PERANCANGAN DAN IMPLEMENTASI APLIKASI SIMULATOR SISTEM MANAJEMEN PELAYANAN PUSAT KESEHATAN MASYARAKAT (PUSKESMAS) KECAMATAN GAMBIRAN MENGGUNAKAN METODE *DISCRETE - EVENT SIMULATION*

SKRIPSI

oleh :

Ahmad Baihaqi

112410101061

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2015

PERANCANGAN DAN IMPLEMENTASI APLIKASI SIMULATOR SISTEM MANAJEMEN PELAYANAN PUSAT KESEHATAN MASYARAKAT (PUSKESMAS) KECAMATAN GAMBIRAN MENGGUNAKAN METODE *DISCRETE - EVENT SIMULATION*

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan Pendidikan Sarjana (S1) Program Studi Sistem Informasi dan mencapai gelar Sarjana Komputer

oleh :

Ahmad Baihaqi

112410101061

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2015

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Kedua Orangtua saya, Ayahanda (Alm) Ahmad Basuni dan Ibunda Siti Mukamilah;
2. Saudara-saudaraku berserta seluruh keluarga besar;
3. Guru-guruku sejak sekolah dasar hingga perguruan tinggi;
4. Seluruh teman-teman yang selalu memberikan bantuan dan dukungan;
5. Almamater Program Studi Sistem Informasi Universitas Jember.

MOTO

"Berilmu Sebelum Berkata dan Beramal"

(Al-Bukhari)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Ahmad Baihaqi

NIM : 112410101061

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Perancangan dan Implementasi Aplikasi Simulator Sistem Manajemen Pelayanan Pusat Kesehatan Masyarakat (PUSKESMAS) Kecamatan Gambiran Menggunakan Metode *Discrete-Event Simulation*", adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, 27 Agustus 2015

Yang menyatakan,

Ahmad Baihaqi

PENGESAHAN PEMBIMBING

Skripsi berjudul "Perancangan dan Implementasi Aplikasi Simulator Sistem Manajemen Pelayanan Pusat Kesehatan Masyarakat (PUSKESMAS) Kecamatan Gambiran Menggunakan Metode *Discrete-Event Simulation*", telah diuji dan disahkan pada :

hari, tanggal : 27 Agustus 2015

tempat : Program Studi Sistem Informasi Universitas Jember.

Disetujui oleh :

Pembimbing I,	Pembimbing II,
Dr. Saiful Bukhori, ST., M.Kom	Nelly Oktavia A, S.Si., MT.
NIP 196811131994121001	NIP 198410242009122008

SKRIPSI

PERANCANGAN DAN IMPLEMENTASI APLIKASI SIMULATOR SISTEM MANAJEMEN PELAYANAN PUSAT KESEHATAN MASYARAKAT (PUSKESMAS) KECAMATAN GAMBIRAN MENGGUNAKAN METODE *DISCRETE - EVENT SIMULATION*

oleh :

Ahmad Baihaqi

112410101061

Pembimbing :

Dosen Pembimbing Utama : Dr. Saiful Bukhori, ST., M.Kom
Dosen Pembimbing Pendamping : Nelly Oktavia A, S.Si., MT.

PENGESAHAN

Skripsi berjudul "Perancangan dan Implementasi Aplikasi Simulator Sistem Manajemen Pelayanan Pusat Kesehatan Masyarakat (PUSKESMAS) Kecamatan Gambiran Menggunakan Metode *Discrete-Event Simulation*", telah diuji dan disahkan pada :

hari, tanggal : 27 Agustus 2015

tempat : Program Studi Sistem Informasi Universitas Jember.

Penguji I,	Penguji II,
Anang Andrianto ST., MT	Windi Eka Yulia R, S.Kom., MT
NIP 196906151997021002	NIP 198403052010122002

Mengesahkan
Ketua Program Studi,

Prof. Drs. Slamin, M.Comp.Sc.,Ph.D
NIP 196704201992011001

RINGKASAN

Antrian terjadi apabila kebutuhan akan layanan melebihi kemampuan kapasitas pelayanan atau waktu proses lebih besar dari waktu pelayanan. Peningkatan jumlah pasien yang signifikan dan tidak menentu membuat pihak Puskesmas harus bisa meningkatkan pelayanan terhadap pasien, sehingga pendistribusian pasien dalam hal antrian pelayanan tetap efisien dan optimal. Pada penelitian ini dibangun sebuah simulasi sistem pelayanan puskesmas dengan tujuan untuk mengetahui nilai keoptimalan pelayanan pada Puskesmas yang dalam hal ini adalah Puskesmas Gambiran. Ukuran performa dari sistem ini adalah nilai utilitas loket dan dokter, rata-rata mengantri pasien, rata-rata lama layanan pasien dan total lama menganggur loket dan dokter. Berdasarkan ukuran performa dari simulasi yang dilakukan inilah dapat digunakan untuk suatu usulan terhadap sistem yang telah ada sehingga diperoleh sistem yang lebih baik.

Metode yang digunakan dalam membuat simulasi pelayanan puskesmas adalah metode *discrete-event simulation* karena metode ini cocok dalam membahas model suatu sistem yang selalu berkembang karena adanya representasi perubahan variabel-variabel pada kondisi tertentu disaat tertentu. Berdasarkan penelitian yang dilakukan diperoleh hasil kesimpulan bahwa penambahan atau pengurangan loket dan dokter diberlakukan apabila nilai utilitas dari loket atau dokter dianggap masih belum memenuhi nilai optimal. Penambahan loket atau dokter dilakukan jika nilai utilitas yang dirasakan terlalu besar, sedangkan pengurangan loket atau dokter dilakukan jika nilai utilitas lebih kecil dari nilai optimal yaitu diatas 70%.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "Perancangan dan Implementasi Aplikasi Simulator Sistem Manajemen Pelayanan Pusat Kesehatan Masyarakat (PUSKESMAS) Kecamatan Gambiran Menggunakan Metode *Discrete-Event Simulation*". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Prof. Drs. Slamin, M.Comp.Sc., Ph.D., selaku Ketua Program Studi Sistem Informasi Universitas Jember;
2. Dr. Saiful Bukhori, ST., M.Kom selaku Dosen Pembimbing Utama dan Nelly Oktavia A, S.Si., MT selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi;
3. Ibu Windi Eka Yulia R, S.Kom.,MT selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa;
4. Seluruh Bapak dan Ibu dosen beserta staf karyawan di Program Studi Sistem Informasi Universitas Jember;
5. Teman-teman seperjuangan dan seangkatan.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, oleh sebab itu penulis mengharapkan adanya masukan yang bersifat membangun dari semua pihak. Penulis berharap skripsi ini dapat bermanfaat bagi semua pihak.

Jember, 27 Agustus 2015

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1. PENDAHULUAN

Bab ini merupakan langkah awal dari penulisan tugas akhir ini. Bab ini berisi latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah, metodologi penelitian, dan sistematika penulisan.

1.1. Latar Belakang

Di Indonesia Puskesmas merupakan tulang punggung pelayanan kesehatan masyarakat tingkat pertama (Effendi, 2009). Puskesmas memiliki peranan yang cukup penting dalam dunia kesehatan yang ada di Indonesia. Hal itu dikarenakan kemudahan akses untuk mendapatkan pelayanan kesehatan di puskesmas bagi seluruh lapisan kalangan masyarakat.

Dalam mengelola pelayanan kepada pasien, puskesmas haruslah memberikan pelayanan secara optimal dan efisien. Pelayanan yang optimal akan memberikan kepuasan dan kenyamanan pada pasien serta setidaknya dapat membantu pasien agar cepat sembuh dari penyakitnya. Akan tetapi ada beberapa permasalahan yang sering muncul dalam proses manajemen pelayanan yang ada di puskesmas.

Salah satu permasalahan yang umum dan sering terjadi di puskesmas yaitu masalah antrian yang berkepanjangan. Antrian pada puskesmas sering kita temui pada fasilitas - fasilitas yang ada di puskesmas seperti antrian pada fasilitas pendaftaran, antrian pada fasilitas dokter umum, antrian pada fasilitas dokter anak hingga antrian pada fasilitas apotik. Antrian tersebut timbul disebabkan oleh kebutuhan akan layanan melebihi kemampuan kapasitas pelayanan atau fasilitas layanan, sehingga pengguna fasilitas yang tiba tidak bisa segera mendapat layanan disebabkan kesibukan layanan (Ekoanindiyo, 2011). Kasus antrian pelayanan di puskesmas sebenarnya terjadi dalam segmen waktu tertentu dengan kurangnya manajemen sumber daya yang tersedia di puskesmas.

Dengan adanya permasalahan antrian yang panjang di puskesmas, salah satu dampaknya yaitu pasien yang menunggu untuk dilayani tidak mendapat tempat duduk saat mengantri pada jam-jam tertentu. Akibatnya pasien yang sakit merasa tidak nyaman. Pada umumnya, puskesmas mengatasi masalah tersebut dengan menambah jumlah sumber daya yang ada di puskesmas baik yang berupa jumlah operator pendaftaran, dokter maupun menambah jumlah kursi tunggu di antrian. Tetapi hal itu bukanlah solusi yang efektif karena antrian hanya terjadi pada waktu-waktu tertentu. Apabila jumlah operator dan dokter terlalu sedikit maka akan menimbulkan jumlah antrian yang panjang dan lama. Sedangkan apabila jumlah jumlah operator dan dokter terlalu banyak, maka akan menyebabkan banyaknya waktu operator dan dokter menganggur. Karena tidak adanya sistem simulasi manajemen pelayanan,penentuan jumlah petugas loket pendaftaran (operator) dan jumlah dokter masih bersifat intuitif (Warmadewa, 2009).

Penggunaan sistem simulasi memiliki banyak sekali manfaat. Hal itu dikarenakan sistem simulasi digunakan untuk merepresentasikan suatu model dari sistem nyata. Metode yang digunakan untuk membuat suatu sistem simulasi terdiri dari bermacam - macam metode. Salah satunya yaitu metode *discrete-event simulation.* Metode ini banyak digunakan dalam kasus model simulasi yang memiliki variabel yang selalu berkembang dan berubah-ubah dalam kondisi tertentu.

Berdasarkan permasalahan diatas, maka di perlukan penelitian untuk menangani dan memberikan solusi terbaik untuk mengatasi permasalahan antrian di puskesmas. Penulis mengadakan penelitian tentang perancangan dan implementasi simulasi sistem manajemen pelayanan puskesamas kecamatan Gambiran menggunakan metode *discrete-event simulation.* Dimana nanti dalam penelitian ini diharapkan dapat memberikan solusi terhadap permasalahan antrian dalam proses pelayanan di puskesmas.

1.2. Perumusan Masalah

Berdasarkan latar belakang diatas maka dirumuskan permasalahan sebagai berikut :

1. Bagaimana menerapkan metode *Discrete-Event Simulation* pada aplikasi simulasi sistem pelayanan PUSKESMAS di Kecamatan Gambiran.
2. Bagaimana merancang dan membangun aplikasi untuk simulasi sistem pelayanan PUSKESMAS di Kecamatan Gambiran.

1.3. Tujuan dan Manfaat

Tujuan dan manfaat berisi tentang tujuan dari penelitian penerapan metode *Discrete-Event Simulation* untuk simulasi sistem pelayanan PUSKESMAS. Sedangkan pada bagian manfaat berisi tentang manfaat apa yang akan diperoleh pada penelitian ini, baik bagi peneliti sendiri maupun bagi objek pada penelitian ini.

1.3.1. Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Merancang dan membangun aplikasi untuk simulasi sistem manajemen pelayanan PUSKESMAS.
2. Merancang dan membangun aplikasi untuk simulasi sistem manajemen pelayanan PUSKESMAS di Kecamatan Gambiran dengan menerapkan metode *Discreate-Event Simulation.*

1.3.2. Manfaat

Manfaat diperoleh dari adanya ini adalah aplikasi simulator sistem pelayanan Puskesmas adalah sebagai berikut:

a. Manfaat Akademis

Hasil penelitian ini diharapkan dapat memberikan kontribusi dan masukan bagi siapa saja yang membutuhkan informasi yang berhubungan dengan judul penelitian ini. Selain itu, hasil penelitian ini merupakan suatu upaya untuk menambah varian judul penelitian yang ada di Program Studi Sistem Informasi Universitas Jember.

b. Manfaat bagi peneliti

1. Mengetahui bagaimana proses penerapan metode *Discrete-Event Simulation* untuk simulasi sistem pelayanan PUSKESMAS.
2. Membantu instansi untuk melakukan simulasi sistem pelayanan PUSKESMAS.

c. Manfaat bagi objek penelitian

1. Memberikan inovasi baru kepada instansi tempat penelitian dilakukan mengenai penggunaan aplikasi simulasi sistem pelayanan PUSKESMAS.
2. Membantu instansi untuk melakukan simulasi sistem pelayanan PUSKESMAS sehingga dapat memberikan alternatif solusi bagi permasalahan pelayanan PUSKESMAS.

1.4. Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Program simulasi ini digunakan untuk mensimulasikan sistem manajemen pelayanan PUSKESMAS.
2. Program simulasi ini hanya mengacu pada data kedatangan, lama pendaftaran dan lama pemeriksaan pasien.
3. Aplikasi simulasi sistem manajemen pelayanan PUSKESMAS di buat menggunakan metode *Discrete-Event Simulation.*
4. Aplikasi dibangun berbasis *desktop* dengan menggunakan bahasa pemrogaman *java programming*.

1.5. Sistematika Penulisan

Sistematika penulisan dalam penyusunan tugas akhir ini adalah sebagai berikut:

a. Pendahuluan

Bab ini terdiri atas latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah dan sistematika penulisan.

b. Tinjauan Pustaka

Bab ini berisi tentang kajian materi, penelitian terdahulu dan informasi apa saja yang digunakan dalam penelitian ini. Dimulai dari kajian pustaka mengenai pengertian dari sistem hingga metode *Discrete-Event Simulation.*

c. Metodologi Penelitian

Bab ini menguraikan tentang metode apa yang dilakukan selama penelitian. Dimulai dari tahap pencarian permasalahan hingga pengujian aplikasi simulasi sistem manajemen pelayanan PUSKESMAS yang akan dibuat.

d. Hasil dan Pembahasan

Bab ini menjelaskan tentang hasil dan pembahasan dari penelitian yang telah dilakukan dengan memaparkan hasil penelitian dan hasil percobaan pengimplementasian sistem.

e. Penutup

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Pada bagian ini akan dijelaskan teori-teori dan pustaka yang digunakan dalam penelitian. Teori yang dibahas meliputi teori tentang sistem, aplikasi perangkat lunak, java, pemodelan, simulasi, PUSKESMAS, antrian *(Queueing)*, uji *kolmogorov-smirnov*, bilangan *random uniform*, model *waterfall* dan metode *discrete-event simulation*.

2.1 Pengertian Sistem

Sistem merupakan komponen–komponen yang bekerja sama untuk mencapai tujuan tertentu. Sistem adalah suatu kesatuan usaha yang terdiri dari bagian-bagian yang berkaitan satu sama lain yang berusaha mencapai suatu lingkungan kompleks (Marimin, 2009). Jadi dengan demikian dapat disimpulkan bahwa sistem adalah media yang didukung oleh komponen-komponen yang memiliki keterkaitan satu sama lain & dibatasi oleh suatu aturan tertentu untuk mencapai aturan tertentu.

2.2 Pengertian Aplikasi

Menurut Hendrayudi (2009), aplikasi adalah kumpulan perintah program yang dibuat untuk melakukan pekerjaan – pekerjaan tertentu (khusus).Sedangkan menurut Jogiyanto (2004), aplikasi merupakan program yang berisikan perintah-perintah untuk melakukan pengolahan data. Jadi aplikasi secara umum adalah suatu proses dari cara manual yang ditransformasikan ke komputer dengan membuat sistem atau program agar data diolah lebih berdaya guna secara optimal.

Jadi dapat disimpulkan bahwa aplikasi adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya aplikasi merupakan suatu perangkat komputer yang siap pakai bagi user.

2.3 Java

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat program sebagaimana membuatnya dengan bahasa seperti Pascal atau C++.

Java juga mendukung sumber daya internet dan juga Java mendukung aplikasi klien/server, baik dalam jaringan lokal maupun jaringan berskala luas (Kadir, 2004).

Sebutan Java 2 diberikan untuk Java versi 1.2 dan versi berikutnya. Java 2 terbagi dalam 3 kategori , yaitu:

1. Java 2 Standart Edition (J2SE) merupakan edisi standar (basis) dari Java2. J2SE lebih difokuskan pada pemrograman Desktop dan Applet(aplikasi yang dapat dijalankan di browser web).
2. Java 2 Enterprise Edition (J2EE) merupakan edisi perluasan dari J2SE (Superset dari J2SE), aplikasi yang dibuat dengan edisi ini untuk aplikasi berskala besar (Enterprise), seperti pemrograman memakai database dan diatur di server.
3. Java 2 Mobile Edition (J2ME) merupakan edisi khusus dari Java dan subset dari edisi J2SE. Edisi ini untuk pemrograman dengan peralatanperalatan kecil atau terbatas, seperti PDA, handphone, pager, dan lain lain (Kadir, 2004).

2.4 Pemodelan

Model merupakan representasi sistem dalam kehidupan nyata. Pemodelan suatu sistem merupakan suatu proses penyaringan dan penyeleksian yang dilakukan sedemikian rupa terhadap berbagai data sehingga didapatkan beberapa data atau komponen sistem yang dapat dimodelkan, dan yang dianggap kurang penting atau tidak relevan dapatlah diasumsikan mampu mendukung tujuan yang ingin dicapai (Djati, 2007). Jadi sebuah model tidak hanya merupakan perwujudan tujuan, namun juga merupakan asumsi untuk mendukung tujuan tersebut.

Dalam kehidupan, model yang digunakan untuk mengenal suatu sistem (studi terhadap sistem) dibedakan berdasarkan data yang diperoleh dan hal tersebut dapat diklasifikasikan menjadi seperti berikut:

1. Model Fisik

Didasarkan pada analogi dari sistem dengan sistem. Pemodelan yang seperti ini atribut atau *field* (data) dari sistem didapatkan dari pengukuran, seperti jarak yang

ditempuh oleh truk dengan beban tertentu dan kecepatan tertentu yang mempangaruhi kemampuan mesin, dengan beban bervariasi dan kecepatan tertentuseberapa jauh pesawat dapat meninggalkan landasan, dan masih banyak lagi contoh lain.

2. Model Matematika

Pada model ini simbol-simbol matematika dan persamaan-persamaan matematika digunakan untuk menggambarkan sistem. Atribut atau field dari sistem dipresentasikan oleh aktivitas-aktivitas setiap variabel yang dideklarasikan (diidentifikasi lebih awal) dan kemudian dengan fungsi-fungsi matematika maka dari seluruh variabel tersebut akan dihasilkan aktivitas-aktivitas yang diharapkan.

2.5 Simulasi

Menurut Khosnevis dalam buku pemodelan dan simulasi(Suryani, 2006), simulasi merupakan proses aplikasi membangun model dari sistem nyata atau usulan sistem, melakukan eksperimen dengan model tersebut untuk menjelaskan perilaku sistem, mempelajari kinerja sistem, atau untuk membangun sistem baru sesuai dengan kinerja yang diinginkan. Dari penjelasan diatas sehingga dengan demikian dapat diartikan Simulasi merupakan proses perancangan model dari suatu sistem nyata dan pelaksanaan eksperimen-eksperimen dengan model ini untuk tujuan memahami tingkah laku sistem atau untuk menyusun strategi (dalam suatu batas atau limit yang ditentukan oleh satu atau beberapa kriteria) sehubungan dengan sistem operasi tersebut.

2.6 Puskesmas

Pusat kesehatan masyarakat (PUSKESMAS) adalah suatu kesatuan organisasi kesehatan fungsional yang merupakan pusat pengembangan kesehatan masyarakat yang juga membina peran serta masyarakat disamping memberikan pelayanan secara menyeluruh dan terpadu kepada masyarakat di wilayah kerjanya dalam bentuk kegiatan pokok (Efendi & Makhfudli 2009). PUSKESMAS juga dapat didefinisikan sebagai unit pelaksana teknis dinas kesehatan kabupaten/kota yang bertanggung jawab menyelenggarakan pembangunan kesehatan di suatu wilayah kerja (Depkes RI,

2004). Puskesmas mempunyai wewenang dan tanggung jawab atas pemeliharaan kesehatan masyarakat dalam wilayah kerjanya.

2.7 Antrian *(Queuing)*

Antrian *(Queuing)* timbul disebabkan oleh kebutuhan akan layanan melebihi kemampuan (kapasitas) pelayanan atau fasilitas layanan, sehingga pengguna fasilitas yang tiba tidak bisa segera mendapat layanan disebabkan kesibukan layanan (Ekoanindiyo, 2011). Menurut Bronson (Farkhan, 2013), proses antrian (*queueing process*) adalah suatu proses yang berhubungan dengan kedatangan seorang pelanggan pada suatu fasilitas pelayanan, kemudian menunggu dalam suatu baris (antrian) jika semua pelayannya sibuk, dan akhirnya meninggalkan fasilitas tersebut. Jadi dapat disimpulkan bahwa antrian terjadi karena sumber daya yang tersedia kurang dan terjadi karena lonjakan jumlah pengguna/pelanggan secara signifikan pada periode tertentu.

2.8 *Discrete-Event Simulation*

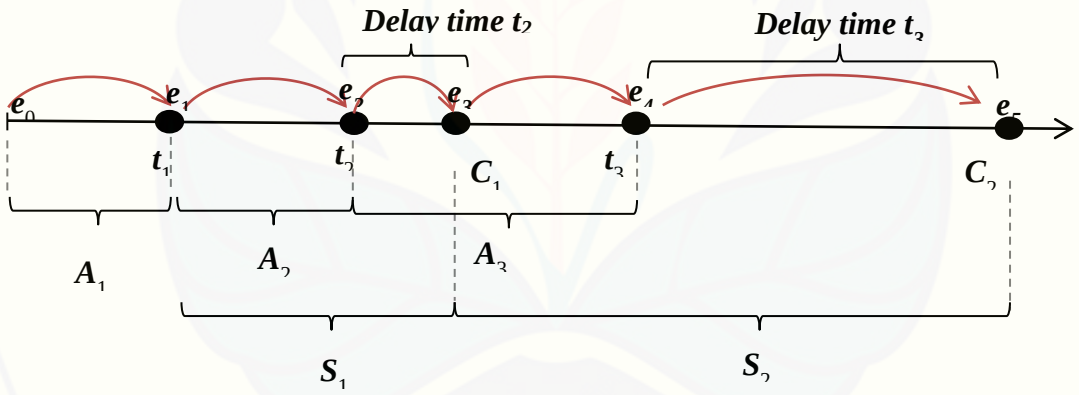
Discrete-Event Simulation (simulasi kejadian diskrit) adalah simulasi yang membahas model suatu sistem yang selalu berkembang karena adanya representasi perubahan variabel-variabel pada kondisi tertentu di saat tertentu. Kondisi tertentu ini merupakan kejadian di mana suatu peristiwa terjadi dan *event* (kejadian) didefinisikan sebagai kejadian atau peristiwa yang pada saat yang sama dapat mengubah kondisi sistem (Djati, 2007). Secara konsep perhitungan pada metode *discreate-event* dapat menggunakan perhitungan secara manual, tetapi karena begitu banyak data yang akan diproses di mana dibutuhkan media penyimpanan atas proses-proses tersebut maka penyelesaian permasalahan menggunakan *discreate-event simulation* disarankan untuk menggunakan media komputer.

Berdasarkan mekanisme pemajuan waktu simulasi, maka *discreate-event simulation* dibedakan menjadi 2, yaitu *Next-event Time Advance* dan *Fixed-Increment Time Advance*

1. *Next-Event Time Advance*

Dengan menggunakan pendekatan *next-event time advance*, waktu simulasi diinisialisasikan dengan 0 (nol) dan waktu kejadian pada *event* selanjutnya dideklarasikan, yaitu pada saat *server* melayani sejumlah permintaan dan pesanan. Dalam pendekatan ini, batas-batas waktu pelayanan kejadianya selalu berubah. Peristiwa tersebut termasuk dalam model *discreate-event simulation* karena dalam suatu periode aktivitasnya sistem selalu diputar dengan melompati waktu dari satu waktu kejadian tiap periode kewaktu kejadian yang lain.

Perilaku *next-event time advance* mirip dengan perilaku *single-server queue system* dan *first-in first-out system.* Pendekatan menggunakan *next-event time advance* tampak pada Gambar 2.1



Gambar 2.1 Ilustrasi model *next-event time advance* (Djati, 2007)

Dari Gambar 2.1 diatas perlu didefinisikan simbol-simbol yang dapat menggambarkan tiap-tiap kejadian akan disimulasikan.

t_1 = waktu kedatangan pelanggan yang ke-*i* ($t_0 = 0$)

A_i = t_{i-1} = waktu antar kedatngan yang (*i* - 1) dan ke-*i* adalah kedatangan pelanggan.

S_i = waktu yang dibutuhkan server dalam melayani pelanggan yang ke-*i* (pelanggan yang harus dilayani lebih dahulu dalam antrian).

D_i = waktu tunggu yang dibutuhkan pelanggan yang ke-*i*

C_i = $t_i + D_i + S_i$ = waktu yang dibutuhkan untuk melayani pelanggan yang ke-*i* sampai meninggalkan tempat pelayanan.

e_i = waktu kedatangan kejadian yang ke-*i* yang bervariasi (i adalah nilai dari waktu simulasi, $e_0 = 0$)

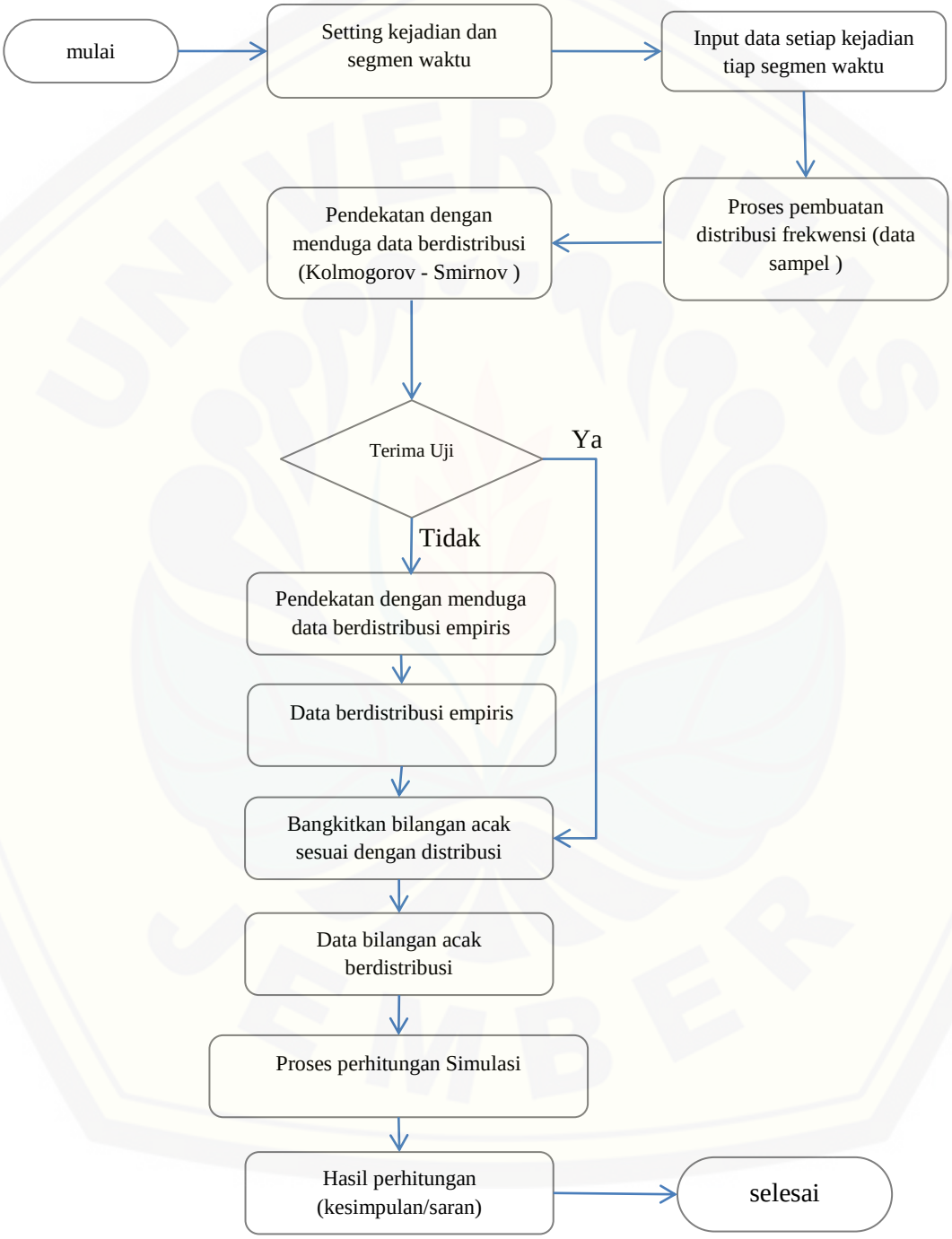
2. *Fixed-Increment Time Advance*

Dalam pendekatan *fixed-increment time advance* , waktu simulasi ditambah sebanyak Δt dengan tepat tanpa diubah untuk dijadikan acuan penambahan waktu untuk simulasi berikutnya. Pendekatan menggunakan *fixed-increment time advance* dapat dilihat pada Gambar 2.2.

Gambar 2.2 Ilustrasi model fixed-increment time advance (Djati, 2007)

Dari Gambar 2.2 dijelaskan bahwa tanda lengkungan menggambarkan percepatan waktu simulasi dan e_i (i =1,2,3.....) adalah waktu yang sebenarnya dari kejadian *i*th dari suatu kondisi (bukan nilai *i*th pada waktu simulasi). Pada jangka waktu (0, Δt), suatu kejadian akan muncul pada waktu e_i tetapi dianggap kejadian tersebut muncul pada waktu Δt menurut model. Ditunjukkan juga didalam gambar bahwa diantara waktu (Δt, **2**Δt) tidak terdapat kejadian yang muncul, tetapi proses ini tetap harus dijalankan untuk mengetahui dan menentukan ada atau tidaknya kejadian pada waktu tersebut. Pada waktu simulasi berikutnya antara (**2**Δt, **3**Δt) terdapat kejadian kejadian yang muncul yang terdapat pada waktu e_2 dan *e3* tetapi perubahan tersebut dianggap muncul pada waktu **3**Δt dan seterusnya.

Untuk menggambarkan urutan pengerjaan simulasi dengan menggunakan metode *discreate-event simulation* lebih lengkapnya dapat di lihat pada Gambar 2.3 dibawah ini.



Gambar 2.3 *Workflow* Penyelesaian Metode *Discrete-Event* (Djati, 2007)

2.9 Distribusi Frekuensi

Distribusi frekuensi merupakan suatu teknik pengelompokan data kedalam beberapa kelompok atau kelas dan kemudian dihitung banyaknya data yang masuk kedalam tiap kelas (Dzati, 2007). Distribusi frekuensi menunjukan jumlah atau banyaknya *item* dalam setiap kategori atau kelas. Didalam distribusi frekuensi ini data kualitatif maupun kuantitatif isajikan dalam bentuk ringkas dan jelas.

Tahapan untuk menentukan kelas dalam distribusi frekuensi meliputi untuk menentukan jumlah kelas dan perkiraan besarnya kelas menggunakan persamaan *kriterium sturges* yaitu antara lain :

1. Mencari kelas

   $K = 1 + 3{,}33 \log n$ **.....(2.1)**

   Dimana :

   K = banyaknya kelas

   n = banyaknya nilai pengamatan

2. Mencari perkiraan besarnya kelas (*range)*

   $$h = \frac{\max - min}{k}$$ **.....(2.2)**

   Dimana :

   h = perkiraan besarnya kelas

   k = banyaknya kelas

   max = nilai data observasi terbesar

   min = nilai data observasi terkecil

2.10 Uji Keselarasan *Kolmogorov-Smirnov*

Uji *kolmogorov-smirnov* digunakan untuk mengetahui apakah distribusi nilai-nilai sampel yang teramati sesuai dengan distribusi teoritis tertentu (normal,eksponensial). Prinsip dari uji *kolmogorov-smirnov* adalah menghitung selisih absolut antara fungsi distribusi frekuensi kumulatif sampel [S(x)] dan fungsi distribusi kumulatif teoritis [F(x)] pada masing-masing interval kelas (Djati, 2007).

Untuk menguji distribusi normal digunakan uji keselarasan *kolmogorov-smirnov* normal dan untuk menguji distribusi *eksponensial* digunakan uji keselarasan *kolmogorov-smirnov ekponensial.*

Langkah - langkah pengujian keselarasan *kolmogorov-smirnov* normal antara lain :

1. Cari nilai Fi (frekuensi kumulatif) dari masing-masing kelas.

   $$\boldsymbol{Fi(n) = Fi(n) + Fi(n-1)} \quad \textbf{(2.3)}$$

2. Cari nilai Xi (nilai tengah) untuk masing-masing kelas.

   $$\boldsymbol{Xi = \frac{Batas\ bawah + Batas\ atas}{2}} \quad \textbf{(2.4)}$$

3. Cari nilai rata - rata (Xbar) yang dapat didekati.

   $$\boldsymbol{Xbar = \frac{total(Fi * Xi)}{n}} \quad \textbf{(2.5)}$$

4. Selanjutnya cari nilai simpangan baku dari data sampel.

   $$\boldsymbol{Simpangan\ Baku = \frac{total(Fi * [Xi - Xbar]^2)}{n}} \quad \textbf{(2.6)}$$

5. Cari nilai S(x) dari masing-masing kelas.

   $$\boldsymbol{Sx = \frac{Fi}{n}} \quad \textbf{(2.7)}$$

6. Terakhir cari nilai Z untuk masing-masing kelas. Dari nilai Z kemudian dicari tabel distribusi normal dengan menggunakan Z tabel normal standar yang menghasilkan nilai F(x). Kemudian nilai F(x) dikurangi dengan nilai S(x) dan selanjutnya dicari nilai terbesarnya.

$$\boldsymbol{Z = \frac{Xi+Xbar}{simpangan\ baku}} \quad \textbf{(2.8)}$$

7. Menetapkan α (taraf signifikansi), α = 0,05.
8. Menentukan daerah penolakan, $W_{1-\alpha}$

$$\boldsymbol{W_{1-\alpha} = \frac{1.36}{\sqrt{n}}} \quad \textbf{(2.9)}$$

9. Jika nilai T_{hitung} (max [F(x)-S(x)]) > $W_{1-\alpha}$ maka hasil ditolak (data tidak berdistribusi normal) dan nilai T_{hitung} (max [F(x)-S(x)]) < $W_{1-\alpha}$ maka hasil diterima (data berdistribusi normal).

Sedangkan langkah - langkah pengujian keselarasan *kolmogorov-smirnov eksponensial* antara lain :

1. Cari nilai Fi (frekuensi kumulatif) dari masing-masing kelas.
2. Cari nilai Xi (nilai tengah) untuk masing-masing kelas.
3. Cari nilai frekuensi dikalikan dengan nilai tengah masing-masing kelas.
4. Cari nilai S(x) dari masing-masing kelas.
5. Terakhir cari nilai F(x) untuk masing-masing kelas. Nilai F(x) kemudian dikurangi dengan nilai S(x), dan selanjutnya dicari nilai yang terbesar.
6. Menetapkan α (taraf signifikansi), α = 0,05.
7. Menentukan daerah penolakan, $W_{1-\alpha}$
8. Menentukan nilai F(x) eksponensial

$$\boldsymbol{F(x) = 1 - 2.71^{\frac{-xi}{xbar}}} \quad \textbf{(2.10)}$$

9. Jika nilai T_{hitung} (max [F(x)-S(x)]) > $W_{1-\alpha}$ maka hasil ditolak (data tidak berdistribusi ekponensial) dan nilai T_{hitung} (max [F(x)-S(x)]) < $W_{1-\alpha}$ maka hasil diterima (data berdistribusi ekponensial).

Jika pengujian keselarasan *kolmogorov-smirnov* normal dan *eksponensial* hasilnya ditolak, maka langkah terakhir yaitu dengan melakukan uji distribusi *empiris.* Langkah-langkah pengujian *empiris* antara lain :

1. Bagilah nilai frekuensi masing-masing kelas dengan jumlah data (n)

2. Terakhir cari nilai presentase kumulatif (Yj)..

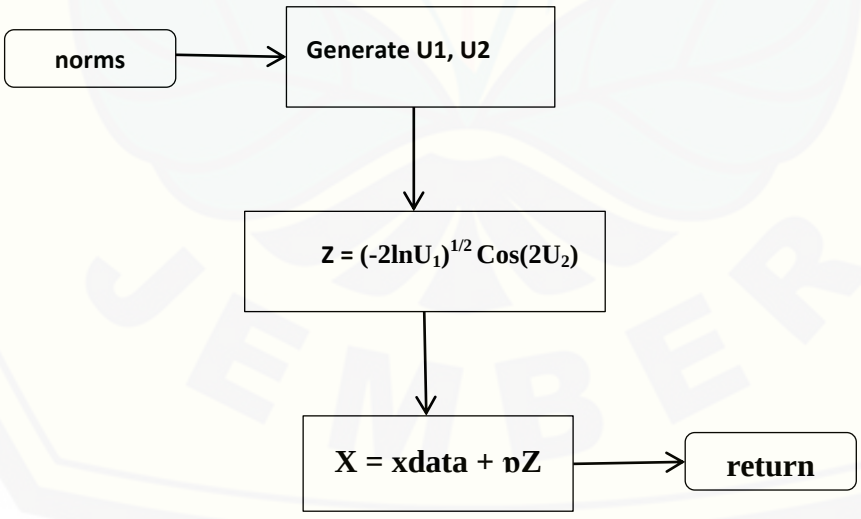
2.11 Bilangan *Random Uniform*

Bilangan *random uniform* atau biasa disimbolkan dengan U(0,1) merupakan suatu bilangan acak yang sekali dan rutin dalam pembangkitanya (Wahyudi, 2012). Bilangan *random* ini merupakan bilangan acak yang nilainya seragam. Kriteria yang harus dipenuhi yaitu :

1. Bilangan acak harus memiliki distribusi sama (*uniform*). Beberapa bilangan acak yang diambil haruslah memiliki *probabilitas* atau peluang terambil sama besar.
2. Masing masing bilangan acak tidak saling tergantung atau independen.

Metode pembangkitan bilangan acak dengan data berdistribusi normal langkah algoritmanya antara lain :

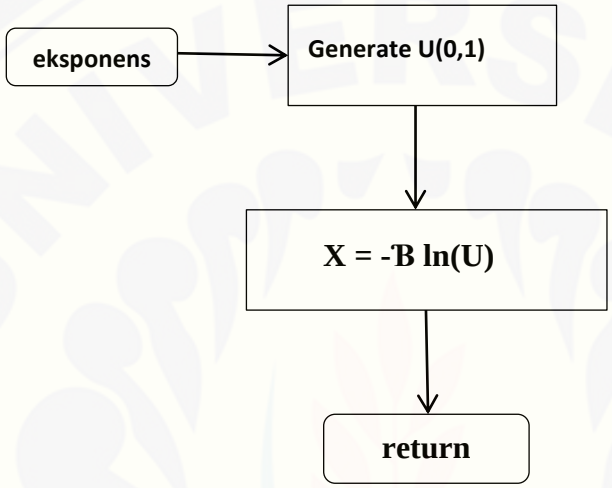
1. Data berdistribusi normal
2. *Generate* U_1 dan U_2
3. Cari nilai Z dengan cara mengalikan $(-2\ln U_1)^{1/2}$ dengan $\text{Cos}(2U_2)$
4. Cari nili X data dengan cara mengalikan simpangan hasil distribusi data normal dengan nili Z lalu tambahkan dengan nilai data sampel yang akan di *random*



Gambar 2.4 *Workflow* perhitungan bilangan random berdistribusi normal (Dzati,2007)

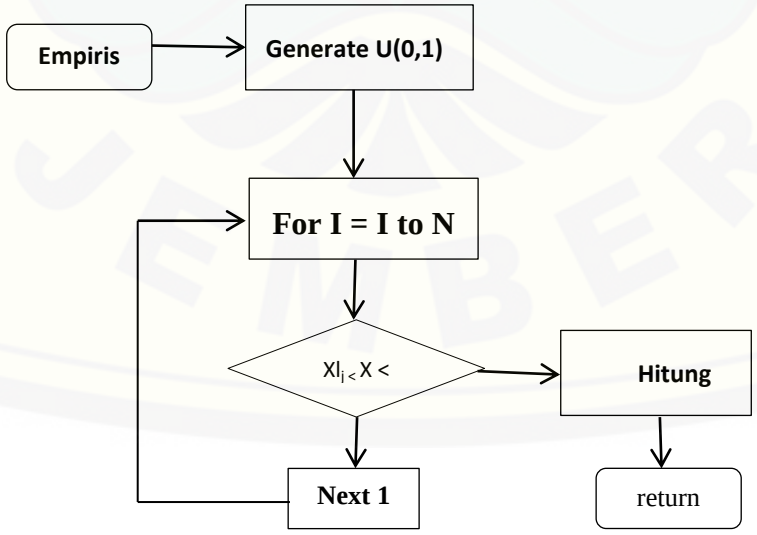
Metode pembangkitan bilangan acak dengan data berdistribusi *eksponensial* langkah algoritmanya antara lain :

1. Data berdistribusi *eksponensial*
2. *Generate* U(0,1)
3. Dapatkan nilai X = -B ln(U)



Gambar 2.5 *Workflow* perhitungan bilangan random berdistribusi *eksponensial* (Dzati,2007)

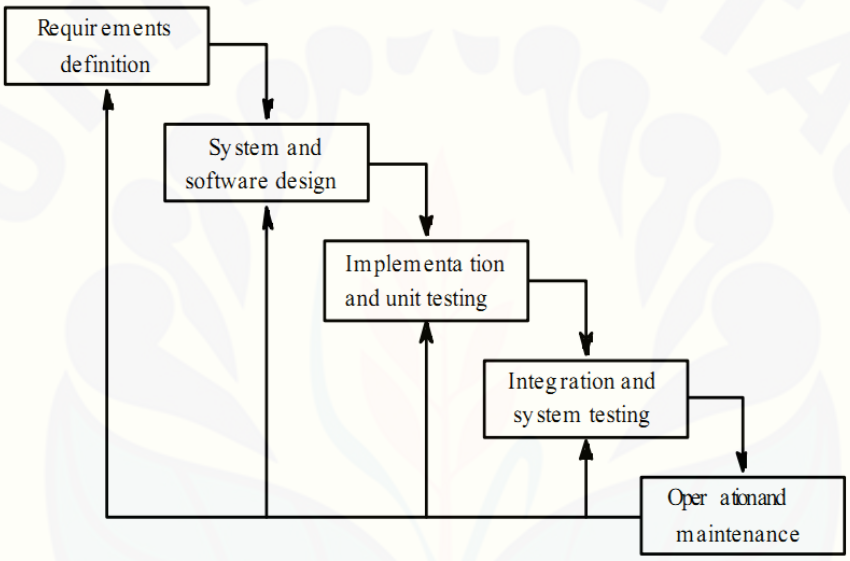
Metode pembangkitan bilangan acak dengan data berdistribusi *empiris* alurnya dapat dilihat pada Gambar 2.6



Gambar 2.6 *Workflow* perhitungan bilangan random berdistribusi empiris (Dzati,2007)

2.12 Model *Waterfall*

Menurut Pressman (2012), model *waterfall* merupakan sebuah pendekatan pengembangan perangkat lunak yang sistematik dan sekunsial yang dimulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain, kode, pengujian, dan pemeliharaan. Alur *life cycle* pengembangan perangkat lunak pada model waterfall diselesaikan pertahapan dan berurutan. Alur dari model *waterfall* dapat dilihat padaGambar 2.7 Ilustrasi Model *Waterfall*



Gambar 2.7 Ilustrasi Model *Waterfall* (Pressman, 2012)

Keterangan dari skema di atas adalah :

1. Analisis Kebutuhan

Menganalisis kebutuhan yang akan digunakan dalam pembuatan aplikasi. Meliputi pengumpulan data kebutuhan fungsional dan non-fungsional dari aplikasi yang akan kita bangun. Setelah itu, menentukan fungsi dan fasilitas apa saja yang akan dibuat dalam aplikasi.

2. Desain Sistem

Jika proses analisis kebutuhan telah diketahui maka proses selanjutnya adalah pada tahapan desain sistem. Proses pendesainan sistem dari aplikasi yang akan kita

bangun yaitu dengan menggunakan *Unified Modeling Language* (UML). Penggunaan UML karena sudah menggunakan konsep *Object Oriented Design* yang tentunya akan sangat memudahkan developer untuk membangun sebuah sistem. UML diagram yang akan dibuat antara lain:

a. Business Process

Business Proses digunakan untuk menggambarkan inputan data yang dibutuhkan sistem, output dari sistem serta tujuan dari pembuatan sistem.

b. Use Case Diagram

Use case adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor.

c. Scenario

Scenario diagram digunakan untuk menjelaskan atau menceritakan fitur atau isi yang ada di *use case* diagram. *Scenario* menjelaskan alur sistem dan keadaan yang akan terjadi ketika terjadi suatu event tertentu.

d. Squence Diagram

Sequence diagram (diagram urutan) adalah suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut termasuk pengguna, *display*, dan sebagainya berupa pesan/message.

e. Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram mempunyai fungsi yang sama dengan scenario namun diimplementasikan dalam diagram alir.

f. Class Diagram

Class diagram digunakan untuk menggambarkan struktur statis class dalam sistem. Class Diagram dibuat untuk memudahkan dalam proses pengkodean.*Entity Relationship Diagram*

g. ERD

merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.

3. *Coding* (Pengkodean)

Setelah proses desain system dikerjakan, proses selanjutnya adalah *coding* atau penulisan kode program. Bahasa pemrograman yang dipakai adalah *Java* dengan *tool* / IDE yang digunakan berupa NetBean. Proses penkodean menggunakan konsep OOP (*Object Oriented Programming*) dengan tujuan programmer akan lebih mudah dalam melakukan *coding,* karena konsep OOP mengikuti model yang telah ada dalam kehidupan nyata. Semua bagian (*entity*) dari suatu permasalahan adalah objek adalah konsep dari OOP. Objek-objek ini kemudian juga dapat berupa gabungan dari beberapa objek yang lebih kecil. Objek besar dapat dibentuk dengan menggabungkan beberapa objek-objek dalam bahasa pemrograman. Objek-objek tersebut berkomunikasi dengan saling interaksi kepada objek lain.

4. Pengujian / Testing

Pengujian wajib dilakukan untuk menguji apakah sistem ini sudah sesuai dengan kebutuhan dari user atau belum. Dan apakah masih ada kesalahan maupun kelemahan terhadap sistem yang kami bangun tersebut. Diharapkan proses pengujian / testing dapat menyempurnakan sistem yang kami buat. Pengujian yang dilakukan melibatkan semua aspek sistem meliputi *hardware*, *software* aplikasi, *environment software*, penempatan aplikasi, dan *user* yang menggunakan aplikasi ini. Pengujian perangkat lunak menggunakan dua metode yakni :

1. *Black Box Testing*

Black Box Testing adalah metode pengujian perangkat lunak yang memeriksa fungsionalitas dari aplikasi yang berkaitan dengan struktur internal atau kerja. Pengetahuan khusus dari kode aplikasi atau struktur internal dan pengetahuan

pemrograman pada umumnya tidak diperlukan. Metode ini memfokuskan pada keperluan fungsionalitas dari *software* (Wildan Agissa, 2013).

Pada pengujian *black box* ini, aplikasi yang dibangun pada penelitian ini akan diuji dengan mengujikan langsung *running aplikasi* dan melakukan kegiatan pengujian dengan menganalisis proses input dan output yang dihasilkan aplikasi.

Adapun tabel pengujian disusun seperti Tabel 2.1 berikut :

Tabel 2.1 Tabel pengujian *blackbox*

Kelas Uji	**Butir Uji**	**Jenis Pengujian**

Keterangan Tabel :

a. **Kelas Uji** **:** Merupakan fungsi aplikasi yang akan diujikan.

b. **Butir Uji** **:** Rincian fitur yang diuji dari fungsi yang terdapat pada aplikasi.

c. **Jenis Pengujian** **:** Metode pengujian yang dilakukan , yaitu *black box.*

Dalam metode *black box* juga dilakukan pengujian dengan cara memasukkan data normal dan data salah , dari penginputan ini nantinya akan dilakukan analisis terhadap reaksi yang muncul pada aplikasi. Contoh tabel pengujian untuk *event* yang terjadi ketika ada data masukan dapat dilihat pada Tabel 2.2 dibawah ini.

Tabel 2.2 Tabel pengujian data normal dan salah

Kasus dan Hasil Uji (Data Normal)			
Data Masukan	Yang Diharapkan	Pengamatan	Kesimpulan

Username:admin Password : 12345 Klik tombol login	Aplikasi menampilkan semua menu yang dapat diakses oleh user , yaitu menu input segmen waktu, data poli, data operator, uji distribusi, simulasi dan report	Dapat masuk ke tampilan utama dan menampilkan menu sesuai yg diharapkan	[x]diterima [ ]ditolak

Kasus dan Hasil Uji (Data Salah)

Data Masukan	Yang Diharapkan	Pengamatan	Kesimpulan
Username:admin Password : 1234 Klik tombol login	Tidak dapat login dan masuk ke tampilan utama serta menampilkan peringatan bahwa username dan password salah	Tidak dapat login dan masuk ke tampilan utama serta menampilkan peringatan bahwa username dan password salah	[x]diterima [ ]ditolak

2. *White Box Testing*

Merupakan cara pengujian dengan melihat modul untuk yang telah dibuat dengan progam – progam yang ada. Dan menganalisa apakah terjadi kesalahan atau tidak pada penulisan kode progam. Pengujian ini dilakukan oleh *(develeoper)* pembuat progam. Jika ada modul yang menghasilkan output yang tidak sesuai, maka baris-baris progam, variabel dan parameter yang terlibat pada unit tersebut satu persatu akan di cek dan diperbaiki, kemudian di compile ulang (Agissa 2013). Menurut Presman (2002) pengujian *white box* menrupakan teknik pengujian jalur dasar yang digunakan untuk menentukan kompleksitas logis dengan menentukan rangkaian dasar jalur eksekusinya. Tahapan teknik pengujian jalur dasar meliputi dari

mulai listing program, grafik alir, kompleksitas siklomatik, jalur program independen dan pengujian basis set.

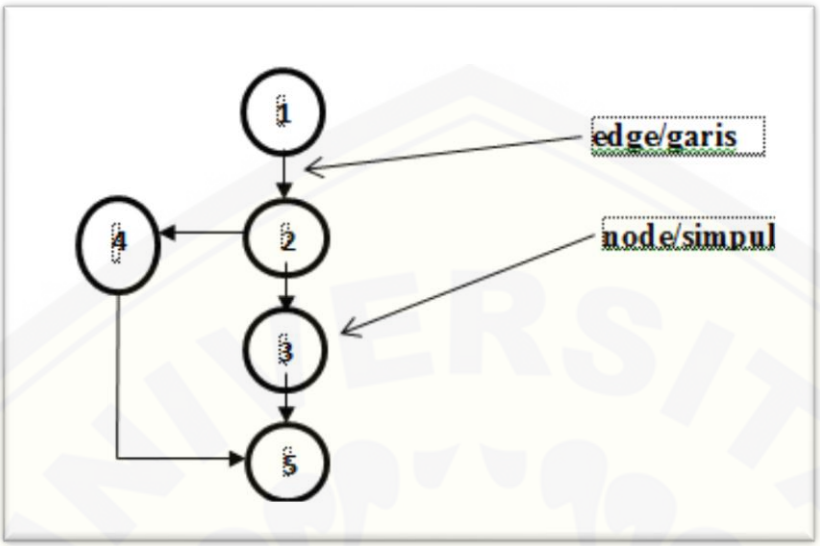
a. Listing Program

Merupakan baris-baris kode yang nantinya akan diuji. Setiap langkah dari kode-kode yang ada diberi nomor baik menjalankan *statement* biasa atau penggunaan kondisi dalam program. Contoh penerapan tahapan ini dapat dilihat pada Gambar 2.8

Gambar 2.8 Contoh *Listing Program* (Pressman, 2012)

b. Grafik Alir

Menurut Pressman (2012) Grafik alir merupakan Sebuah notasi sederhana yang digunakan untuk merepresentasikan aliran kontrol. Aliran kontrol yang digambarkan merupakan hasil penomoran dari listing program. Grafik alir digambarkan dengan *node-node* (simpul) yang dihubungkan dengan *edge-edge* (garis) yang menggambarkan alur jalannya program. Contoh penggambaran diagram alir dapat dilihat pada gambar 3.4.



Gambar 2.9 Contoh Diagram Alir (Pressman, 2012)

c. Kompleksitas Siklomatik

Kompleksitas Siklomatik merupakan metrik perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logis suatu program (Pressman, 2012). Bila digunakan dalam konteks teknik pengujian jalur dasar, nilai yang dihitung untuk kompleksitas siklomatik mendefinisikan jumlah jumlah jalur independen dalam basis ser suatu program (Pressman, 2012). Rumus yang digunakan untuk menghitung kompleksitas siklomatika yaitu:

$$V(G) = E - N + 2$$

Keterangan :

V(G): Kompleksitas Siklomatik

E : Jumlah Edge

N : Jumlah Node

Berdasarkan grafik alir yang ada pada tahapan kedua diketahui jumlah edge adalah 5 dan jumlah node adalah 5, sehingga dapat dihitung kompleksitas siklomatisk *V(G) = E – N + 2 = 5 – 5 + 2 = 2*. Jadi jumlah jalur independen adalah 2 jalur.

d. Jalur Program Independen

Jalur independen adalah setiap jalur yang melalui program yang memperkenalkan setidaknya satu kumpulan pernyataan-pernyataan pemrosesan atau kondisi baru (Pressman, 2012). Bila dinyatakan dalam grafik alir, jalur independen harus bergerak setidaknya sepanjang satu *edge* yang belum dilintasi sebelum jalur tersebut didefinisi (Pressman, 2012). Dari perhitungan kompleksitas siklomatik *Basis set* yang dihasilkan dari jalur independent secara linier adalah 2 jalur, yaitu:

Jalur 1 : 1-2-3-5

Jalur 2 : 1-2-4-5

e. Pengujian Basis Set

Pada bagian ini diberikan contoh data yang akan memaksa pelaksanaan jalur di *basis set*. Data yang dieksekusi dimasukkan ke dalam grafik alir apakah sudah melewati *basis set* yang tersedia. Sistem telah memenuhi syarat kelayakan software jika salah satu jalur yang dieksekusi setidaknya satu kali. Dari tahap sebelumnya telah diketahui 2 *basis set* Jika kemudian diuji dengan memasukkan data panjang = 5 dan lebar 3, maka basis set jalur yang digunakan adalah 1-2-4-5. Dapat dilihat bahwa jalur telah dieksekusi satu kali. Berdasarkan ketentuan tersebut dari segi kelayakan software, sistem ini telah memenuhi syarat.

5. Maintenance

Perawatan diadakan untuk mengatasi masalah pada sistem dilain waktu ketika aplikasi sudah dapat digunakan oleh *user*. Selama *user* menemui *bug* pada aplikasi

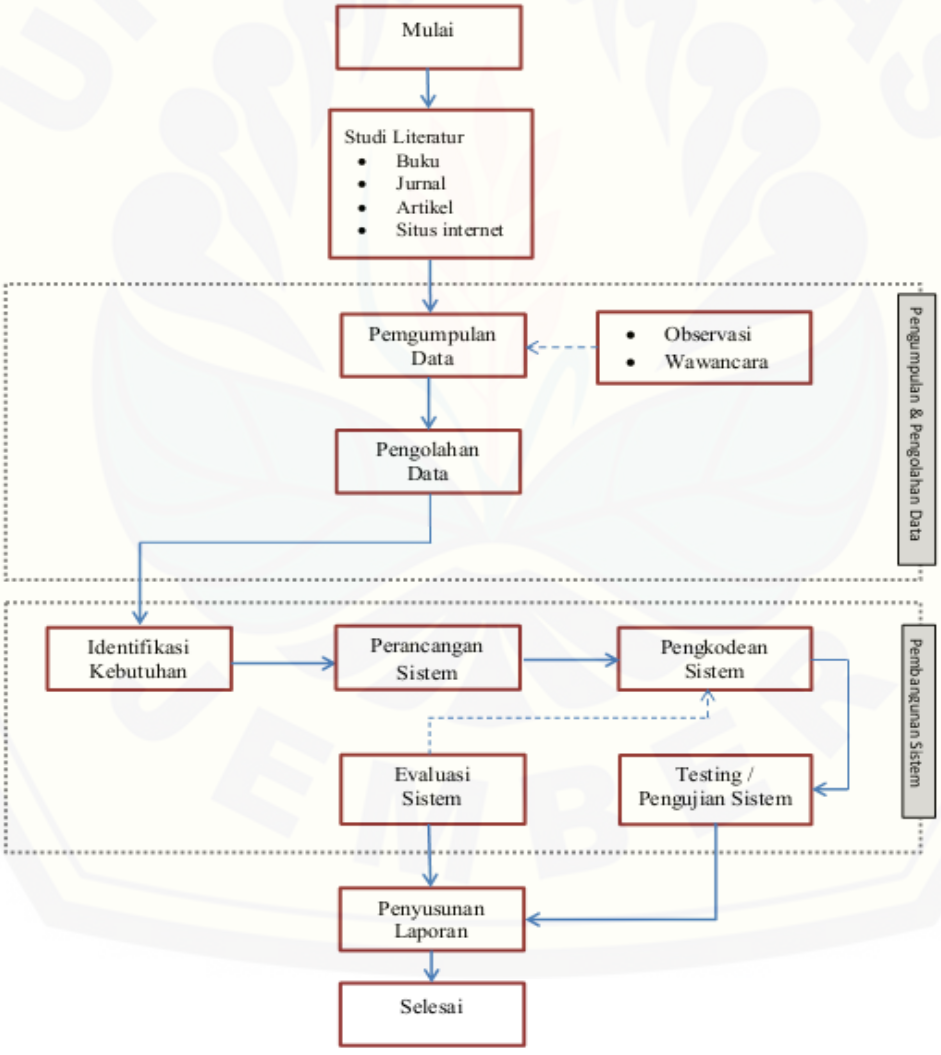
ini, maka *user* langsung dapat mengkonfirmasi kepada *developer* untuk segera ditangani oleh *developer*.

BAB 3. METODOLOGI PENELITIAN

Pada bab ini dijelaskan tentang metode-metode yang digunakan selama penelitian dilakukan, seperti jenis penelitian, studi literatur, data dan sumber data penelitian, serta tahapan analisis hingga model perancangan sistem.

3.1 Tahapan Penelitian

Penelitian akan dilaksanakan dalam beberapa tahap, tahapan alur penelitian untuk membuat aplikasi simulasi pelayanan puskesmas dapat dilihat pada Gambar 3.1 dibawah ini..



Gambar 3.1 Diagram Alir Penelitian (Sumber:Hasil Analisis, 2014)

3.2 Jenis Penelitian

Pada penelitian ini digunakan dua jenis penelitian, yaitu penelitian kualitatif dan penelitian kuntitatif. Jenis penelitian kualitatif digunakan karena penelitian ini menganalisa studi kasus yang berada pada Pusat Kesehatan Masyarakat (PUSKESMAS) Kecamatan Gambiran dan jenis penelitian kuantitatif digunakan karena dalam penelitian ini menerapkan serta mengkaji teori yang sudah ada sebelumnya.

3.3 Teknik Pengumpulan Data

Proses untuk mendapatkan data yang dibutuhkan untuk membangun aplikasi simulasi pelayanan puskesmas menggunakan dua cara, yaitu Studi Literatur dan wawancara.

3.3.1 Studi Literatur

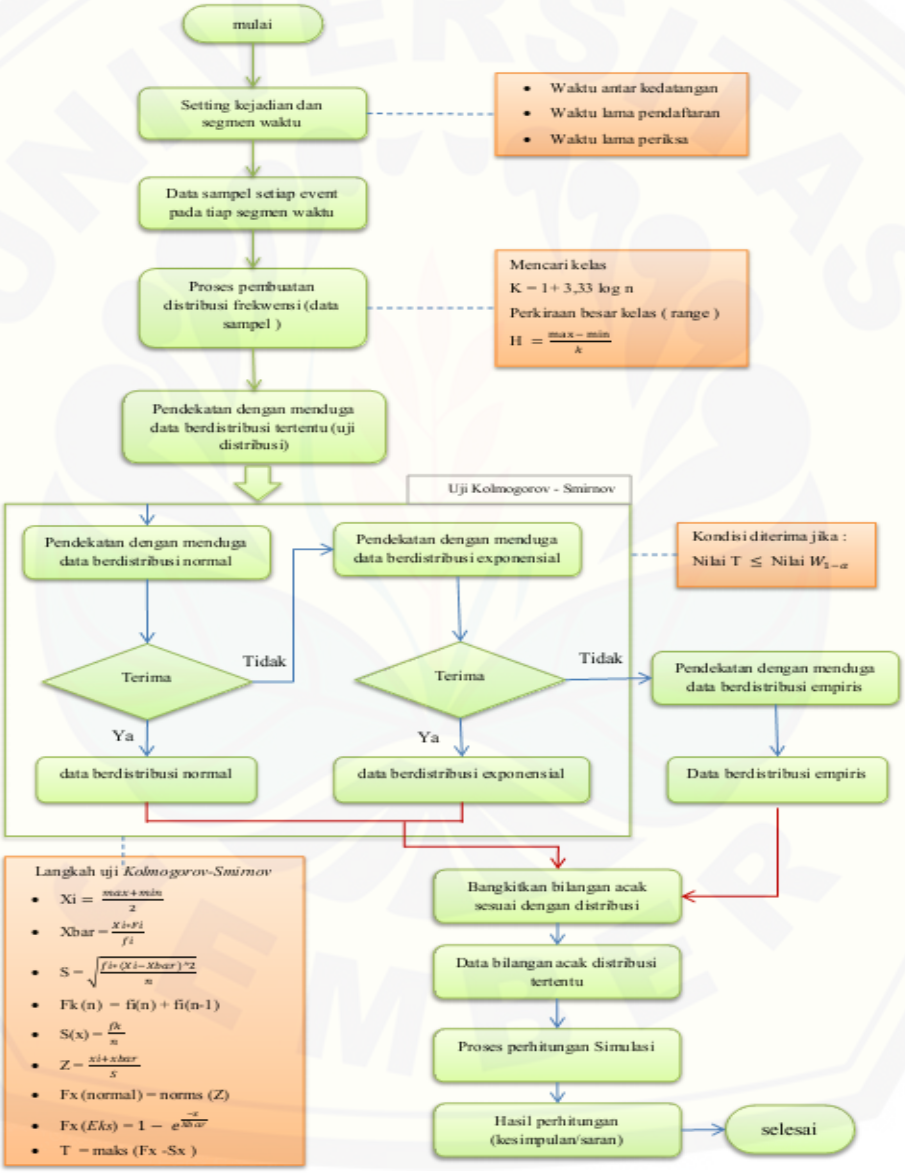
Pada tahap ini dilakukan pengumpulan data dan informasi yang diperlukan untuk proses perancangan sistem. Data dan informasi dapat diperoleh dari lokasi penelitian yaitu di Puskesmas Gambiran. Selain itu, studi literatur juga dapat diperoleh dari *paper*, jurnal ilmiah, serta buku-buku referensi yang berkaitan dengan penelitian.

3.3.2 Wawancara

Teknik penyerapan pengetahuan terdiri atas dua bagian utama, yaitu identifikasi proyek dan penyerapan pengetahuan. Pelaksanaan penyerapan pengetahuan bisanya dilakukan dengan wawancara (*interview*). Metode wawancara yang digunakan adalah diskusi bebas (*talk through*), pembicaraan atas dasar kasus yang menarik (*critical incident technique*) dan reklasifikasi dari tujuan yang akan diraih. Penulis dalam penelitian ini melakukan wawancara secara langsung pada pihak managerial Puskesmas Gambiran.

3.4 Tahap Analisis

Tahap analisis dilakukan setelah melakukan pengumpulan data mengenai managerial pelayanan Puskesmas. Data yang diperoleh akan dianalisa dengan metode simulasi *Discret-Event Simulation.* Proses pembuatan program simulasi pelayanan puskesmas dengan menggunakan metode *Discret-Event Simulation* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram alir sistem (sumber:hasil analisis, 2014)

3.5 Tahap Pengembangan Sistem

Didalam pembuatan aplikasi simulasi pelayanan puskesmas ini mengikuti tahapan *software development life cycle* (SDLC) *waterfall*. Penggunaan SDLC *waterfall* bertujuan untuk memudahkan alur pembuatan *software*. Tahapan SDLC dengan metode waterfall meliputi tahapan analisis, desain, implementasi, pengujian, dan pemeliharaan.

Setelah tahap pengumpulan data selesai, selanjutnya data akan dianilisis menggunakan metode *Discret-Event Simulation*. Kemudian akan dilanjutkan ke perancangan sistem dengan menggunakan konsep berbasis objek dengan pemodelan *Unified Modelling Language* (UML). Pemodelan UML yang digunakan pada penelitian ini antara lain, *Business Process*, *Usecase Diagram*, *Scenario*, *Sequence Diagram*, *Activity Diagram*, *Class diagram* dan *Entity Relationship Diagram* (ERD). Setelah tahap perancangan selesai, dilanjutkan dengan tahap implementasi menggunakan bahasa pemrograman *Java programming*. Hasil perancangan dan implementasi kemudian akan ditesting menggunakan *White Box* dan *Black Box*.

BAB 4. ANALISIS DAN PENGEMBANGAN SISTEM

Bab ini akan membahas tentang analisis dan pengembangan untuk membuat aplikasi simulator sistem manajemen pelayanan Pusat Kesehatan Masyarakat (PUSKESMAS) menggunakan metode *discrete-event simulation.* Tahapan-tahapan perancangan dilaksanakan berdasarkan metode waterfall, dimulai dari analisis kebutuhan fungsional dan non-fungsional sistem, dilanjutkan dengan pembuatan usecase diagram, skenario, activity diagram, sequence diagram, class diagram dan *entity relation diagram* (ERD).

4.1 Pengumpulan Data

Pengumpulan data dilakukan agar sistem yang dibangun sesuai dengan kebutuhan user. Pengumpulan data dilakukan dengan teknik wawancara langsung kepada pihak puskesmas dan analisa dokumen data jumlah pasien yang pernah menggunakan fasilitas atau memanfaatkan pelayanan yang ada pada puskesmas Gambiran.

4.2 Penerapan Metode *Discrete-Event Simulation*

Metode yang digunakan untuk membuat program simulasi pelayanan puskesmas ini adalah metode *Discrete-Event Simulation.* Proses pertama dalam metode ini yaitu menentukan *event* (kejadian) apa saja yang akan digunakan dalam proses simulasi pelayanan puskesmas. *Event* yang digunakan dalam hal ini antara lain : *event* waktu antar kedatangan, waktu lama pendaftaran dan waktu lama pemeriksaan pasien. Langkah selanjutnya yaitu menetapkan segmen waktu yang akan digunakan dalam simulasi. Sebelum mentukan segmen waktu yang akan digunakan, langkah pertama yang harus dilakukan adalah menentukan jumlah lama waktu per-segmen waktu yang ada. Antar segmen waktu haruslah memiliki lama waktu yang sama.

Untuk lebih jelasnya contoh tabel daftar *event* dapat dilihat pada Tabel 4.1 dan segmen waktu dapat dilihat pada Tabel 4.2.

Tabel 4.1 Tabel daftar *event* (Kejadian)

No Event	Nama Event
1	Antar Kedatangan
2	Lama Pendaftaran
3	Lama Pemeriksaan

Tabel 4.2 Tabel segmen waktu

No Segmen	Nama Segmen	Waktu Awal	Waktu Akhir	Interval/-detik
1	Segmen 1	07.00	10.00	3600
2	Segmen 2	10.00	13.00	3600
3	Segmen 3	13.00	16.00	3600

Setelah daftar *event* dan segmen waktu yang digunakan telah ditentukan, langah selanjutnya adalah mengelompokkan data tiap *event* kejadian dan segmen waktu dengan menggunakan distribusi frekuensi. Data dibawah ini pada Tabel 4.3 adalah contoh data *event* antar kedatangan pasien pada segmen 2.

Tabel 4.3 Tabel data antar kedatangan pasien dalam detik

No	Data /-detik	No	Data/-detik
1	120	11	60
2	197	12	1687
3	302	13	779
4	533	14	164
5	658	15	732
6	362	16	359
7	240	17	61
8	1080	18	663
9	1718	19	119
10	140	20	536
		n = 20	

Dari data waktu antar kedatangan akan dilakukan pengelompokkan data kedalam beberapa kelompok atau kelas dan kemudian dihitung banyaknya data yang masuk kedalam tiap kelas atau disebut distribusi frekuensi dengan cara mencari kelas menggunakan persamaan 2.1 dan mencari besarnya kelas menggunakan persamaan 2.2. Setelah diketahui jumlah kelas dan range maka selanjutnya dihitung distribusi frekuensi datanya. Distribusi frekuensi data antar kedatangan pada segmen 2 dapat dilihat pada Tabel 4.4.

Tabel 4.4 Tabel distribusi frekuensi data antar kedatangan pada segmen 2

No	Batas Bawah	Batas Atas	Frekuensi
1	60.0	392.0	11
2	393.0	725.0	4
3	726.0	1058.0	2
4	1059.0	1391.0	1
5	1392.0	1724.0	2

Setelah diketahui hasil dari nilai distribusi frekuensi data maka langkah selanjutnya yaitu menguji sifat dari data distribusi. Pengujian dilakukan dengan menggunakan pendugaan distribusi data secara normal, eksponensial dan jika semua ditolak maka akan menggunakan pendugaan data berdistribusi empiris. Untuk mengetahui distribusi data diterima atau ditolak maka yang menjadi acuan nilainya adalah nilai uji keselarasan *kolmogorov-smirnov*. Jika nilai yang dihitung (distribusi data) hasilnya lebih besar dari nilai *kolmogorov-smirnov* hasilnya maka ditolak dan jika lebih kecil maka hasilnya diterima.

Dalam kasus distribusi data antar kedatangan seperti pada Tabel 4.4 diatas penulis menduga data berdistribusi *eksponensial.* Hasil dari perhitungan uji distribusi eksponensial dapat dilihat pada Tabel 4.5 Tabel Distribusi *eksponensial* data antar kedatangan pada segmen .

Tabel 4.5 Tabel Distribusi *eksponensial* data antar kedatangan pada segmen 2

Batas Bawah	Batas Atas	frek	Xi	Fi*Xi	Frek. Kumulatif	S(x)	F(x)	F(x)-S(x)
60	392	11	226	2486	11	0.55	0.34	-0.210
393	725	4	559	2236	15	0.75	0.642	-0.108
726	1058	2	892	1784	17	0.85	0.806	-0.044
1059	1391	1	1225	1225	18	0.9	0.895	-0.005
1392	1724	2	1558	1558	20	1	0.943	-0.057

Untuk nilai *kolmogorov-smirnov* dari data antar kedatanganya dengan n = 20 yaitu sebesar 0.304 yang didapat dengan perhitungan dari persamaan 2.7. untuk nilai F(x) menggunakan perhitungan dari persamaan 2.8. Berdasarkan Tabel 4.5 diatas didapatkan nilai terbesar dari F(x) - S(x) bernilai -0.005. Jadi karena nilai *kolmogorov* lebih besar dari nilai F(x) - S(x) terbesar maka hasilnya uji diterima dan data berdistribusi *eksponensial.*

Setelah selesai melakukan uji distribusi data pada *event* antar kedatangan langkah selanjutnya yaitu membangkitkan bilangan *random* sesuai distribusi data. Karena pada proses uji distribusi frekuensi sebelumnya pada data antar kedatangan di Poli Umum pada segmen 2 menghasilkan data berdistribusi *eksponensial* maka otomatis pembangkitan bilangan *random*nya juga menggunakan pembangkitan bilangan *random eksponensial.* Gunakan algoritma seperti Gambar 2.5 untuk membangkitan bilangan *random* secara *eksponensial.* Hasil pembangkitan bilangan

random dapat dilihat pada Tabel 4.6. Hasil dari bilangan *random*(X) itulah yang nantinya akan digunakan dalam proses simulasi. untuk membangkitkan bilangan random *ekponensial* dapat dengan menggunakan algoritma seperti pada Gambar 2.5.

Tabel 4.6 Tabel hasil pembangkitan bil acak

No	Data	U(Acak)	X
1	120	0.66	49
2	197	0.75	55
3	302	0.60	153
4	533	0.74	158
5	658	0.11	1478
6	362	0.26	482
7	240	0.32	272
8	1080	0.44	886
9	1718	0.65	978
10	140	0.36	141
11	60	0.43	50
12	1687	0.62	799
13	779	0.52	505
14	164	0.81	33
15	732	0.08	1842
16	359	0.37	354
17	61	0.61	30
18	663	0.10	1560
19	119	0.05	368
20	536	0.94	31

Langkah selanjutnya setelah selesai melakukan uji ditribusi data pada semua variabel *event* yang ada (antar kedatangan, lama pendaftaran dan lama pemeriksaan) dan telah membangkitkan *random seed* sesuai uji distribusi yaitu proses simulasi data. Pada proses simulasi, data dari *event - event* yang berupa antar kedatangan, lama pendaftaran dan lama pemeriksaan akan diolah atau disimulasikan sesuai dengan

kondisi yang diinginkan. Hasil dari proses simulasi berupa informasi informasi yang antara lain, nilai utilitas kerja loket dan dokter, rata - rata lama antrian pasien, rata - rata lama pelayanan pasien dan total lama menganggur loket dan dokter.Untuk kasus simulasi, penulis mencoba untuk mensimulasikan pelayanan pasien pada segmen 2. Hasil proses simulasi dapat dilihat pada Tabel 4.7

Tabel 4.7 Simulasi dengan 1-Loket dan 2-Dokter aktif

No	Wtd	Wtm	WtLd	WtLp	TL	Wml	Wtpd	Wpmd	Wpsd	Wmd	Wtpp	Wpmp	Wsl	nL	nD
116	49	49	1143	4437	5580	49	0	49	1192	1192	0	1192	5629	1	1
117	55	104	223	1673	1896	0	1088	1192	1415	1415	0	1415	3088	1	2
118	153	257	1332	787	2119	0	1158	1415	2747	0	341	3088	3875	1	2
119	158	415	62	1093	1155	0	2332	2747	2809	0	1066	3875	4968	1	2
120	1478	1893	543	2856	3399	0	916	2809	3352	0	1616	4968	7824	1	2
121	482	2375	1498	1736	3234	0	977	3352	4850	0	779	5629	7365	1	1
122	272	2647	94	4906	5000	0	2203	4850	4944	0	2421	7365	12271	1	1
123	886	3533	651	41	692	0	1411	4944	5595	0	2229	7824	7865	1	2
124	978	4511	27	1311	1338	0	1084	5595	5622	0	2243	7865	9176	1	2
125	141	4652	489	2301	2790	0	970	5622	6111	0	3065	9176	11477	1	2
126	50	4702	236	1460	1696	0	1409	6111	6347	0	5130	11477	12937	1	2
127	799	5501	208	668	876	0	846	6347	6555	0	5716	12271	12939	1	1
128	505	6006	143	1935	2078	0	549	6555	6698	0	6239	12937	14872	1	2
129	33	6039	593	392	985	0	659	6698	7291	0	5648	12939	13331	1	1
130	1842	7881	148	1460	1608	590	0	7881	8029	0	5302	13331	14791	1	1
131	354	8235	1075	104	1179	206	0	8235	9310	0	5481	14791	14895	1	1
132	30	8265	357	517	874	0	1045	9310	9667	0	5205	14872	15389	1	2
133	1560	9825	34	176	210	158	0	9825	9859	0	5036	14895	15071	1	1
134	368	10193	10	913	923	334	0	10193	10203	0	4868	15071	15984	1	1
135	31	10224	189	1753	1942	21	0	10224	10413	0	4976	15389	17142	1	2

Keterangan :

Wtd : waktu antar kedatangan

Wtm : jam masuk pasien

WtLd : waktu lama daftar

WtLp : waktu lama periksa

TL : total layanan

Wml : waktu menganggur loket

Wtpd : waktu tunggu pasien daftar

Wpmd : waktu pasien mulai daftar

Wpsd : waktu pasien selesai daftar

Wmd : waktu tunggu pasien periksa

Wtpp : waktu menganggur dokter

Wpmp : waktu pasien mulai periksa

Wsl : waktu selesai layanan

nL : no loket yang melayani

nD : no dokter yang melayani

Alur proses Perhitungan simulasi pada Tabel 4.7 adalah sebagai berikut :

1. Total pelayanan puskesmas mulai dibuka pada pukul 07:00 -16.00. Segmen waktu pelayanan dibagi menjadi 3 segmen. Jadi per-segmen total kerjanya adalah 3 jam atau sebesar 10800 (dalam detik).
2. Pada segmen 2 pasien pertama datang pada detik ke 49. Maka dapat dihitung total menganggur dari loket sebesar 49 detik . Begitu seterusnya.
3. Waktu menganggur loket didapat dari pengurangan waktu masuk pasien ke (n) dikurangi waktu selesai daftar pasien ke (n - 1). Sedangkan waktu masuk pasien sendiri didapat dari waktu antar kedatangan pasien ke (n) ditambah waktu masuk pasien ke (n - 1).
4. Pasien pertama pada saat daftar karena dia datang disaat loket sedang tidak melakukan pelayanan maka pasien pertama tidak perlu mengantri untuk daftar sehingga dapat diketahui waktu tunggu pasien pertama daftar sebesar 0. Waktu tunggu daftar sendiri dapat dihitung dari waktu masuk pasien ke (n) dikurangi waktu selesai daftar pasien ke (n - 1) * (-1).
5. Pasien pertama selesai daftar pada detik 1192. Karena ke dua dokter kosong sedang tidak melayani maka pasien pertama langsung masuk periksa ke dokter 1. Maka dapat diketahui waktu menganggur dokter 1 sebesar 1192 detik.Begitu seterusnya.
6. Waktu menganggur dokter didapat dari pengurangan waktu pasien ke (n) mulai periksa dikurangi waktu selesai layanan pasien ke (n - 1). Sedangkan waktu pasien mulai periksa sendiri didapat dari waktu selesai daftar pasien ke (n) dikurangi waktu selesai layanan pasien ke (n - 1). Jika hasilnya ≤ 0 maka waktu pasien mulai periksa (n) = waktu selesai layanan pasien ke (n - 1).
7. Waktu tunggu pasien periksa didapat dari waktu selesai daftar pasien ke (n) dikurangi waktu selesai layanan pasien ke (n - 1) * (-1)
8. Ketika pasien kedua datang loket ternyata sedang melayani pasien pertama jadi pasien kedua menunggu sampai pasien pertama selesai dilayani. Jadi

pasien kedua masuk dalam antrian. Sedangkan saat akan mulai periksa pasien pertama dilayani oleh dokter 1, sedangkan dokter 2 kosong sedang tidak melayani pasien maka pasien kedua langsung masuk mulai periksa dilayani oleh dokter 2 tanpa melakukan pengantrian. Inti dari pelayananya yang memiliki waktu selesai layanan lebih sedikit yang akan dimasuki oleh pasien terlebih dahulu.

9. Pada kasus pasien no 3 saat akan melakukan pemeriksaan ternyata antara dokter 1 dan dokter 2 yang kosong adalah dokter 2 maka pasien 3 langsung dilayani oleh dokter 2 tanpa menunggu dokter 1. Pasien langsung masuk layanan yang kosong terlebih dahulu. Begitu seterusnya
10. Setelah semua pasien selesai dilayani maka dapat dihitung nilai utilitas dari masing-masing loket dan dokter dengan cara (total waktu kerja - total waktu menganggur loket) / total waktu kerja yang kemudian hasilnya dikalikan 100%.

Penambahan atau pengurangan loket dan dokter diberlakukan apabila nilai utilitas dari loket atau dokter dianggap masih belum memenuhi nilai optimal. Penambahan loket atau dokter dilakukan jika nilai utilitas yang dirasakan terlalu besar, sedangkan pengurangan loket atau dokter dilakukan jika nilai utilitas lebih kecil dari nilai optimal yaitu diatas 70%.

Pada kasus simulasi seperti pada Tabel 4.7 didapatkan hasil total menganggur loket sebesar 1745 dan total menganggur dokter 1 sebesar 1192 dan total menganggur loket 2 sebesar 2395 jadi dapat dihitung :

$$\boldsymbol{Nilai\ Utilitas\ Loket} = \frac{\boldsymbol{total\ waktu\ kerja - total\ waktu\ menganggur\ loket}}{\boldsymbol{total\ waktu\ kerja}}\ \boldsymbol{x}\ \mathbf{100\%}$$

$$\boldsymbol{Nilai\ Utilitas\ Loket} = \frac{\mathbf{10800 - 1745}}{\mathbf{10800}}\ \boldsymbol{X}\ \mathbf{100}\ \%$$

$$= \mathbf{83.84\ \%}$$

$$\textbf{Nilai Utilitas Dokter 1} = \frac{\textbf{total waktu kerja} - \textbf{total waktu menganggur dokter 1}}{\textbf{total waktu kerja}} \textbf{ x 100\%}$$

$$\textbf{Nilai Utilitas Dokter 1} = \frac{\textbf{10800} - \textbf{1192}}{\textbf{10800}} \textbf{ X 100 \%}$$

$$= \textbf{84.96 \%}$$

$$\textbf{Nilai Utilitas Dokter 2} = \frac{\textbf{total waktu kerja} - \textbf{total waktu menganggur dokter 2}}{\textbf{total waktu kerja}} \textbf{ x 100\%}$$

$$\textbf{Nilai Utilitas Dokter 2} = \frac{\textbf{10800} - \textbf{1415}}{\textbf{10800}} \textbf{ X 100 \%}$$

$$= \textbf{88.82 \%}$$

$$\textbf{Total Utilitas Dokter} = \frac{\textbf{Nilai Utilitas Dokter 1} + \textbf{ Nilai Utilitas Dokter 2}}{\textbf{n dokter}}$$

$$\textbf{Total Utilitas Dokter} = \frac{\textbf{84 . 96} + \textbf{88. 82}}{\textbf{2}}$$

$$= \textbf{86.89 \%}$$

Dari perhitungan diatas dapat diketahui hasil utilitas loket sebesar 83.84%, nilai utilitas dokter 1 sebesar 84.96% dan nilai utilitas dokter 2 sebesar 88.82%. Berdasarkan hasil tersebut dapat disimpulkan bahwa nilai utilitas lebih besar dari nilai optimal (70%) jadi pelayanan pendistribusian pasien bernilai optimal.

4.3 Pengembangan Sistem

Pengembangan sistem yang digunakan oleh penulis adalah pengembangan model *waterfall*. Pengembangan model *waterfall* terdiri dari beberapa tahapan yaitu analisis kebutuhan, desain sistem, penulisan kode program, *testing*, penerapan program dan *maintenance*.

4.3.1 Statement of Purpose

Aplikasi simulator sistem manajemen pelayanan Pusat Kesehatan Masyarakat (PUSKESMAS) membantu pihak puskesmas dalam mensimulasikan manajemen pelayanan kepada pasien. Sistem ini akan membantu manager dalam menentukan jumlah sumber daya yang seharusnya tersedia untuk pelayanan yang optimal. Simulator sistem manajemen pelayanan Puskesmas ini memiliki beberapa fitur antara lain : fitur mengelola data antrian, mengelola data pendaftaran, mengelola data pemeriksaan, mengelola master data user, data segmen waktu, data poli, fitur untuk menguji distribusi data, membangkitkan bilangan random dan fitur untuk simulasi pelayanan Puskesmas serta report untuk menampilkan laporan data. Fitur-fitur tersebut diharapkan dapat memberikan alternatif solusi bagi pihak manager untuk menentukan jumlah sumber daya yang optimal dalam pelayanan Puskesmas.

4.3.2 Analisis Kebutuhan

Analisis kebutuhan perangkat lunak dalam penelitian ini yaitu dengan cara mengidentifikasi permasalahan yang ada untuk kemudian dicatat dan dijadikan bahan untuk mulai membangun simulator sistem pelayanan Puskesmas. Analisis kebutuhan yang dilakukan meliputi proses pengumpulan data kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional adalah kebutuhan yang berisi proses-proses apa saja yang nantinya dilakukan oleh sistem. Sedangkan kebutuhan nonfungsional adalah kebutuhan yang menitikberatkan pada properti prilaku yang dimiliki oleh sistem. Berikut adalah kebutuhan fungsional dan non-fungsional sistem :

a. Kebutuhan fungsional dari sistem yang akan dibangun
    1. Sistem dapat mengelola data antrian pasien.
    2. Sistem dapat mengelola data pendaftaran pasien.
    3. Sistem dapat mengelola data pemeriksaan pasien.
    4. Sistem dapat mengelola data master user, segmen waktu dan data poli.

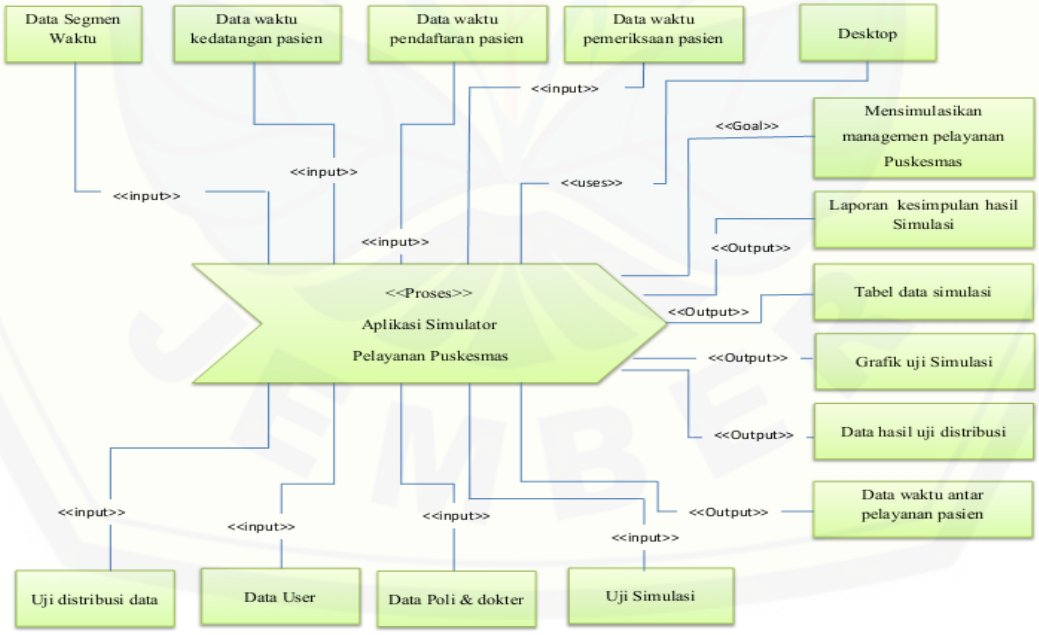
5. Sistem dapat mengelola uji ditribusi data.
6. Sistem dapat mengelola data *random seed* sesuai uji distribusi data.
7. Sitsem dapat mengelola simulasi data.
8. Sistem dapat menampilkan kesimpulan hasil simulasi data yang berupa nilai utilitas pelayanan Puskesmas.
9. Sitem dapat menampilkan detail data hasil simulasi.
10. Sistem dapat menampilkan *report* atau laporan data.

b. Kebutuhan non-fungsional dari sistem yang akan dibangun

1. Sistem berbentuk *desktop*
2. Sistem menggunakan *java programming* dengan konsep OOP (*Object Oriented Programming*)

4.3.3 *Bussiness Process*

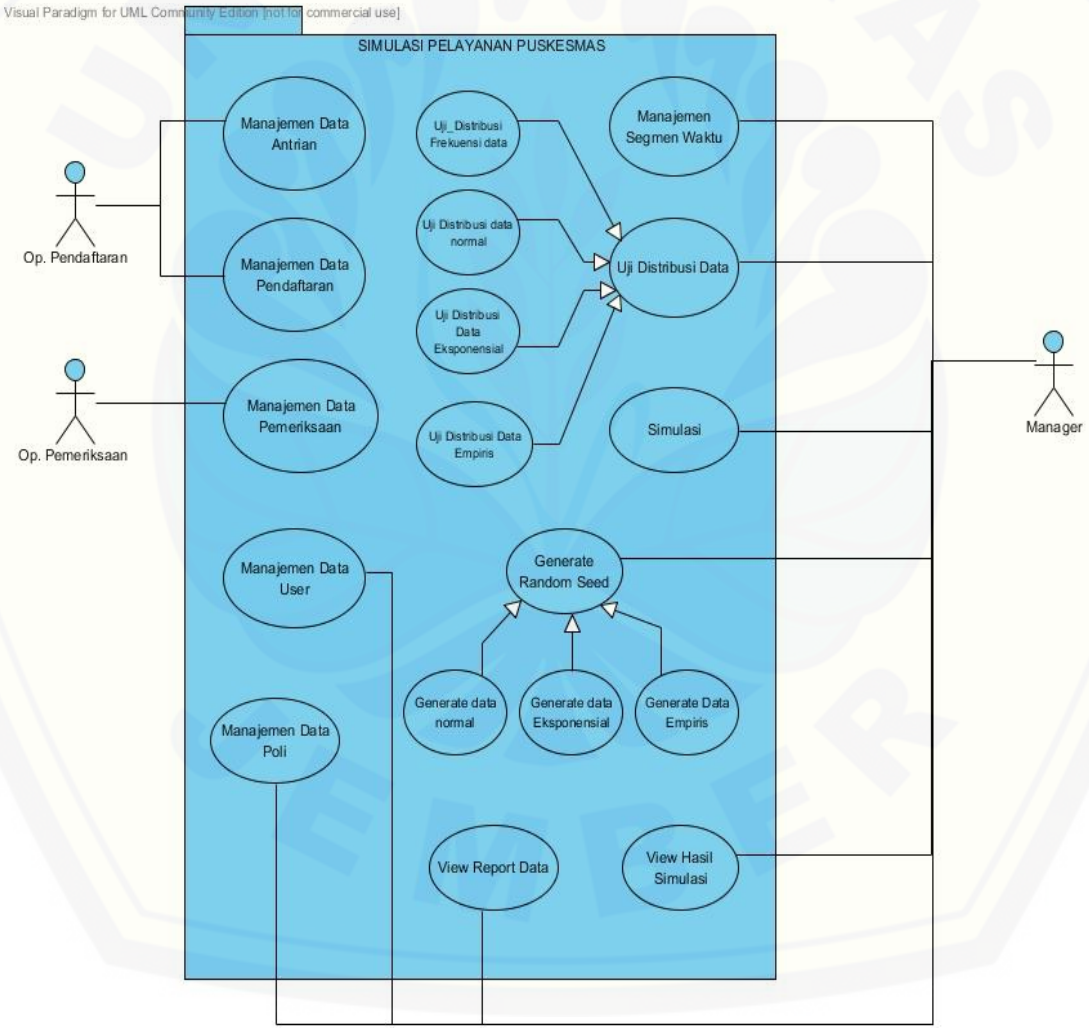
Bussines Process digunakan untuk menggambarkan inputan data yang dibutuhkan sistem, output dari sistem serta tujuan dari pembuatan sistem. *Bussines Process* sistem dapat dilihat pada Gambar 4.1



Gambar 4.1 *Business proces* simulator sistem pelayanan Puskesmas

4.3.4 *Usecase Diagram*

Usecase Diagram adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. *Usecase Diagram* berfungsi untuk menggambarkan fitur apa saja yang akan dijalankan pada *software* yang akan dibuat.*Usecase Diagram* simulator sistem pelayanan Puskesmas dapat diilihat pada Gambar 4.2. Definisi *usecase* dapat dilihat pada Tabel 4.8 dan definisi actor dapat dilihat pada Tabel 4.9



Gambar 4.2 *Usecase diagram* simulator sistem pelayanan Puskesmas

Tabel 4.8 Definisi *usecase*

No.	Usecase	Deskripsi
1.	Manajemen data antrian	Operator pendaftaran dapat menambah data antrian yang berupa data nomer antrian dan waktu kedatangan pasien.
2.	Manajemen data pendaftaran	Proses manajemen data oleh operator pendaftaran untuk menambah atau mengedit data pendaftaran pasien yang berupa data pasien dan data waktu mulai hingga waktu selesai pasien daftar.
3.	Manajemen data pemeriksaan	Operator pemeriksaan dapat menambah data pemeriksaan pasien yang berupa data waktu mulai periksa dan waktu selesai periksa pasien
4.	Manajemen data user	Manajer dapat menambah , update dan menghapus user yang meliputi user operator pendaftaran, operator pemeriksaan dan manajer
5.	Manajemen data poli	Manajer dapat menambah , update atau menghapus data poli yang berupa nama poli dan data jumlah dokter pada poli.
6.	Manajemen data segmen waktu	Manajer dapat menambah , update atau menghapus data segmen waktu yang berupa nama segmen, waktu awal dan waktu akhir dari segmen waktu.
7.	Uji distribusi data	Proses pengujian data sampel dari waktu antar kedatangan, lama daftar dan lama periksa pasien yang berupa pengujian distribusi frekuensi data, dan jenis distribusi suatu data (normal, eksponesial, empiris) yang dilakukan oleh manajer.
8.	*Generate random seed*	Proses pembangkitan bilangan acak oleh manajer dari data sampel waktu antar kedatangan, lama

		daftar dan lama periksa pasien sesuai dengan jenis distribusi datanya yang mana hasil dari bilangan acak tersebut akan digunakan untuk proses simulasi.
9.	Simulasi	Proses mensimulasikan data oleh manajer yang mana manajer akan menginputkan jumlah sumber daya (loket pendaftaran dan dokter) lalu akan dilakukan perhitungan data simulasi.
10.	*View* hasil simulasi	Manajer dapat melihat hasil kesimpulan simulasi yang berupa nilai utilitas dari manajemen pelayanan puskesmas.
11.	*View report* data	Manajer dapat melihat keseluruhan rekap data yang berupa data pasien, data pegawai, data poli, data segmen waktu dan data simulasi.

Tabel 4.9 Definisi Aktor

No.	Usecase	Deskripsi
1.	Manajer	Operator sistem yang mengelola master data (user, segmen waktu, poli), melakukan uji distribusi data dan melakukan simulasi data.
2.	Operator pendaftaran	Operator sistem yang mengelola data antrian dan data pendaftaran pasien yang berupa data waktu antar kedatangan, lama daftar dan data pasien
3.	Operator Pemeriksaan	Operator sistem yang mengelola data pemeriksaan pasien yang berupa data waktu lama periksa pasien

4.3.5 *Scenario*

Scenario berfungsi untuk menggambarkan alur sistem beserta alternatif alur yang akan dijalankan oleh user pada simulator sistem pelayanan Puskesmas. Berikut adalah skenario dari sistem yang akan dibuat.

Tabel 4.10 Skenario manajemen data segmen waktu

ID	:	UCS-06
Name	:	Manajemen data segmen waktu
Participating Aktor	:	Manajer
Triger	:	-
Entry Condition	:	Manajer login ke sistem
Exit Condition	:	Manajer sukses menambahkan data, mengubah data atau menghapus data segmen waktu.

SKENARIO UTAMA	
Actor	***System***
1. Klik menu *master data*	
2. Klik submenu data segmen waktu	
	3. Menampilkan form data segmen waktu beserta field form data segmen waktu , tombol tambah,edit,delete,reset dan tabel data segmen waktu
4. Ketika ingin menambah segmen waktu	
5. Mengisi form data segmen waktu	
6. Klik tombol tambah	
	7. Simpan data kedalam *database* sistem

	8. Menampilkan *message* data berhasil ditambah
	9. Menampilkan kembali form data segmen waktu dengan tabel yang telah terisi data segmen waktu yang baru diinputkan
10. Ketika ingin mengubah segmen waktu	
11. Pilih data *row* tabel yang ingin di*update*	
	12. Menampilkan informasi data segmen waktu yang dipilih di *field form*
13. *Input update* form data segmen waktu	
14. Klik tombol Edit	
	15. *Update* data pada *database* sistem
	16. Menampilkan *message* data berhasil di*update*
17. Ketika ingin menghapus segmen waktu	
18. Pilih data *row* tabel yang ingin di*delete*	
	19. Menampilkan informasi data segmen waktu yang dipilih di *field form*
20. Klik tombol *delete*	
	21. *Delete* data pada *database* sistem
	22. Menampilkan *message* data berhasil di*delete*
	23. Menampilkan kembali form data segmen

	waktu dengan tabel data segmen waktu
SKENARIO ALTERNATIF	
5. Ketika klik tombol tambah dan belum mengisi form segmen waktu	
	6. Menampilkan error message data masih ada yang kosong atau belum lengkap
14.Ketika klik tombol *edit* dan belum memilih *row* data pada tabel	
	15. Menampilkan error message belum ada data yang dipilih
16.Ketika klik tombol *edit* dan belum mengisi semua data atau masih ada data yang kosong di *field form*	
	17.Menampilkan error message data masih ada yang kosong atau belum lengkap
20. Ketika klik tombol *delete* dan belum memilih *row* data pada tabel	
	21. Menampilkan error message belum ada data yang dipilih

Tabel 4.11 Skenario uji distribusi frekuensi data (uji *eksponensial*)

ID	:	UCS-07
Name	:	Uji distribusi frekuensi data (uji *eksponensial*)
Participating Aktor	:	Manajer
Triger	:	-
Entry Condition	:	Manajer login ke sistem
Exit Condition	:	Manajer sukses menguji distribusi data dengan uji distribusi *eksponensial*

SKENARIO UTAMA	
Actor	***System***
1. Klik menu uji distribusi	
	2. Menampilkan frame uji distribusi data dengan variabel event tanggal data,jenis data, segmen waktu, poli ,tabel data sampel, tabel data hasil perhitungan distribusi frekuensi dan tombol set data
3. Pilih tanggal data, jenis data, poli dan segmen waktu data pada *combo box* yang tersedia	
4. Klik tombol set data	
	5. Menampilkan data sampel berdasarkan variabel (tanggal,jenis, segmen waktu dan poli) yang dipilih pada tabel data sampel (persiapan data) dan data perhitungan distribusi frekuensi dari data sampel pada tabel distribusi frekuensi beserta grafik distribusi frekuensi data

	6. Simpan data ke dalam database
7. Klik tombol uji eksponensial	
	8. Menampilkan frame hasil uji data distribusi eksponensial dengan tabel perhitungan data uji eksponensial beserta hasil kesimpulan perhitungan data
	9. Simpan data ke dalam database
SKENARIO ALTERNATIF	
4. Ketika klik tombol set data dan belum memilih semua variabel event data	
	5. Menampilkan error message semua event harus dipilih terlebih dahulu
6. Ketika klik tombol uji *eksponensial* dan belum melakukan uji normal	
	7. Menampilkan error message pilih uji normal terlebih dahulu

Tabel 4.12 Skenario *generate random seed*

ID	:	UCS-08
Name	:	*Generate random seed*
Participating Aktor	:	Manajer
Triger	:	-
Entry Condition	:	Manajer login ke sistem, Manajer telah melakukan uji distribusi data
Exit Condition	:	Manajer sukses membangkitkan bilangan random dari sampel data berdasarkan uji distribusi

SKENARIO UTAMA	
Actor	***System***
1. Klik menu uji distribusi	
	2. Menampilkan frame uji distribusi data dengan variabel event tanggal data,jenis data, segmen waktu, poli ,tabel data sampel, tabel data hasil perhitungan distribusi frekuensi dan tombol set data
3. Pilih tanggal data, jenis data, poli dan segmen waktu data pada *combo box* yang tersedia	
4. Klik tombol set data	
	5. Menampilkan data sampel berdasarkan variabel (tanggal,jenis, segmen waktu dan poli) yang dipilih pada tabel data sampel (persiapan data) dan data perhitungan distribusi frekuensi dari data sampel pada

	tabel distribusi frekuensi beserta grafik distribusi frekuensi data
	6. Simpan data ke dalam database
7. Ketika melakukan uji distribusi normal	
8. Klik tombol uji distribusi normal	
	9. Menampilkan frame hasil uji data distribusi normal dengan tabel perhitungan data uji normal beserta hasil kesimpulan perhitungan data.
	10. Simpan data ke dalam database
	11. Menampilkan tombol *generate random seed*
12. Klik tombol *generate random seed*	
	13. Simpan data ke dalam database.
	14. Menampilkan hasil perhitungan bilangan acak *(random seed)* berdistribusi normal pada tabel bilangan acak.
15. Ketika melakukan uji distribusi *eksponensial*	
16. Klik tombol uji distribusi *eksponensial*	
	17. Menampilkan frame hasil uji data distribusi *eksponensial* dengan tabel perhitungan data uji normal beserta hasil kesimpulan perhitungan data.

	18. Simpan data ke dalam database
	19. Menampilkan tombol *generate random seed*
20. Klik tombol *generate random seed*	
	21. Simpan data ke dalam database.
	22. Menampilkan hasil perhitungan bilangan acak *(random seed)* berdistribusi *ekponensial* pada tabel bilangan acak.
23. Ketika melakukan uji distribusi *empiris*	
24. Klik tombol uji distribusi *empiris*	
	25. Menampilkan frame hasil uji data distribusi *empiris* dengan tabel perhitungan data uji normal beserta hasil kesimpulan perhitungan data.
	26. Simpan data ke dalam database
	27. Menampilkan tombol *generate random seed*
28. Klik tombol *generate random seed*	
	29. Simpan data ke dalam database.
	30. Menampilkan hasil perhitungan bilangan acak *(random seed)* berdistribusi *empiris* pada tabel bilangan acak.
SKENARIO ALTERNATIF	
4. Ketika klik tombol set data	

dan belum memilih semua variabel event data	
	5. Menampilkan error message semua event harus dipilih terlebih dahulu

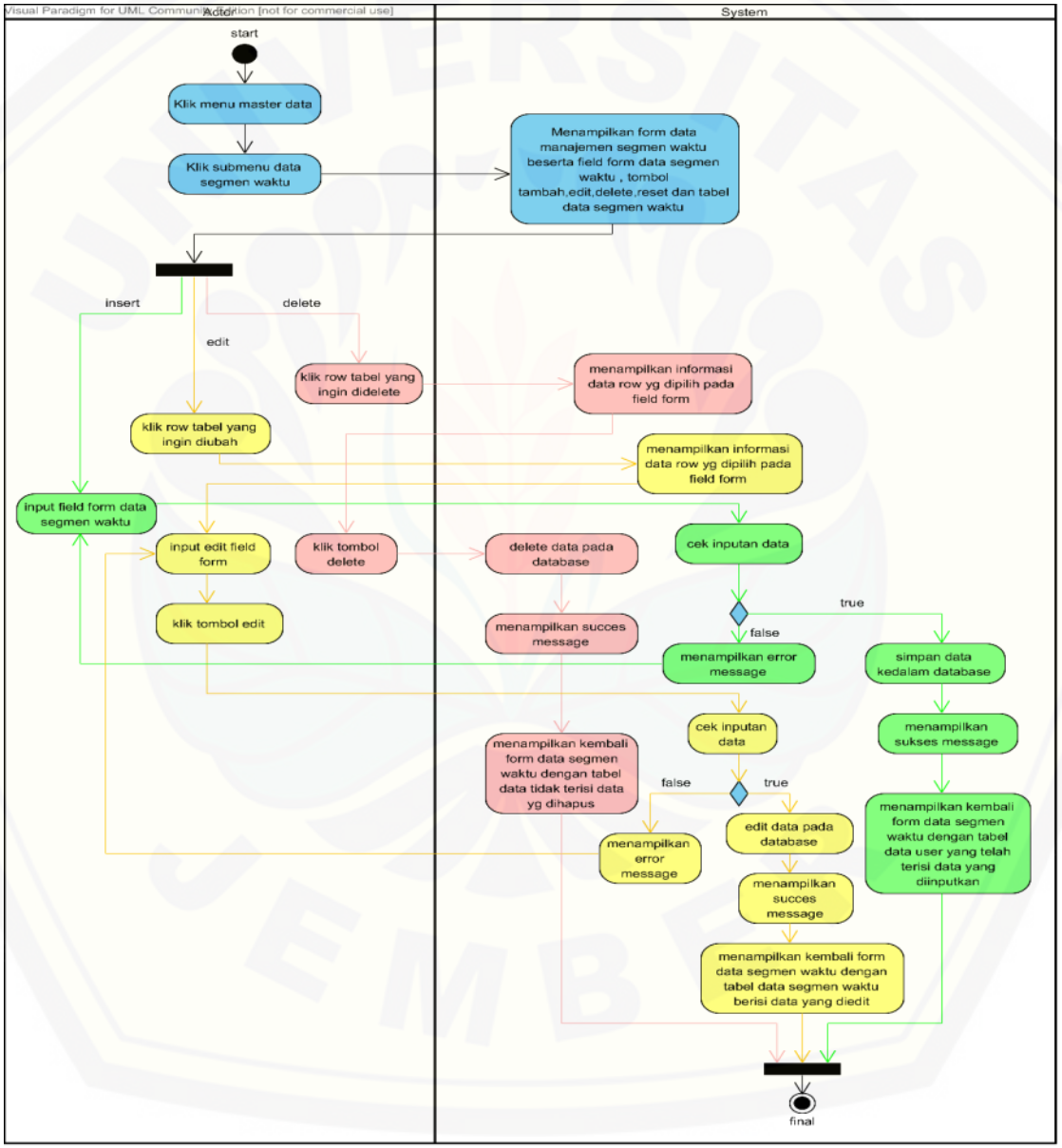
Tabel 4.13 Skenario simulasi

ID	:	UCS-09
Name	:	Simulasi
Participating Aktor	:	Manajer
Triger	:	-
Entry Condition	:	Manajer login ke sistem
Exit Condition	:	Manajer sukses melakukan simulasi
SKENARIO UTAMA		
Actor		***System***
1. Klik menu simulasi		
		2. Menampilkan *frame form* simulasi dengan tombol set data, *process* dan tabel perhitungan simulasi
3. Pilih variabel data yang mau disimulasikan (tanggal, segmen waktu dan poli)		
4. Klik tombol set data		
		5. Menampilkan hasil set data (persiapan data) pada tabel perhitungan simulasi dan *textfield* inputan jumlah loket beserta *textfield* jumlah dokter

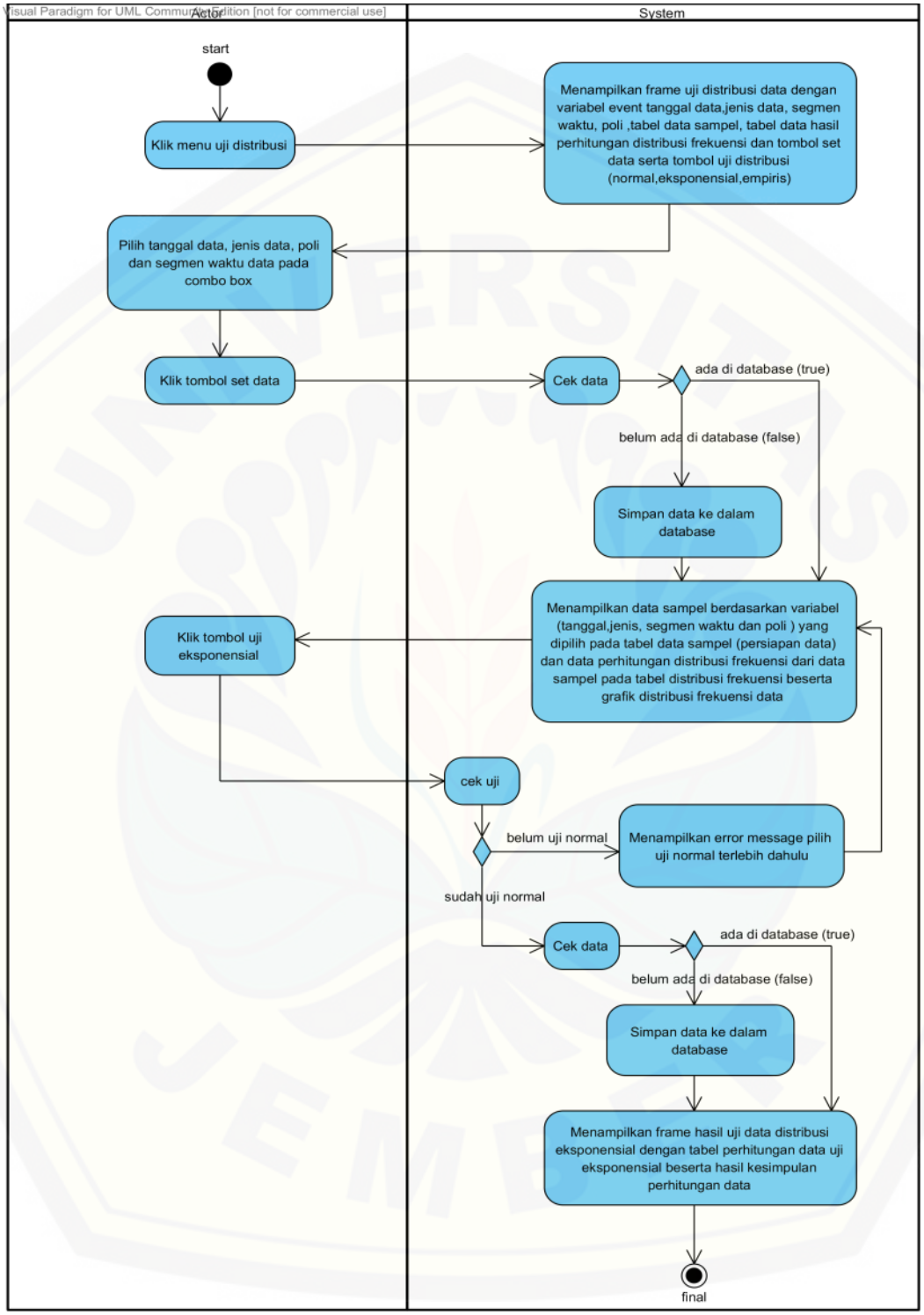
6. Input jumlah loket dan dokter yang mau disimulasikan	
7. Klik tombol *process*	
	8. Simpan data kedalam *database*
	9. Menampilkan perhitungan simulasi pada tabel simulasi beserta nilai utilitas loket dan dokter
10. Klik tombol detail simulasi	
	11. Simpan data kedalam *database*
	12. Menampilkan frame kesimpulan dan detail hasil simulasi
SKENARIO ALTERNATIF	
4. Ketika klik tombol set data dan belum memilih semua variabel data	
	5. Menampilkan error message semua event harus dipilih terlebih dahulu
7. Ketika klik tombol *process* dan belum menginputkan jumlah loket atau dokter yang mau disimulasikan	
	7. Menampilkan error message masih ada data yang kosong atau kurang

4.3.6 *Activity diagram*

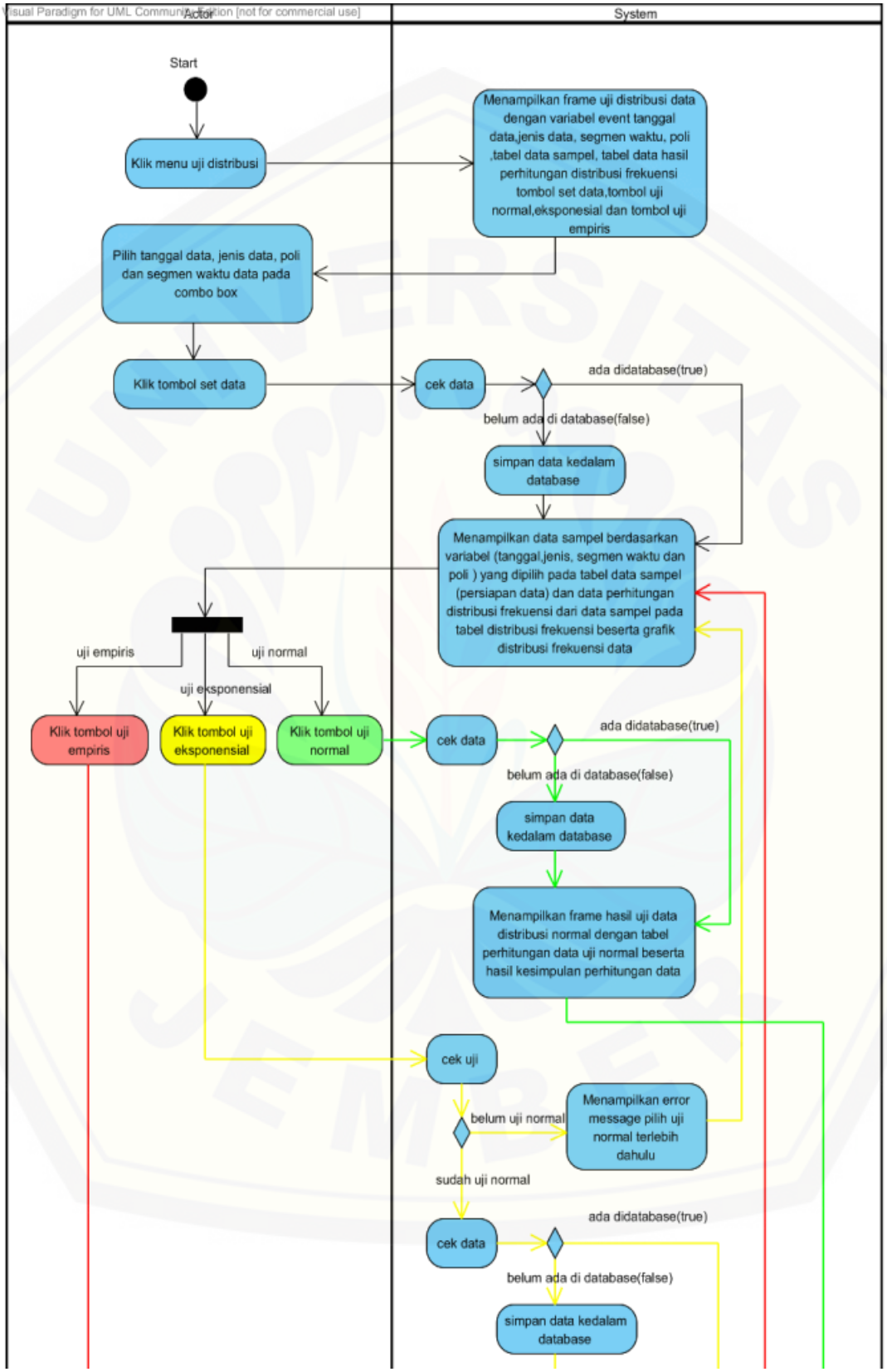
Activity diagram simulator sistem pelayanan Puskesmas ini berfungsi untuk menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. Activity Diagram sistem adalah sebagai berikut :

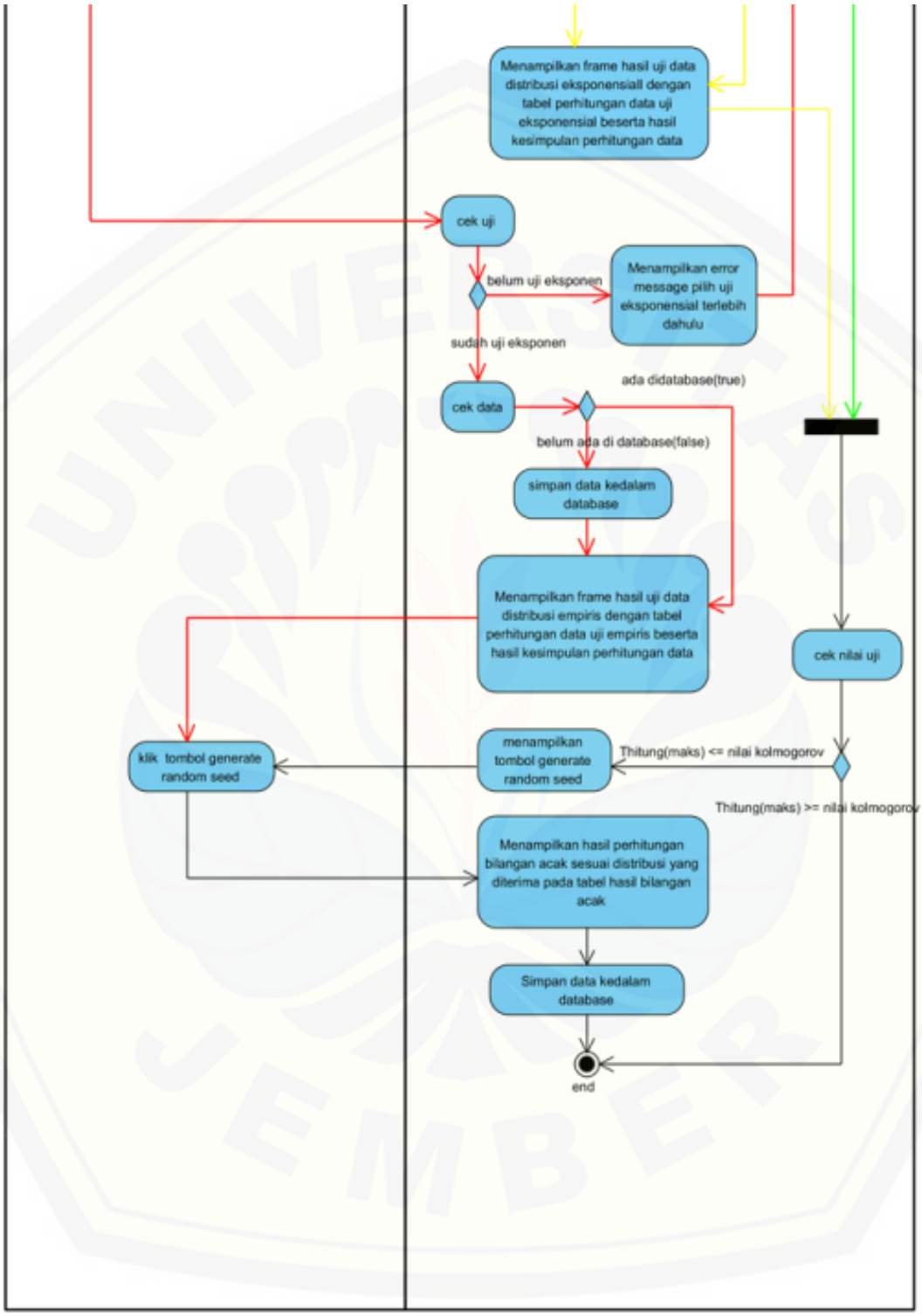


Gambar 4.3 *Activity diagram* manajemen data segmen waktu

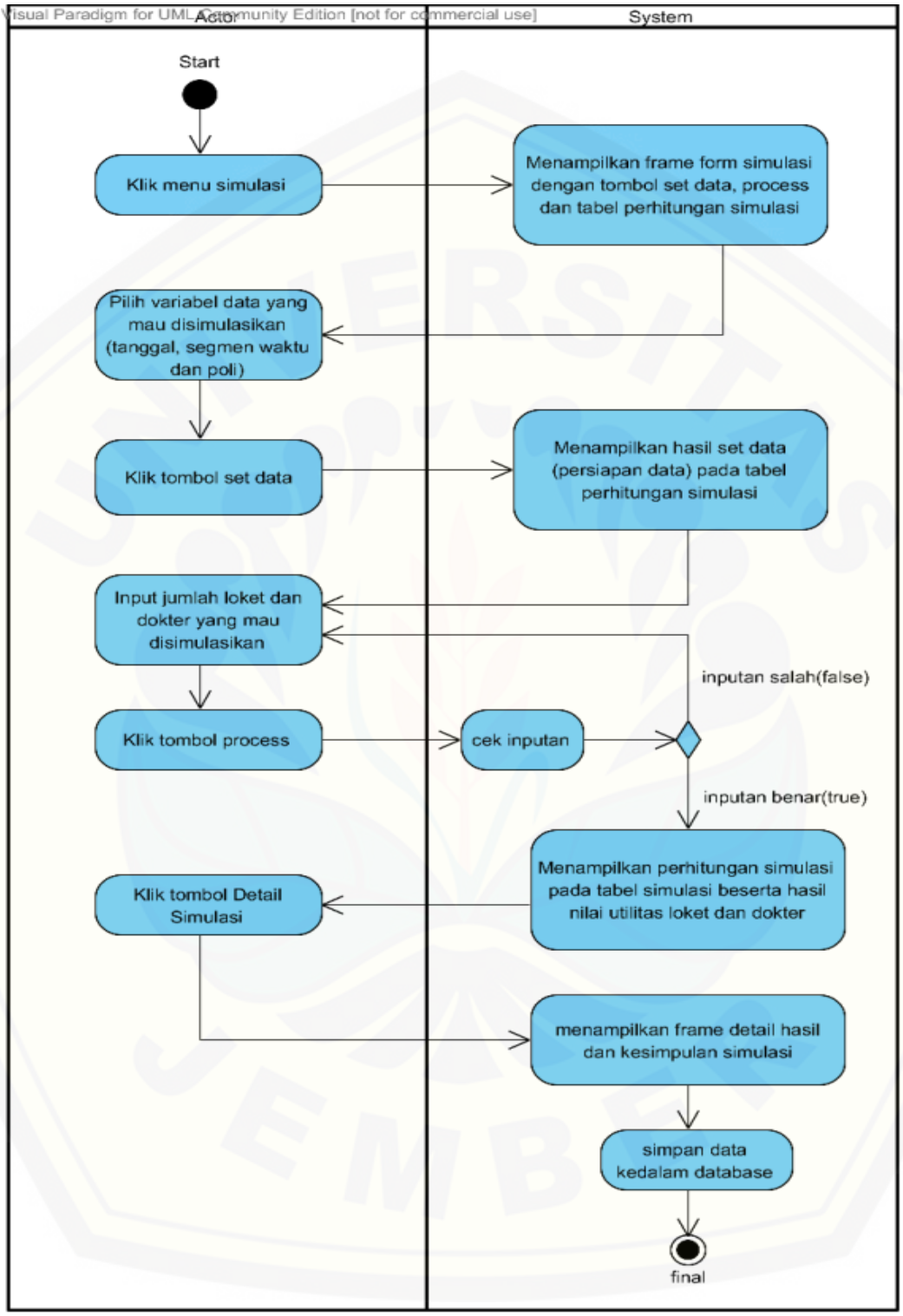


Gambar 4.4 *Activity diagram* uji distribusi data *(eksponensial)*





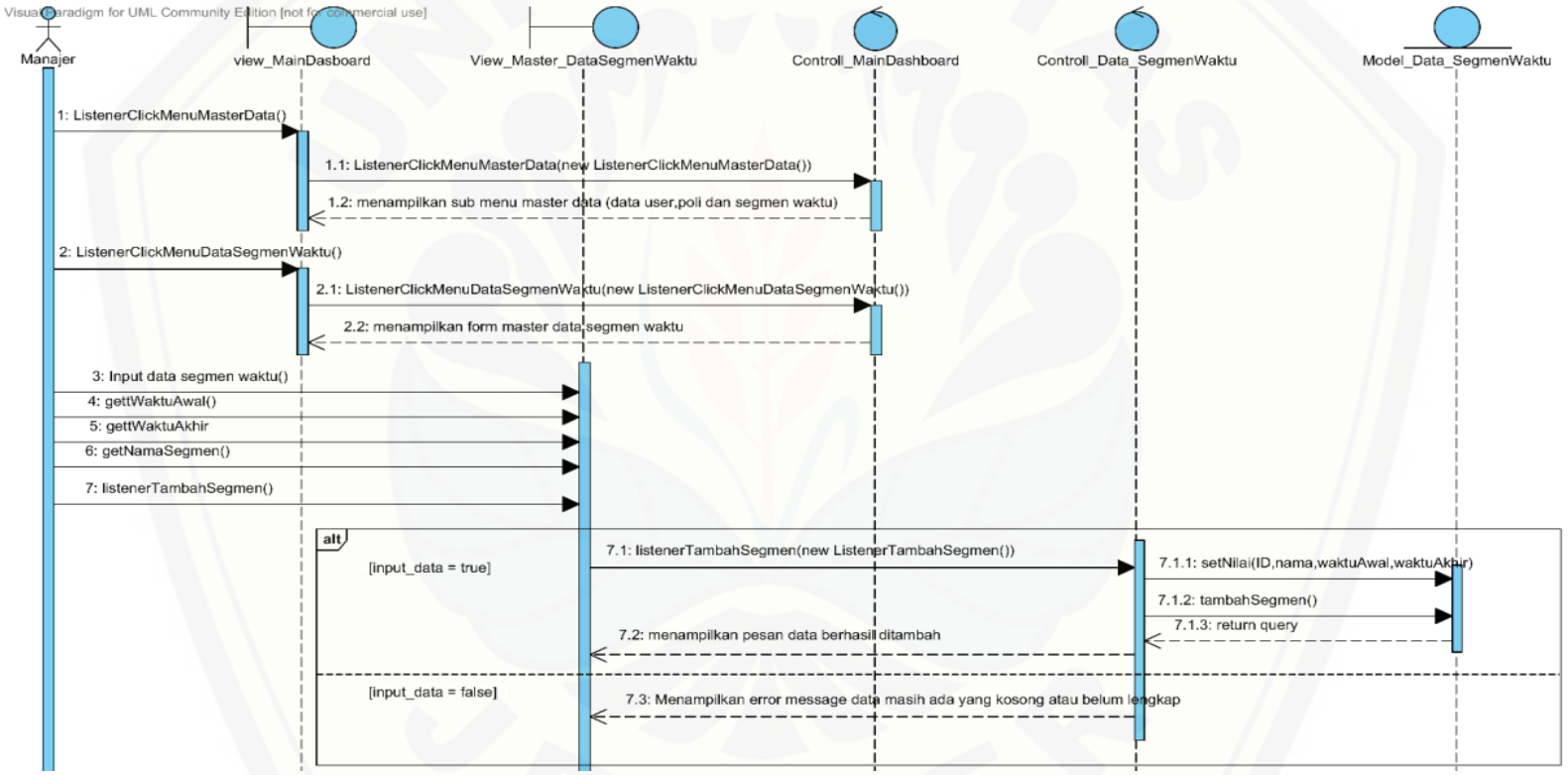
Gambar 4.5 *Activity diagram generate random seed*

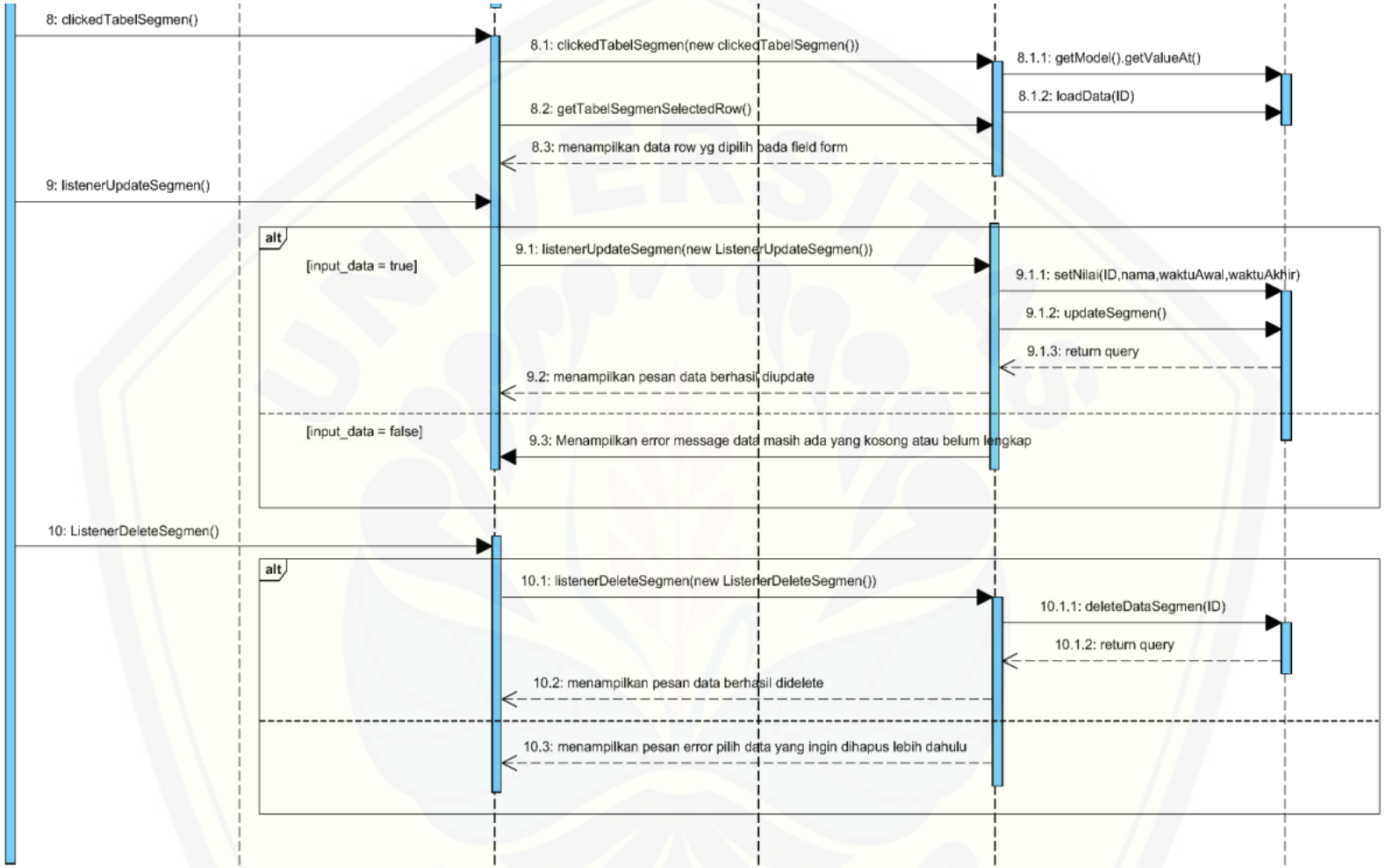


Gambar 4.6 *Activity diagram* Simulasi

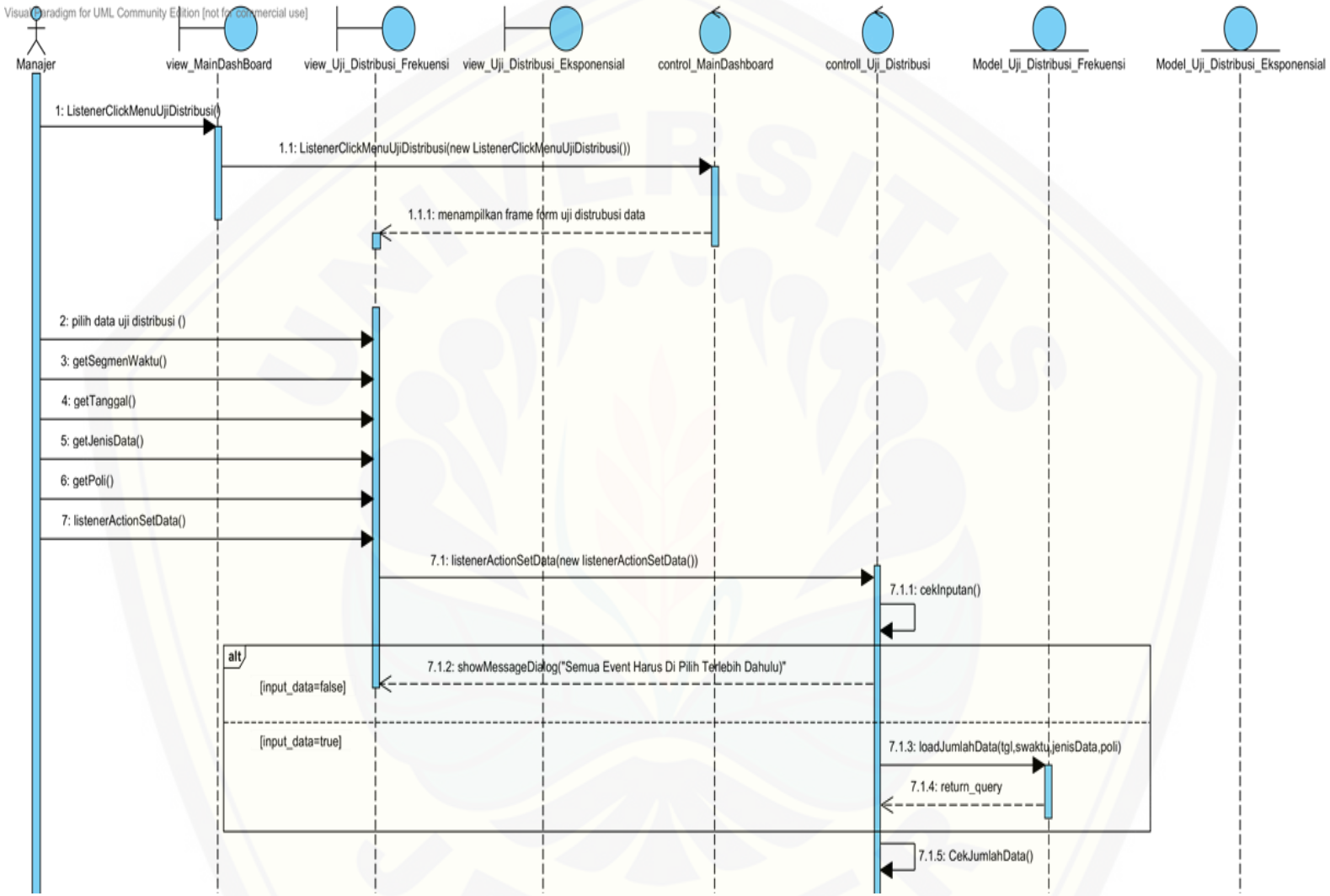
4.3.7 *Sequence diagram*

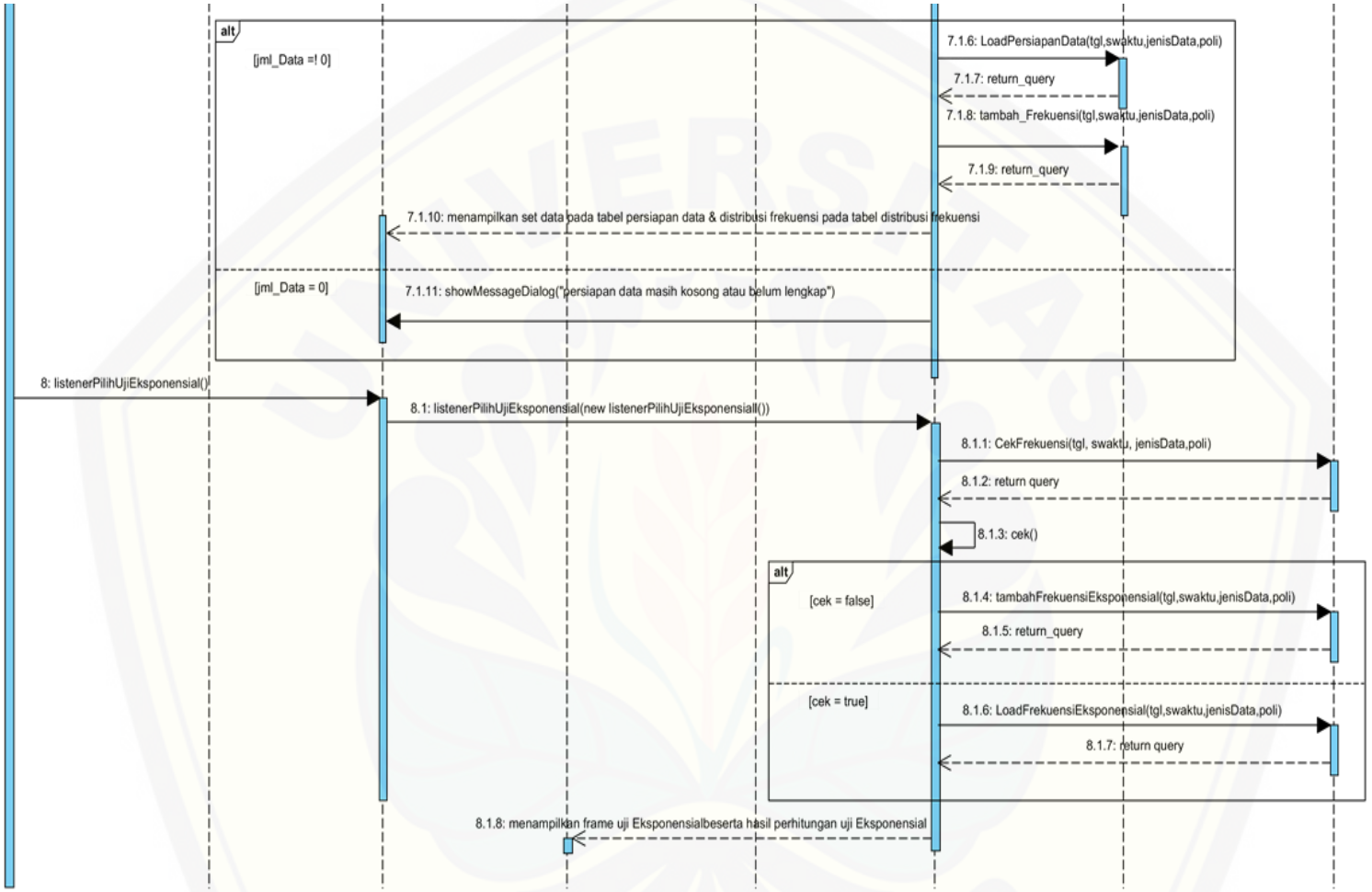
Sequence diagram pada simulator sistem pelayanan Puskesmas ini digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai sebuah respon dari suatu kejadian/even untuk menghasilkan output tertentu. *Sequence Diagram* diawali dari apa yang me-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. *Sequence Diagram* dari sistem adalah sebagai berikut :



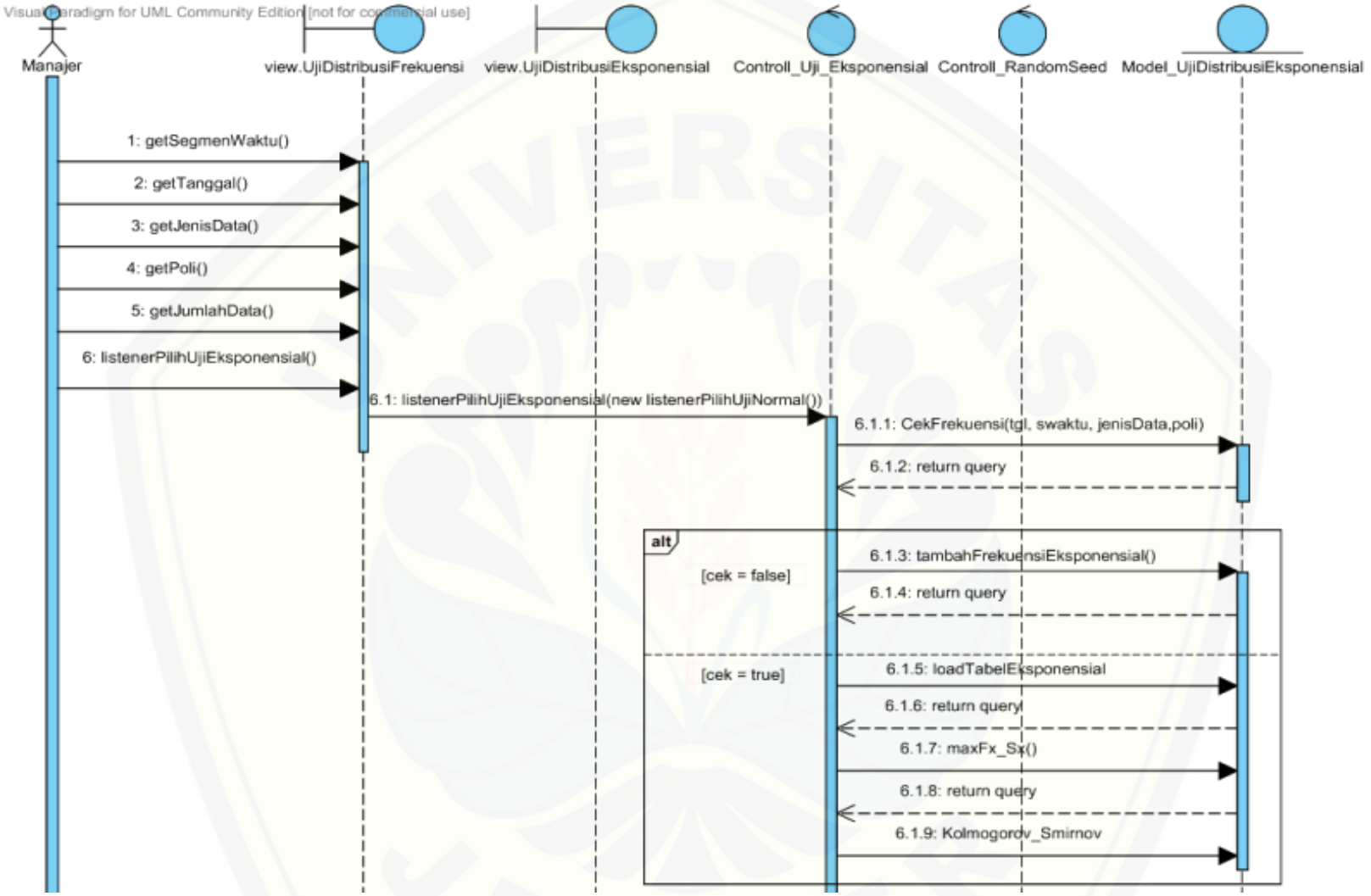


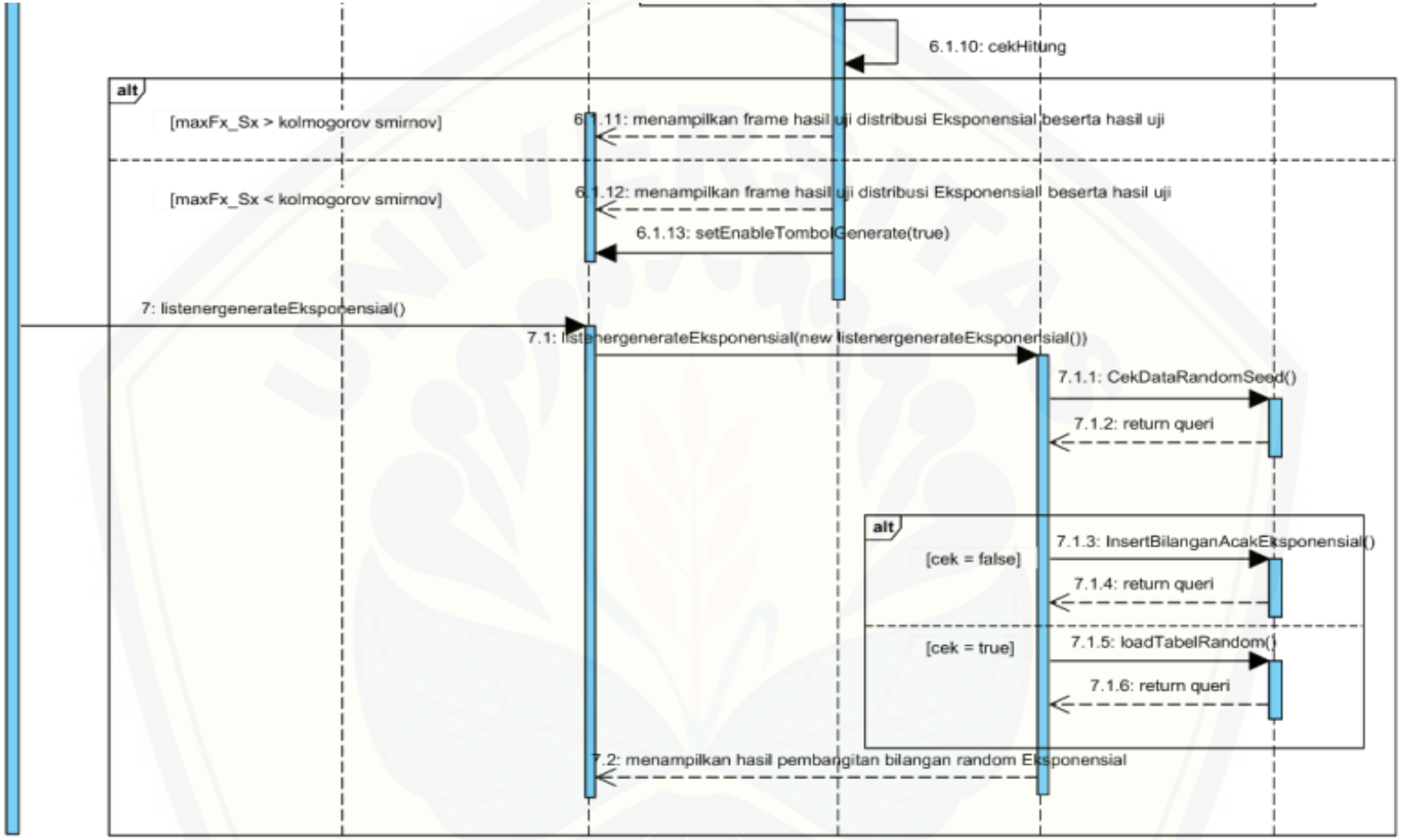
Gambar 4.7 *Sequence diagram* manajemen segmen waktu



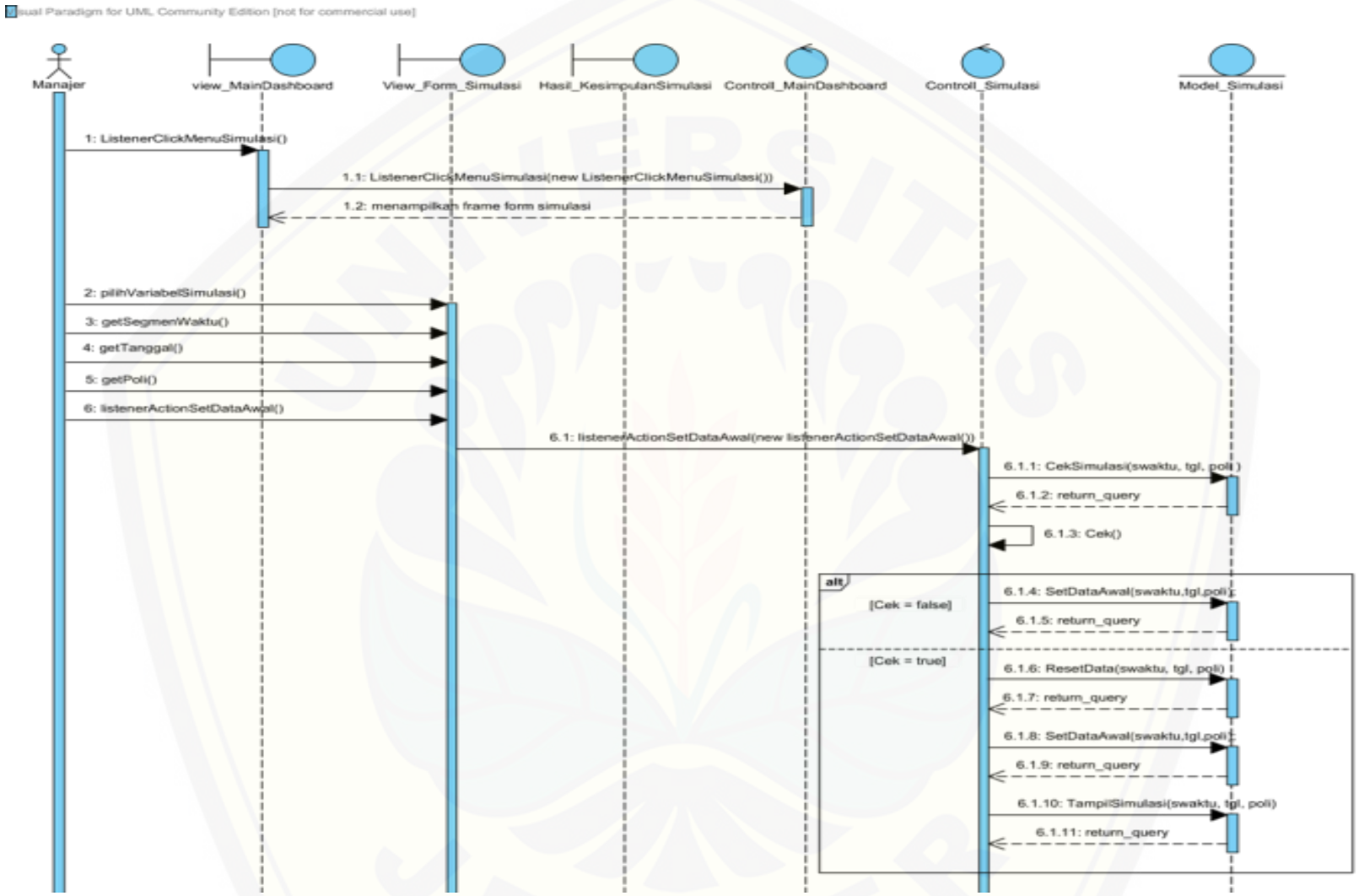


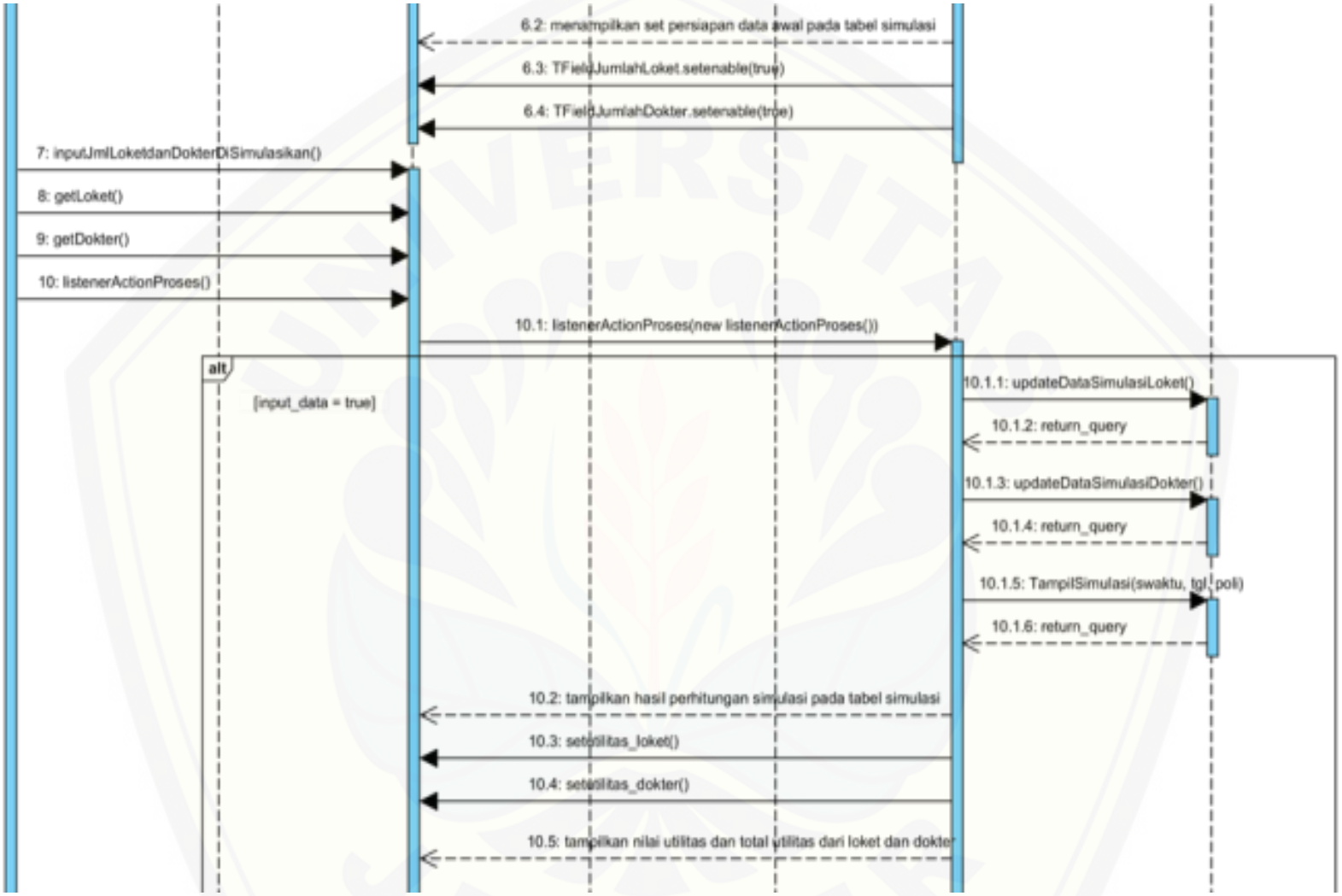
Gambar 4.8 *Sequence diagram* uji distribusi data (*eksponensial*)

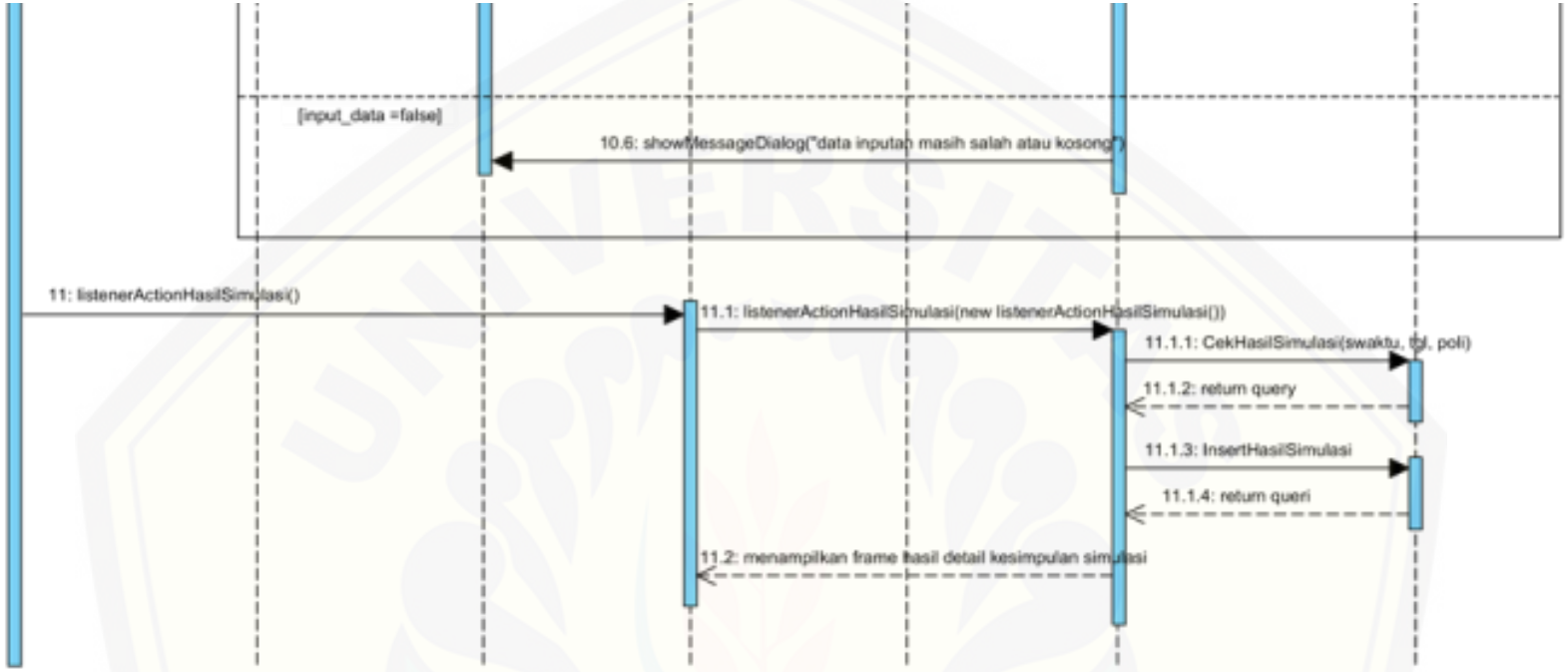




Gambar 4.9 *Sequence diagram generate random seed (eksponensial)*



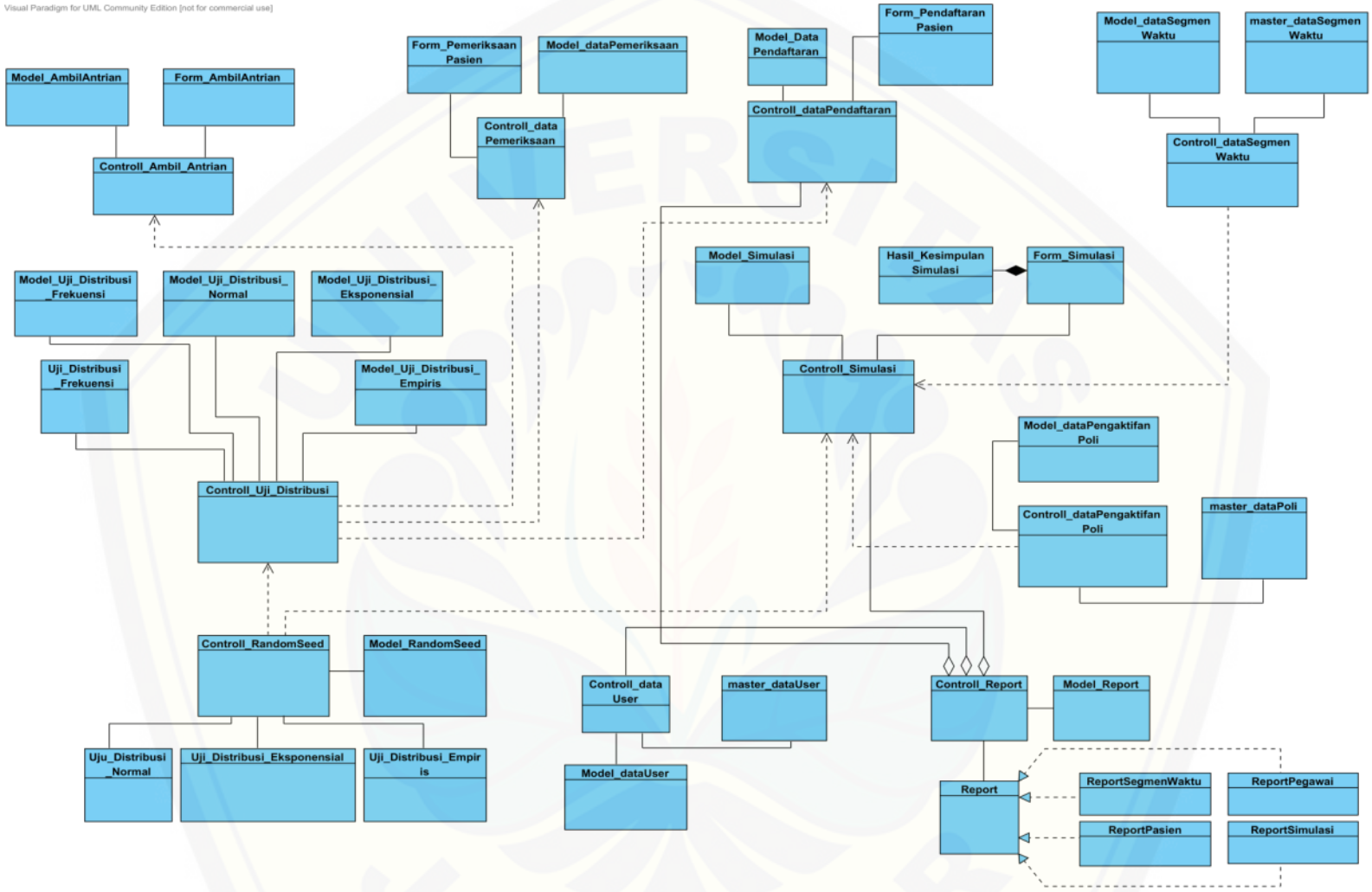




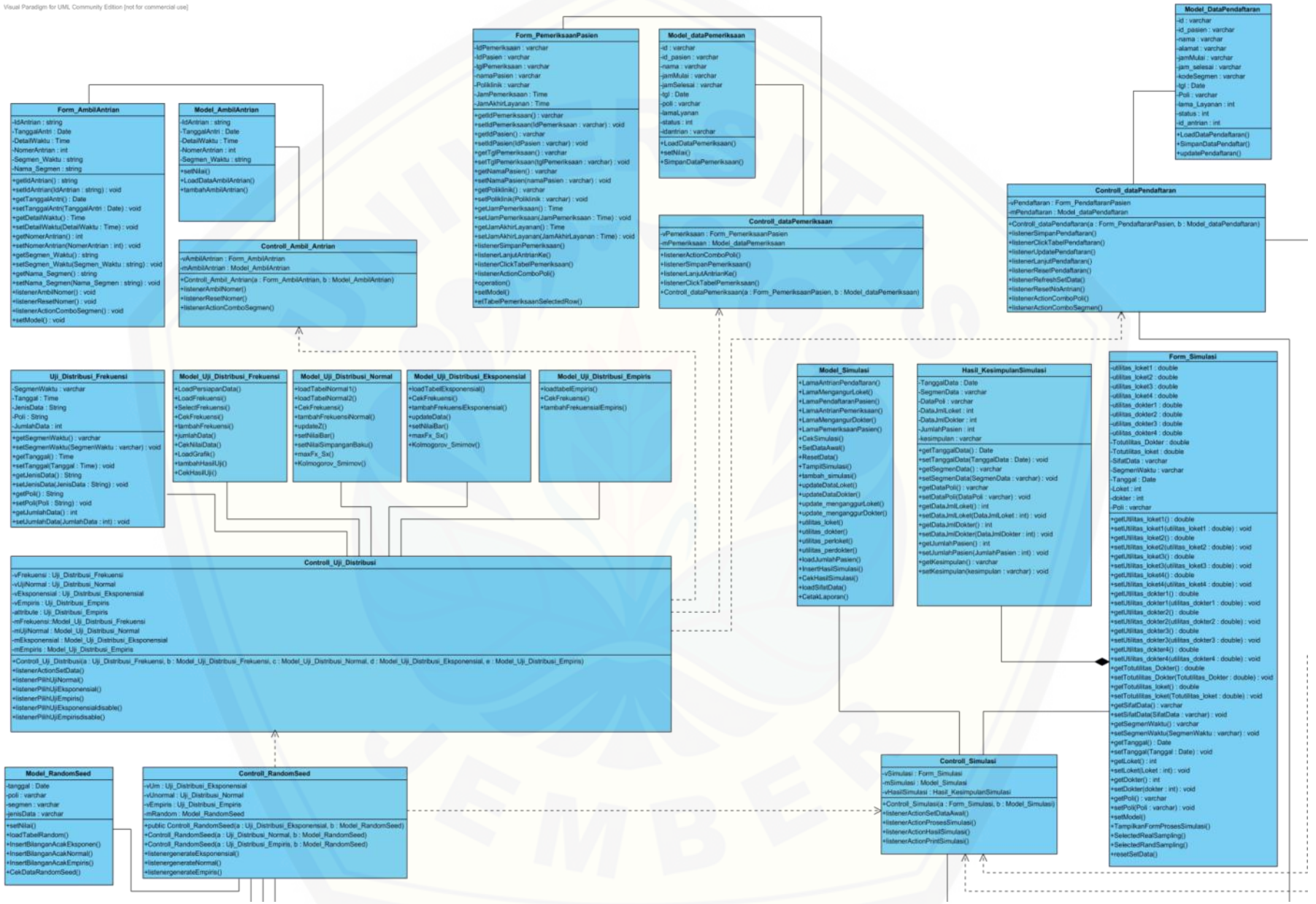
Gambar 4.10 *Sequence diagram* simulasi data

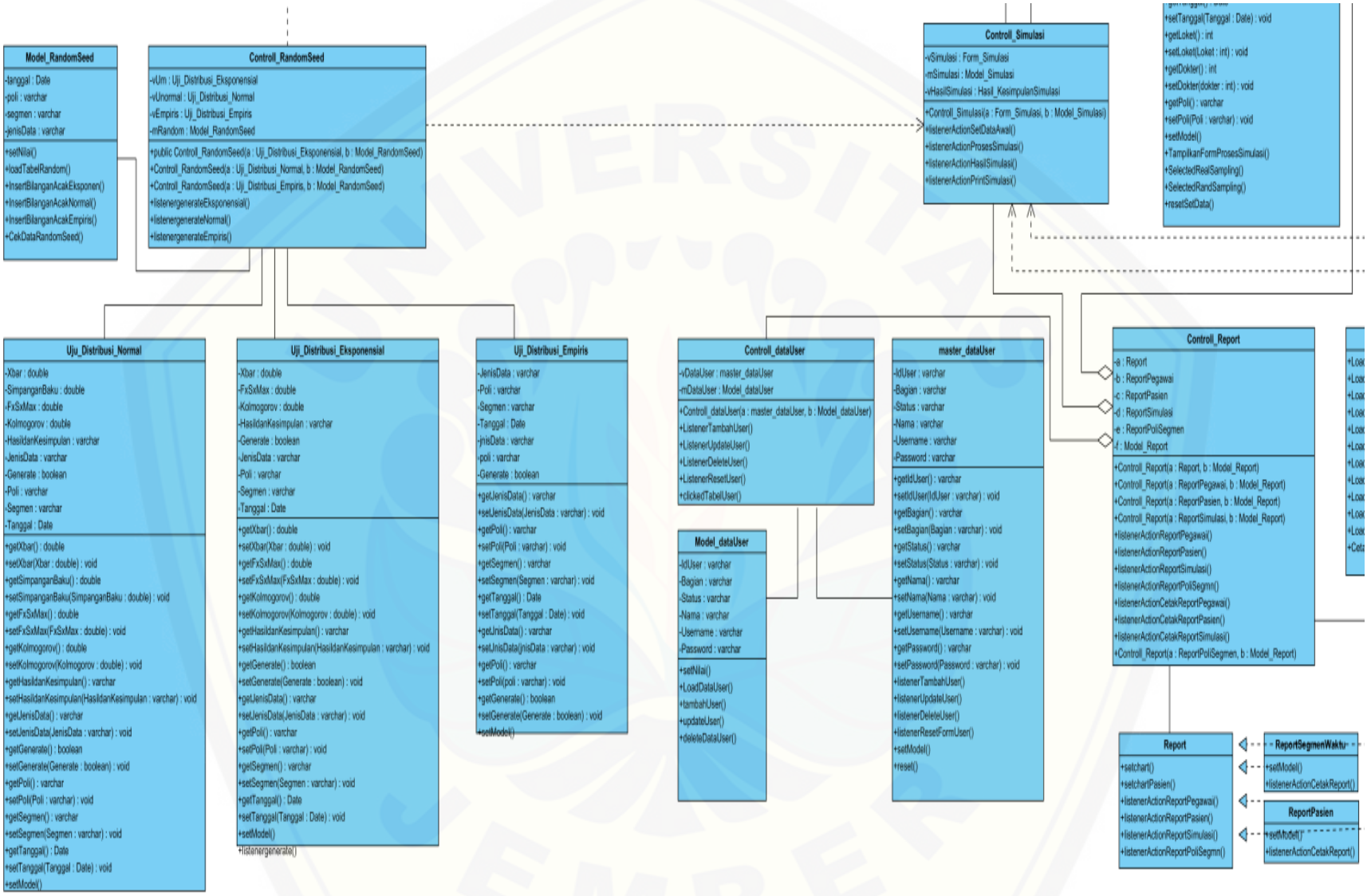
4.3.8 *Class diagram*

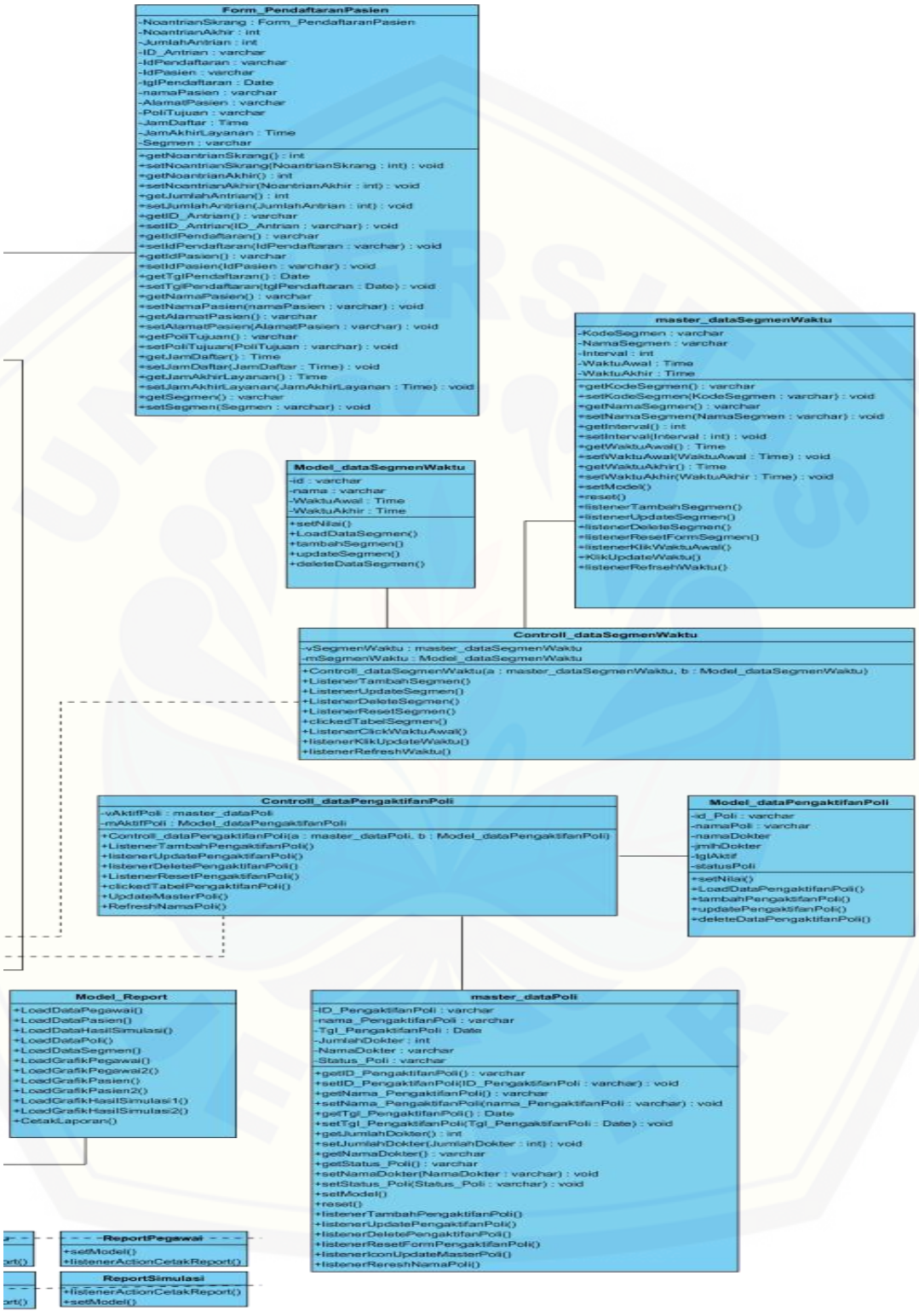
Class diagram menggambarkan struktur dan penjelasan class, paket, dan objek serta hubungan satu sama lain seperti pewarisan, asosiasi, dal lain-lain. Selain itu class diagram juga menjelaskan hubungan antar class dalam sebuah sistem yang sedang dirancang sehingga bagaimana caranya setiap class saling berkalaborasi untuk mencapai sebuah tujuan. *Class diagram* sistem dapat dilihat pada Gambar 4.11.



Gambar 4.11 *Class diagram simulator* pelayanan Puskesmas

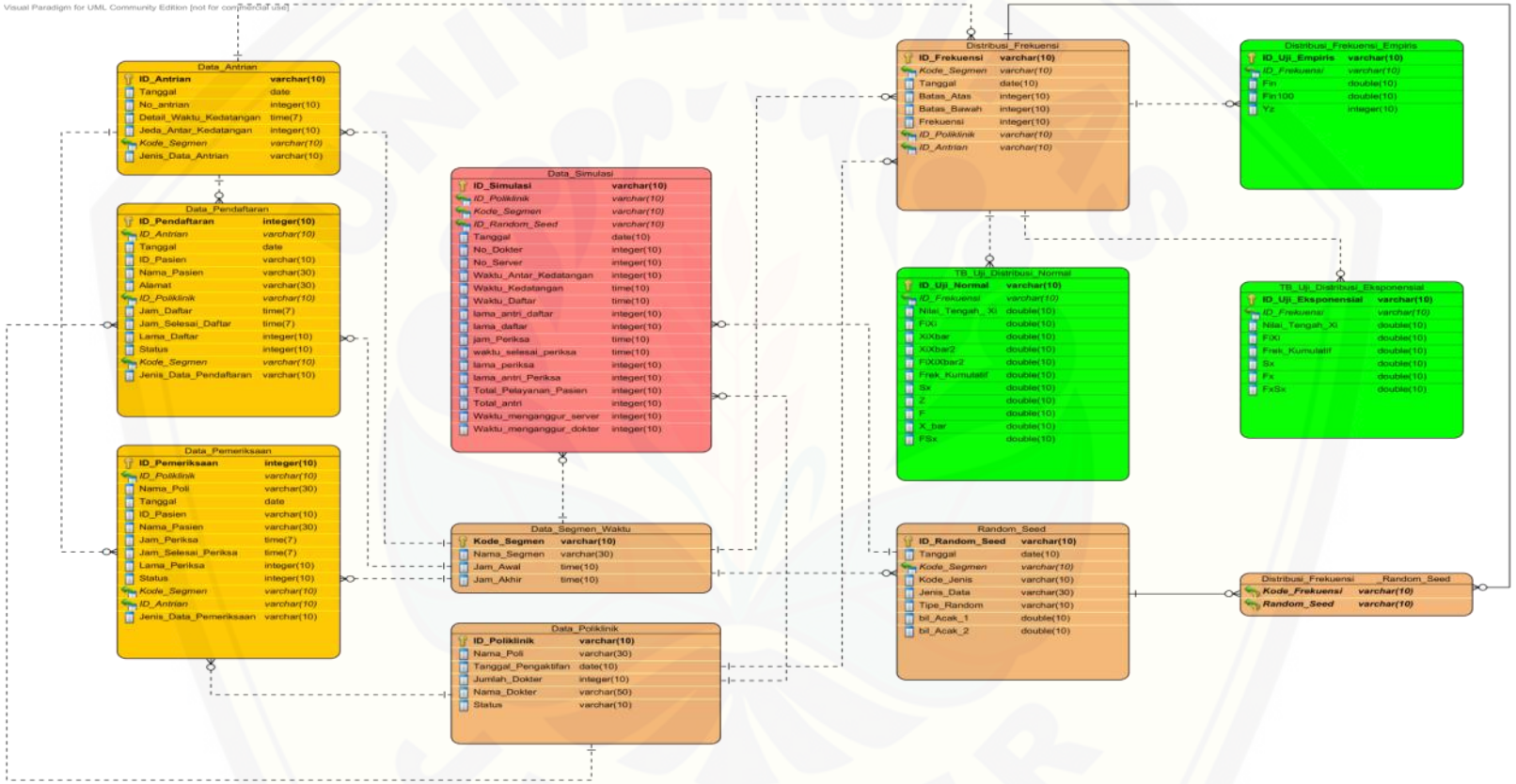






4.3.9 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) pada aplikasi simulator sistem pelayanan Puskesmasmenjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi. ERD aplikasi ditunjukkan pada Gambar 4.12



Gambar 4.12 *Entity Relationship Diagram* (ERD) sistem

4.4 Penulisan Kode Program

Setelah tahap desain perancangan selesai, tahap selanjutnya dalam penelitian ini yaitu tahap pengimplementasian desain perancangan ke dalam bahasa pemrograman. Desain sistem dari semua fitur sistem yang telah dibuat menggunakan pemodelan UML akan diimplementasikan kedalam kode program. Penulisan kode program menggunakan bahasa pemrograman *Java programming* menerapkan konsep teknik pemrogaman OOP *(Object Oriented programming)* MVC dengan menggunakan *database* My-SQL.

4.5 Pengujian Sistem

Pada penelitian ini penulis menggunakan dua metode pengujian sistem yaitu *Black Box Testing* dan *White Box Testing*. Berikut adalah hasil pengujian sistem :

4.5.1 *White Box Testing*

Pengujian *white box* testing terdiri dari listing program, diagram alir, *cyclomatic complexcity*, jalur program independen dan *test case*. Pada tahap ini fitur yang diuji adalah sebagai berikut:

1. Pengujian *White Box Testing* CRUD Segmen Waktu

Pengujian *White Box Testing* CRUD segmen waktu meliputi fitur tambah segmen waktu, edit segmen waktu, dan hapus segmen waktu. Pengujian *White Box Testing* CRUD segmen waktu adalah sebagai berikut :

a. *Listing* program fitur manajemen segmen waktu

```
class ListenerTambahSegmen implements MouseListener{
    @Override
    public void mouseClicked(MouseEvent e) {

        String ID          = vSegmenWaktu.getKodeSegmen();
        String nama        = vSegmenWaktu.getNamaSegmen();
        Object waktuAwal   = vSegmenWaktu.gettWaktuAwal();
        Object waktuAkhir  = vSegmenWaktu.gettWaktuAkhir();

    if(waktuAwal.equals("~Pilih~")||waktuAkhir.equals("~Pilih~")||nama.equals("")){
    JOptionPane.showMessageDialog(null, "Masih ada inputan kosong!","Fail",JOptionPane.ERROR_MESSAGE);
    }
    else {
    mSegmenWaktu.setNilai(ID,nama,waktuAwal,waktuAkhir);
    mSegmenWaktu.tambahSegmen();
    vSegmenWaktu.setKodeSegmen(Integer.toString(mSegmenWaktu.loadKode_Segmen()));
    vSegmenWaktu.reset();
    }
    }
```

Gambar 4.13 Kode Program Tambah Segmen

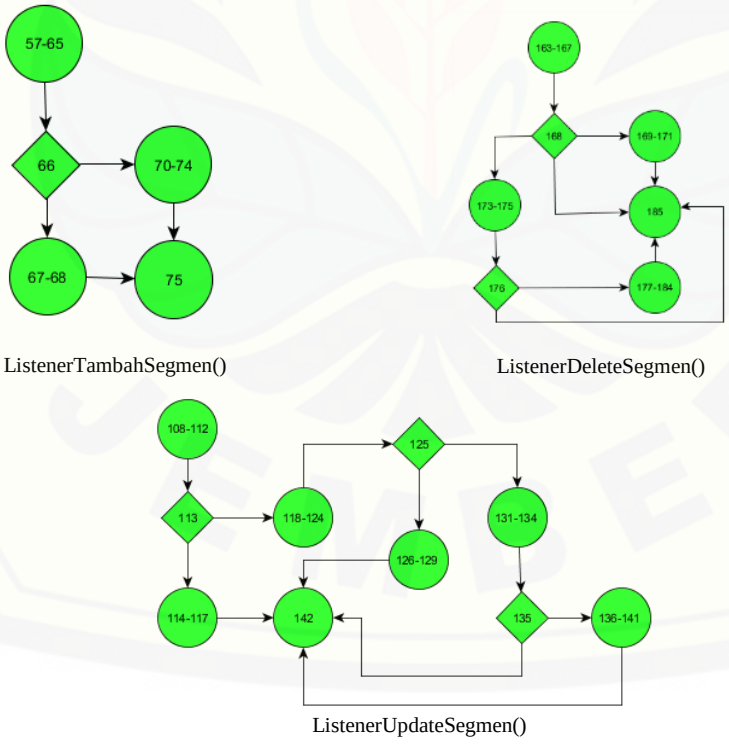
```
class ListenerUpdateSegmen implements MouseListener{

    @Override
    public void mouseClicked(MouseEvent e) {
         int i = vSegmenWaktu.getTabelSegmenSelectedRow();
      if(i == -1){ //tidak ada baris terseleksi
      JOptionPane.showMessageDialog(null,"PILIH BARIS DATA TERLEBIH DAHULU","ERROR",
            JOptionPane.ERROR_MESSAGE);

     }else{

        //ambil id yang terseleksi
         String ID          = (String)mSegmenWaktu.getModel().getValueAt(i, 0);
         String nama        = vSegmenWaktu.getNamaSegmen();
         Object waktuAwal   = vSegmenWaktu.gettWaktuAwal();
         Object waktuAkhir  = vSegmenWaktu.gettWaktuAkhir();

        if(waktuAwal.equals("~Pilih~")||waktuAkhir.equals("~Pilih~")||nama.equals(""))
        {
        JOptionPane.showMessageDialog(null, "Masih ada inputan kosong!","Fail",
            JOptionPane.ERROR_MESSAGE);
        }
        else{

        int l=JOptionPane.showConfirmDialog(null, "Yakin Update Data ?", "Confirm",
            JOptionPane.YES_NO_OPTION);

        if(l==0){

        mSegmenWaktu.setNilai(ID, nama, waktuAwal, waktuAkhir);
        mSegmenWaktu.updateSegmen();
        vSegmenWaktu.setKodeSegmen(Integer.toString(mSegmenWaktu.loadKode_Segmen()));
        vSegmenWaktu.reset();
        JOptionPane.showMessageDialog(null, "Sukses update data","Complete !!",1);
        }}}}
```

Gambar 4.14 Kode Program Ubah Segmen Waktu

```
163  class ListenerDeleteSegmen implements MouseListener{
164
165      @Override
166      public void mouseClicked(MouseEvent e) {
167          int i = vSegmenWaktu.getTabelSegmenSelectedRow();
168      if(i == -1){ //tidak ada baris terseleksi
169           JOptionPane.showMessageDialog(null,"PILIH BARIS DATA TERLEBIH DAHULU","ERROR",
170                  JOptionPane.ERROR_MESSAGE);
171
172      }else{
173
174         int l=JOptionPane.showConfirmDialog(null, "Yakin Delete Data ?", "Confirm",
175                 JOptionPane.YES_NO_OPTION);
176         if(l == 0){
177
178         //ambil id yang terseleksi
179         String ID = (String)mSegmenWaktu.getModel().getValueAt(i, 0);
180         mSegmenWaktu.deleteDataSegmen(ID);
181         vSegmenWaktu.reset();
182         JOptionPane.showMessageDialog(null, "Sukses delete data","Complete !!",1);
183
184         }
185         }}
```

Gambar 4.15 Kode Program Hapus Segmen Waktu

b. Diagram alir fitur manajemen manajemen segmen waktu



Gambar 4.16 Diagram alir fitur manajemen segmen waktu

c. Perhitungan *Cyclomatic Complexity* fitur manajemen segmen waktu

Perhitungan diagram alir pada manajemen data manajemen segmen waktu menggunakan *Cyclomatic Complexity* adalah sebagai berikut:

method ListenerTambahSegmen() : V(G) = E – N + 2 = 5 – 5 + 2 = 2

method ListenerUpdateSegmen () : V(G) = E – N + 2 = 12 – 10 + 2 = 4

method ListenerDeleteSegmen () : V(G) = E – N + 2 = 9 – 7 + 2 = 4

d. Pengujian jalur program fitur manajemen data segmen waktu

Pengujian jalur program fitur manajemen data segmen waktu berdasarkan diagram alir fitur manajemen kriteria :

method ListenerTambahSegmen () : jalur 1 : [57-65] - 66 - [70-74] - 75
jalur 2 : [57-65] - 66 - [67-68] - 75

method ListenerUpdateSegmen () : jalur 1 : [108-112] - 113 - [114-117] - 142
jalur 2 : [108-112] - 113 - [118-124] - 125 - [126-129] - 142
jalur 3 : [108-112] - 113 - [118-124] - 125 - [131-134] - 135 - 142
jalur 4 : [108-112] - 113 - [118-124] - 125 - [131-134] - 135 - [136 - 141] -142

method ListenerDeleteSegmen () : jalur 1 : [163-167] - 168 -185
jalur 2 : [163-167] - 168 - [169-171] -185
jalur 3 : [163-167] - 168 - [173-175] - 176 -185
jalur 4 : [163-167] - 168 - [173-175] - 176 - [177-184] - 185

e. *Test Case* fitur manajemen data segmen waktu

Tabel 4.14 *Test Case* fitur manajemen segmen waktu

Test Case method ListenerTambahSegmen ()	
Jalur 1	
Test Case	Jika Tambah Segmen waktu berhasil
Target yang diharapkan	Menyimpan data kriteria ke *database*
Hasil pengujian	Benar
Path/Jalur	[57-65] - 66 - [70-74] - 75
Jalur 2	
Test Case	Jika Tambah Segmen waktu gagal
Target yang diharapkan	Menampilkan JoptionPane Error Message
Hasil pengujian	Benar
Path/Jalur	[57-65] - 66 - [67-68] - 75
Test Case method ListenerUpdateSegmen ()	
Jalur 4	
Test Case	Jika ubah Segmen waktu berhasil
Target yang diharapkan	Mengubah data Segmen waktu di *database*
Hasil pengujian	Benar
Path/Jalur	[108-112] - 113 - [118-124] - 125 - [131-134] - 135 - [136 - 141] -142
Jalur 2	
Test Case	Jika ubah Segmen waktu data gagal
Target yang diharapkan	Menampilkan JoptionPane Error Message
Hasil pengujian	Benar
Path/Jalur	[108-112] - 113 - [118-124] - 125 - [126-129] - 142
Jalur 3	
Test Case	Jika ubah Segmen waktu data gagal
Target yang diharapkan	Menampilkan JoptionPane Error Message
Hasil pengujian	Benar
Path/Jalur	[108-112] - 113 - [118-124] - 125 - [126-129] - 142
Jalur 1	
Test Case	Jika ubah data gagal
Target yang diharapkan	Menampilkan JoptionPane Error Message
Hasil pengujian	Benar
Path/Jalur	[108-112] - 113 - [114-117] - 142
Test Case method ListenerDeleteSegmen ()	
Jalur 4	
Test Case	Jika hapus Segmen waktu berhasil

Target yang diharapkan	Menghapus data Segmen waktu dari *database*
Hasil pengujian	Benar
Path/Jalur	[163-167] - 168 - [173-175] - 176 -[177-184] - 185
Jalur 1	
Test Case	Jika hapus Segmen waktu gagal
Target yang diharapkan	Menampilkan JoptionPane Error Message
Hasil pengujian	Benar
Path/Jalur	[163-167] - 168 -185
Jalur 2	
Test Case	Jika hapus Segmen waktu gagal
Target yang diharapkan	Menampilkan JoptionPane Error Message
Hasil pengujian	Benar
Path/Jalur	[163-167] - 168 - [169-171] -185
Jalur 3	
Test Case	Jika hapus Segmen waktu gagal
Target yang diharapkan	Menampilkan JoptionPane Error Message
Hasil pengujian	Benar
Path/Jalur	[163-167] - 168 - [173-175] - 176 -185

2. Pengujian *White Box Testing* Simulasi

a. *Listing* program proses Simulasi

```
106  public class listenerActionProsesSimulasi implements  ActionListener{
107
108      @Override
109      public void actionPerformed(ActionEvent e) {
110
111          swaktu      = vSimulasi.getSegmenWaktu();
112          tgl         = vSimulasi.getTanggal();
113          poli        = vSimulasi.getPoli();
114
115        if(vSimulasi.getLoket().equals("") || vSimulasi.getDokter().equals("") ){
116            JOptionPane.showMessageDialog(vSimulasi,"Jumlah loket / dokter masih ada yang "
117                    + "belum di inputkan","Error", JOptionPane.ERROR_MESSAGE);
118        }else{
119
120          int banyak_loket     = Integer.parseInt(vSimulasi.getLoket());
121          int banyak_dokter    = Integer.parseInt(vSimulasi.getDokter());
122
123
124          if(banyak_loket > 4 || banyak_loket < 1 || banyak_dokter > 4 || banyak_dokter < 1){
125
126            JOptionPane.showMessageDialog(vSimulasi,"Isian jumlah loket / dokter di luar "
127                    + "range 1 - 4","Error", JOptionPane.ERROR_MESSAGE);
128          }else{
129
130           //simulasi loket
131          loket               = new String[banyak_loket];
132          status_loket        = new String[banyak_loket];
133          akhir_layanan       = new int[banyak_loket];
134
135          for (int i = 0; i < loket.length; i++) {
136              loket[i]            = Integer.toString(i+1);
137              status_loket[i]     = "aktif";
138              akhir_layanan[i]    = 0;
139           }
```

```
140
141             Object segmen           = mSimulasi.getKode(swaktu);
142             Object id_pengaktifan   = mSimulasi.getID_Pengaktifan(poli);
143              try {
144                  mSimulasi.updateDataLoket(tgl, segmen, id_pengaktifan, loket, status_loket, akhir_layanan);
145                  mSimulasi.update_menganggurLoket(banyak_loket,tgl, segmen, id_pengaktifan);
146                  mSimulasi.TampilSimulasi(segmen, tgl, id_pengaktifan,loket.length,0);
147              } catch (SQLException ex) {
148                  Logger.getLogger(Controll_Simulasi.class.getName()).log(Level.SEVERE, null, ex);
149              }
150
151             //simulasi dokter
152             dokter= new String[banyak_dokter];
153             status_dokter= new String[banyak_dokter];
154             akhir_layanan_periksa= new int[banyak_dokter];
155
156              for (int i = 0; i < dokter.length; i++) {
157                  dokter[i]                   = Integer.toString(i+1);
158                  status_dokter[i]            = "aktif";
159                  akhir_layanan_periksa[i]    = 0;
160              }
161
162              try {
163                  mSimulasi.updateDataDokter(tgl, segmen, id_pengaktifan, dokter, status_dokter, akhir_layanan_periksa);
164                  mSimulasi.update_menganggurDokter(banyak_dokter,tgl, segmen, id_pengaktifan);
165                  mSimulasi.TampilSimulasi(segmen, tgl, id_pengaktifan, loket.length,dokter.length);
166              } catch (SQLException ex) {
167                  Logger.getLogger(Controll_Simulasi.class.getName()).log(Level.SEVERE, null, ex);
168              }
169
170
171              for (int i = 0; i < loket.length; i++) {
172                  System.out.println("utilitas loket "+loket[i]+" = "+mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[i])
173
174              }
```

```
175
176             for (int i = 0; i < dokter.length; i++) {
177                 System.out.println("utilitas dokter "+dokter[i]+" = "+mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,
178                         dokter[i])+" %");
179             }
180 //              vSimulasi.setutilitas_loket(mSimulasi.utilitas_loket(tgl, segmen, id_pengaktifan)+" %");
181             vSimulasi.setutilitas_dokter(mSimulasi.utilitas_dokter(tgl, segmen, id_pengaktifan)+" %");
182
183             if(loket.length == 1){
184
185             vSimulasi.setutilitas_loket(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0])+"%");
186             vSimulasi.setutilitas_loket2("   - ");
187             vSimulasi.setutilitas_loket3("   - ");
188             vSimulasi.setutilitas_loket4("   - ");
189
190             vSimulasi.setTotutilitas_loket(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0])+"%");
191
192             }
193
194
195             if(loket.length == 2){
196
197             vSimulasi.setutilitas_loket(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0])+"%");
198             vSimulasi.setutilitas_loket2(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[1])+"%");
199             vSimulasi.setutilitas_loket3("   - ");
200             vSimulasi.setutilitas_loket4("   - ");
201
202             double totLoket1 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0]));
203             double totLoket2 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[1]));
204
205             double totUtilitasLoket = (totLoket1+totLoket2)/loket.length;
206
207             vSimulasi.setTotutilitas_loket(totUtilitasLoket+"%");
208
209             }
```

```
211             if(loket.length == 3){
212
213             vSimulasi.setutilitas_loket(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0])+"%");
214             vSimulasi.setutilitas_loket2(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[1])+"%");
215             vSimulasi.setutilitas_loket3(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[2])+"%");
216             vSimulasi.setutilitas_loket4("   -  ");
217
218             double totLoket1 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0]));
219             double totLoket2 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[1]));
220             double totLoket3 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[2]));
221
222             double totUtilitasLoket = (totLoket1+totLoket2+totLoket3)/loket.length;
223
224             DecimalFormat formatter  = new DecimalFormat(".##");
225
226             String b = formatter.format(totUtilitasLoket);
227             String a = b.replace(',', '.');
228
229             vSimulasi.setTotutilitas_loket(""+a+" %");
230
231             }
232
233             if(loket.length == 4){
234             vSimulasi.setutilitas_loket(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0])+"%");
235             vSimulasi.setutilitas_loket2(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[1])+"%");
236             vSimulasi.setutilitas_loket3(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[2])+"%");
237             vSimulasi.setutilitas_loket4(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[3])+"%");
238
239             double totLoket1 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[0]));
240             double totLoket2 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[1]));
241             double totLoket3 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[2]));
242             double totLoket4 = Double.parseDouble(mSimulasi.utilitas_perloket(tgl, segmen, id_pengaktifan,loket[3]));
243
244             double totUtilitasLoket = (totLoket1+totLoket2+totLoket3+totLoket4)/loket.length;
245             vSimulasi.setTotutilitas_loket(totUtilitasLoket+"%");
```

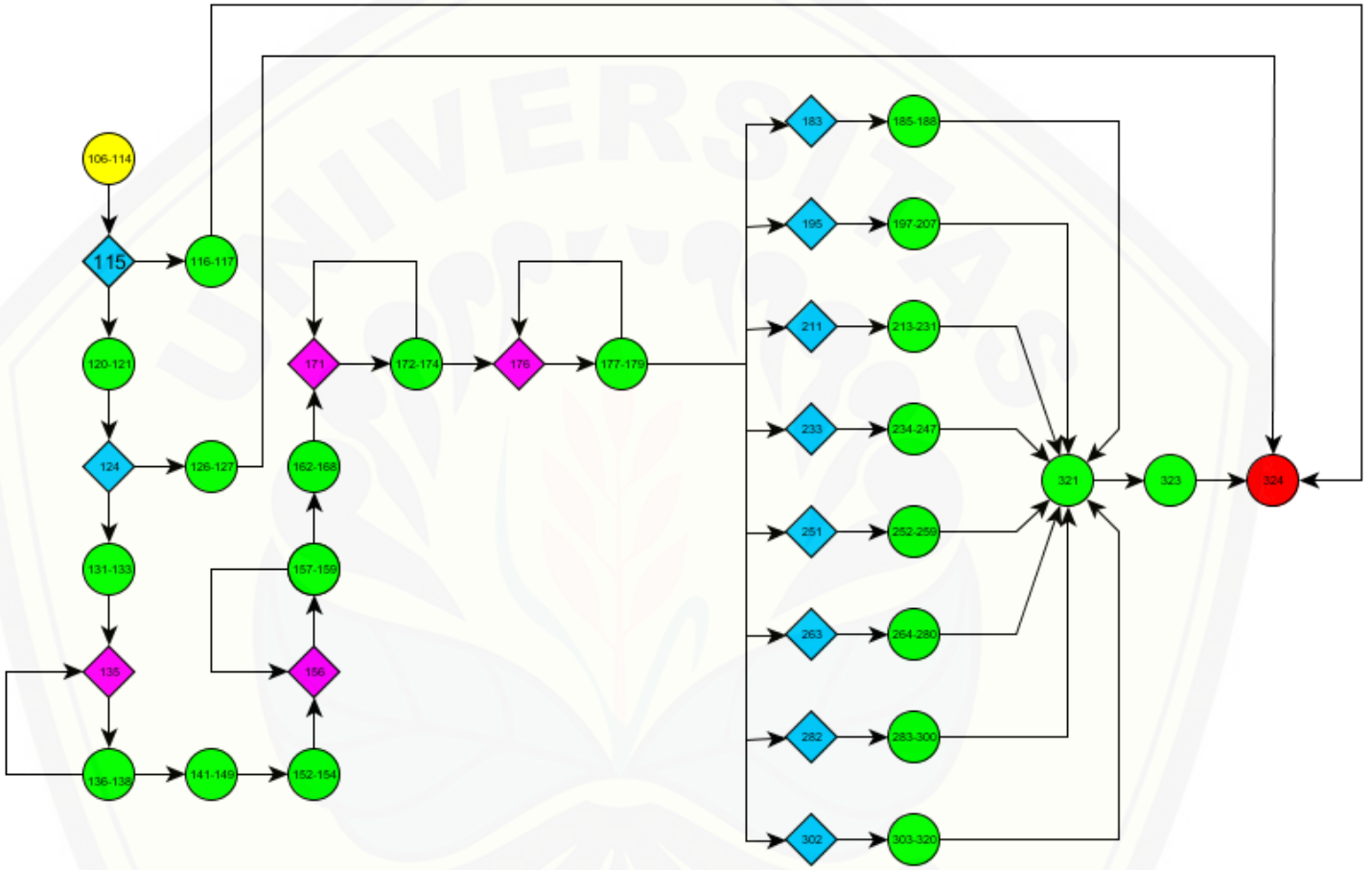
```
246
247             }
248
249             //dokter
250
251             if(dokter.length == 1){
252             vSimulasi.setutilitas_dokter(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0])+"%");
253             vSimulasi.setutilitas_dokter2("    -  ");
254             vSimulasi.setutilitas_dokter3("    -  ");
255             vSimulasi.setutilitas_dokter4("    -  ");
256
257             vSimulasi.setTotutilitas_Dokter(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0])+"%");
258
259             }
260
261             DecimalFormat formatter  = new DecimalFormat(".##");
262
263             if(dokter.length == 2){
264             vSimulasi.setutilitas_dokter(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0])+"%");
265             vSimulasi.setutilitas_dokter2(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[1])+"%");
266             vSimulasi.setutilitas_dokter3("    -  ");
267             vSimulasi.setutilitas_dokter4("    -  ");
268
269             double totDokter1 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0]));
270             double totDokter2 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[1]));
271
272             double totUtilitasDokter = (totDokter1+totDokter2)/ dokter.length;
273
274             double b = totUtilitasDokter;
275             String ba = formatter.format(b);
276             String a = ba.replace(',', '.');
277
278             vSimulasi.setTotutilitas_Dokter(""+a+"%");
279
280             }
```

```
282             if(dokter.length == 3){
283             vSimulasi.setutilitas_dokter(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0])+"%");
284             vSimulasi.setutilitas_dokter2(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[1])+"%");
285             vSimulasi.setutilitas_dokter3(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[2])+"%");
286             vSimulasi.setutilitas_dokter4("   -  ");
287
288             double totDokter1 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0]));
289             double totDokter2 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[1]));
290             double totDokter3 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[2]));
291
292             double totUtilitasDokter = (totDokter1+totDokter2+totDokter3)/ dokter.length;
293
294             double b = totUtilitasDokter;
295             String ba = formatter.format(b);
296             String a = ba.replace(',', '.');
297
298             vSimulasi.setTotutilitas_Dokter(""+a+"%");
299
300             }
301
302             if(dokter.length == 4){
303             vSimulasi.setutilitas_dokter(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0])+"%");
304             vSimulasi.setutilitas_dokter2(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[1])+"%");
305             vSimulasi.setutilitas_dokter3(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[2])+"%");
306             vSimulasi.setutilitas_dokter4(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[3])+"%");
307
308             double totDokter1 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[0]));
309             double totDokter2 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[1]));
310             double totDokter3 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[2]));
311             double totDokter4 = Double.parseDouble(mSimulasi.utilitas_perdokter(tgl, segmen, id_pengaktifan,dokter[3]));
312
313             double totUtilitasDokter = (totDokter1+totDokter2+totDokter3+totDokter4)/ dokter.length;
314
315             double b = totUtilitasDokter;
316             String ba = formatter.format(b);
```

```
317             String a = ba.replace(',', '.');
318
319             vSimulasi.setTotutilitas_Dokter(""+a+"%");
320             }
321            }}
322
323         }
324         }
```

Gambar 4.17 Kode Program Proses Simulasi

b. Diagram alir Proses Simulasi



Gambar 4.18 Diagram alir listenerActionProsesSimulasi()

c. Perhitungan *Cyclomatic Complexity* proses simulasi

Perhitungan diagram alir pada proses simulasi menggunakan *Cyclomatic Complexity* berdasarkan diagram alir method listenerActionProsesSimulasi() pada Gambar 4.18 adalah sebagai berikut:

Method listenerActionProsesSimulasi () :

$$V(G) = E – N + 2 = 48 – 37 + 2 = 13$$

d. Pengujian jalur program fitur simulasi

Pengujian jalur program simulasi berdasarkan diagram alir simulasi padaGambar 4.18 adalah sebagai berikut :

Method listenerActionProsesSimulasi () :

jalur 1 : [106-114] - 115 - [120-121] -[131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 183 - [185-188] - 321 - 323 - 324

jalur 2 : [106-114] - 115 - [120-121] -[131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 195 - [197-207] - 321 - 323 - 324

jalur 3 : [106-114] - 115 - [120-121] -[131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 211 - [213-231] - 321 - 323 - 324

jalur 4 : [106-114] - 115 - [120-121] -[131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 233 - [234-247] - 321 - 323 - 324

jalur 5 : [106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 251 - [252-259] - 321 - 323 - 324

jalur 6 : [106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 263 - [264-280] - 321 - 323 - 324

jalur 7 : [106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 282 - [283-300] - 321 - 323 - 324

jalur 8 : [106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 302 - [303-320] - 321 - 323 - 324

jalur 9 : [106-114] - 115 - [116-117] - 324

jalur 10 : [106-114] - 115 - [120-121] - 124 - [126-127] - 324

jalur 11 : [106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - 156

jalur 12 : [106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [177-179] -171 - [177-179]

jalur 13 : [106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - 135 - [136-138]

e. *Test Case* proses Simulasi

Tabel 4.15 *Test Case* proses simulasi

Test Case method listenerActionProsesSimulasi ()	
Jalur 1	
Test Case	Jika jumlah loket = 1
Target yang diharapkan	Menampilkan utilitas Loket 1
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] -[131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 183 - [185-188] - 321 - 323 - 324
Jalur 2	
Test Case	Jika jumlah loket = 2
Target yang diharapkan	Menampilkan utilitas Loket 1, utilitas Loket2
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] -[131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 195 - [197-207] - 321 - 323 - 324
Jalur 3	
Test Case	Jika jumlah loket = 3
Target yang diharapkan	Menampilkan utilitas Loket 1, utilitas Loket2, Loket3
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] -[131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 211 - [213-231] - 321 - 323 - 324
Jalur 4	
Test Case	Jika jumlah loket = 4
Target yang diharapkan	Menampilkan utilitas Loket 1, utilitas Loket2, Loket3,Loket4
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] -[131-133] - 135 -

	[136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 233 - [234-247] - 321 - 323 - 324
Jalur 5	
Test Case	Jika jumlah dokter = 1
Target yang diharapkan	Menampilkan utilitas Dokter1
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 251 - [252-259] - 321 - 323 - 324
Jalur 6	
Test Case	Jika jumlah dokter = 2
Target yang diharapkan	Menampilkan utilitas Dokter1, Dokter2
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 263 - [264-280] - 321 - 323 - 324
Jalur 7	
Test Case	Jika jumlah dokter = 3
Target yang diharapkan	Menampilkan utilitas Dokter1, Dokter2,Dokter3
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 282 - [283-300] - 321 - 323 - 324
Jalur 8	
Test Case	Jika jumlah dokter = 4
Target yang diharapkan	Menampilkan utilitas Dokter1, Dokter2,Dokter3,Dokter4
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [131-133] -

	135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [172-174] -176 - [177-179] - 302 - [303-320] - 321 - 323 - 324
Jalur 9	
Test Case	Jika tidak mengisi jumlah loket atau dokter
Target yang diharapkan	Menampilkan error message
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [116-117] - 324
Jalur 10	
Test Case	Jika mengisi jumlah loket atau dokter diluar range atau tidak sesuai format
Target yang diharapkan	Menampilkan error message
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [126-127] - 324
Jalur 11	
Test Case	Jika set jumlah dokter aktif sesuai yang diinputkan
Target yang diharapkan	Jumlah Dokter aktif = dokter.length
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - 156
Jalur 12	
Test Case	Jika set utilitas perloket sesuai yang diinputkan
Target yang diharapkan	Jumlah Utilitas loket = loket.length
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - [141-149] - [152-154] - 156 - [157-159] - [162-168] - 171 - [177-179] -171 - [177-179]
Jalur 13	
Test Case	Jika set jumlah loket aktif sesuai yang diinputkan
Target yang diharapkan	Jumlah loket aktif = loket.length
Hasil pengujian	Benar
Path/Jalur	[106-114] - 115 - [120-121] - 124 - [131-133] - 135 - [136-138] - 135 - [136-138]

4.5.2 *Black Box Testing*

Pengujian *black box* menitik beratkan pada fungsionalitas sistem. Pengujian ini tidak melihat kinerja internal dari sistem, jadi hanya berfokus pada kinerja sistem sesuai dengan spesifikasi dan kebututuhan yang dianalisis pada bab perancangan. Hasil pengujian *black box* dapat dilihat pada Tabel 4.16.

Tabel 4.16 Pengujian *Black Box* fitur manajemen segmen waktu

No	Fitur	Kasus	Hasil	Keterangan
1.	Tambah Segmen waktu	• Ketika klik tombol tambah dan semua *field* sudah terisi dengan benar • Ketika klik tombol tambah dan belum mengisi secara lengkap / *field* masih ada yang kosong • Ketika klik tombol tambah dan mengisi *field* tidak sesuai *format* isian dari *field*	• Menambah data segmen waktu sesuai yang diinputkan • Menampilkan peringatan *error message* • Menampilkan peringatan *error message*	[√] Berhasil [ ] Gagal
2.	Ubah Segmen waktu	• Ketika klik tombol edit dan semua *field* sudah terisi dengan benar • Ketika klik tombol edit dan belum mengisi secara lengkap / *field* masih ada yang kosong • Ketika klik tombol edit dan mengisi *field* tidak sesuai *format* isisan dari *field* • Ketika klik tombol edit	• *Update* data segmen waktu sesuai yang diinputkan • Menampilkan peringatan *error message* • Menampilkan peringatan *error message* • Menampilkan peringatan *error*	[√] Berhasil [ ] Gagal

		dan belum memilih row data yang akan diedit	*message*	
3.	Hapus Segmen waktu	• Ketika klik tombol delete dan sudah memilih row data yang akan didelete • Ketika klik tombol delete dan belum memilih row data yang akan didelete	• *Delete* data segmen waktu sesuai yang dipilih • Menampilkan peringatan *error message*	[√] Berhasil [ ] Gagal

Tabel 4.16 merupakan hasil pengujian *Black Box Testing* pada fitur manajemen segmen waktu. Hasil pengujian menunjukkan bahwa hasil yang sesuai dengan yang dihapakan oleh pengguna.

Tabel 4.17 Pengujian *Black Box* fitur uji distribusi

No	Fitur	Kasus	Hasil	Keterangan
1.	Set Persiapan Data Uji Disribusi Frekuensi	• Ketika klik tombol set data dan semua *field combo box* telah terpilih. • Ketika klik tombol set data dan ada *field combo box* yang belum dipilih. • Ketika klik tombol set data dan semua *field combo box* telah terpilih tetapi data belum ada.	• Menampilkan data persiapan data sesuai variabel data yang dipilih sebelumnya dan menampilkan data hasil perhitungan distribusi frekuensi. • Menampilkan peringatan *error message* semua event harus dipilih. • Menampilkan	[√] Berhasil [ ] Gagal

			peringatan *error message* persiapan data masih kosong	
2.	Uji Distribusi Normal	• Ketika klik tombol uji normal dan telah selesai menjalankan uji distribusi frekuensi • Ketika klik tombol uji normal dan belum selesai menjalankan uji distribusi frekuensi	• Menampilkan *frame* hasil perhitungan uji normal • Menampilkan peringatan *error message*	[√] Berhasil [ ] Gagal
3.	Uji Distribusi Ekponensial	• Ketika klik tombol uji eksponensial dan telah selesai menjalankan uji distribusi normal • Ketika klik tombol uji eksponensial dan belum selesai menjalankan uji distribusi normal	• Menampilkan *frame* hasil perhitungan uji eksponensial • Menampilkan peringatan *error message* pilih uji normal terlebih dahulu	[√] Berhasil [ ] Gagal
3.	Uji Distribusi Empiris	• Ketika klik tombol uji Empiris dan telah selesai menjalankan uji distribusi Empiris • Ketika klik tombol uji Empiris dan belum selesai menjalankan uji distribusi eksponensial	• Menampilkan *frame* hasil perhitungan uji Empiris • Menampilkan peringatan *error message* pilih uji ekponensial terlebih dahulu	[√] Berhasil [ ] Gagal

Tabel 4.17 merupakan hasil pengujian *Black Box Testing* pada fitur uji distribusi. Hasil pengujian menunjukkan bahwa hasil yang sesuai dengan yang dihapakan oleh pengguna.

Tabel 4.18 Pengujian *Black Box* fitur *generate random seed*

No	Fitur	Kasus	Hasil	Keterangan
1.	*Generate random seed* data normal	• Ketika klik tombol *generate* dan hasil uji distribusi normal diterima	• Menampilkan *data random seed* pada tabel bilangan acak	[√] Berhasil [ ] Gagal
2.	*Generate random seed* data *eksponensial*	• Ketika klik tombol *generate* dan hasil uji distribusi *eksponensial* diterima	• Menampilkan *data random seed* pada tabel bilangan acak	[√] Berhasil [ ] Gagal
3.	*Generate random seed* data *empiris*	• Ketika klik tombol *generate* dan hasil uji distribusi *empiris* diterima	• Menampilkan *data random seed* pada tabel bilangan acak	[√] Berhasil [ ] Gagal

Tabel 4.18 merupakan hasil pengujian *Black Box Testing* pada fitur *generate random seed*. Hasil pengujian menunjukkan bahwa hasil yang sesuai dengan yang dihapakan oleh pengguna.

Tabel 4.19 Pengujian *Black Box* fitur simulasi

No	Fitur	Kasus	Hasil	Keterangan
1.	Set Data	• Ketika klik tombol set	• Menambah data	

	Simulasi	data dan telah memilih semua *field* data serta data telah tersedia didatabase • Ketika klik tombol set data dan belum memilih secara lengkap *field* yang tersedia • Ketika klik tombol set data dan data persiapan simulasi belum tersedia di *database*	persiapan simulasi kedalam tabel simulasi dan menampikan sifat data beserta *enabled field* input simulasi dan tombol proses • Menampilkan peringatan *error message* • Menampilkan peringatan *error message*	[√] Berhasil [ ] Gagal
2.	Proses Simulasi	• Ketika klik tombol proses dan sudah mengisi *field data* secara lengkap dan benar/sesuai format • Ketika klik tombol proses dan belum mengisi *field data* secara lengkap dan benar/sesuai format • Ketika klik tombol proses dan belum melakukan uji distribusi data	• Memulai perhitungan simulasi dan menyimpan hasilnya kedalam database • Menampilkan peringatan *error message* • Menampilkan peringatan *error message*	[√] Berhasil [ ] Gagal
3.	Menampilkan hasil detail simulasi	• Ketika klik detail hasil simulasi dan sudah melakukan proses simulasi	• Menampilkan form informasi hasil kesimpulan simulasi	[√] Berhasil [ ] Gagal

	• Ketika klik detail hasil simulasi dan belum melakukan proses simulasi	• Menampilkan *error message*	

Tabel 4.19 merupakan hasil pengujian *Black Box Testing* pada fitur simulasi. Hasil pengujian menunjukkan bahwa hasil yang sesuai dengan yang dihapakan oleh pengguna.

BAB 5. HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil pembuatan sistem dan juga pembahasannya. Penelitian ini menghasilkan sebuah simulator sistem pelayanan puskesmas yang digunakan untuk mensimulasikan pelayanan yang ada dipuskesmas Gambiran. Pembahasan bertujuan untuk menjelaskan bagaimana penelitian ini menjawab perumusan masalah serta tujuan dan manfaat dari aplikasi simulator sistem pelayanan puskesmas ini.

5.1 Simulator Sistem Pelayanan Puskesmas

Simulator sistem pelayanan puskesmas memiliki tiga hak akses yaitu manajer sebagai admin, operator pendaftaran, dan operator pemeriksaan. Bagian manajer memiliki beberapa fitur utama yaitu antara lain fitur manajemen (*input,update,delete*) master data yang berupa data *user*, data poli dan data segmen waktu, fiur uji distribusi data yang berupa uji distribusi frekuensi, uji distribusi normal, uji distribusi *eksponensial* dan uji distribusi *empiris,* fitur *generate random seed* dan fitur simulasi. Bagian operator pendaftaran memiliki fitur utama yaitu *input* antrian dan manajemen (*input, update*) data pendaftaran. Sedangkan bagian operator pemeriksaan memiliki fitur utama yang berupa *input* data pemeriksaan pasien.

5.2 Hasil Implementasi Simulator Sistem Pelayanan Puskesmas

Hasil implementasi simulator sistem pelayanan puskesmas yang dibangun pada penelitian ini terdiri atas beberapa fitur yang dapat diakses oleh user (manajer,operator pendaftaran dan operator pemeriksaan). Sistem ini dapat memudahkan manajer untuk mensimulasikan pelayanan puskesmas yang nantinya menghasilkan kesimpulan simulasi yang berupa nilai utilitas pelayanan dan informasi-informasi lainya. Simulator sistem ini memiliki beberapa fitur yang antara lain :

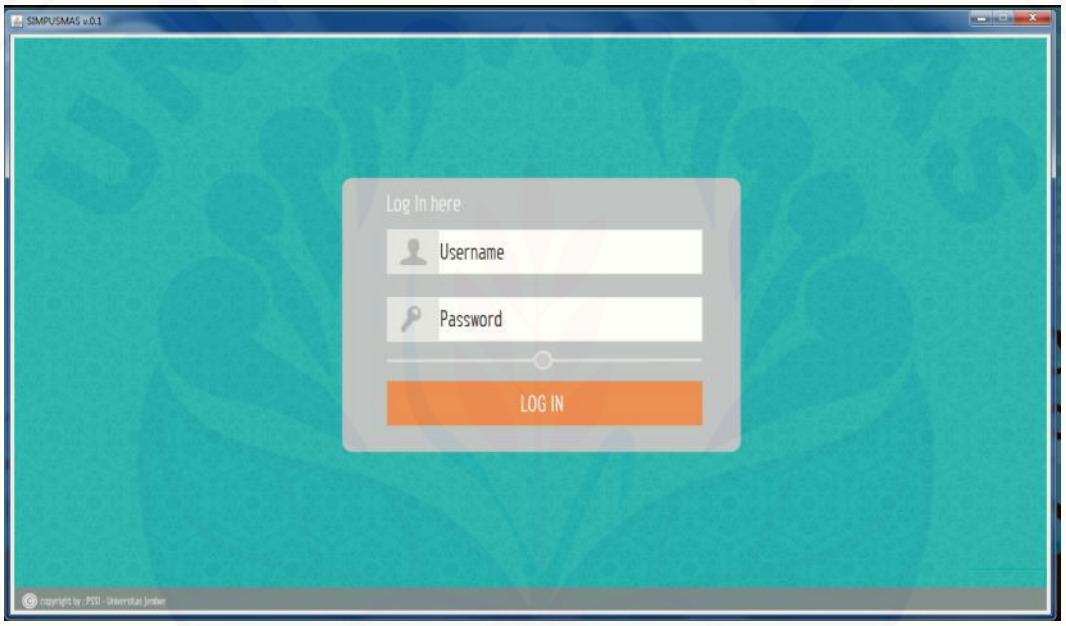
5.2.1 Tampilan *Splash Screen*

Gambar 5.1 merupakan tampilan awal saat aplikasi simulator sistem pelayanan puskesmas mulai dijalankan. Pada saat proses aktivitas *splash screen* dijalankan, dilakukan pengkoneksian data kedalam *database* sistem. Setelah *splash screen* selesai dijalankan maka akan ditampilkan tampilan *login* seperti pada Gambar 5.2

Gambar 5.1 Tampilan *Splash Screen*

5.2.2 Tampilan *login*

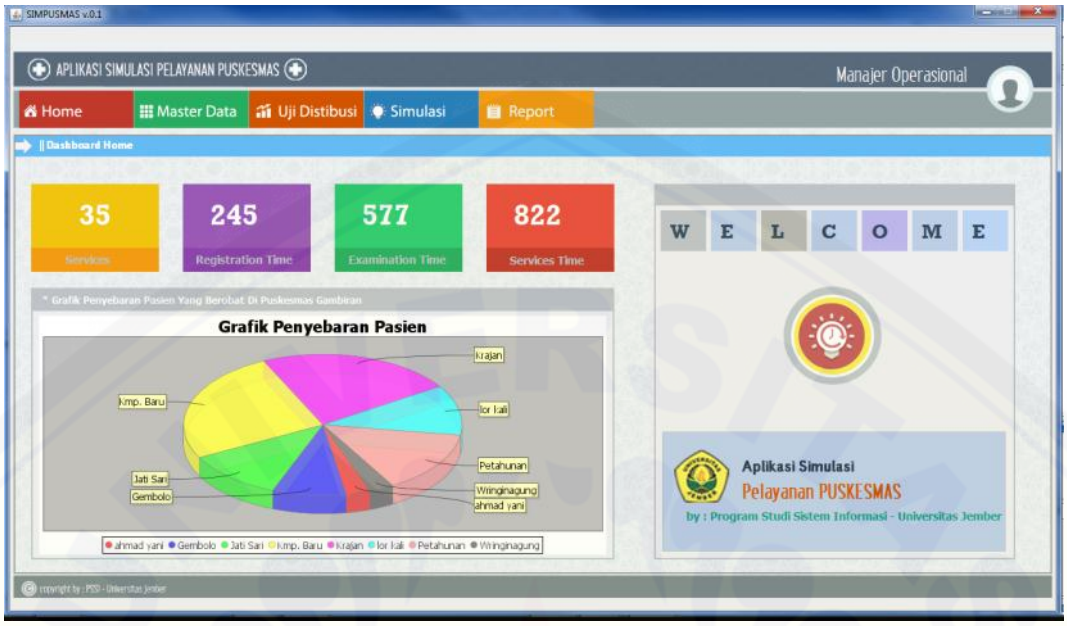
Setelah *user* mulai menjalankan aplikasi simulator sistem pelayanan puskesmas dan tampilan *splash screen* menghilang maka akan ditampilkan tampilan *form login* seperti pada Gambar 5.2 dibawah. Pada *form login* ada dua variabel *inputan* yaitu *username* dan *password* serta satu tombol yaitu tombol *login* yang berfungsi sebagai cek *username* dan *password* sesuai yang ada pada *database* atau tidak . Login ini berguna sebagai hak akses *user*.



Gambar 5.2 Tampilan *login*

5.2.3 Tampilan Menu *Dashboard home*

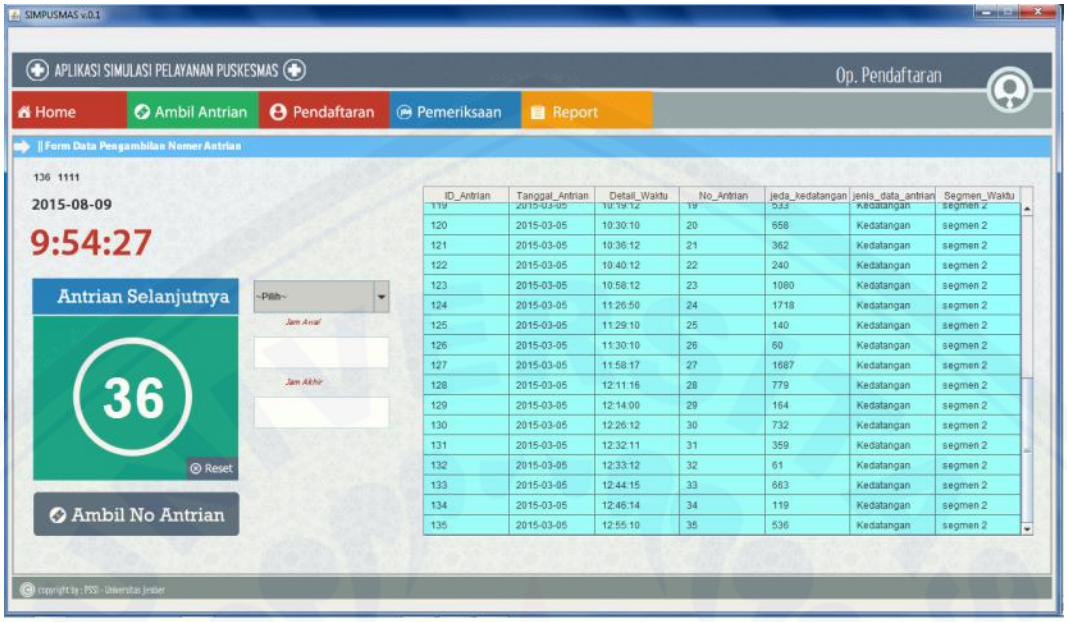
Setelah *user* berhasil *login* maka selanjutnya user akan masuk kedalam tampilan menu *dashboard home* seperti pada Gambar 5.3 Tampilan *Dashboard home.* Tampilan ini merupakan tampilan awal sebelum *user* mengakses fitur - fitur lainya. Pada *dashboard home* terdapat informasi yang berupa grafik *piechart* penyebaran pasien yang berobat di Puskesmas Gambiran.



Gambar 5.3 Tampilan *Dashboard home*

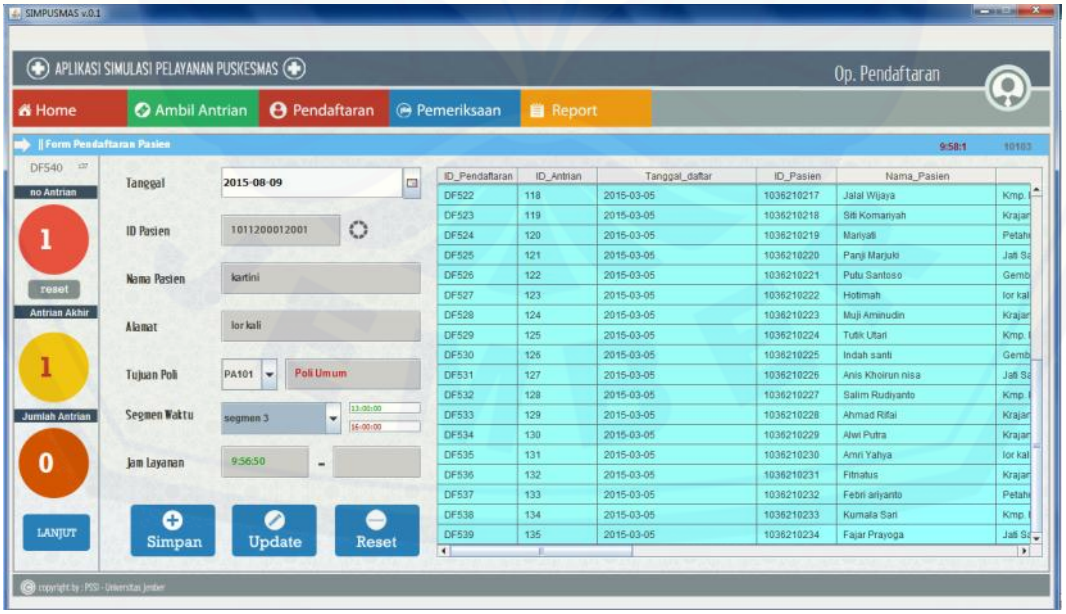
5.2.4 Tampilan Menu Ambil Antrian

Fungsi utama dari fitur ambil antrian ini adalah mencatat data kedatangan antrian pasien yang akan memanfaatkan fasilitas pelayanan di Puskesmas. Data antrian disimpan didalam *database* sistem yang nantinya akan digunakan dalam proses simulasi.



Gambar 5.4 Tampilan Menu Ambil Antrian

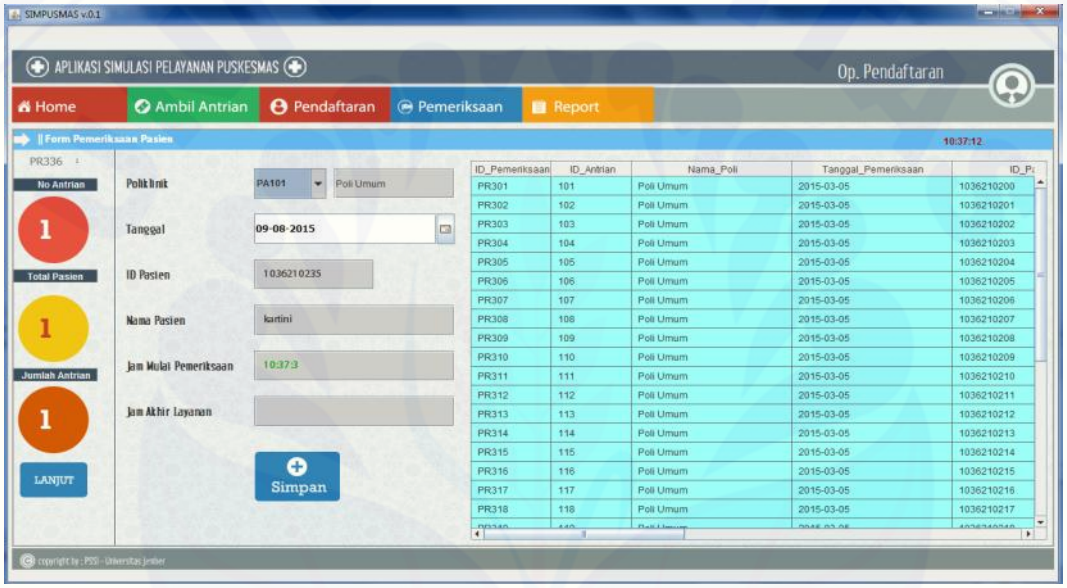
Ketika *user* Operator pendaftaran memilih menu pendaftaran maka akan ditampilkan tampilan *form* pendaftaran pasien seperti pada Gambar 5.5. Fungsi utama dari fitur pendaftaran adalah mencatat informasi data dari pasien mulai dari identitas hingga waktu daftar sampai waktu selesai daftar.



Gambar 5.5 Tampilan *form* Pendaftaran pasien

5.2.6 Tampilan Menu Pemeriksaan

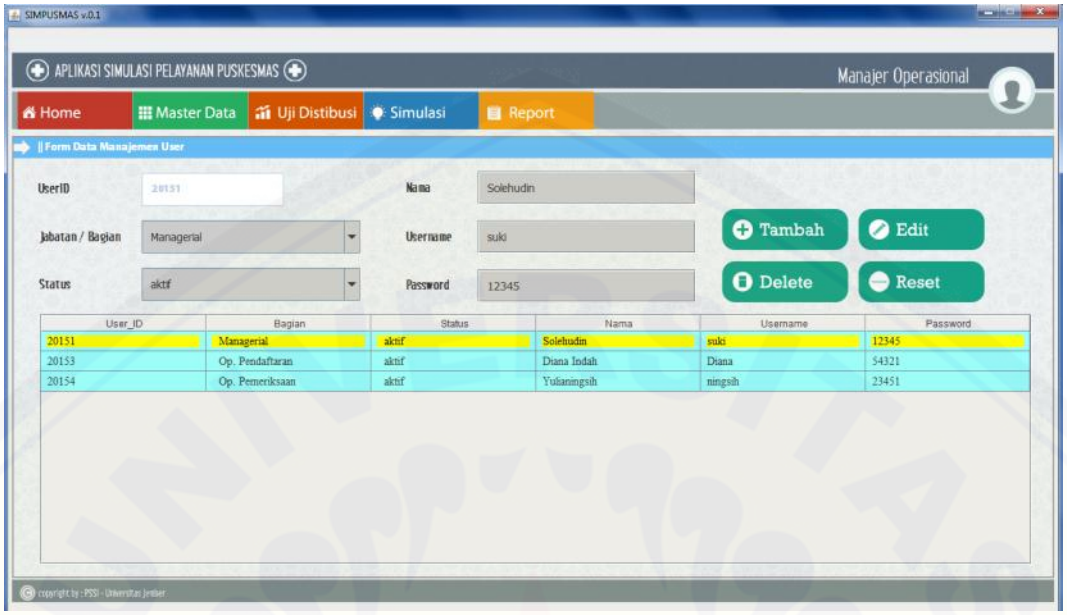
Setelah *user* berhasil *login* dan jika *user* tersebut adalah operator pemeriksaan maka selanjutnya user tersebut akan masuk kedalam tampilan menu pemeriksaan seperti pada gambar Gambar 5.6. Fungsi utama dari fitur pemeriksaan ini adalah mencatat waktu lama periksa pasien di Puskesmas. Data pemeriksaan disimpan didalam *database* sistem yang nantinya akan digunakan dalam proses simulasi.



Gambar 5.6 Tampilan *form* Pemerikasaan

5.2.7 Tampilan Menu Master Data *User*

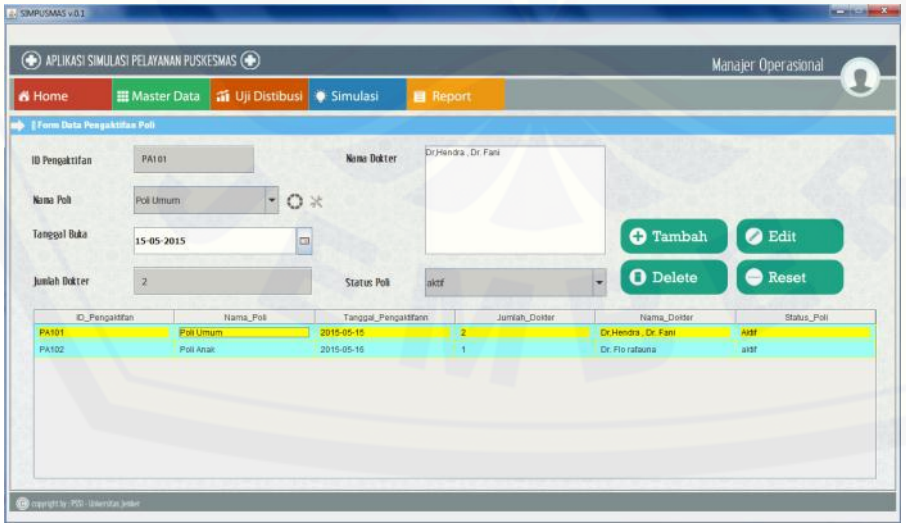
Tampilan master data *user* hanya dapat diakses jika *user* yang login adalah manajer. Pada master data *user* ini manajer dapat melakukan *input user* baru, *update* data *user* dan menghapus data *user* yang dapat menggunakan atau memiliki hak akses terhadap sistem. Tampilan *form* master data *user* dapat dilihat pada Gambar 5.7



Gambar 5.7 Tampilan *form* master data *user*

5.2.8 Tampilan Menu Master Data Poli

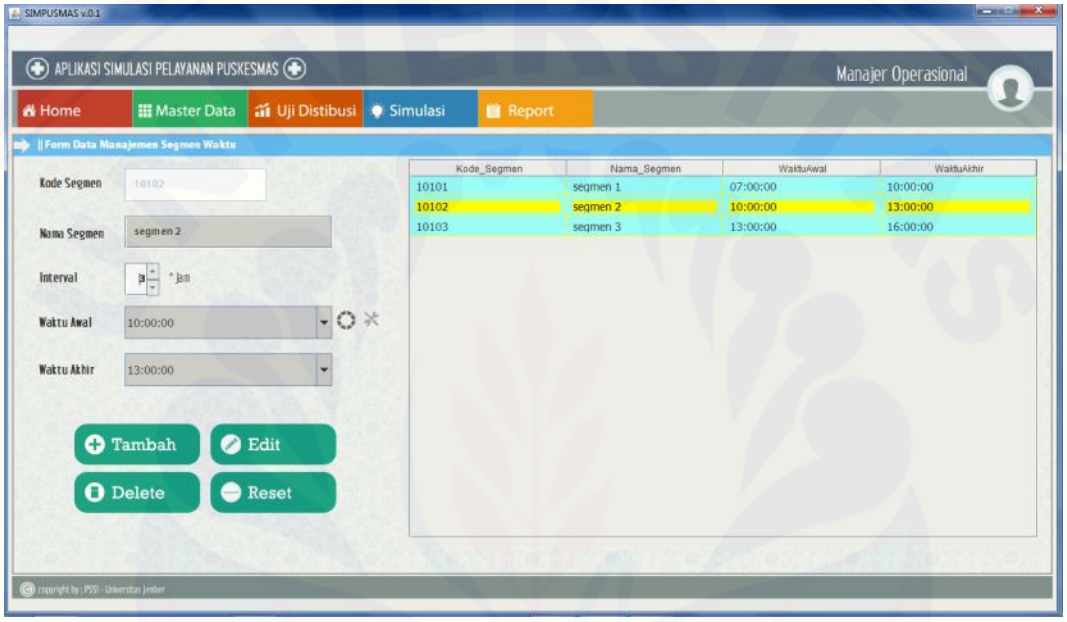
Tampilan master data poli hanya dapat diakses jika *user* yang login adalah manajer. Pada master data poli ini manajer dapat melakukan pengaktifan poli baru, *update* data poli dan menghapus data poli. Tampilan *form* master data poli dapat dilihat pada Gambar 5.8



Gambar 5.8 Tampilan *form* data poli

5.2.9 Tampilan Menu Master Data Segmen Waktu

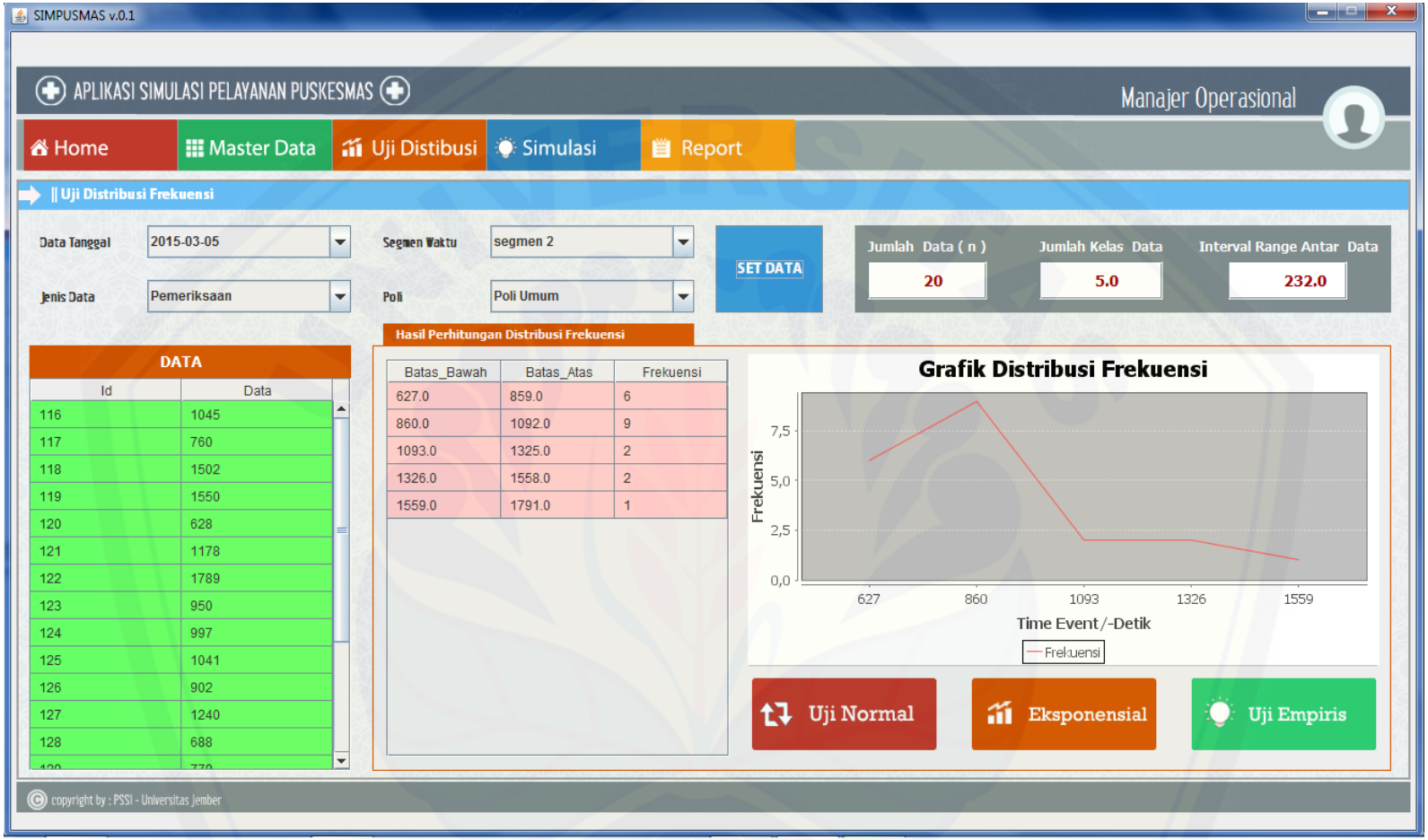
Tampilan master data segmen waktu hanya dapat diakses jika *user* yang login adalah manajer. Pada master data segmen waktu ini manajer dapat melakukan *input* segmen waktu, *update* data segmen waktu dan menghapus data segmen waktu. Tampilan *form* master data segmen waktu dapat dilihat pada Gambar 5.9.



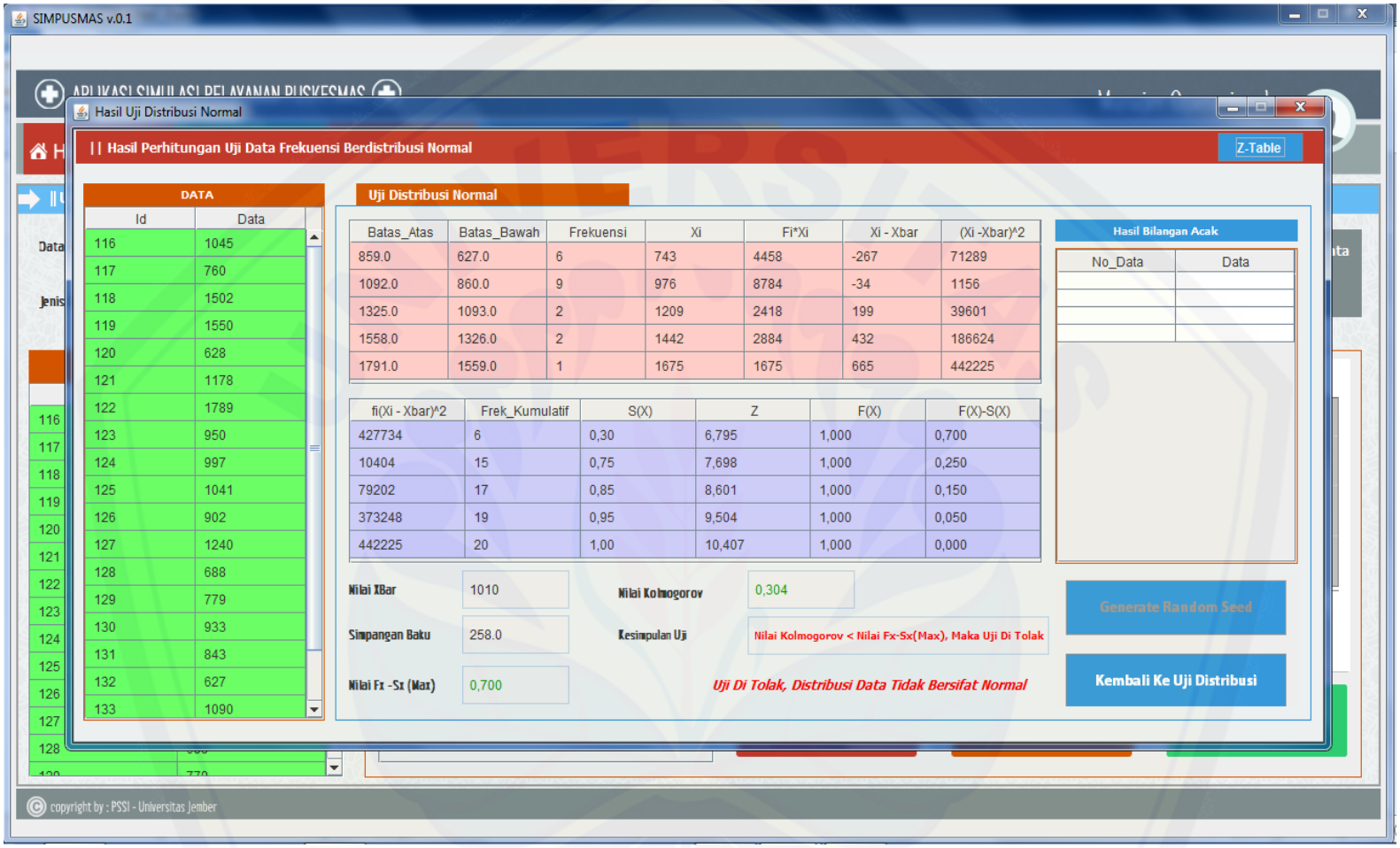
Gambar 5.9 Tampilan *form* data segmen waktu

5.2.10 Tampilan Menu Uji Distribusi Data

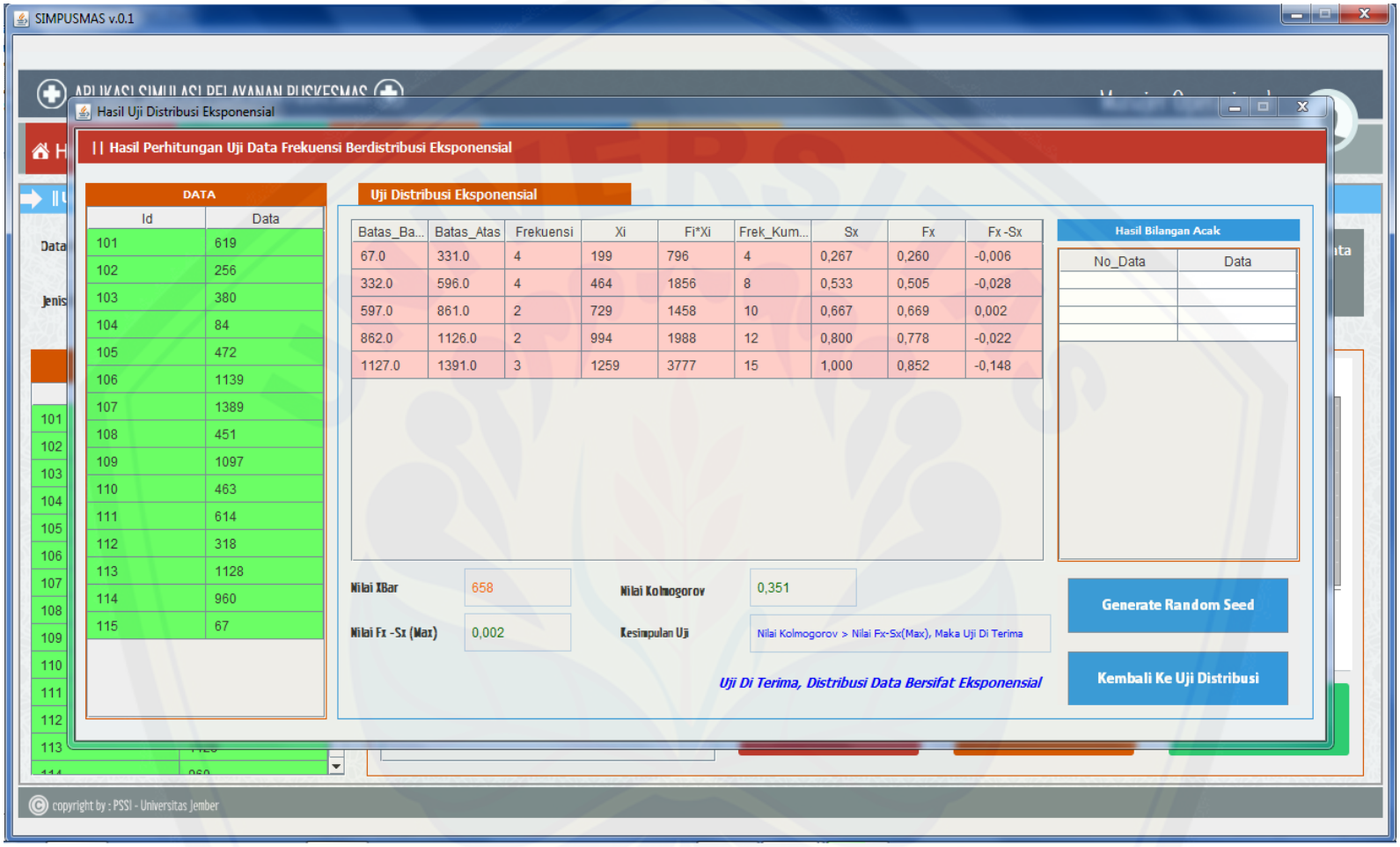
Menu uji distribusi data hanya dapat diakses ketika user yang login adalah manajer. Uji distribusi data digunakan untuk mengetahui sifat - sifat data yang ada dari setiap event. Pengujianya dibagi menjadi tiga yaitu normal, *ekponensial* dan *empiris.* Sebelum dilakukan pengujian sifat data, langkah pertama yang harus dilakukan adalah mencari distribusi frekuensi data. Distribusi frekuensi data ditampilkan kedalam tabel distribusi frekuensi dan grafik distribusi frekuensi seperti yang terlihat pada Gambar 5.10 dan tabel uji distribusi normal pada Gambar 5.11, tabel uji distibusi *eksponensial* pada Gambar 5.12 serta tabel uji distribusi empiris pada Gambar 5.13.



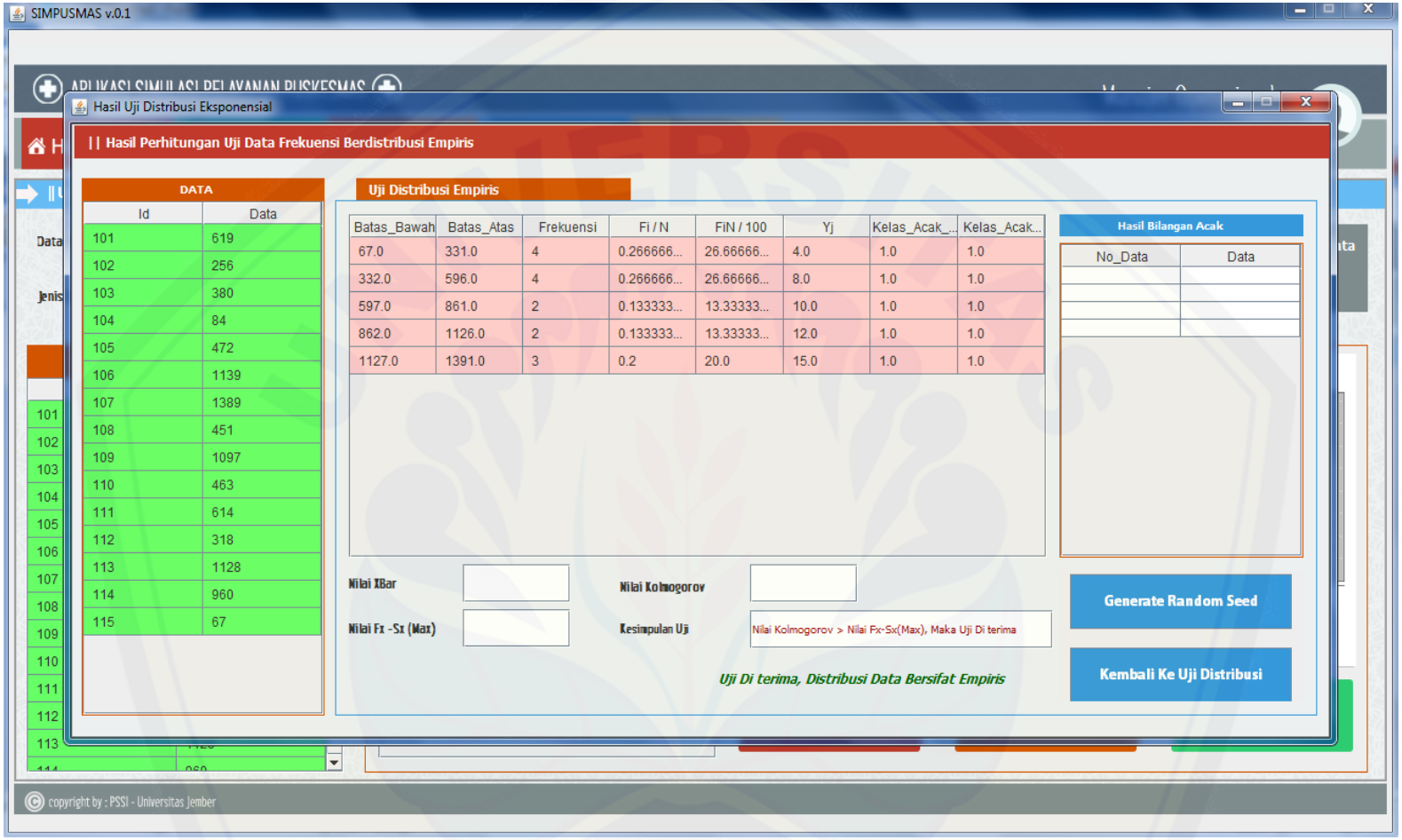
Gambar 5.10 Tampilan uji distribusi frekuensi



Gambar 5.11 Tampilan uji distribusi normal



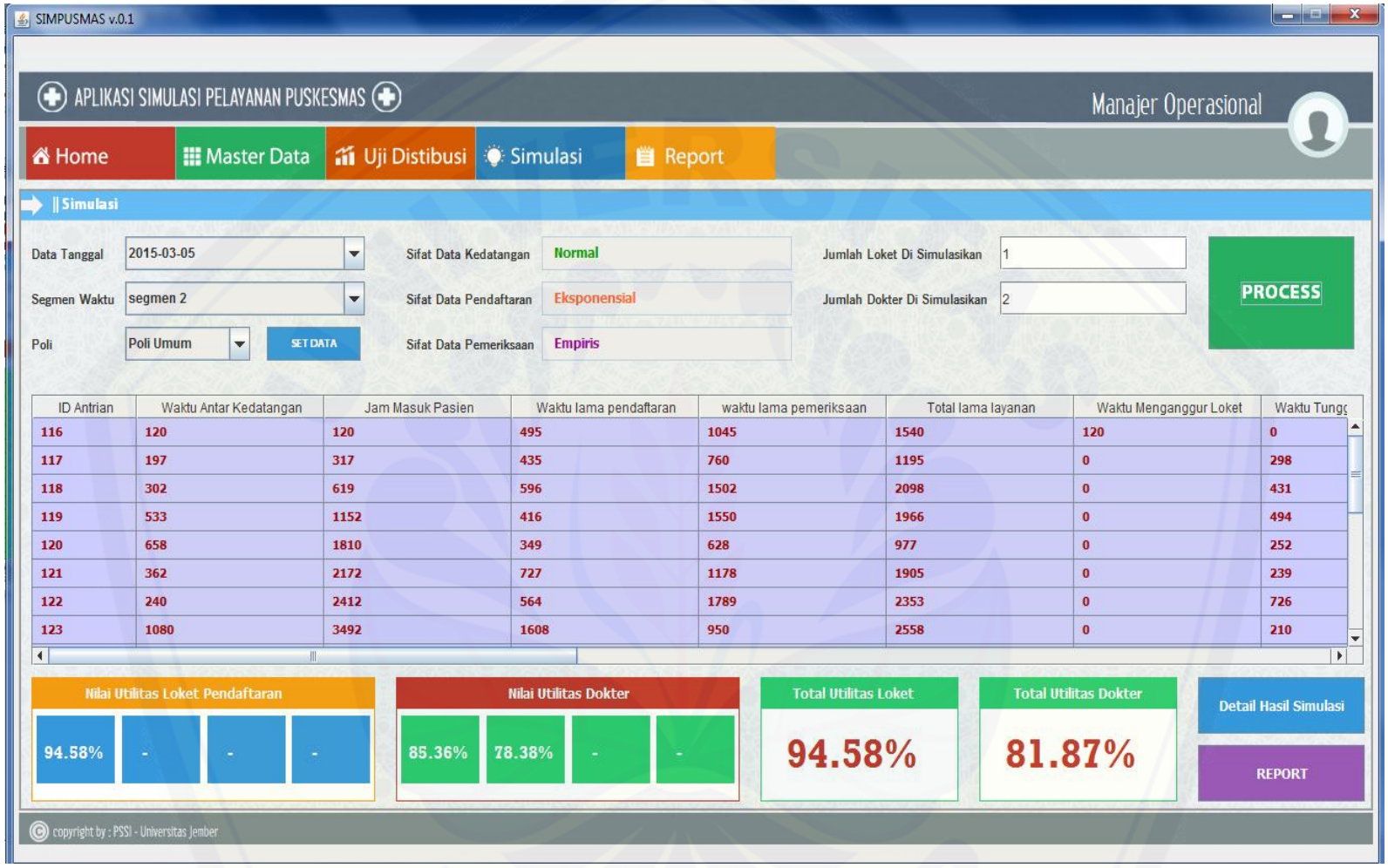
Gambar 5.12 Tampilan uji distribusi *ekponensial*



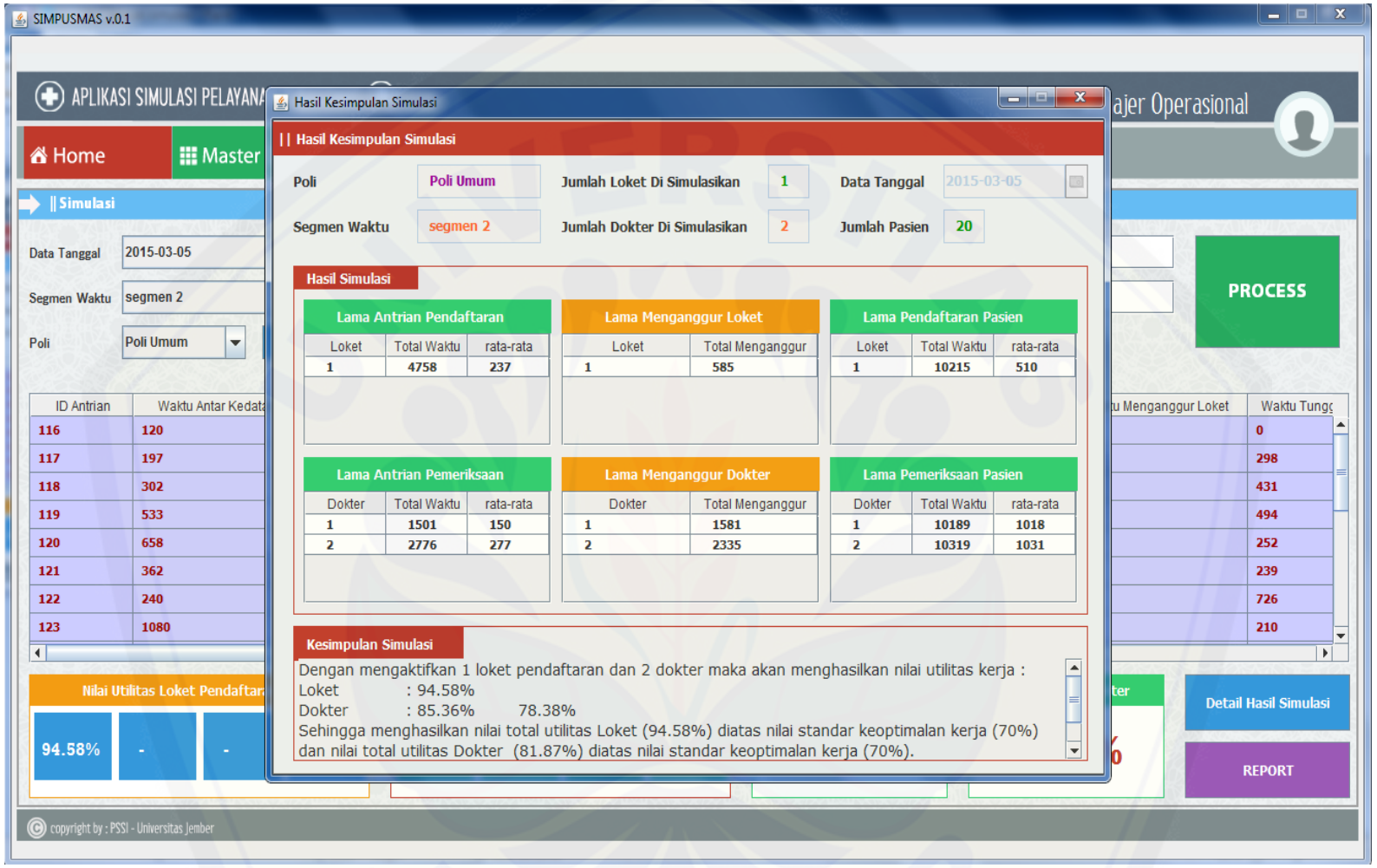
Gambar 5.13 Tampilan uji distribusi *empiris*

5.2.11 Tampilan Menu Simulasi

Fitur simulasi hanya dapat diakses ketika yang *login* adalah manajer. Pada tampilan simulasi seperti pada Gambar 5.14 terdapat lima variabel yang diinputkan oleh manajer untuk melakukan simulasi yaitu, tanggal data, poli, segmen waku, jumlah loket yang disimulasikan dan jumlah dokter yang disimulasikan. Selain itu terdapat tabel simulasi yang mana isinya berupa data - data proses simulasi. terdapat juga hasil nilai utilitas dari proses simulasi dan tombol detail hasil simulasi untuk melihat hasil kesimpulan dari simulasi seperti pada Gambar 5.15 yang menampilkan informasi kesimpulan simulasi dan enam tabel hasil simulasi yang berupa tabel lama antrian pendaftaran, tabel lama menganggur loket, tabel lama pendaftaran, tabel lama antrian pemeriksaan, tabel lama waktu menganggur dokter dan tabel lama waktu pemeriksaan pasien.



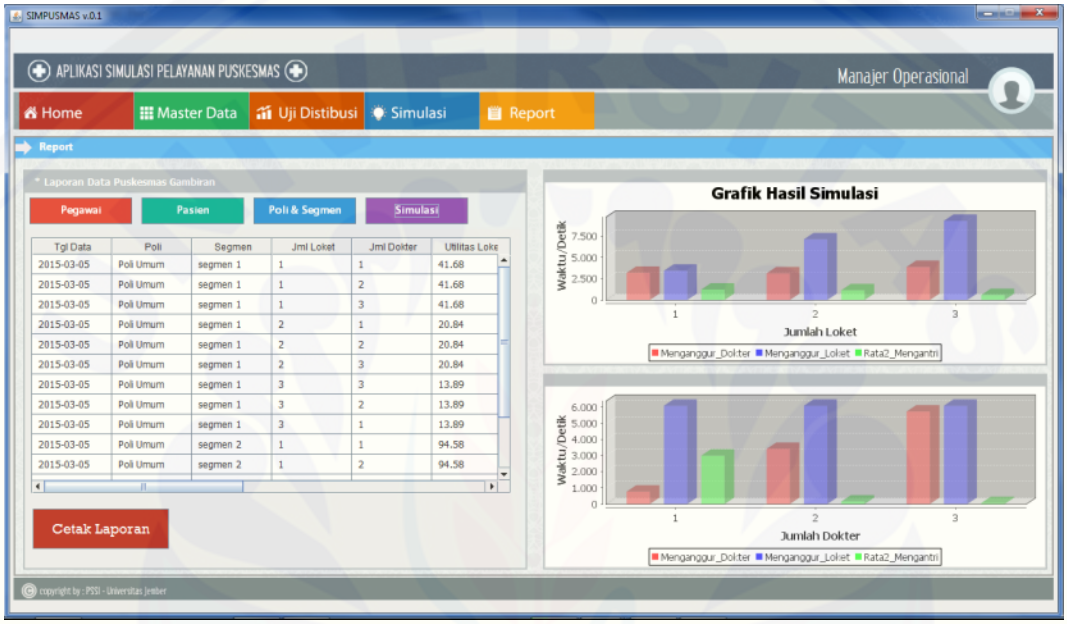
Gambar 5.14 Tampilan simulasi



Gambar 5.15 Tampilan hasil kesimpulan simulasi

5.2.12 Tampilan *Report*

Fitur utama dari *report* adalah menampilkan rekap data Puskesmas yang berupa data pegawai, data pasien, data segmen waktu, data poli dan data hasil simulasi. Pada report juga terdapat tombol cetak yang digunakan ketika ingin mencetak laporan. Menu *report* dapat dilihat pada Gambar



Gambar 5.16 Tampilan *report*

5.3 Pembahasan Metode *Discrete-Event Simulation* Pada Aplikasi Simulator Pelayanan Puskesmas

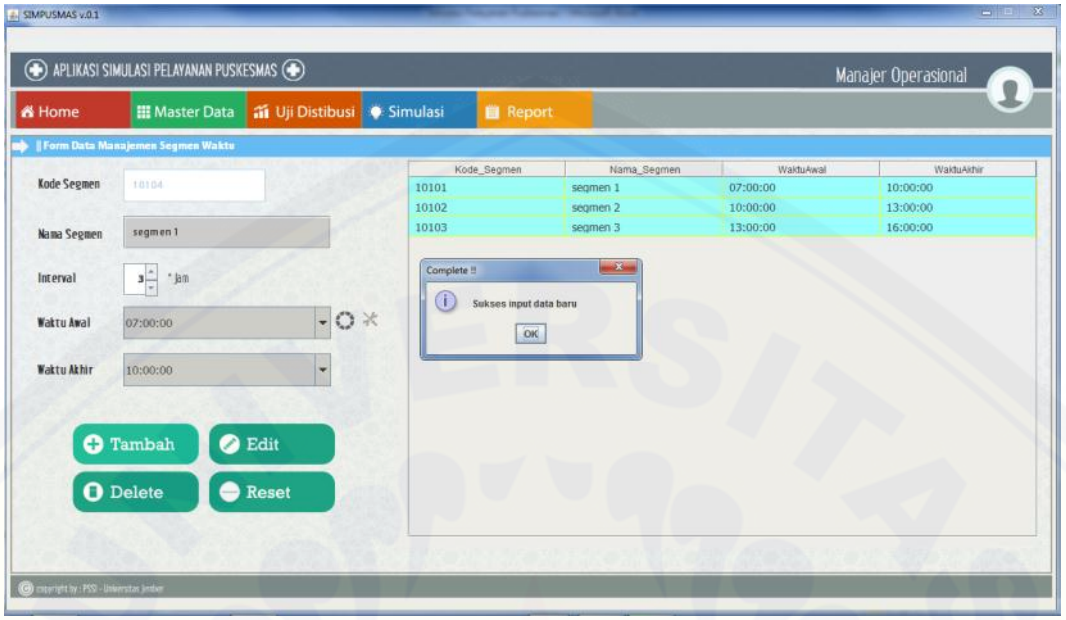
Discrete-event simulation digunakan dalam suatu model sistem yang selalu berubah - ubah dan bersifat terus menerus (*continu*). Pada simulasi dengan metode *discrete-event simulation* proses simulasinya didasarkan pada perubahan *event* (kejadian) pada periode tertentu. Implementasi metode *discrete-event simulation* pada aplikasi simulasi sistem pelayanan Puskesmas dalam penelitian ini terdapat pada pembagian *event* dalam simulasinya, pembagian periode simulasi, pengujian distribusi dan pencarian perhitungan nilai utiltas hasil dari pelayanan berdasarkan

variabel waktu menganggur dari objek yang disimulasikan yaitu loket pendaftaran dan dokter.

Pada pembagian periode simulasi, dalam sistem terdapat fitur manajemen segmen waktu. Fitur manajemen segmen waktu hak aksesnya hanya terbatas untuk user manajer yang bertindak sebagai admin. Manajer disini dapat menambahkan, merubah ataupun menghapus data segmen waktu. *Source code* yang dieksekusi dalam menambah segmen waktu dapat dilihat pada Gambar 5.17 dan untuk hasil dapat dilihat pada Gambar 5.18

```
190      public void tambahSegmen(){
191          //cek kondisi inputan
192
193          try{
194
195              java.sql.Connection c = KoneksiDataBase.getKoneksi();
196              String sql ="INSERT INTO data_segmen VALUES (?, ?, ?, ?)";
197
198              java.sql.PreparedStatement p =  c.prepareStatement(sql);
199              p.setString(1,id);
200              p.setString(2,nama);
201              p.setObject(3,WaktuAwal);
202              p.setObject(4,WaktuAkhir);
203
204              p.executeUpdate();
205              p.close();
206              JOptionPane.showMessageDialog(null, "Sukses input data baru","Complete !!",1);
207              }catch(SQLException e){
208              JOptionPane.showMessageDialog(null,"Kode sudah di pakai, Input kembali","Fail",
209                      JOptionPane.ERROR_MESSAGE);
210              System.out.println(e.getMessage());
211
212              }finally{
213              LoadDataSegmen();
214              }
215  }
```

Gambar 5.17 *Execute code* tambah segmen



Gambar 5.18 sukses tambah segmen

Setelah segmen waktu diinputkan selanjutnya data dari *event* pelayanan puskesmas yang berupa *event* antar kedatangan, lama pendaftaran dan lama pemeriksaan dikelompokkan persegmen waktunya dalam suatu kelas. Untuk mengelompokkanya maka dinutuhkan nilai kelas, batas atas dan batas bawah, range antar batas dan juga frekuensi kemunculanya. Kode program untuk mengelompokkan data dalam suatu kelas dapat dilihat pada Gambar 5.19

```
double jumlah_Kelas_awal     = 1 + (3.33 *(Math.log10(JumlahData)));
jumlah_Kelas                 = Math.round(jumlah_Kelas_awal);


nilaiMax                 = mFrekuensi.CekNilaiData("max",tgl,swaktu,jenisData,poli);
nilaiMin                 = mFrekuensi.CekNilaiData("min",tgl,swaktu,jenisData,poli);

double Rangetmp          = (nilaiMax - nilaiMin)/jumlah_Kelas;
Range                    = Math.round(Rangetmp);

boolean cek=mFrekuensi.CekFrekuensi(tgl, swaktu, jenisData,poli);
if(cek==true){
    vFrekuensi.setModelFrekuensi(mFrekuensi.getModelFrekuensi());
    mFrekuensi.LoadFrekuensi(tgl, swaktu, jenisData,poli);
     }
else{
    for (int i = 0; i < jumlah_Kelas; i++) {
    if (i==0) {
        batasbawah= nilaiMin;
        batasatas = batasbawah+Range;
    }
    else{
        batasbawah= batasatas+1;
        batasatas = batasbawah+Range;
    }

    int frekuensi = mFrekuensi.SelectFrekuensi(batasatas, batasbawah, tgl, swaktu, jenisData,poli);
    //insert
    String kode = mFrekuensi.getKode(swaktu);
    String id_poli = mFrekuensi.getID_Pengaktifan(poli);

    mFrekuensi.tambahFrekuensi(batasatas, batasbawah, frekuensi, tgl, kode,jenisData,id_poli);
    }
    vFrekuensi.setModelFrekuensi(mFrekuensi.getModelFrekuensi());
    mFrekuensi.LoadFrekuensi(tgl, swaktu, jenisData,poli);
    }
```

Gambar 5.19 Kode program pengelompokkan data dalam kelas

Setelah data telah dikelompokkan maka selanjutnya data dapat diuji dengan pengujian *kolmogorov-smirnov* (normal, *eksponensial*) atau jika tidak ada hasil uji yang memenuhi maka menggunakan uji data empiris. Kode program uji data dapat dilihat pada Gambar 5.20 untuk uji distribusi normal

```
java.sql.Connection c = KoneksiDataBase.getKoneksi();
String ambilXbar ="SELECT (sum(`Fi_Xi`) / sum(`Frekuensi`)) as Xbar FROM "
        + "`tabel_uji_distribusi_normal` where Tanggal = '"+tgl+"' and "
        + "kode_Segmen = '"+swaktu+"' and jenis_data_antrian = '"+jData+"' "
        + "and id_pengaktifan ='"+pol+"'";

java.sql.Statement a =  c.createStatement();
ResultSet rs = a.executeQuery(ambilXbar);

int Xbar = 0;
if(rs.next()){
    Xbar = rs.getInt("Xbar");
    System.out.println(Xbar);
}

String Select ="SELECT id_frekuensi,Xi,frekuensi FROM `tabel_uji_distribusi_normal` "
        + "where Tanggal = '"+tgl+"' and kode_Segmen = '"+swaktu+"' and jenis_data_antrian = '"+jData+"'";

java.sql.Statement z =  c.createStatement();
ResultSet zz = z.executeQuery(Select);

while(zz.next()){

    double Xi       = zz.getDouble("Xi");
    String id       = zz.getString("id_frekuensi");
    int frekuensi   = zz.getInt("frekuensi");
    double XiXbar   = Xi - Xbar;
    double XiXbar2  = XiXbar * XiXbar;
    double FiXiXbar2  = frekuensi * XiXbar2;

    String update ="Update `tabel_uji_distribusi_normal` set Xi_Xbar="+XiXbar+","
            + "Xi_Xbar2="+XiXbar2+",Fi_Xi_Xbar2="+FiXiXbar2+" where id_frekuensi='"+id+"'";
    java.sql.Statement o =  c.createStatement();
    o.executeUpdate(update);

   }

String totalXbar2 ="SELECT sum(Fi_Xi_Xbar2) as n FROM `tabel_uji_distribusi_normal` where "
       + "Tanggal = '"+tgl+"' and kode_Segmen = '"+swaktu+"' and jenis_data_antrian = '"+jData+"'";

java.sql.Statement v =  c.createStatement();
ResultSet vr = v.executeQuery(totalXbar2);

int totXbar2 = 0;
if(vr.next()){
    totXbar2 = vr.getInt("n");
   }
```

```
double simpanganBaku =Math.round(Math.sqrt(totXbar2/n));

String Select2 ="SELECT id_frekuensi,Xi,Sx FROM `tabel_uji_distribusi_normal`"
        + " where Tanggal = '"+tgl+"' and kode_Segmen = '"+swaktu+"' and "
        + "jenis_data_antrian = '"+jData+"'";
java.sql.Statement z2 =  c.createStatement();
ResultSet zz2 = z2.executeQuery(Select2);
while(zz2.next()){
    //totXbar2 = vr.getInt("n");
    double Xi       = zz2.getDouble("Xi");
    String id       = zz2.getString("id_frekuensi");
    double Sx       = zz2.getDouble("Sx");
    double Z = (Xi + Xbar)/simpanganBaku;
    String Zx = Double.toString(Z);
    NumberFormat formatter = new DecimalFormat("#0.0000");
    double FX = ((double)(ModelRumusNormsDist.calculateBelowStandardDistribution
              ((Z), 1000, 0, 1)) * 10000) / 10000;
    double FxSx= FX- Sx;
    String update ="Update `tabel_uji_distribusi_normal` set Z="+Z+", FX="+FX+", "
            + "FX_SX="+FxSx+" where id_frekuensi='"+id+"'";
    java.sql.Statement o =  c.createStatement();
    o.executeUpdate(update);

   }
```

Gambar 5.20 Perhitungan uji normal

Jika uji normal tidak memenuhi maka menggunakan uji *eksponensial* yang kode programnya terlihat pada Gambar 5.21

```
while(zz2.next()){
    //totXbar2 = vr.getInt("n");
    double Xi       = zz2.getDouble("Xi");
    String id       = zz2.getString("id_frekuensi");
    double Sx       = zz2.getDouble("Sx");
    double FX       = 1 - (Math.pow(2.71, -( Xi / Xbar)));
    double FxSx     = FX- Sx;


    String update ="Update `tabel_uji_distribusi_Eksponensial` "
            + "set FX="+FX+", FX_SX="+FxSx+" where id_frekuensi='"+id+"'";
    java.sql.Statement o =  c.createStatement();
    o.executeUpdate(update);

  }
```

Gambar 5.21 Kode program uji *eksponensial*

Setelah uji normal maupun eksponensial tidak ada yang memenuhi maka langkah uji terakhir yaitu dengan menggunakan uji distribusi empiris. Kode program uji empiris dapat dilihat pada Gambar 5.22

```
boolean cek=mEmpiris.CekFrekuensi(tgl, swaktu, jenisData,poli);
if(cek==false){
   int Yz=0;
   int frekuensi=0;
   for (int i = 0; i < jumlah_Kelas; i++) {
   if (i==0) {
       batasbawah= nilaiMin;
       batasatas = batasbawah+Range;
       frekuensi = mFrekuensi.SelectFrekuensi(batasatas, batasbawah, tgl, swaktu, jenisData,poli);
       Yz=frekuensi;
   }
   else{
       batasbawah= batasatas+1;
       batasatas = batasbawah+Range;
       frekuensi = mFrekuensi.SelectFrekuensi(batasatas, batasbawah, tgl, swaktu, jenisData,poli);
       Yz=Yz+frekuensi;
   }

   mEmpiris.tambahFrekuensialEmpiris(batasatas, batasbawah, frekuensi,Yz, tgl, kode, jenisData,poliID);
   boolean cekHasilUji = mFrekuensi.CekHasilUji((Date)tgl, (String)jenisData,(String) swaktu,(String) poli)

    if(cekHasilUji == false){
    mFrekuensi.tambahHasilUji((Date)tgl, (String)jenisData,(String) swaktu,(String) poli, "Empiris");
    }

   }
}
```

```
double FiN      = (double)frekuensi/n;
double FiN_100  = FiN * 100;
double Yj       = Yjsebelum;
```

Gambar 5.22 Kode program uji empiris

Langkah selanjutnya setelah data telah diuji dan diketahui sifatnya yaitu melakukan *generate random seed.* Tujuanya yaitu untuk mendapatkan bilangan acak *uniform* yang mana hasilnya akan digunakan dalam variabel simulasi. Bilangan acak ini berfungsi untuk mewakili data sampel. Perhitungan metode pembangkitan bilangan acak didalam program antara lain :

```
StdRandom Random = new StdRandom();
double U1= Random.StdRandom();
double U2= Random.StdRandom();
double lnU1    = (int) (-(2)* Math.log(U1/Math.log10(10))) ;
double Z = Math.pow(lnU1, 1/2)*(Math.cos(2*U2));
int xrand    = (int) (data+ (simpangBaku*Z));
```

Gambar 5.23 Kode program perhitungan *generate random seed* normal

```
java.sql.PreparedStatement p =  c.prepareStatement(sql2);

StdRandom Random = new StdRandom();
double rand = Random.StdRandom();
int xrand    = (int) (-(data)* Math.log(rand/Math.log10(10))) ;
```

Gambar 5.24 Kode program perhitungan *generate random seed* eksponensial

Setelah data telah diuji dan didapat bilangan acaknya langkah selanjutnya yaitu menghitung simulasi data. ketika disimulasikan komponen utama yang dicari nilainya waktu menganggur loket, waktu menganggur dokter, lama mengantri dan lama pelayanan. Dari nilai nilai tersebut maka bisa dicari hasil kesimpulan simulasi berdasarkan nilai utilitas.

Kode program untuk mencari nilai - nilai diatas antara lain :

```
try {
String Select ="SELECT id_simulasi,Jam_Pasien_Mulai_Daftar,"
        + "Jam_Pasien_Selesai_Daftar from `simulasi` where "
        + "Tanggal = '"+tgl+"' and kode_Segmen = '"+swaktu+"' "
        + "and id_pengaktifan ='"+pol+"' and no_loket='"+i+"'";
java.sql.Connection c = KoneksiDataBase.getKoneksi();
java.sql.Statement s;
s = c.createStatement();
ResultSet r = s.executeQuery(Select);
int x=0;
int waktu_nganggur=0;
int temp_selesai=0;
while(r.next()){
String id = r.getString("id_simulasi");
int jam_mulai= r.getInt("Jam_Pasien_Mulai_Daftar");
int jam_selesai= r.getInt("Jam_Pasien_Selesai_Daftar");

if(x==0){
waktu_nganggur=jam_mulai;
temp_selesai=jam_selesai;
}else{
waktu_nganggur=jam_mulai-temp_selesai;
temp_selesai=jam_selesai;
}
```

Gambar 5.25 Kode program hitung waktu menganggur loket

```
try {
String Select ="SELECT id_simulasi,Jam_Pasien_Mulai_Periksa,"
        + " Jam_Selesai_Layanan from `simulasi` where Tanggal = '"+tgl+"'"
        + " and kode_Segmen = '"+swaktu+"' and "
        + "id_pengaktifan ='"+pol+"' and no_dokter='"+i+"'";
java.sql.Connection c = KoneksiDataBase.getKoneksi();
java.sql.Statement s;
s = c.createStatement();
ResultSet r = s.executeQuery(Select);
int x=0;
int waktu_nganggur=0;
int temp_selesai=0;
while(r.next()){
String id = r.getString("id_simulasi");
int jam_mulai= r.getInt("Jam_Pasien_Mulai_Periksa");
int jam_selesai= r.getInt("Jam_Selesai_Layanan");
if(x==0){
waktu_nganggur=jam_mulai;
temp_selesai=jam_selesai;
}else{
waktu_nganggur=jam_mulai-temp_selesai;
temp_selesai=jam_selesai;
}
```

Gambar 5.26 Kode programhitung waktu menganggur dokter

```
try{
java.sql.Connection c = KoneksiDataBase.getKoneksi();
java.sql.Statement s = c.createStatement();
String sql = "SELECT sum(Waktu_Menganggur_Loket) as total_nganggur, "
+ "max(Jam_Pasien_Selesai_Daftar) as selesai_layanan from simulasi "
+ "where Tanggal = '"+tgl+"' and kode_Segmen = '"+swaktu+"' "
        + "and id_pengaktifan ='"+pol+"' and no_loket='"+loket+"'";
ResultSet rs = s.executeQuery(sql);
DecimalFormat formatter  = new DecimalFormat(".##");
while(rs.next()){
        double total_kerja = 3*3600;
        double selesai_layanan = total_kerja - rs.getInt("selesai_layanan");
        double total_nganggur = rs.getInt("total_nganggur")+ selesai_layanan;
         double utilitas =(total_kerja-total_nganggur);
        double hasil= (utilitas/total_kerja)*100;
        String b = formatter.format(hasil);
        a = b.replace(',', '.');

        }
```

Gambar 5.27 Kode program hitung utilitas loket

```
try{
java.sql.Connection c = KoneksiDataBase.getKoneksi();
java.sql.Statement s = c.createStatement();
String sql = "SELECT sum(Waktu_Menganggur_Dokter) as total_nganggur, "
+ "max(Jam_Selesai_Layanan) as selesai_layanan from simulasi where "
        + "Tanggal = '"+tgl+"' and kode_Segmen = '"+swaktu+"' and "
        + "id_pengaktifan ='"+pol+"' and no_dokter='"+dokter+"'";
ResultSet rs = s.executeQuery(sql);
DecimalFormat formatter  = new DecimalFormat(".##");
while(rs.next()){
        double total_kerja = 3*3600;
        double selesai_layanan = rs.getInt("selesai_layanan");
        if(selesai_layanan>total_kerja){
            selesai_layanan=0;
        }else{
            selesai_layanan=total_kerja-selesai_layanan;
        }
       double total_nganggur = rs.getInt("total_nganggur")+selesai_layanan;
        double utilitas =(total_kerja-total_nganggur);
        double hasil= (utilitas/total_kerja)*100;
        String b = formatter.format(hasil);
        a = b.replace(',', '.');

        }
```

Gambar 5.28 Kode program hitung utilitas dokter

5.4 Pengujian Aplikasi Simulasi Pelayanan Puskesmas

Pengujian aplikasi simulasi sistem pelayan puskesmas bertujuan untuk menilai kinerja dari aplikasi yang dibuat. Pengujian dilakukan dengan melakukan simulasi data dengan memasukkan beragam jumlah nilai loket dan dokter yang akan disimulasikan. Banyaknya nilai jumlah loket dan jumlah dokter mempengaruhi hasil nilai utilitas kerja loket/dokter.

Keterangan :

Wtd : waktu antar kedatangan
Wtm : jam masuk pasien
WtLd : waktu lama daftar
WtLp : waktu lama periksa
TL : total layanan
Wml : waktu menganggur loket
Wtpd : waktu tunggu pasien daftar
Wpmd: waktu pasien mulai daftar
Wpsd : waktu pasien selesai daftar
Wmd : waktu menganggur dokter
Wtpp : waktu tunggu pasien periksa
Wpmp: waktu pasien mulai periksa
Wsl : waktu selesai layanan
nL : no loket yang melayani
nD : no dokter yang melayani

Percobaan simulasi data tanggal 03-05-2015 pada segmen 2 di poli umum dengan jumlah loket = 1 dan jumlah dokter = 1

Tabel 5.1 Simulasi 1 loket - 1 dokter aktif

No	Wtd	Wtm	WtLd	WtLp	TL	Wml	Wtpd	Wpmd	Wpsd	Wmd	Wtpp	Wpmp	Wsl	nL	nD
116	49	49	1143	4437	5580	49	0	49	1192	1192	0	1192	5629	1	1
117	55	104	223	1673	1896	0	1088	1192	1415	0	4214	5629	7302	1	1
118	153	257	1332	787	2119	0	1158	1415	2747	0	4555	7302	8089	1	1
119	158	415	62	1093	1155	0	2332	2747	2809	0	5280	8089	9182	1	1
120	1478	1893	543	2856	3399	0	916	2809	3352	0	5830	9182	12038	1	1
121	482	2375	1498	1736	3234	0	977	3352	4850	0	7188	12038	13774	1	1
122	272	2647	94	4906	5000	0	2203	4850	4944	0	8830	13774	18680	1	1
123	886	3533	651	41	692	0	1411	4944	5595	0	13085	18680	18721	1	1
124	978	4511	27	1311	1338	0	1084	5595	5622	0	13099	18721	20032	1	1
125	141	4652	489	2301	2790	0	970	5622	6111	0	13921	20032	22333	1	1
126	50	4702	236	1460	1696	0	1409	6111	6347	0	15986	22333	23793	1	1
127	799	5501	208	668	876	0	846	6347	6555	0	17238	23793	24461	1	1
128	505	6006	143	1935	2078	0	549	6555	6698	0	17763	24461	26396	1	1
129	33	6039	593	392	985	0	659	6698	7291	0	19105	26396	26788	1	1
130	1842	7881	148	1460	1608	590	0	7881	8029	0	18759	26788	28248	1	1
131	354	8235	1075	104	1179	206	0	8235	9310	0	18938	28248	28352	1	1
132	30	8265	357	517	874	0	1045	9310	9667	0	18685	28352	28869	1	1
133	1560	9825	34	176	210	158	0	9825	9859	0	19010	28869	29045	1	1
134	368	10193	10	913	923	334	0	10193	10203	0	18842	29045	29958	1	1
135	31	10224	189	1753	1942	21	0	10224	10413	0	19545	29958	31711	1	1

Waktu menganggur loket : **1745** Waktu menganggur dokter: **1192**

Rata - rata lama antrian = (4766 + 832) / 2 **2799** Rata - rata lama layanan = (452 + 1525) / 2 = **989**

Utilitas Loket = ((to.kerja - w.menganggur loket)/tot.kerja)*100% = 83.84 %

U. dokter= ((to.kerja - w.menganggur dokter)/tot.kerja)*100 %= 88.9 %

Percobaan simulasi data tanggal 03-05-2015 pada segmen 2 di poli umum dengan jumlah loket = 2 dan jumlah dokter = 2

Tabel 5.2 Simulasi 2 loket - 2 dokter aktif

No	Wtd	Wtm	WtLd	WtLp	TL	Wml	Wtpd	Wpmd	Wpsd	Wmd	Wtpp	Wpmp	Wsl	nL	nD
116	49	49	1143	4437	5580	49	0	49	1192	1192	0	1192	5629	1	1
117	55	104	223	1673	1896	104	0	104	327	327	0	327	2000	2	2
118	153	257	1332	787	2119	0	70	327	1659	0	341	2000	2787	2	2
119	158	415	62	1093	1155	0	777	1192	1254	0	1533	2787	3880	1	2
120	1478	1893	543	2856	3399	639	0	1893	2436	0	1444	3880	6736	1	2
121	482	2375	1498	1736	3234	716	0	2375	3873	0	1756	5629	7365	2	1
122	272	2647	94	4906	5000	211	0	2647	2741	0	3995	6736	11642	1	2
123	886	3533	651	41	692	792	0	3533	4184	0	3181	7365	7406	1	1
124	978	4511	27	1311	1338	327	0	4511	4538	0	2868	7406	8717	1	1
125	141	4652	489	2301	2790	114	0	4652	5141	0	3576	8717	11018	1	1
126	50	4702	236	1460	1696	829	0	4702	4938	0	6080	11018	12478	2	1
127	799	5501	208	668	876	360	0	5501	5709	0	5933	11642	12310	1	2
128	505	6006	143	1935	2078	297	0	6006	6149	0	6161	12310	14245	1	2
129	33	6039	593	392	985	1101	0	6039	6632	0	5846	12478	12870	2	1
130	1842	7881	148	1460	1608	1732	0	7881	8029	0	4841	12870	14330	1	1
131	354	8235	1075	104	1179	206	0	8235	9310	0	4935	14245	14349	1	2
132	30	8265	357	517	874	1633	0	8265	8622	0	5708	14330	14847	2	1
133	1560	9825	34	176	210	515	0	9825	9859	0	4490	14349	14525	1	2
134	368	10193	10	913	923	334	0	10193	10203	0	4322	14525	15438	1	2
135	31	10224	189	1753	1942	21	0	10224	10413	0	4434	14847	16600	1	1

Waktu menganggur loket 1 :	**5984**	Waktu menganggur dokter1:	**1192**
Waktu menganggur loket 2 :	**6561**	Waktu menganggur dokter2:	**327**

Rata - rata lama antrian = (66 +636) / 4 **176**

Rata - rata lama layanan = (344 + 706 + 1540 + 1511) / 4 = **1025**

Utilitas Loket1 = ((to.kerja - w.menganggur loket)/tot.kerja)*100% = 44.59 %

U. dokter1= ((to.kerja - w.menganggur dokter)/tot.kerja)*100 %= 78%

Utilitas Loket2 = ((to.kerja - w.menganggur loket)/tot.kerja)*100% = 39.25 %

U. dokter2= ((to.kerja - w.menganggur dokter)/tot.kerja)*100 %= 97%

Percobaan simulasi data tanggal 03-05-2015 pada segmen 2 di poli umum dengan jumlah loket = 1 dan jumlah dokter = 2

Tabel 5.3 Simulasi 1 loket - 2 dokter aktif

No	Wtd	Wtm	WtLd	WtLp	TL	Wml	Wtpd	Wpmd	Wpsd	Wmd	Wtpp	Wpmp	Wsl	nL	nD
116	49	49	1143	4437	5580	49	0	49	1192	1192	0	1192	5629	1	1
117	55	104	223	1673	1896	0	1088	1192	1415	1415	0	1415	3088	1	2
118	153	257	1332	787	2119	0	1158	1415	2747	0	341	3088	3875	1	2
119	158	415	62	1093	1155	0	2332	2747	2809	0	1066	3875	4968	1	2
120	1478	1893	543	2856	3399	0	916	2809	3352	0	1616	4968	7824	1	2
121	482	2375	1498	1736	3234	0	977	3352	4850	0	779	5629	7365	1	1
122	272	2647	94	4906	5000	0	2203	4850	4944	0	2421	7365	12271	1	1
123	886	3533	651	41	692	0	1411	4944	5595	0	2229	7824	7865	1	2
124	978	4511	27	1311	1338	0	1084	5595	5622	0	2243	7865	9176	1	2
125	141	4652	489	2301	2790	0	970	5622	6111	0	3065	9176	11477	1	2
126	50	4702	236	1460	1696	0	1409	6111	6347	0	5130	11477	12937	1	2
127	799	5501	208	668	876	0	846	6347	6555	0	5716	12271	12939	1	1
128	505	6006	143	1935	2078	0	549	6555	6698	0	6239	12937	14872	1	2
129	33	6039	593	392	985	0	659	6698	7291	0	5648	12939	13331	1	1
130	1842	7881	148	1460	1608	590	0	7881	8029	0	5302	13331	14791	1	1
131	354	8235	1075	104	1179	206	0	8235	9310	0	5481	14791	14895	1	1
132	30	8265	357	517	874	0	1045	9310	9667	0	5205	14872	15389	1	2
133	1560	9825	34	176	210	158	0	9825	9859	0	5036	14895	15071	1	1
134	368	10193	10	913	923	334	0	10193	10203	0	4868	15071	15984	1	1
135	31	10224	189	1753	1942	21	0	10224	10413	0	4976	15389	17142	1	2

Waktu menganggur loket 1 :	**1745**	Waktu menganggur dokter1:	**1192**
		Waktu menganggur dokter2:	**1415**

Rata - rata lama antrian = (832+330+306) / 3 **490**

Rata - rata lama layanan = (452+1643 + 1429) / 3 = **1175**

Utilitas Loket1 = ((to.kerja - w.menganggur loket)/tot.kerja)*100% = 83.84%

U. dokter1= ((to.kerja - w.menganggur dokter)/tot.kerja)*100 %= 84%

U. dokter2= ((to.kerja - w.menganggur dokter)/tot.kerja)*100 %= 88%

BAB 6. PENUTUP

Pada bab ini merupakan bagian akhir di dalam penulisan skripsi, berisi tentang kesimpulan dan saran. Kesimpulan yang ditulis merupakan kesimpulan dari hasil penelitian yang telah dilakukan dan saran lanjutan untuk dilakukan pada penelitian selanjutnya.

6.1 Kesimpulan

Kesimpulan dari penelitian yang telah dilakukan adalah sebagai berikut:

1. Aplikasi simulator sistem pelayanan puskesmas dengan menggunakan metode *discrete-event simulation* dibuat dengan menggunakan 3 hak akses yaitu manajer sebagai admin, operator pendaftaran dan operator pemeriksaan. Manajer memiliki beberapa fitur yaitu manajemen data user, data poli, data segmen, uji distribusi data dan fitur simulasi. Operator pendaftaran memiliki fitur manajemen data antrian dan manajemen data pendaftaran. Sedangkan operator pemeriksaan memiliki fitur untuk manajemen data pemeriksaan pasien
2. Aplikasi simulator sistem pelayanan puskesmas dibangun dengan menerapkan metode *discrete-event simulation.* Penerapan metode ini terdapat pada pembagian periode segmen waktu dalam tiap *event.* Pengujian uji distribusi data dan simulasi untuk mencari nilai utilitas kerja.
3. Pada percobaan dalam penelitian, dilakukan simulasi pada data sampel data tanggal 03-05-2015 pada segmen 2 di poli umum. Ketika yang disimulasikan 1 loket dan 1 dokter maka akan menghasilkan nilai utilitas loket sebesar 83.84% dan utiltas dokter sebesar 88.9% dengan waktu menganggur loket sebesar 1745 detik dan waktu menganggur dokter sebesar 1192 detik dengan rata-rata antrian sebesar 2799 detik/pasien dan rata - rata layanan sebesar 989 detik/pasien. Ketika yang disimulasikan 2 loket dan 2 dokter maka akan menghasilkan nilai utilitas loket 1 sebesar 44.59% loket 2 sebesar 39.25% dan utiltas dokter 1 sebesar 78%, dokter 2 sebesar 97% dengan waktu menganggur loket 1 sebesar

5984 detik, loket 2 sebesar 6561 detik dan waktu menganggur dokter 1 sebesar 1192 detik, menganggur dokter 2 sebesar 327 detik dengan rata-rata antrian sebesar 176 detik/pasien dan rata - rata layanan sebesar 1025 detik/pasien. Sedangkan jika yang disimulasikan 1 loket dan 2 dokter maka akan menghasilkan nilai utilitas loket sebesar 83.84% dan utiltas dokter 1 sebesar 84%, dokter 2 sebesar 88% dengan waktu menganggur loket sebesar 1745 detik dan waktu menganggur dokter 1 sebesar 1192 detik, menganggur dokter 2 sebesar 1415 detik dengan rata-rata antrian sebesar 490 detik/pasien dan rata - rata layanan sebesar 1175 detik/pasien.

4. Penambahan atau pengurangan loket dan dokter diberlakukan apabila nilai utilitas dari loket atau dokter dianggap masih belum memenuhi nilai optimal. Penambahan loket atau dokter dilakukan jika nilai utilitas yang dirasakan terlalu besar, sedangkan pengurangan loket atau dokter dilakukan jika nilai utilitas lebih kecil dari nilai optimal yaitu diatas 70%..

6.2 Saran

Beberapa saran dan masukan berikut diharapkan dapat memberikan perbaikan dalam penelitian selanjutnya, yaitu :

1. Melakukan penelitian simulasi terhadap semua fasilitas yang ada didalam Puskesmas
2. Melakukan penelitian simulasi ke instansi atau objek lain yang memiliki pola antrian yang serupa dengan data yang lebih komplek.
3. Inputan jumlah objek yang disimulasikan bersifat dinamis.

DAFTAR PUSTAKA

Agissa, W. (2013). White Box and Black Box Testing. *http://bangwildan.web.id/berita-176-white-box-testing--black-box-testing.html*.

Departemen Kesehatan RI. (2009). *Sistem Kesehatan Nasional.*

Djati, B. S. (2007). *Simulasi Teori dan Aplikasinya.* Yogyakarta: Andi.

Effendi, F., & Makhfudli. (2009). *Keperawatan Kesehatan Masyrakat.* Jakarta: Salemba Medika.

Ekoanindiyo, F. A. (2012). Pemodelan Sistem Antrian Dengan Menggunakan Simulasi. *Dinamika Teknik*, 72 - 85.

Farkhan, F., Hendikawati, P., & Arifudin, R. (2013). Aplikasi Teori antrian dan Simulasi Pada Pelayanan Teller Bank. *UNNES Journal of Mathematics (UJM)*, 18-23.

Hendrayudi. (2008). *VB 2008 Untuk Berbagai Keperluan Programming* . Jakarta: PT Elex Media Komputindo.

Jogiyanto, H. M. (2004). *Pengenalan Komputer: Dasar Ilmu Komputer, Pemrogaman, Sistem Informasi dan Intelegensi Buatan.* Yogyakarta: C.V ANDI OFFSET .

Marimin, Tanjung, H., & Prabowo, H. (2006). *Sistem Informasi Manajemen Sumber Daya Manusia.* Jakarta: Grasindo.

Pressman, R. (2012). *Rekayasa Perangkat Lunak Pendekatan Praktisi.* Yogyakarta: ANDI OFFSET.

Suryani, E. (2006). *Pemodelan dan Simulasi.* Yogyakarta: Graha Ilmu.

Wahyudi, V. G., Sinulingga, S., & Firdaus, F. (2012). Perancangan Sistem Simulasi Antrian Kendaraan Bermotor Pada Stasium Pengisian Bahan-Bakar (SPBU) Menggunakan Metode Distribusi Eksponensial. *Jurnal Elektronik Ilmu Komputer - Universitas Udayana (JELIKU)*, 104-113.

Warmadewa, I. M. (2011). Rancang Bangun Aplikasi Simulasi Pelayanan Pelanggan Dengan Menggunakan Metode Discrete-Event Pada PT. PLN (PERSERO) Cabang Surabaya. *http://digilib.stikom.edu/detil.php?id=2206&q*.