transaksi

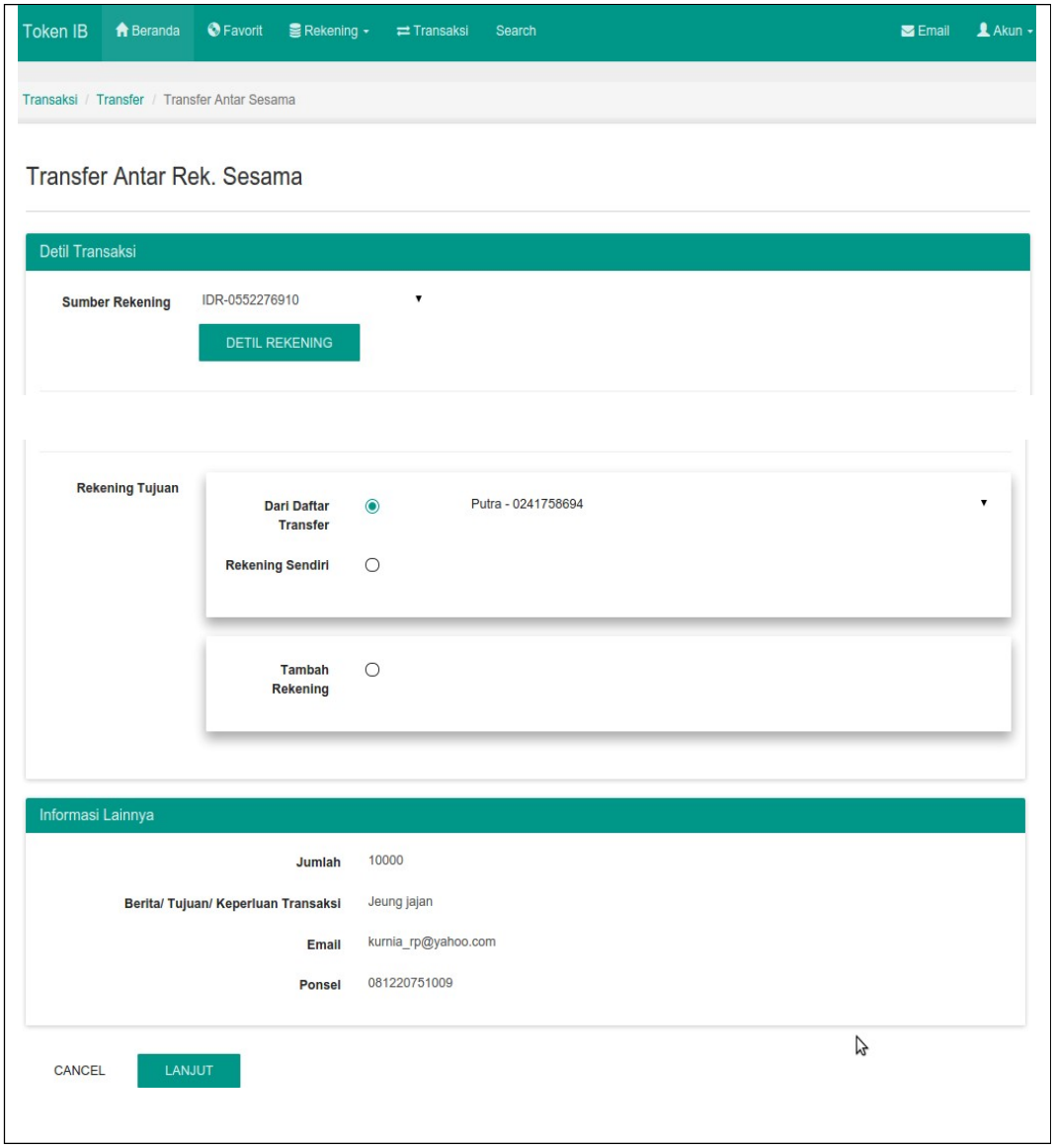
User menekan tombol *Print*, kemudian akan muncul *pop up* menyimpan ke dalam file .pdf seperti pada Gambar 76.



Gambar 76. Hasil pengujian fungsionalitas Mencetak Histori Transaksi

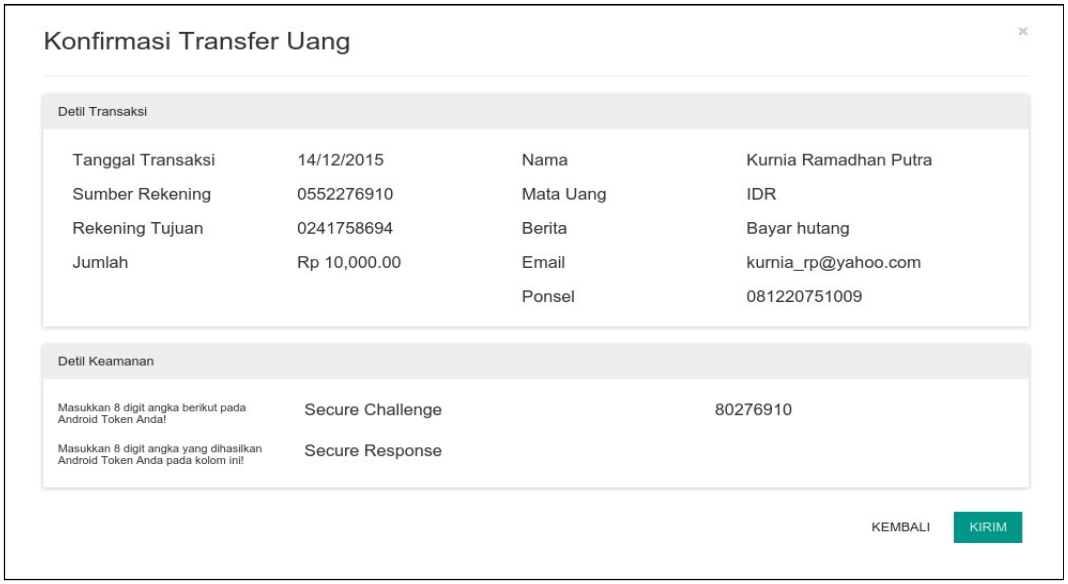
4.3.8 Pengujian Fungsionalitas Mentransfer ke Rekening Sesama

Pengujian fungsionalitas Mentransfer ke Rekening Sesama berfungsi untuk mengirim uang ke sebuah rekening dengan bank yang sama apakah berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 36 Pengujian Fungsi Mentransfer ke Rekening Sesama. Pengujian fungsionalitas Mentransfer ke Rekening Sesama ditunjukkan pada Gambar 77.



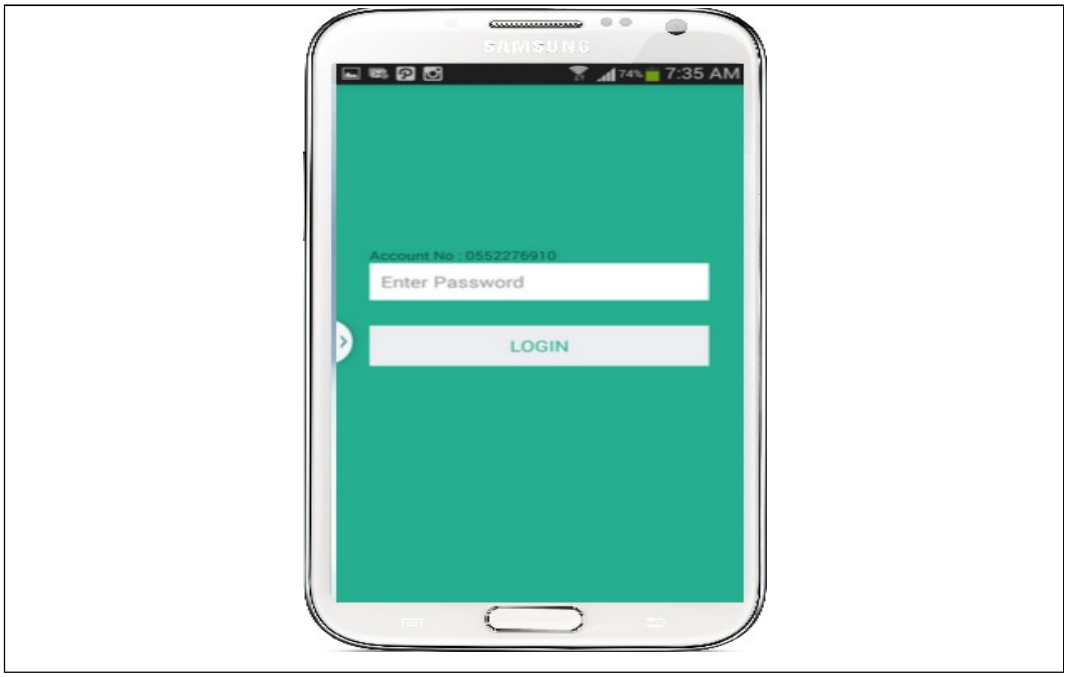
Gambar 77. Proses pengujian fungsionalitas Mentransfer ke Rekening Sesama

User menekan tombol Lanjut, maka akan muncul detil transaksi dan untuk memproses pengiriman user menekan tombol Kirim seperti pada Gambar 78.



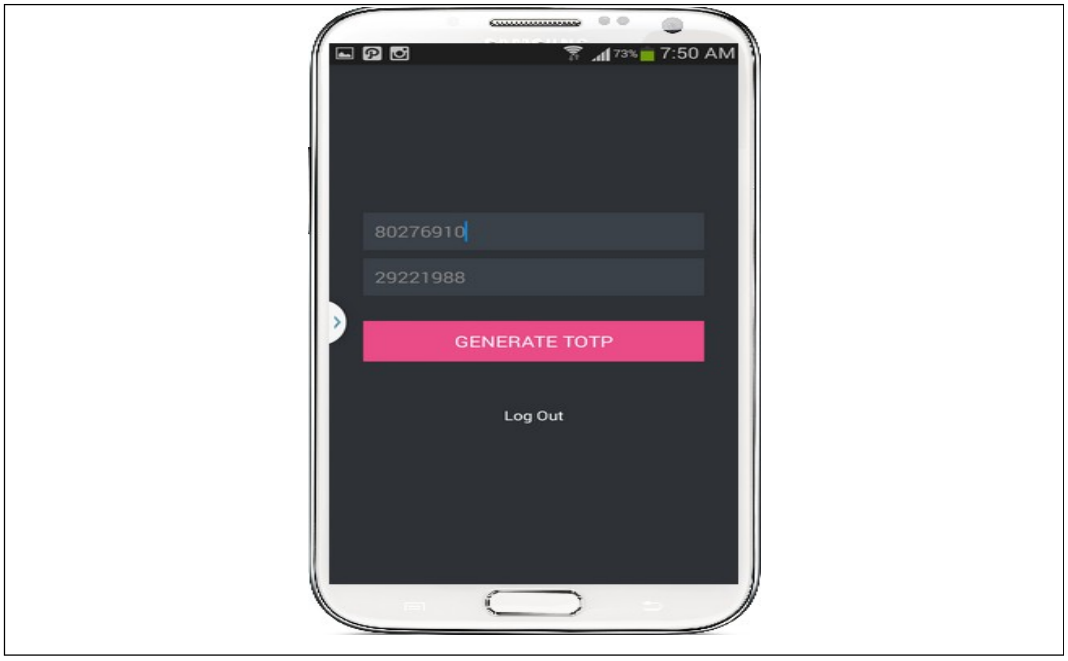
Gambar 78. Detil transaksi pengiriman uang

User memasukkan *secure challenge* ke dalam *android* token untuk mendapatkan *secure response* seperti pada Gambar 79.



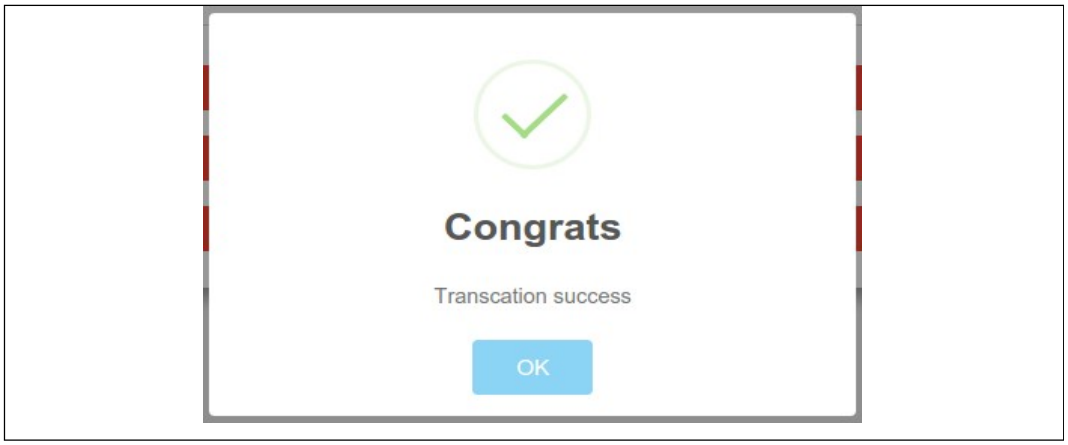
Gambar 79. Proses login pada android token

User memasukkan *password android* token kemudian menekan tombol *Login*, maka akan muncul *form* untuk memasukkan *secure challenge* seperti pada Gambar 80.



Gambar 80. Proses menghasilkan secure response

User memasukkan *secure response* pada *web internet banking* dan setelah menekan tombol Kirim, maka akan muncul *message dialog* yang menampilkan bahwa transaksi berhasil seperti pada Gambar 81.

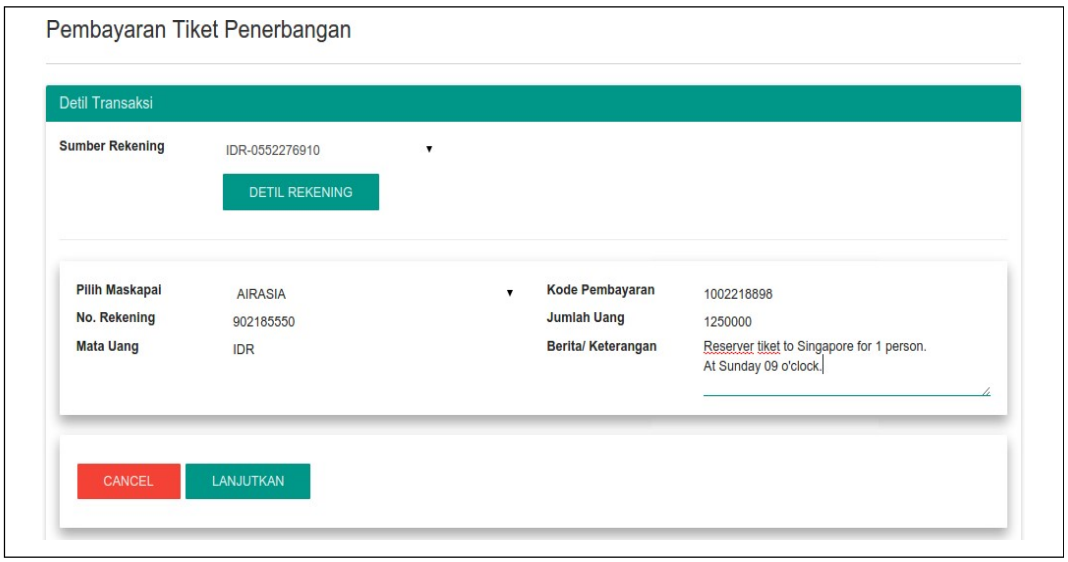


Gambar 81. Hasil pengujian fungsionalitas Mentransfer ke Rekening Sesama

4.3.9 Pengujian Fungsionalitas Pembayaran Tiket Penerbangan

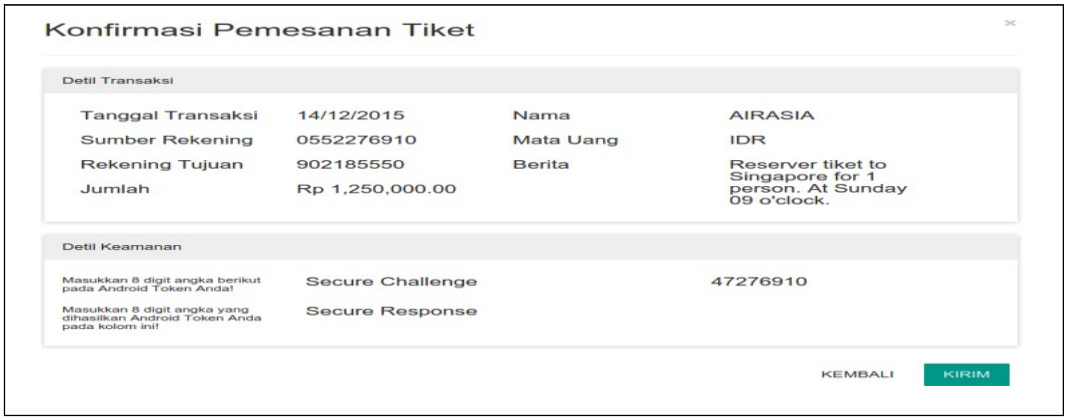
Pengujian fungsionalitas Pembayaran Tiket Penerbangan berfungsi untuk melakukan pembayaran terhadap tiket penerbangan yang sudah dipesan apakah berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran

37 Pengujian Fungsi Pembayaran Tiket Penerbangan. Pengujian fungsionalitas Pembayaran Tiket Penerbangan ditunjukkan pada Gambar 82.



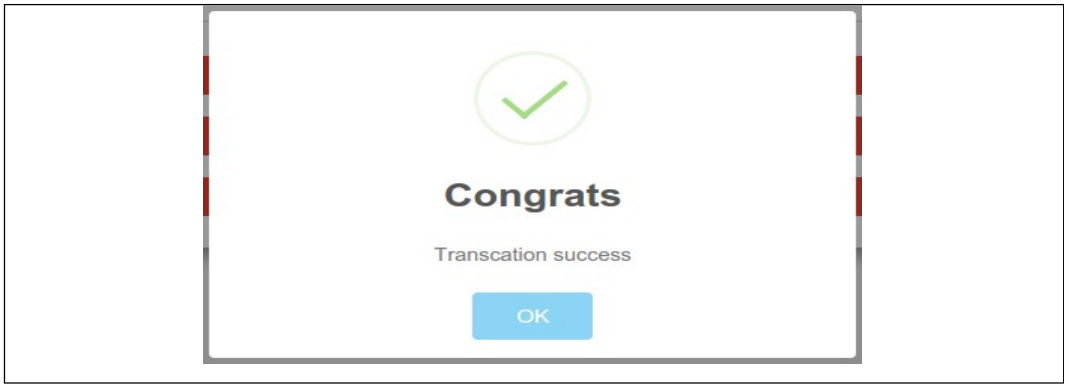
Gambar 82. Proses pengujian fungsionalitas Pembayaran Tiket Penerbangan

User menekan tombol Lanjutkan, maka akan muncul detil transaksi dan untuk memproses pembayaran, user menekan tombol Kirim seperti pada Gambar 83.



Gambar 83. Detil transaksi pembayaran tiket penerbangan

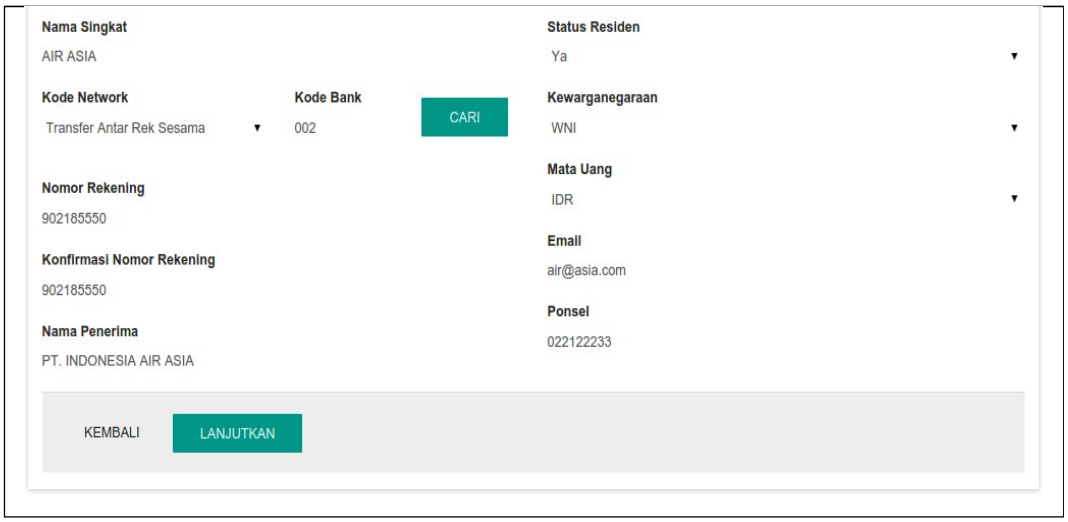
User menekan tombol Kirim, maka muncul *message dialog* yang menampilkan pesan bahwa transaksi berhasil seperti pada Gambar 84.



Gambar 84. Hasil pengujian fungsional Pembayaran Tiket Penerbangan

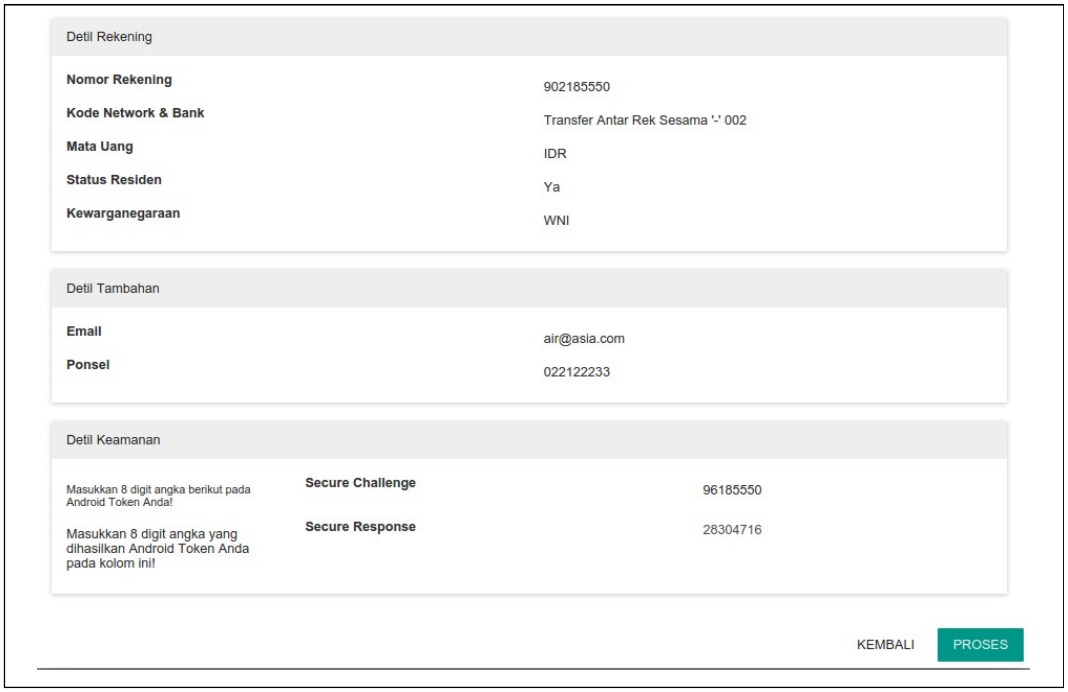
4.3.10 Pengujian Fungsionalitas Menambah Rekening Tujuan

Pengujian fungsionalitas Menambah Rekening Tujuan berfungsi untuk menambah rekening tujuan ke dalam daftar transfer apakah berfungsi dengan baik atau tidak. Pengujian dijabarkan pada Tabel Lampiran 38 Pengujian Fungsi Penambahan Rekening Tujuan. Pengujian fungsionalitas Menambah Rekening Tujuan ditunjukkan pada Gambar 85.



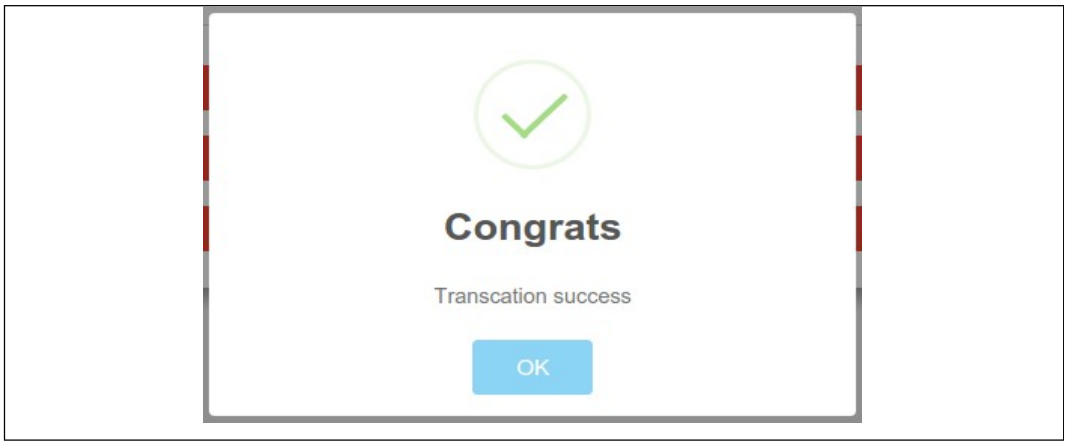
Gambar 85. Proses pengujian fungsionalitas Menambah Rekening Tujuan

User menekan tombol Lanjutkan, maka akan muncul detil informasi dari rekening yang akan ditambahkan dan untuk memproses penambahan rekening, user menekan tombol Proses seperti pada Gambar 86.



Gambar 86. Detil penambahan rekening tujuan

User menekan tombol Proses, maka akan muncul *message dialog* yang menampilkan pesan bahwa rekening berhasil ditambahkan seperti pada Gambar 87.



Gambar 87. Hasil pengujian fungsionalitas Menambah Rekening Tujuan

4.4 Pengujian Algoritma TOTP

Pengujian ini bertujuan untuk menguji kehandalan algoritma TOTP sebagai implementasi dari metode otentikasi *Two Factor Authentication* sehingga perlu diuji beberapa parameter penting yang terdapat pada algoritma TOTP.

4.4.1 Skenario Pengujian

Pengujian yang dilakukan pada algorima TOTP dapat dilihat pada Tabel 24.

Tabel 24. Daftar Skenario Pengujian Algoritma TOTP

No	Nama Skenario	Fungsi
1	Skenario Pengujian *Secret Key*	Membuktikan apakah setiap *secret key* untuk masing-masing token virtual memiliki nilai yang unik.
2	Skenario Pengujian *Challenge Code*	Membuktikan apakah setiap *challenge code* yang dihasilkan oleh *server* memiliki nilai yang unik.
3	Skenario Pengujian *Times*	Membuktikan apakah *password* yang dihasilkan oleh token memiliki masa berlaku selama 3 menit.
4	Skenario Pengujian *Epoch*	Membuktikan apakah proses sinkronisasi berjalan dengan baik dengan cara mengubah waktu antara token virtual dengan server.
5	Skenario Pengujian *Password* OTP	Membuktikan apakah *password* TOTP hanya dapat digunakan untuk sekali pemakaian.
6	Skenario Pengujian 2FA	Membuktikan apakah transaksi dapat dilakukan tanpa menggunakan 2FA.
7	Skenario Pengujian *Brute Force Attack*	Melakukan serangan terhadap token *virtual* untuk mencuri *password* OTP dengan cara mencoba segala kemungkinan kombinasi angka dari *password* OTP.

4.4.1.1 Skenario Pengujian *Secret Key*

Inisialisasi *secret key* diambil dari nomor seri token yang dihasilkan secara *random* dengan panjang sebesar 10 digit dan tiap token memiliki nilai yang unik.

Berikut contoh nilai *secret key* untuk 25 unit token *virtual* seperti pada Tabel 25.

Tabel 25. Hasil Generate Secret Key Tiap User

No	Nama	No.Rekening	Nilai Secret Key
1	Kurnia Ramadhan Putra	0241758694	310567127
2	Uung Ungkawa	0552276910	430925415
3	Dompet Dhuafa	0058333279	847863299
4	Badan Amil Zakat	0995555554	427483987
5	MNC Sky Vision	2173019190	577780340
6	Muhammad Nurchalish	1212010092	555000978
7	Aditya Maryada	0331227681	299313565
8	Ersa Triansyah	4410012331	237221822
9	Muhammad Kurniawan	0121778892	505912260
10	Irfan Afandi	2211688442	359013733
11	Tatan Hartanto Nusa	1211678345	348770396
12	Fairuz Nisa Fauziah	4211622488	544603049
13	Yogi Gustaman	0311681590	822440552
14	Syntia Ula Muiza	3811502432	766932161
15	Fajar Ramadhan	5516684427	402017053
16	Syafitri Putri Iskandar	7111688120	387567064
17	Dicky Hasan Mubaroq	3232688422	196452725
18	Windhy Yulianty Kartika	0812654442	170502039
19	Surya Adikarya Andrian	4401377110	177745060
20	Devi Sulastri	8002668125	827905424
21	Isykariman Ismail	1211678148	963542603
22	Zahwa Fitria Gumilang	4411699459	251196046
23	Farid Ahmad Fakhrullah	3211667676	919319429
24	Amanda Rahmalia Syafitri	4511611486	309030788
25	Muhammad Fathan	3711604224	750450987

Analisis *Secret Key* :

Nilai *secret key* merupakan kombinasi dari 10 digit angka. Total kemungkinan kombinasi angka tersebut dihitung menggunakan faktorial.
Dimana : 10 ! = 10x9x8x7x6x5x4x3x2x1 = 3.628.800 *secret key* yang berbeda antar token. Cara menghasilkan *secret key* adalah :

1. Menentukan nilai minimal = 0
2. Menentukan nilai maksimal = 9
3. Menentukan *length* (panjang data) = 10
4. Melakukan *looping* dengan batas jumlah *secret key.*

Pada Tabel 25 diambil sampel untuk 25 orang nasabah yang menggunakan token *virtual* untuk transaksi *internet banking*, diperoleh *secret key* yang unik untuk masing-masing token *virtual*.

4.4.1.2 Skenario Pengujian *Challenge Code*

Berikut hasil perbandingan kemungkinan *challenge code* yang dihasilkan saat melakukan transaksi seperti pada Tabel 26 dan Tabel 27.

Tabel 26. Kemungkinan challenge code pada transaksi User A

Nama	**No. Rekening**	***Random***	***Challenge Code***
Kurnia Ramadhan Putra	0241758694	00-09	(00-09)758694
		10-19	(10-19)758694
		20-29	(20-29)758694
		39-39	(39-39)758694
		40-49	(40-49)758694
		50-59	(50-59)758694
		60-69	(60-69)758694
		70-79	(70-79)758694
		80-89	(80-89)758694
		90-99	(90-99)758694

Tabel 27. Kemungkinan challenge code pada transaksi User B

Nama	**No. Rekening**	***Random***	***Challenge Code***
Uung Ungkawa	0552276910	00-09	(00-09)276910
		10-19	(10-19)276910
		20-29	(20-29)276910
		39-39	(39-39)276910
		40-49	(40-49)276910
		50-59	(50-59)276910
		60-69	(60-69)276910
		70-79	(70-79)276910
		80-89	(80-89)276910
		90-99	(90-99)276910

Kemudian dilakukan pengujian untuk melihat *challenge code* muncul secara berulang dalam beberapa kali transaksi untuk satu rekening tertentu seperti pada Tabel 28.

Tabel 28. Hasil pengujian challenge code muncul berulang

No. Rekening	**Transaksi**	***Challenge Code***	***Password* OTP**
0241758694	Transaksi ke - 1	72758694	06267046
	Transaksi ke - 2	**23758694**	**95509888**
	Transaksi ke - 3	77758694	01057116
	Transaksi ke - 4	69758694	46018512
	Transaksi ke - 5	75758694	22027475
	Transaksi ke - 6	31758694	71896918
	Transaksi ke - 7	69758694	42329392
	Transaksi ke - 8	56758694	05867466
	Transaksi ke - 9	26758694	15956433
	Transaksi ke - 10	29758694	33156327
	Transaksi ke - 11	**23758694**	**05867466**

Analisis *Challenge Code*:

Nilai *challenge code* yaitu 8 digit angka yang diambil dari 2 digit secara *random* dan 6 digit dari 6 digit terakhir nomor rekening dan memiliki nilai yang unik. Cara menghasilkan *challenge code* adalah :

1. Menentukan nilai minimal = 0
2. Menentukan nilai maksimal = 9
3. Menentukan *length* (panjang data) = 8
4. Melakukan *looping* terhadap 2 digit angka pertama sebanyak $10^2 = 100$ kombininasi.
5. Mengambil 6 digit terakhir nomor rekening.
6. Menggabungkan hasil kombinasi dengan 6 digit nomor rekening.

Pada Tabel 9 dan Tabel 10 dilakukan perbandingan *challenge code* nilainya selalu akan bernilai unik untuk masing-masing *user*. Pada Tabel 11 dilakukan pengujian kemungkinan *challenge code* muncul berulang pada transaksi yang dilakukan oleh *user*. Dari hasil pengujian *challenge code* muncul berulang saat transaksi ke -2 dengan transaksi ke - 11 tetapi dengan nilai OTP yang berbeda.

4.4.1.3 Skenario Pengujian *Times*

Times adalah waktu yang digunakan untuk menentukan masa berlaku *password* OTP dengan rentang waktu sebesar 3 menit.

Berikut pengujian masa berlaku *password* OTP seperti pada Tabel 29.

Tabel 29. Pengujian masa berlaku password OTP

Waktu	*Epoch* Awal (EAw)	*Epoch* Akhir (EAk)	*Range (mdt)* EAk-EAw	*Password* OTP
10 : 57 : 0	1451923020140	1451923021136	996	51435870
10 : 57 : 12	1451923032318	1451923033431	1113	51435870
10 : 57 : 28	1451923048982	1451923049901	919	51435870
10 : 57 : 42	1451923062366	1451923063356	990	51435870
10 : 57 : 56	1451923076498	1451923077498	1000	51435870
10 : 58 : 12	1451923092844	1451923093986	1142	51435870
10 : 58 : 25	1451923105570	1451923106680	1110	51435870
10 : 58 : 38	1451923118199	1451923119302	1103	51435870
10 : 58 : 50	1451923130343	1451923131333	990	51435870
10 : 59 : 1	1451923141419	1451923142578	1159	51435870
10 : 59 : 14	1451923154455	1451923155520	1065	51435870
10 : 59 : 26	1451923166414	1451923167606	1192	51435870
10 : 59 : 39	1451923179402	1451923180310	908	51435870
10 : 59 : 56	1451923196008	1451923197109	1101	51435870
11 : 0 : 8	**1451923208076**	**1451923209116**	**1040**	**33410644**

Analisis *Times* :

Pada Tabel 12 dapat dijelaskan masing-masing atribut :

1. Waktu adalah gabungan dari jam, menit dan detik saat akan memulai menghasilkan *password* OTP.
2. *Epoch* Awal adalah jumlah detik dari waktu saat ini sebelum *password* OTP dihasilkan.
3. *Epoch* Akhir adalah jumlah detik dari waktu saat ini setelah *password* OTP dihasilkan.
4. *Range* adalah jangka waktu yang dibutuhkan untuk menghasilkan password OTP, dimana : *Range* = Epoch Akhir - Epoch Awal.

Dari hasil pengujian pada Tabel 12 dapat dilihat dari waktu **10:57:0 - 10:59:56** (2 menit, 56 detik) memperoleh *password* TOTP yang sama yaitu 51435870 sedangkan pada waktu **11:0:8** (3 menit, 8 detik) menghasilkan

password yang berbeda yaitu 33410644 karena sudah melebihi batas masa berlaku *password* OTP sebesar 3 menit.

4.4.1.4 Skenario Pengujian *Epoch*

Pengujian *Epoch* dilakukan untuk sinkronisasi waktu antara *server* dengan token virtual. Berikut dilakukan pengujian terhadap waktu sebelum sampai sesudah *password* OTP dihasilkan pada Tabel 30.

Tabel 30. Proses sinkronisasi waktu antara server dan client

Keterangan	Inisialisasi Waktu Server	Waktu Client	*OTP Server*	*OTP Client*	Hasil
Waktu Client < Server	6 : 39 : 0	6 : 38 : 0	26410380	20727102	*Invalid*
Waktu Client < Server	6 : 39 : 0	6 : 38 : 59	26410380	20727102	*Invalid*
Waktu Client = Server	**6 : 39 : 0**	**6 : 39 : 0**	**26410380**	**26410380**	***Valid***
Waktu Client = Server	**6 : 39 : 0**	**6 : 39 : 59**	**26410380**	**26410380**	***Valid***
Waktu Client = Server	**6 : 39 : 0**	**6 : 40 : 0**	**26410380**	**26410380**	***Valid***
Waktu Client = Server	**6 : 39 : 0**	**6 : 40 : 0**	**26410380**	**26410380**	***Valid***
Waktu Client = Server	**6 : 39 : 0**	**6 : 40 : 59**	**26410380**	**26410380**	***Valid***
Waktu Client = Server	**6 : 39 : 0**	**6 : 41 : 0**	**26410380**	**26410380**	***Valid***
Waktu Client = Server	**6 : 39 : 0**	**6 : 41 : 59**	**26410380**	**26410380**	***Valid***
Waktu Client > Server	6 : 39 : 0	6 : 42 : 0	26410380	97431993	*Invalid*

Analisis *Epoch* :

Pada Tabel 30 dilakukan pengujian sinkronisasi waktu antara *server* dengan *client* (token *virtual*) dengan penjelasan sebagai berikut :

1. Waktu saat *password* OTP dihasilkan pada server adalah **6:39:0** menghasilkan *password* OTP = 26410380.
2. Waktu pada token *virtual* diubah menjadi **6:38:0** (*time before*) menghasilkan password OTP = 20727102 dan hasil tidak *valid*.
3. Waktu pada token *virtual* diubah menjadi **6:38:59** menghasilkan *password* OTP = 20727102 dan hasil tidak *valid*.
4. Dilakukan sinkronisasi waktu antara token *virtual* dengan *server* menjadi **6:39:0** menghasilkan *password* OTP = 26410380 dan hasil *valid*.

5. Dalam rentang waktu antara **6:39:0 - 6:41:59** menghasilkan *password* OTP = 26410380 dengan hasil *valid*.
6. Waktu pada token *virtual* diubah menjadi **6:42:0** (*time after*) menghasilkan *password* OTP = 97431993 dengan hasil tidak *valid*.

4.4.1.5 Skenario Pengujian *Password* OTP

Pengujian *password* OTP dilakukan untuk beberapa transaksi seperti pada Tabel 31.

Tabel 31. Pengujian password OTP

Transaksi	**Inisialisasi Waktu Server**	***Waktu Client***	***Challenge Code***	***OTP Server***	***OTP Client***
Transaksi 1	10 : 15 : 30	10 : 16 : 5	58758694	73743656	73743656
Transaksi 2	10 : 15 : 30	10 : 16 : 25	58758694	73743656	73743656
Transaksi 3	10 : 15 : 30	10 : 16 : 39	58758694	73743656	73743656
Transaksi 4	10 : 15 : 30	10 : 16 : 55	58758694	73743656	73743656
Transaksi 5	10 : 15 : 30	10 : 17 : 13	58758694	73743656	73743656
Transaksi 6	10 : 15 : 30	10 : 17 : 29	58758694	73743656	73743656
Transaksi 7	10 : 15 : 30	10 : 17 : 54	58758694	73743656	73743656
Transaksi 8	10 : 15 : 30	10 : 18 : 10	58758694	73743656	73743656
Transaksi 9	**10 : 15 : 30**	**10 : 19 : 41**	**58758694**	**73743656**	**54784581**
Transaksi 10	**10 : 15 : 30**	**10 : 20 : 40**	**23758695**	**73743656**	**52952279**

Analisis *Password* OTP :

Pada Tabel 31 dijabarkan sebagai berikut :

1. Nilai *challenge code* dan *password* OTP yang dihasilkan oleh *server* pertama kali adalah 58758694 dan 73743656.
2. Pada Transaksi 1 sampai transaksi 8, token *virtual* menghasilkan *password* yang valid dengan nilai *challenge code* = 58758694 dan nilai OTP = 73743656.

3. Pada transaksi 9, token *virtual* menghasilkan *password* yang tidak *valid* dengan nilai *challenge code* = 58758694 dan nilai OTP = 54784581. Walaupun dengan *challenge code* yang sama tetapi masa berlaku token sudah habis atau melewati batas waktu 3 menit.
4. Pada transaksi 10, token virtual menghasilkan password yang tidak valid dengan *challenge code* = 23758695 dan nilai OTP = 52952279 karena nilai *challenge code* token *virtual* berbeda dengan dengan nilai *chalenge code server*.

4.4.1.6 Skenario Pengujian *Two Factor Authentication* (2FA)

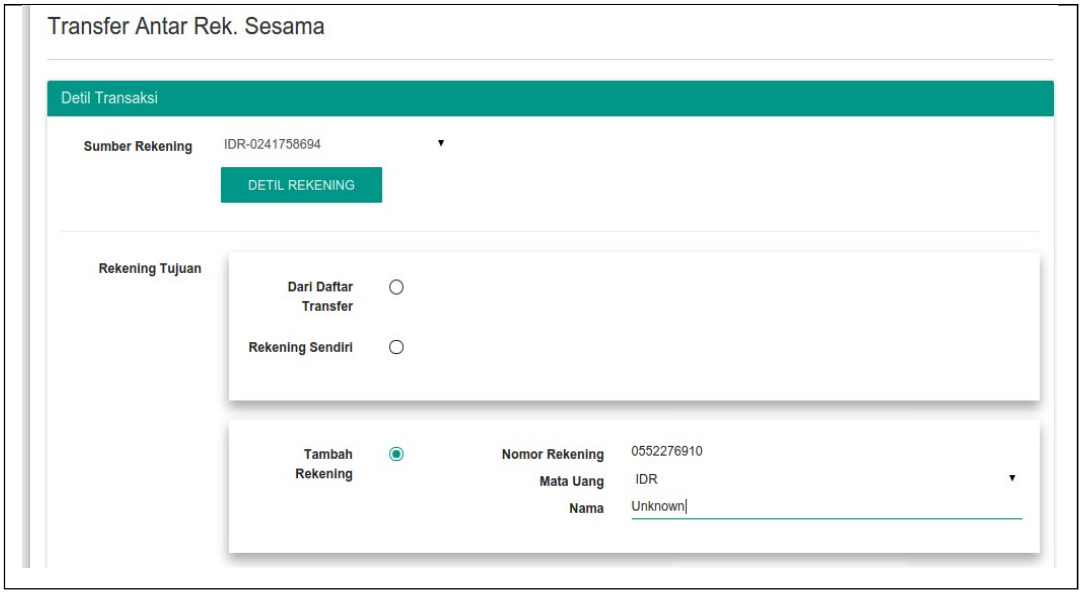
Skenario pengujian 2FA digunakan untuk membuktikan bahwa penggabungan dua otentikasi berbeda secara bersamaan membuat transaksi menjadi lebih aman.

Alur Skenario :

1. Seorang *hacker* berhasil mencuri *username* dan *password* salah satu nasabah dan melakukan *login* pada *web internet banking* seperti pada Gambar 88.

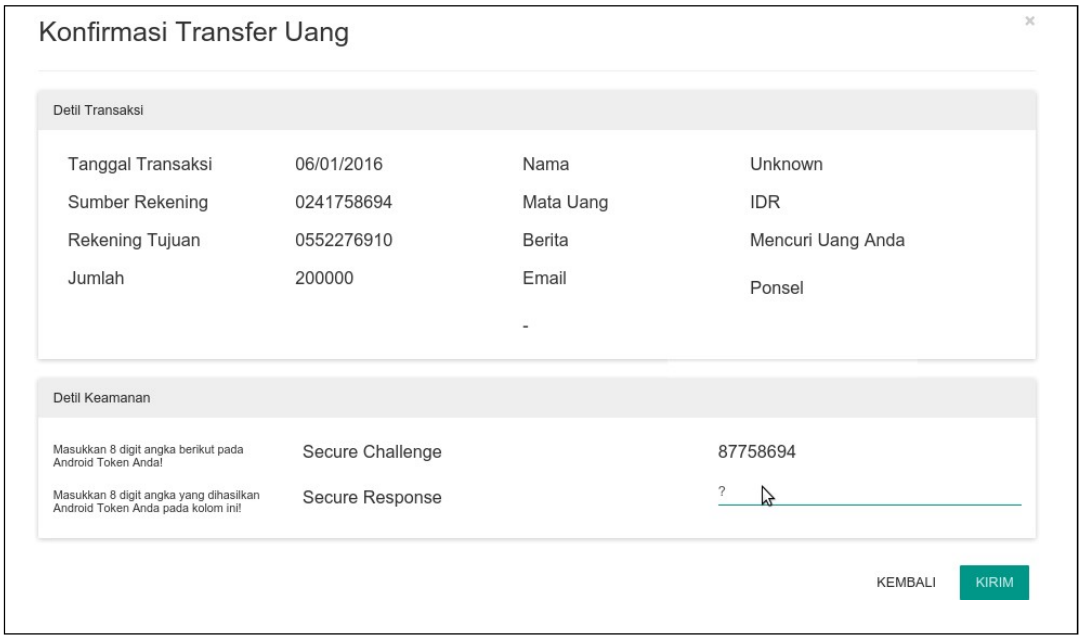
Gambar 88. Hacker melakukan login pada web internet banking

2. *Hacker* melakukan transaksi pengiriman uang dari rekening nasabah yang ditujukan ke rekening *hacker* seperti Gambar 89.



Gambar 89. Hacker melakukan transfer uang dari rekening nasabah

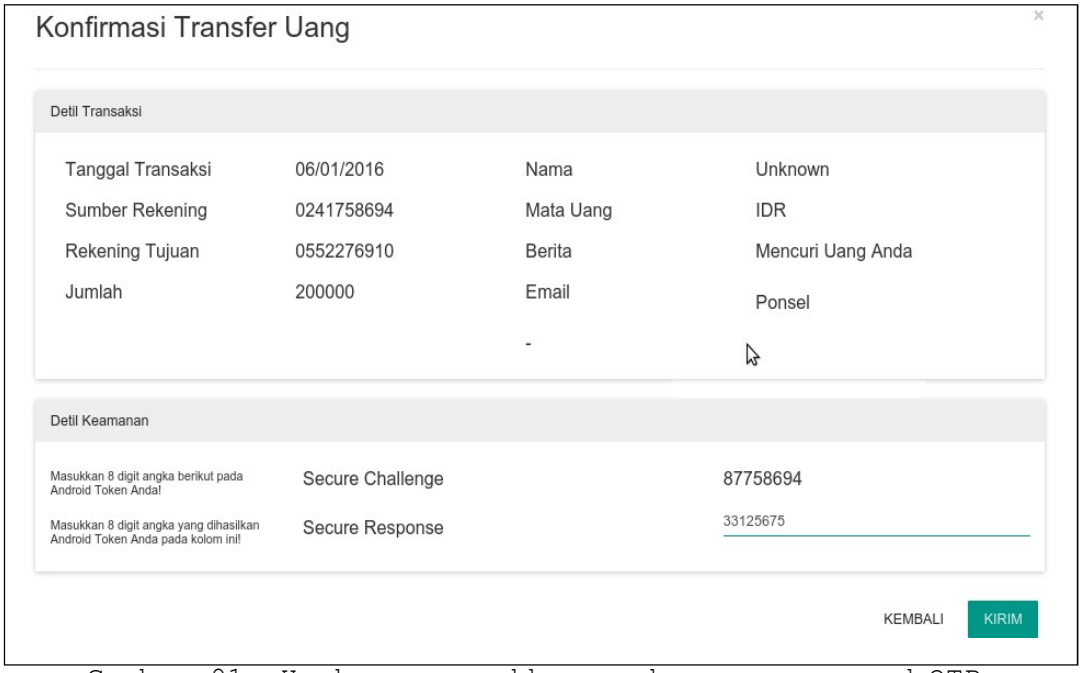
3. Sistem meminta *password* OTP yang dihasilkan oleh token virtual Gambar 90.



Gambar 90. Sistem meminta *password* OTP sebagai validasi

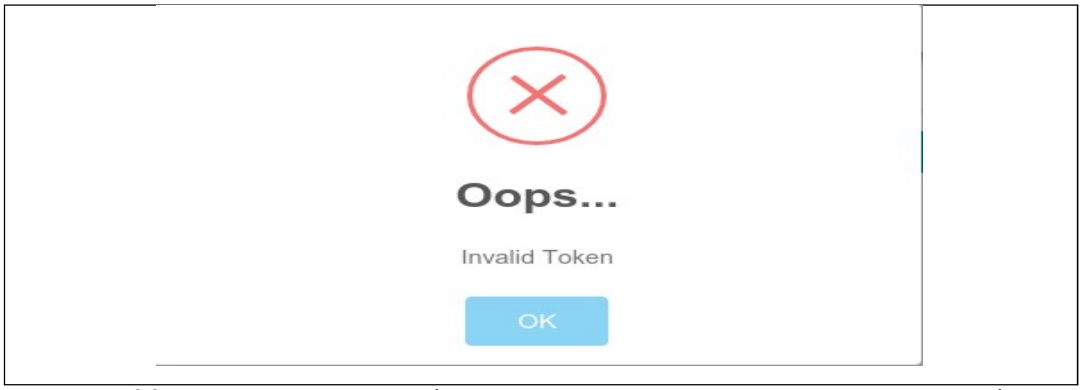
4. *Hacker* tidak dapat melakukan transaksi karena tidak mempunyai alat otentikasi tambahan yaitu token *virtual* untuk menghasilkan *password* OTP.

5. *Hacker* mencoba memasukkan sembarang *password* OTP ke dalam sistem dan menekan tombol Kirim seperti pada Gambar 91.



Gambar 91. Hacker memasukkan sembarang password OTP

6. Sistem melakukan validasi *password* OTP dan menampilkan transaksi ditolak karena *password* OTP antar server dengan client tidak cocok seperti pada Gambar 92.



Gambar 92. Server menampilkan pesan bahwa password OTP tidak valid.

7. *Hacker* mencoba melakukan teknik *Brute Force Attack* untuk menghasilkan *password* OTP dan berhasil mendapatkan password tetapi

telah melewati masa berlaku password OTP sehingga server menganggap bahwa *password* OTP tidak valid seperti pada Tabel 32.

Tabel 32. Hasil Brute Force Attack dari hacker untuk mendapatkan password OTP

Challenge	Password OTP	Brute Force Attack	Menit
87758694	**85859858**	35857242	1
		35858314	2
		35859395	3
		45856477	4
		45857560	5
		45858642	6
		45859723	7
		55856805	8
		55857887	9
		55858970	10
		55859049	11
		65856131	12
		65857213	13
		65858295	14
		65859377	15
		75856457	16
		75857540	17
		75858622	18
		75859701	19
		85856781	20
		85857864	21
		85858947	22
		85859858	23

Analisis *Two Factor Authentication* (2FA):

1. *Hacker* tidak dapat melakukan transaksi karena tidak memiliki alat otentikasi tambahan yaitu token *virtual* untuk menghasilkan *password* OTP walaupun *hacker* sudah berhasil mencuri *password* statis yang digunakan untuk *login* pada *web internet banking*.

2. *Hacker* tidak dapat melakukan transaksi karena memasukkan sembarang *password* OTP dan *server* mengganggap *password* tersebut tidak valid.
3. *Hacker* tidak dapat melakukan transaksi walaupun sudah mendapatkan *password* OTP yang *valid* menggunakan teknik *Brute Force Attack* tetapi masa berlaku token sudah habis karena *password* didapatkan pada menit ke 23.

4.4.1.7 Skenario Pengujian Serangan *Brute Force Attack*

Brute Force Attack adalah tipe serangan secara eksternal yang dilakukan oleh *hacker* dengan mencoba segala kombinasi kode atau angka yang dihasilkan oleh token *virtual*. Berikut dilakukan beberapa kali perobaan *Brute Force Attack* terhadap password OTP seperti pada Tabel 33.

Tabel 33. Pengujian Brute Force Attack

Detik	Durasi	Password OTP	Brute Force Attack	Waktu
188	3 menit 8 detik	10000000	00080303	1 : 40 : 11
			03924408	1 : 41 : 25
			06781806	1 : 42 : 11
			09398627	1 : 43 : 10
			10000000	1 : 43 : 19
194	3 menit 14 detik	10000000	00001091	2 : 10 : 9
			03242582	2: 11 : 12
			06125171	2 : 12 : 9
			08978407	2 : 13 : 2
			10000000	2 : 13 : 23
191	3 menit 11 detik	10000000	00000340	2 : 21 : 55
			02141235	2 : 22 : 41
			03898802	2 : 23 : 15
			05479678	2 : 23 : 44
			06542761	2 : 24 : 4
			0979425	2 : 25 : 3
			10000000	2 : 25 : 7

Analisa Serangan *Brute Force Attack*:

Secara sistematis, langkah-langkah yang dilakukan *Brute Force Attack* adalah melakukan pencocokan satu per satu dengan angka yang menjadi target yaitu *password* OTP. Langkah kerja *Brute Force Attack* adalah :

1. Menyimpan pola angka yang menjadi target.
2. Mencocokkan angka yang dihasilkan dengan angka yang menjadi target secara satu per satu dari kiri ke kanan sampai kondisi berikut terpenuhi :
   a. Angka yang dihasilkan dengan pola tidak cocok (*missmatch*).
   b. Semua angka yang dihasilkan cocok dengan pola dan memberitahukan penemuan posisi (dalam waktu).
3. Algoritma terus mencocokkan dengan pola sampai batas panjang data.
4. Berikut *pseudocode Brute Force Attack* dapat dilihat pada Tabel 34.

Tabel 34. Pseudocode brute force attack

```
procedure BruteForceSearch (
      input m,n : integer;
      input P : array[0..n-1] of String
      input T : array[0..m-1] of String
      output ketemu : array[0..m-1] of Boolean)
Deklarasi :
      i, j : integer
Algoritma :
      for (i=0 to m-1) do
            j=0
            while (j<n and T(i+j) = P[j]) do
                  j=j+1
            end while
            if (j>= n) then
                  ketemu[i] = true
            end if
      end for
```

Pada Tabel 33 dilakukan pengujian sebanyak 3 kali terhadap *password* OTP terkecil yaitu **10000000**. *Brute Force Attack* berhasil melakukan pencarian *password* setelah melewati batas masa berlaku token sebesar 3 menit.

BAB V
PENUTUP

Pada bab ini dibahas tentang kesimpulan dari hasil penelitian yang telah dilakukan sesuai dengan analisis dan perancangan pada Bab III.

5.1 Kesimpulan

Kesimpulan yang dapat diperoleh dari penelitian ini berdasarkan fakta pengujian adalah :

1. Dari hasil pengujian 10 kali transaksi diperoleh waktu rata-rata 2 menit 47 detik untuk setiap transaksi sehingga rentang waktu 3 menit cukup untuk melakukan transaksi.
2. Berdasarkan pengujian sinkronisasi token yaitu waktu token diatur lebih kecil dari waktu server, waktu token sama dengan waktu server dan waktu token lebih besar dari waktu server maka password OTP yang dihasilkan valid jika hanya waktu token sama dengan waktu server.
3. Dari hasil pengujian 100 kombinasi *secret key* yang dihasilkan secara acak menggunakan library java.util.Random tidak menghasilkan angka secara berulang tetapi mempunyai presentasi kemiripan sebanyak 3/100 atau 0,03 %.
4. Berdasarkan hasil pengujian *password* yang dihasilkan oleh Algoritma TOTP bersifat unik karena password tidak muncul secara berulang dan selalu berubah dalam periode tertentu.
5. Berdasarkan hasil pengujian 2FA, *hacker* berhasil mencuri *password login user internet banking* tetapi tidak dapat melakukan transaksi finansial karena meminta user untuk memasukkan *password* OTP yang dihasilkan oleh alat otentikasi tambahan token *virtual* dan hacker mencoba memasukkan sembarang password OTP dan setelah dilakukan validasi *password* OTP tersebut ditolak oleh *server*.

DAFTAR PUSTAKA

1. Dzulqaidah, Aisyah. 2012. *Pembangkitan Kombinasi Kode pada Google Authenticator*. Bandung: Institut Teknologi Bandung.
2. Eastlake 3rd, D., Schiller, J., and S. Crocker. 2005. *Randomness Recommendations for Security.* BCP 106, RFC 4086.
3. Krawczyk, H., Bellare, M., and R. Canetti. 1997. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104.
4. Krisna, Masagus. 2015. *Memoles e-Banking Guna Menangkal Serangan Sinkronisasi Token*. Bandung : Lembaga Riset Telematika Sharing Vision.
5. M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen. 2005. *HOTP: An HMAC-Based One-Time Password Algorithm*. RFC 4226.
6. Pasanda, Obeth. 2013. *Pemanfaatan One Time Password dan Algoritma HMAC-SHA1 pada USB Device*. Yogyakarta: Universitas Kristen Duta Wacana.
7. Peyrott, Sebastian. *Learn how to easily 2FA/ MFA to your Node.js + Express.js apps using TOTP*. Diakses : 14 September 2015. *https://auth0.com.*
8. S.M Abukeshipa, Amna. 2014. *Implementing and Comprising of OTP Techniques (TOTP,HOTP,CROTP) to Prevent Replay Attack in RADIUS Protocol*. Gaza : The Islamic University.
9. Sulistyo, Budi. 2015. *Memoles e-Banking Guna Menangkal Serangan Sinkronisasi Token*. Bandung : Lembaga Riset Telematika Sharing Vision.
10. Wedagama, Made Bayu. Tritoasmoro, Iwan Iwut. Kurniawan Usman, Uke. 2009. *Desain dan Implementasi Sistem Keamanan Algoritma Rindjael (AES) dengan One Time Password Untuk Optimasi SMS Banking*. Bandung: Telkom University.

A.1 Skenario Utama

Skenario Utama adalah skenario *use case* dari proses yang dijalankan oleh sistem dengan kondisi berhasil.

A.1.1 Skenario *Use Case* Fungsionalitas *Login*

Skenario *use case* yang ada pada sisi *user* untuk melakukan proses *login* diuraikan pada Tabel Lampiran 1.

Tabel Lampiran 1. Skenario Use Case Fungsionalitas Login

Identifikasi Masalah	
Nomor	SUC-01
Nama	Fungsionalitas Login
Tujuan	Aktor dapat masuk ke dalam sistem untuk mengakses menu *web internet banking*.
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu login.
Aksi Aktor	**Reaksi Sistem**
1. User mengisi *username* dan *password* pada *form login*. 2. User menekan tombol Login	3. Sistem memverifikasi username dan password yang telah diisi. 4. Sistem memberi respon kepada user bahwa login valid.
Kondisi Akhir	Aktor dapat melakukan login pada sistem dan mengakses menu utama *web internet banking*.

A.1.2 Skenario *Use Case* Fungsionalitas Menambah Aktifitas Favorit

Skenario *use case* yang ada pada sisi *user* untuk melakukan proses penambahan aktifitas favorit diuraikan pada Tabel Lampiran 2.

Tabel Lampiran 2. Skenario Use Case Fungsionalitas Menambah Aktifitas Favorit

Identifikasi Masalah	
Nomor	SUC-02
Nama	Fungsionalitas Menambah Aktifitas Favorit
Tujuan	Aktor dapat menambahkan aktifitas yang sering dilakukan dalam mengakases web internet banking untuk menjadi aktifitas favorit.
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Favorit
Aksi Aktor	**Reaksis Sistem**
1. User menekan *hyperlink* untuk menambahkan aktifitas favorit. 3. User menekan tombol "arrow left" untuk memindahkan aktifitas dari tabel Aktifitas Non-Favorit ke tabel Aktifitas Favorit. 4. User menekan tombol "Save" untuk menyimpan pengaturan.	2. Sistem memberi respon dengan menampilkan halaman pengaturan aktifitas. 5. Sistem membuka koneksi ke database dan menjalan query penyimpanan data ke tabel UserActivity
Kondisi Akhir	Sistem berhasil menjalankan *query* untuk menyimpan Aktifitas Favorit.

A.1.3 Skenario *Use Case* Fungsionalitas Memilih Aktifitas Finansial atau Non-Finansial

Skenario *use case* yang ada pada sisi *user* untuk melakukan proses pemilihan aktifitas finansial atau non-finansial diuraikan pada Tabel Lampiran 3.

Tabel Lampiran 3. Skenario Use Case Fungsionalitas Memilih Aktifitas Finansial dan Non-Finansial

Identifikasi Masalah	
Nomor	SUC-03
Nama	Fungsionalitas Memilih Aktifitas Finansial atau Non-Finansial
Tujuan	Aktor dapat mengakses dengan cepat halaman transaksi Finansial atau Non-Finansial..
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Favorit
Aksi Aktor	**Reaksis Sistem**
1. User memilih salah satu aktifitas yang ada pada *dropdown* Aktifitas Finansial atau Non-Finansial . 3. User menekan tombol Pilih.	2. Sistem menampilkan aktifitas yang dipilih oleh user pada *dropdown*. 5. Sistem menangkap ID dari aktifitas yang dipilih kemudian membaca hyperlink dari aktifitas tersebut dan mengakses halamannya.
Kondisi Akhir	Aktor dapat mengakses halaman transaksi sesuai Aktifitas Finansial atau Non-Finansial yang dipilih.

A.1.4 Skenario *Use Case* Fungsionalitas Mencari Informasi Saldo Rekening

Skenario *use case* yang ada pada sisi *user* untuk mencari informasi saldo rekening diuraikan pada Tabel Lampiran 4.

Tabel Lampiran 4. Skenario Use Case Fungsionalitas Mencari Informasi Saldo Rekening

Identifikasi Masalah	
Nomor	SUC-04
Nama	Fungsionalitas Mencari Informasi Saldo Rekening
Tujuan	Aktor dapat melihat detil informasi saldo dari rekening yang sedang login.
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Saldo Rekening
Aksi Aktor	**Reaksis Sistem**
1. User mengisi *field* nama singkat atau memilih jenis mata uang pada dropdown. 2. User menekan tombol Search dengan icon kaca pembesar.	3. Sistem menampilkan informasi saldo rekening seperti nomor rekening, nama pemilik rekening dan jumlah saldo pada tabel.
Kondisi Akhir	Aktor dapat melihat detil informasi saldo rekening pada tabel.

A.1.5 Skenario *Use Case* Fungsionalitas Mencetak Informasi Saldo Rekening

Skenario *use case* yang ada pada sisi *user* untuk mencetak informasi saldo rekening diuraikan pada Tabel Lampiran 5.

Tabel Lampiran 5. Skenario Use Case Fungsionalitas Mencetak Informasi Saldo Rekening

Identifikasi Masalah	
Nomor	SUC-05
Nama	Fungsionalitas Mencetak Informasi Saldo Rekening
Tujuan	Aktor dapat mencetak informasi saldo yang ditampilkan pada tabel ke dalam file PDF..
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Saldo Rekening
Aksi Aktor	**Reaksis Sistem**
1. User menekan tombol Print. 3. User menekan tombol Save.	2. Sistem menampilkan *Show Modal Print*. 4. Sistem menyimpan file PDF pada direktori komputer.
Kondisi Akhir	Sistem menyimpan informasi saldo rekening dari tabel menjadi file PDF.

A.1.6 Skenario *Use Case* Fungsionalitas Mencari Histori Transaksi

Skenario *use case* yang ada pada sisi *user* untuk mencari histori transaksi diuraikan pada Tabel Lampiran 6.

Tabel Lampiran 6. Skenario Use Case Fungsionalitas Mencari Histori Transaksi

Identifikasi Masalah	
Nomor	SUC-06
Nama	Fungsionalitas Mencari Histori Transaksi
Tujuan	Aktor dapat melihat histori transaksi penarikan atau pengiriman uang berdasarkan nomor rekening yang dipilih.
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Histori Transaksi
Aksi Aktor	**Reaksis Sistem**
1. User memilih berdasarkan periode atau tanggal awal dan tanggal akhir transaksi pada dropdown. 2. User menekan tombol Search.	3. Sistem menampilkan histori transaksi pada tabel
Kondisi Akhir	Aktor dapat melihat histori transaksi baik penarikan uang maupun pengiriman uang.

A.1.7 Skenario *Use Case* Fungsionalitas Mencetak Histori Transaksi

Skenario *use case* yang ada pada sisi *user* untuk mencari histori transaksi diuraikan pada Tabel Lampiran 7.

Tabel Lampiran 7. Skenario Use Case Fungsionalitas Mencetak Histori Transaksi

Identifikasi Masalah	
Nomor	SUC-07
Nama	Fungsionalitas Mencetak Histori Transaksi
Tujuan	Aktor dapat mencetak histori transaksi pada tabel ke dalam file PDF.
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Histori Transaksi
Aksi Aktor	**Reaksis Sistem**
1. User menekan tombol Print. 3. User menekan tombol Save.	2. Sistem menampilkan Show Modal Print. 4. Sistem menyimpan file PDF ke direktori komputer.
Kondisi Akhir	Sistem menyimpan histori transaksi pada tabel menjadi file PDF.

A.1.8 Skenario *Use Case* Fungsionalitas Mentransfer ke Rekening Sesama

Skenario *use case* yang ada pada sisi *user* untuk melakukan transfer ke rekening sesama diuraikan pada Tabel Lampiran 8.

Tabel Lampiran 8. Skenario Use Case Fungsionalitas Menstransfer ke Rekening Sesama

Identifikasi Masalah	
Nomor	SUC-08
Nama	Fungsionalitas Menstransfer ke Rekening Sesama
Tujuan	Aktor dapat melakukan transfer uang ke sebuah rekening dengan bank yang sama.
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Transfer Antar Rekening Sesama
Aksi Aktor	**Reaksis Sistem**
1. User memilih sumber rekening pada dropdown. 3. User memilih rekening tujuan berdasarkan dari daftar transfer, ke rekening sendiri atau input rekening baru. 5. User memasukkan jumlah uang, berita acara, email dan ponsel pada *field*. 6. User menekan tombol Lanjut 8. User memasukkan 8 digit challenge code pada aplikasi Android Token.	2. Sistem menampilkan nomor rekening yang dipilih pada dropdown 4. Sistem menangkap aksi pilihan rekening tujuan. 7. Sistem menampilkan detil informasi pengiriman uang dan men-*generate* 8 digit *challenge code*. 9. Aplikasi Android Token memberikan respon dengan menghasilkan 8 digit password

Tabel Lampiran 9. Skenario Use Case Fungsionalitas Menstransfer ke Rekening Sesama (Lanjutan)

Aksi Aktor	**Reaksi Sistem**
10. User memasukkan 8 digit TOTP password pada field response pada sistem. 11. User menekan tombol Kirim	12. Sistem memproses transaksi dan memindahkan saldo dari rekening sumber ke rekening tujuan sesuai nominal yang dimasukkan.
Kondisi Akhir	Aktor dapat melakukan transaksi pengiriman uang ke sebuah rekening dengan bank yang sama.

A.1.9 Skenario *Use Case* Fungsionalitas Mentransfer Antar Bank

Skenario *use case* yang ada pada sisi *user* untuk melakukan transfer antar bank diuraikan pada Tabel Lampiran 10.

Tabel Lampiran 10. Skenario Use Case Fungsionalitas Menstransfer Antar Bank

Identifikasi Masalah	
Nomor	SUC-09
Nama	Fungsionalitas Menstransfer Antar Bank
Tujuan	Aktor dapat melakukan transfer uang ke sebuah rekening dengan bank yang berbeda.
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Transfer Online Antar Bank

Tabel Lampiran 11. Skenario Use Case Fungsionalitas Menstransfer Antar Bank (Lanjutan)

Aksi Aktor	**Reaksis Sistem**
1. User memilih rekening sumber pada dropdown.	
	2. Sistem menampilkan nomor rekening sumber yang dipilih pada dropdown
3. User memilih kode bank	
	4. Sistem menampilkan kode bank
4. User memilih rekening tujuan pada dropdown.	
	5.Sistem menampilkan nomor rekening tujuan yang dipilih pada dropdown.
6. User memasukkan jumlah uang, berita acara, email dan ponsel pada *field*.	
	8. Sistem menampilkan detil informasi pengiriman uang dan men-*generate* 8 digit *challenge code*.
7. User menekan tombol Lanjut	
9. User memasukkan 8 digit challenge code pada aplikasi Android Token.	
	10. Aplikasi Android Token memberikan respon dengan menghasilkan 8 digit password TOTP.
11. User memasukkan 8 digit TOTP password pada field response pada sistem.	
12. User menekan tombol Kirim	
	13. Sistem memproses transaksi dan memindahkan saldo dari rekening sumber ke rekening tujuan sesuai nominal yang dimasukkan.
Kondisi Akhir	Aktor dapat melakukan transaksi pengiriman uang ke sebuah rekening dengan bank yang berbeda.

A.1.10 Skenario *Use Case* Fungsionalitas Menambah Rekening Tujuan

Skenario *use case* yang ada pada sisi *user* untuk menambah rekening tujuan diuraikan pada Tabel Lampiran 12.

Tabel Lampiran 12. Skenario Use Case Fungsionalitas Menambah Rekening Tujuan

Identifikasi Masalah	
Nomor	SUC-10
Nama	Fungsionalitas Menambah Rekening Tujuan
Tujuan	Aktor dapat menambahkan rekening tujuan ke dalam sistem
Aktor	User
Skenario Utama	
Kondisi Awal	Aktor berada pada menu Tambah Rekening Tujuan
Aksi Aktor	**Reaksis Sistem**
1. User mengisi field nama singkat, kode network, nomor rekening, konfirmasi nomor rekening, nama penerima, status residen, kewarganegaraan, mata uang, email dan ponsel. 2. User menekan tombol Cari Kode Bank 4. User menekan tombol Pilih. 6. User menekan tombol Lanjutkan	3. Sistem menampilkan Show Modal daftar beberapa kode bank. 5. Sistem menampilkan kode bank pada form tambah rekening tujuan 7. Sistem menampilkan detil informasi rekening yang akan ditambahkan serta men-generate 8 digit challenge code.

Tabel Lampiran 13. Skenario Use Case Fungsionalitas Menambah Rekening Tujuan (lanjutan)

Aksi Aktor	**Reaksis Sistem**
8. User memasukkan 8 digit challenge code pada aplikasi Android Token.	
	9. Aplikasi Android Token memberikan respon dengan menghasilkan 8 digit password TOTP.
10. User memasukkan 8 digit TOTP password pada field response pada sistem.	
11. User menekan tombol Simpan	
	12. Sistem menyimpan data rekening baru ke database.
Kondisi Akhir	Aktor dapat menambahkan rekening tujuan ke dalam sistem.

A.2 Skenario Alternatif

Skenario Alternatif adalah skenario *use case* dari proses yang dijalankan oleh sistem dengan kondisi gagal.

A.2.1 Skenario *Use Case* Fungsionalitas Gagal Melakukan Login

Skenario *use case* yang ada pada sisi *user* saat gagal melakukan *login* diuraikan pada Tabel Lampiran 14.

Tabel Lampiran 14. Skenario Use Case Fungsionalitas Gagal Melakukan Login

Identifikasi Masalah	
Aktor	User
Skenario Utama	
Kondisi Awal	User berada pada menu Login
Aksi Aktor	**Reaksi Sistem**
1. User mengisi *username* dan *password* pada *form login*. 2. User menekan tombol Login	2. Sistem memverifikasi username dan password yang telah diisi. 3. Sistem memberi respon kepada user bahwa login tidak valid.
Kondisi Akhir	Aktor tidak dapat masuk ke menu utama dan harus memasukkan username dan password yang valid

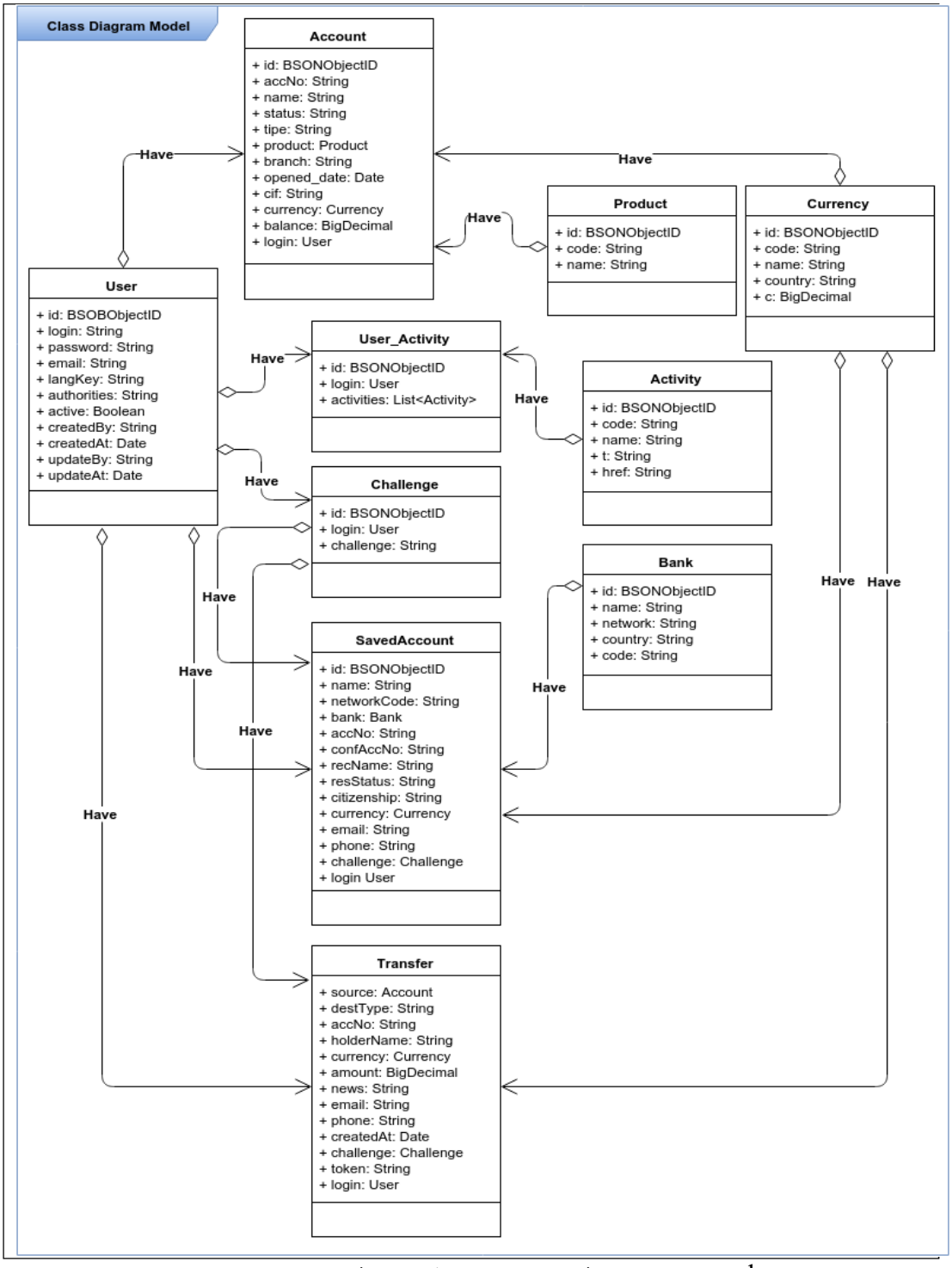
A.2.1 Skenario *Use Case* Fungsionalitas *Password* TOTP Tidak Valid

Skenario *use case* yang ada pada sisi *user* saat memasukkan *password* TOTP yang tidak valid diuraikan pada Tabel Lampiran 15.

Tabel Lampiran 15. Skenario Use Case Fungsionalitas Password TOTP Tidak Valid

Identifikasi Masalah	
Aktor	User
Skenario Utama	
Kondisi Awal	User berada pada menu Transaksi
Aksi Aktor	**Reaksi Sistem**
1. User mengisi field pada form transaksi	
2. User menekan tombol Lanjutkan	3. Sistem menampilkan detil transaksi serta 8 digit challenge code
4. User memasukkan 8 digit challenge code pada aplikasi Android Token.	5. Aplikasi Android Token memberikan respon dengan menghasilkan 8 digit password TOTP.
6. User memasukkan 8 digit TOTP password pada field response pada sistem.	
7. User menekan tombol Kirim	8. Sistem membandingkan 8 digit TOTP password yang dikirim oleh user dengan yang tersimpan di database 9. Sistem memberikan notifikasi bahwa TOTP password tidak valid.
Kondisi Akhir	Sistem tidak dapat memproses transaksi dan meminta TOTP password yang baru.

B.1 Class Diagram Model



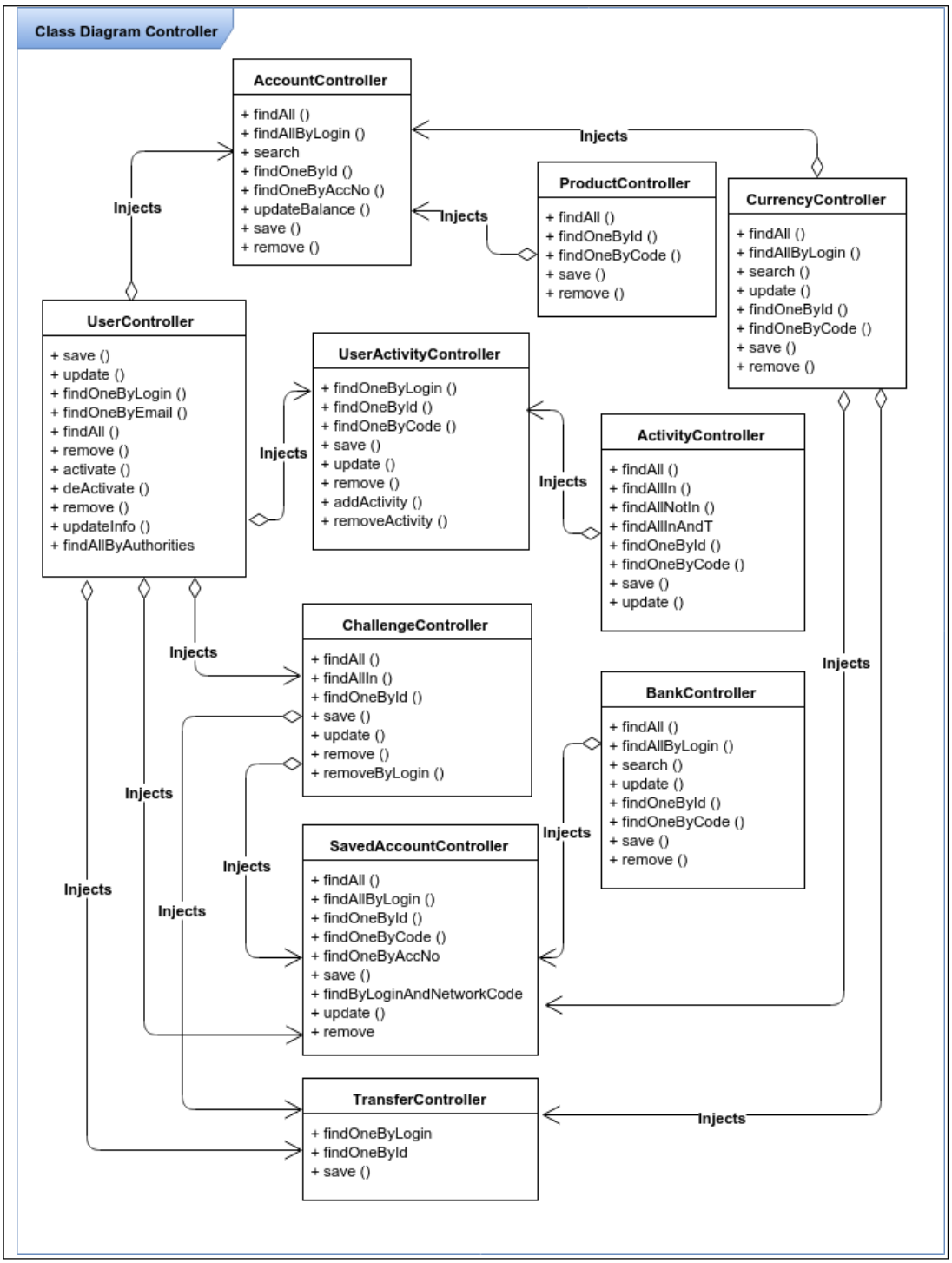
Gambar Lampiran 1. Class Diagram Model

Deskripsi dari masing-masing class pada *Class Diagram Model* diuraikan pada Tabel Lampiran 16.

Tabel Lampiran 16. Deskripsi Class Diagram Model

No.	Nama Class	Fungsi
1	User	Permodelan pada sistem yang mewakili entitas dari user untuk menyimpan data-data user yang mempunyai hak akses untuk login ke dalam sistem.
2	Account	Permodelan pada sistem yang mewakili entitas rekening untuk menyimpan data-data rekening yang dimiliki oleh User.
3	Product	Permodelan pada sistem yang mewakili entitas produk untuk menyimpan data-data produk yang dimiliki oleh Account.
4	Currency	Permodelan pada sistem yang mewakili entitas mata uang untuk menyimpan data-data mata uang.
5	Activity	Permodelan pada sistem yang mewakili entitas layanan untuk menyimpan data-data layanan yang ada sistem.
6	User_Activity	Permodelan pada sistem yang mewakili entitas layanan untuk menyimpan data-data layanan yang sering diakses oleh user pada sistem.
7	Bank	Permodelan pada sistem yang mewakili entitas bank untuk menyimpan data-data bank untuk transfer antar rekening
8	SavedAccount	Permodelan pada sistem yang digunakan untuk menyimpan data-data pada saat user melakukan penambahan rekening tujuan yang baru ke dalam sistem.
9	Transfer	Permodelan pada sistem yang digunakan untuk menyimpan data-data pada saat user melakukan transfer uang ke rekening lain.

B.2 Class Diagram Controller



Gambar Lampiran 2. Class Diagram Controller

Deskripsi dari masing-masing *class* pada *Class Diagram Controller* diuraikan pada Tabel Lampiran 17 sampai Tabel Lampiran 26..

Tabel Lampiran 17. Class Diagram Account Controller

No	Nama Fungsi	Keterangan
1	**save ()**	Menyimpan data user ke database
2	**update ()**	Melakukan update data user dari database
3	**findOneByLogin ()**	Mencari data user berdasarkan username
4	**findOneByEmail ()**	Mencari data user berdasarkan email
5	**findAll ()**	Mencari seluruh data user yang ada pada database
6	**remove ()**	Menghapus data user
7	**activate ()**	Mengaktifkan user
8	**deActivate ()**	Menon-aktifkan user
9	**updateInfo ()**	Melakukan update informasi user
10	**findAllByAuthorities**	Mencari semua data user berdasarkan otoritas

Tabel Lampiran 18. Class Diagram Account Controller

No	Nama Fungsi	Keterangan
1	**findAll ()**	Mencari data semua rekening pada database
2	**findAllByLogin ()**	Mencari data semua rekening berdasarkan username yang login
3	**search ()**	Mencari data rekening dari user yang melakukan login
4	**findOneById ()**	Mencari data sebuah rekening berdasarkan ID
5	**findOneByAccNo ()**	Mencari data sebuah rekening berdasarkan nomor rekening
6	**updateBalance ()**	Melakukan update terhadapa saldo yang tersimpan di dalam rekening jika telah melakukan pengiriman atau penarikan uang.
7	**save ()**	Menyimpan data rekening ke database
8	**remove ()**	Menghapus data rekening dari database

Tabel Lampiran 19. Class Diagram Product Controller

No	Nama Function	Keterangan
1	**findAll ()**	Mencari data semua produk yang dimiliki oleh rekening.
4	**findOneById ()**	Mencari data sebuah produk berdasarkan ID
3	**findOneByCode ()**	Mencari data sebuah rekening berdasarkan kode produk
4	**save ()**	Menyimpan data produk ke database
5	**remove ()**	Menghapus data produk dari database

Tabel Lampiran 20. Class Diagram Currency Controller

No	Nama Function	Keterangan
1	**findAll ()**	Mencari data semua mata uang pada database
2	**findAllByLogin ()**	Mencari data semua mata uang berdasarkan username yang login
3	**search ()**	Mencari data mata uang yang dimiliki oleh user yang melakukan login
4	**update ()**	Melakukan update terhadap perubahan mata uang
5	**findOneById ()**	Mencari data sebuah mata uang berdasarkan ID
6	**findOneByCode ()**	Mencari data sebuah mata uang berdasarkan kode mata uang.
7	**save ()**	Menyimpan data mata uang ke database
8	**remove ()**	Menghapus data mata uang dari database

Tabel Lampiran 21. Class Diagram UserActivity Controller

No	Nama Function	Keterangan
1	**findOneByLogin ()**	Mencari data sebuah aktifitas yang dipilih oleh user sebagai aktifitas favorit berdasarkan username yang melakukan login.
2	**findOneById ()**	Mencari data sebuah aktifitas yag dipilih oleh user sebagai aktifitas favorit berdasarkan ID
3	**findOneByCode ()**	Mencari data sebuah aktifitas yang dipilih user sebagai aktifitas favorit berdasarkan kode
4	**save ()**	Menyimpan pengaturan aktifitas yang telah dipilih oleh user.
5	**update ()**	Melakukan update terhadapa perubahan aktifitas yang dipilih oleh user
6	**remove ()**	Menghapus aktifitas yang dipilih oleh user.
7	**addActivity ()**	Menambahkan aktifitas yang dipilih oleh user
8	**removeActivity ()**	Menghapus data mata uang dari database berdasarkan ID aktifitas.

Tabel Lampiran 22. Class Diagram Activity Controller

No	Nama Function	Keterangan
1	**findAll ()**	Mencari data semua aktifitas pada database.
2	**findAllIn ()**	Mencari data semua aktifitas yang berada di dalam join
3	**findAllNotIn ()**	Mencari data semua aktifitas yang berada di luar join
4	**findAllInAndT ()**	Mencari data semua aktifitas berdasarkan tipe
5	**findOneById ()**	Mencari data sebuah aktifitas berdasarkan ID
6	**findOneByCode ()**	Mencari data sebuah aktifitas berdasarkan kode aktifitas.
7	**save ()**	Menyimpan data aktifitas ke database
8	**update ()**	Melakukan update terhadap perubahan data aktifitas.

Tabel Lampiran 23. Class Diagram Bank Controller

No	Nama Function	Keterangan
1	**findAll ()**	Mencari data semua bank pada database.
2	**findAllByLogin ()**	Mencari data semua bank berdasarkan username yang melakukan login
3	**search ()**	Mencari data bank yang dimiliki oleh user yang melakukan login
4	**update ()**	Melakukan update terhadap perubahan data bank
5	**findOneById ()**	Mencari data sebuah bank berdasarkan ID
6	**findOneByCode ()**	Mencari data sebuah bank berdasarkan kode bank.
7	**save ()**	Menyimpan data bank ke database
8	**remove ()**	Menghapus data bank dari database.

Tabel Lampiran 24. Class Diagram SavedAccount Controller

No	Nama Function	Keterangan
1	**findAll ()**	Mencari data semua rekening yang baru.
2	**findAllByLogin ()**	Mencari data semua rekening yang baru berdasarkan username yang login
3	**findAllById ()**	Mencari data semua rekening yang baru berdasarkan ID
4	**findAllByCode ()**	Mencari data semua rekening yang baru berdasarkan kode
5	**findAllByAccNo ()**	Mencari data semua rekening yang baru berdasarkan nomor rekening
6	**findByLoginAndNetworkCode ()**	Mencari data sebuah rekening berdasarkan kode network.
7	**update ()**	Melakukan update terhadap data rekening yang baru
8	**remove ()**	Menghapus data rekening yang baru dari database

Tabel Lampiran 25. Class Diagram Transfer Controller

No	Nama Function	Keterangan
1	**findOneByLogin ()**	Mencari data sebuah transfer berdasarkan username yang melakukan login.
2	**findOneById ()**	Mencari data sebuah transfer berdasarkan Id
3	**save ()**	Menyimpan data transfer yang dilakukan oleh user ke database

Tabel Lampiran 26. Class Diagram Challenge Controller

No	Nama Function	Keterangan
1	**findAll ()**	Mencari data semua challenge code pada database.
2	**findAllIn ()**	Mencari data semua challengecode yang berada di dalam join
3	**findOneById ()**	Mencari data sebuah challenge code berdasarkan ID
4	**save ()**	Menyimpan challenge code ke database
5	**update ()**	Melakukan update terhadap perubahan challenge code
6	**remove ()**	Menghapus challenge code dari database
7	**removeByLogin**	Menghapus challenge code dari database berdasarkan username yang melakukan login

B.3 Modul View

Deskripsi *class-class* yang terdapat pada modul *view* diuraikan pada `Tabel Lampiran 27`.

`Tabel Lampiran 27. Class-class pada Modul View`

No	Class HTML	Class Javascripts	Fungsi
1	login.html	login.controller.js	Sebagai halaman untuk melakukan login terhadap sistem agar dapat mengakses menu utama.
2	dashboard.html	dashboard.controller.js	Sebagai halaman utama dari sistem yang menampilkan informasi user yang sedang login, informasi email dan informasi berita terkini.
3	favorit.html	favorit.controller.js	Sebagai halaman untuk akses cepat ke halaman transaksi yang diinginkan oleh user.
4	saldo_rekening.html	saldo_rekening.controller.js	Sebagai halaman untuk menampilkan informasi saldo dari user.
5	histori.html	histori.controller.js	Sebagai halaman untuk menampilkan histori transaksi user.
6	transfer_sesama.html	transfer_sesama.controller.js	Sebagai halaman untuk melakukan transfer uang ke suatu rekening dengan bank yang sama
7	atur_rekening.html	atur_rekening.controller.js	Sebagai halaman untuk menambahkan nomor rekening ke daftar tujuan transaksi
8	pembayaran.html	pembayaran.controller.js	Sebagai halaman untuk menampilkan daftar pembayaran tagihan
10	penerbangan.html	pembayaran.controller.js	Sebagai halaman untuk melakukan pembayaran pemesanan tiket pesawat.

B.4 Modul Service

Deskripsi *class-class* yang terdapat pada modul *service* diuraikan pada Tabel Lampiran 28.

Tabel Lampiran 28. Class-class pada Modul Service

No	Nama Class	Fungsi
1	AccountService.scala	Layanan yang digunakan untuk menangani data rekening seperti pembuatan rekening baru dan pergantian password.
2	CommonService.scala	Layanan yang digunakan untuk menghasilkan *challenge code* secara random.
3	MailService.scala	Layanan yang digunakan untuk menangani email pengguna yang terdaftar pada web internet banking seperti menulis email, email masuk dan email keluar.
4	TOTPService.scala	Layanan yang digunakan untuk menghasilkan token menggunakan algoritma Time_Based One Time Password (TOTP)
5	TransferService.scala	Layanan yang digunakan untuk menangani pengiriman uang antar rekening yang terdaftar pada web internet banking.

B.5 Modul Util

Deskripsi *class-class* yang terdapat pada modul *util* diuraikan pada Tabel Lampiran 29.

Tabel Lampiran 29. Class-class pada modul service

No	Nama Class	Fungsi
1	BSONFormat.scala	Sebagai pengatur mapping struktur data dari model ke database dan dari database ke model.
2	JSONFormat.scala	Sebagai pengatur mapping struktur data dari model ke file JSON dan dari file JSON ke model.

Tabel Lampiran 30. Pengujian Fungsi Login

IDENTIFIKASI			
Nomor	**SPA-01**		
Nama	**Login**		
Tujuan	Aktor dapat masuk ke dalam sistem untuk mengakses menu *web internet banking*		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu login		
Pengujian			
Skenario Uji			
1. Mengisi username dan password 2. Menekan tombol Login			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
Username dan password	Sistem dapat menampilkan menu utama.	Sistem menampilkan menu utama	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 31. Pengujian Fungsi Menambah Aktifitas Favorit

IDENTIFIKASI			
Nomor	**SPA-02**		
Nama	**Menambah Aktifitas Favorit**		
Tujuan	Aktor dapat menambahkan aktifitas yang sering dilakukan dalam mengakases web internet banking untuk menjadi aktifitas favorit.		
Aktor	User		
Skenario			
Kondisi Awal	**Kondisi Awal**		
Pengujian			
Skenario Uji			
1. Menekan hyperlink “Klik disini untuk menambahkan aktifitas favorit Anda” 2. Menekan tombol “arrow left” untuk memindahkan dari aktifitas non-favorit ke aktifitas favorit. 3. Menekan tombol Save			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
ID dari setiap aktifitas	Sistem dapat memindahkan data dari tabel aktifitas non-favorit ke tabel aktifitas favorit.	Sistem memindahkan data dari tabel aktifitas non-favorit ke tabel aktifitas favorit.	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 32. Pengujian Fungsi Memilih Aktifitas Finansial atau Non-Finansial

IDENTIFIKASI			
Nomor	**SPA-03**		
Nama	**Memilih Aktfitas Finansial atau Non-Finansial**		
Tujuan	Aktor dapat mengakses dengan cepat halaman transaksi Finansial atau Non-Finansial.		
Aktor	User		
Skenario			
Kondisi Awal	**Kondisi Awal**		
Pengujian			
Skenario Uji			
1. Memilih salah satu aktifitas finansial atau non-finansial yang ada *pada dropdown list* 2. Menekan tombol Pilih			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
ID dari setiap aktifitas	Sistem dapat menuju lokasi halaman web yang dipilih melalui dropdown list.	Sistem menuju lokasi halaman sesuai dengan pilihan pada dropdown list.	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 33. Pengujian Fungsi Mencari Informasi Saldo Rekening

IDENTIFIKASI			
Nomor	**SPA-04**		
Nama	**Mencari Informasi Saldo Rekening**		
Tujuan	Aktor dapat melihat detil informasi saldo dari rekening user yang sedang melakukan login.		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu Saldo Rekening		
Pengujian			
Skenario Uji			
1. Mengisi field nama singkat dan memilih jenis mata uang. 2. Menekan tombol search.			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
Nama singkat dan jenis mata uang	Sistem dapat menampilkan informasi saldo rekening yang dicari pada tabel.	Sistem menampilkan informasi saldo rekening user pada tabel.	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 34. Pengujian Fungsi Mencetak Informasi Saldo Rekening

IDENTIFIKASI			
Nomor	**SPA-05**		
Nama	**Mencetak Informasi Saldo Rekening**		
Tujuan	Aktor dapat mencetak informasi saldo yang ditampilkan pada tabel ke dalam file PDF.		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu Saldo Rekening		
Pengujian			
Skenario Uji			
1. Menekan tombol Print 2. Memberi nama file 3. Menekan tombol Save			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
ID dari tabel saldo rekening	Sistem dapat mengubah tabel ke dalam file pdf dan menyimpannya ke direktori komputer.	Sistem menampilkan jendela file pdf dan menyimpan ke direktori komputer	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 35. Pengujian Fungsi Mencari Histori Transaksi

IDENTIFIKASI			
Nomor	**SPA-06**		
Nama	**Mencari Histori Transaksi**		
Tujuan	Aktor dapat melihat histori transaksi penarikan atau pengiriman uang berdasarkan nomor rekening yang dipilih.		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu Histori Transaksi		
Pengujian			
Skenario Uji			
1. Memilih periode atau mengisi tanggal awal dan tanggal akhir 2. Menekan tombol Search			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
Periode dan tanggal awal dan tanggal akhir	Sistem dapat menampilkan histori transaksi yang dicari pada tabel.	Sistem menampilkan histori transaksi sesuai dengan tanggal yang dicari	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 36. Pengujian Fungsi Mencetak Informasi Saldo Rekening

IDENTIFIKASI			
Nomor	**SPA-07**		
Nama	**Mencetak Histori Transaksi**		
Tujuan	Aktor dapat mencetak histori transaksi pada tabel ke dalam file PDF.		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu Histori Transaksi		
Pengujian			
Skenario Uji			
1. Menekan tombol Print 2. Memberi nama file 3. Menekan tombol Save			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
ID dari tabel histori transaksi	Sistem dapat mengubah tabel ke dalam file pdf dan menyimpannya ke direktori komputer.	Sistem menampilkan jendela file pdf dan menyimpan ke direktori komputer	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 37. Pengujian Fungsi Mentransfer ke Rekening Sesama

IDENTIFIKASI			
Nomor	**SPA-08**		
Nama	**Mentransfer ke Rekening Sesama**		
Tujuan	Aktor dapat melakukan transfer uang ke sebuah rekening dengan bank yang sama.		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu Transfer Antar Rekening Sesama		
Pengujian			
Skenario Uji			
1. Memilih sumber rekening 2. Memilih rekening tujuan dan tambahan informasi lainnya 3. Menekan tombol Lanjut 4. Memasukkan secure challenge ke Android Token 5. Memasukkan secure response pada web internet banking 6. Menekan tombol Kirim			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
Sumber rekening, rekening tujuan,	Sistem berhasil transfer uang. pengirim ke	Sistem memindahkan uang dari rekening	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 38. Pengujian Fungsi Pembayaran Tiket Penerbangan

IDENTIFIKASI			
Nomor	**SPA-09**		
Nama	**Pembayaran Tiket Penerbangan**		
Tujuan	Aktor dapat melakukan pembayaran tiket penerbangan.		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu Pembayaran Tiket Penerbangan		
Pengujian			
Skenario Uji			
1. Memilih sumber rekening 2. Memilih maskapai tujuan 3. Memasukka informasi lainnya 4. Menekan tombol Lanjutkan 5. Memasukkan secure challenge ke Android Token 6. Memasukkan secure response ke web internet banking 7. Menekan tombol Proses			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
no.rekening, informasi lain dan challenge	Sistem dapat melakukan pembayaran	Pembayaran diterima oleh sistem	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 39. Pengujian Fungsi Penambahan Rekening Tujuan

IDENTIFIKASI			
Nomor	**SPA-10**		
Nama	**Penambahan Rekening Tujuan**		
Tujuan	Aktor dapat menambahkan rekening tujuan ke dalam sistem.		
Aktor	User		
Skenario			
Kondisi Awal	Aktor berada pada menu Tambah Rekening Tujuan		
Pengujian			
Skenario Uji			
1. Mengisi field informasi data rekening tujuan 2. Mengisi kode bank 3. Menekan tombol Lanjutkan 4. Memasukkan secure challenge pada Android Token 5. Memasukkan secure response pada web internert banking 6. Menekan tombol Proses			
Kasus dan Hasil Uji			
Masukan	**Harapan**	**Pengamatan**	**Kesimpulan**
Data rekening tujuan, kode bank dan secure challenge	Sistem dapat menambahkan rekening ke database	Sistem menambahkan rekening user ke database	**[V]** Berhasil **[ ]** Gagal

Tabel Lampiran 39. Pengujian Beta (1)

Tanggal	Nama Pemilik Rekening	Tipe	Jumlah	Rekening Tujuan	Saldo	Status
14-12-2015	Uung Ungkawa	Transfer	Rp30.000,-	IDR-902185550	2.420.000,-	Berhasil
13-12-2015	Uung Ungkawa	Transfer	Rp1.000.0000,-	IDR-902185550	Rp2.450.000,-	Berhasil
13-12-2015	Uung Ungkawa	Transfer	Rp50.000,-	IDR- 0241758694	Rp3.450.000,-	Berhasil
12-12-2015	Uung Ungkawa	Transfer	Rp1.500.000,-	IDR- 0241758694	Rp3.500.000,-	Berhasil

Tabel Lampiran 39. Pengujian Beta (2)

Tanggal	Nama Pemilik Rekening	Tipe	Jumlah	Rekening Tujuan	Saldo	Status
16-12-2015	Kurnia R Putra	Transfer	Rp1.000.000,-	IDR-024175884	Rp1.445.000,-	Berhasil
15-12-2015	Kurnia R Putra	Transfer	Rp500.000,-	IDR-024175884	Rp1.945.000,-	Berhasil
14-12-2015	Kurnia R Putra	Transfer	Rp30.000,-	IDR- 902185550	Rp1.975.000,-	Berhasil
14-12-2015	Kurnia R Putra	Transfer	Rp25.000,-	IDR- 902185550	Rp2.000.000,-	Gagal

```
import java.lang.reflect.UndeclaredThrowableException;
import java.security.GeneralSecurityException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.math.BigInteger;
import java.util.TimeZone;


/**
 * This is an example implementation of the OATH
 * TOTP algorithm.
 * Visit www.openauthentication.org for more information.
 *
 * @author Johan Rydell, PortWise, Inc.
 */

public class TOTP {

    private TOTP() {}

    /**
     * This method uses the JCE to provide the crypto algorithm.
     * HMAC computes a Hashed Message Authentication Code with the
     * crypto hash algorithm as a parameter.
     *
     * @param crypto: the crypto algorithm (HmacSHA1, HmacSHA256,
     *                                HmacSHA512)
     * @param keyBytes: the bytes to use for the HMAC key
     * @param text: the message or text to be authenticated
     */

    private static byte[] hmac_sha(String crypto, byte[] keyBytes,
            byte[] text){
        try {
            Mac hmac;
            hmac = Mac.getInstance(crypto);
            SecretKeySpec macKey =
                new SecretKeySpec(keyBytes, "RAW");
            hmac.init(macKey);
            return hmac.doFinal(text);
        } catch (GeneralSecurityException gse) {
            throw new UndeclaredThrowableException(gse);
        }
    }

    /**
     * This method converts a HEX string to Byte[]
     *
     * @param hex: the HEX string
```

```
      *
      * @return: a byte array
      */

     private static byte[] hexStr2Bytes(String hex){
         // Adding one byte to get the right conversion
         // Values starting with "0" can be converted
         byte[] bArray = new BigInteger("10" +
hex,16).toByteArray();

         // Copy all the REAL bytes, not the "first"
         byte[] ret = new byte[bArray.length - 1];
         for (int i = 0; i < ret.length; i++)
             ret[i] = bArray[i+1];
         return ret;
     }

     private static final int[] DIGITS_POWER
     // 0 1  2   3    4     5      6       7        8
     = {1,10,100,1000,10000,100000,1000000,10000000,100000000 };


     /**
      * This method generates a TOTP value for the given
      * set of parameters.
      *
      * @param key: the shared secret, HEX encoded
      * @param time: a value that reflects a time
      * @param returnDigits: number of digits to return
      *
      * @return: a numeric String in base 10 that includes
      *              {@link truncationDigits} digits
      */

     public static String generateTOTP(String key,
             String time,
             String returnDigits){
         return generateTOTP(key, time, returnDigits, "HmacSHA1");
     }


     /**
      * This method generates a TOTP value for the given
      * set of parameters.
      *
      * @param key: the shared secret, HEX encoded
      * @param time: a value that reflects a time
      * @param returnDigits: number of digits to return
      *
      * @return: a numeric String in base 10 that includes
      *              {@link truncationDigits} digits
      */
```

```
    public static String generateTOTP256(String key,
            String time,
            String returnDigits){
        return generateTOTP(key, time, returnDigits, "HmacSHA256");
    }

    /**
     * This method generates a TOTP value for the given
     * set of parameters.
     *
     * @param key: the shared secret, HEX encoded
     * @param time: a value that reflects a time
     * @param returnDigits: number of digits to return
     *
     * @return: a numeric String in base 10 that includes
     *              {@link truncationDigits} digits
     */

    public static String generateTOTP512(String key,
            String time,
            String returnDigits){
        return generateTOTP(key, time, returnDigits, "HmacSHA512");
    }


    /**
     * This method generates a TOTP value for the given
     * set of parameters.
     *
     * @param key: the shared secret, HEX encoded
     * @param time: a value that reflects a time
     * @param returnDigits: number of digits to return
     * @param crypto: the crypto function to use
     *
     * @return: a numeric String in base 10 that includes
     *              {@link truncationDigits} digits
     */

    public static String generateTOTP(String key,
            String time,
            String returnDigits,
            String crypto){
        int codeDigits = Integer.decode(returnDigits).intValue();
        String result = null;

        // Using the counter
        // First 8 bytes are for the movingFactor
        // Compliant with base RFC 4226 (HOTP)
        while (time.length() < 16 )
            time = "0" + time;
```

```
        // Get the HEX in a Byte[]
        byte[] msg = hexStr2Bytes(time);
        byte[] k = hexStr2Bytes(key);

        byte[] hash = hmac_sha(crypto, k, msg);

        // put selected bytes into result int
        int offset = hash[hash.length - 1] & 0xf;

        int binary =
            ((hash[offset] & 0x7f) << 24) |
            ((hash[offset + 1] & 0xff) << 16) |
            ((hash[offset + 2] & 0xff) << 8) |
            (hash[offset + 3] & 0xff);

        int otp = binary % DIGITS_POWER[codeDigits];

        result = Integer.toString(otp);
        while (result.length() < codeDigits) {
            result = "0" + result;
        }
        return result;
    }

    public static void main(String[] args) {
        // Seed for HMAC-SHA1 - 20 bytes
        String seed = "3132333435363738393031323334353637383930";
        // Seed for HMAC-SHA256 - 32 bytes
        String seed32 = "3132333435363738393031323334353637383930"
+
        "31323334353637383930313132";
        // Seed for HMAC-SHA512 - 64 bytes
        String seed64 = "3132333435363738393031323334353637383930"
+
        "3132333435363738393031323334353637383930" +
        "3132333435363738393031323334353637383930" +
        "31323334";
        long T0 = 0;
        long X = 30;
        long testTime[] = {59L, 1111111109L, 1111111111L,
                1234567890L, 2000000000L, 20000000000L};

        String steps = "0";
        DateFormat df = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss");
        df.setTimeZone(TimeZone.getTimeZone("UTC"));
```

```
         try {
             System.out.println(
                     "+--------------------+------------------------------+" +
             "------------------------+-----------+-----------+");
             System.out.println(
                     "|  Time(sec)     |   Time (UTC format)    " +
             "| Value of T(Hex)   |  TOTP  | Mode    |");
             System.out.println(
                     "+--------------------+------------------------------+" +
             "------------------------+-----------+-----------+");

             for (int i=0; i<testTime.length; i++) {
                 long T = (testTime[i] - T0)/X;
                 steps =  Long.toHexString(T).toUpperCase();
                 while (steps.length() < 16) steps = "0" + steps;
                 String fmtTime = String.format("%1$-11s",
testTime[i]);
                 String utcTime = df.format(new
Date(testTime[i]*1000));
                 System.out.print("|  " + fmtTime + "  |  " +
utcTime +
                         "  | " + steps + " |");
                 System.out.println(generateTOTP(seed, steps, "8",
                 "HmacSHA1") + "| SHA1   |");
                 System.out.print("|  " + fmtTime + "  |  " +
utcTime +
                         "  | " + steps + " |");
                 System.out.println(generateTOTP(seed32, steps, "8",
                 "HmacSHA256") + "| SHA256 |");
                 System.out.print("|  " + fmtTime + "  |  " +
utcTime +
                         "  | " + steps + " |");
                 System.out.println(generateTOTP(seed64, steps, "8",
                 "HmacSHA512") + "| SHA512 |");

                 System.out.println(
                         "+--------------------+------------------------------+"
+
                 "------------------------+-----------+-----------+");
             }
         }catch (final Exception e){
             System.out.println("Error : " + e);
         }
     }
 }
```