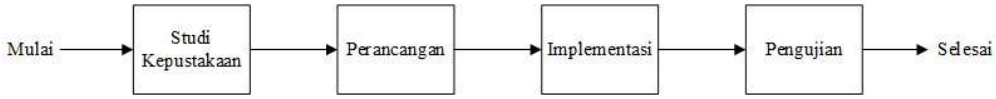
BAB III
ANALISIS DAN PERANCANGAN

3.1 Rancangan Penelitian

Rancangan penelitian dibuat untuk menentukan langkah-langkah dalam menyelesaikan penelitian ini. Adapun diagram mengenai rancangan penelitian ini dapat pada Gambar 8.

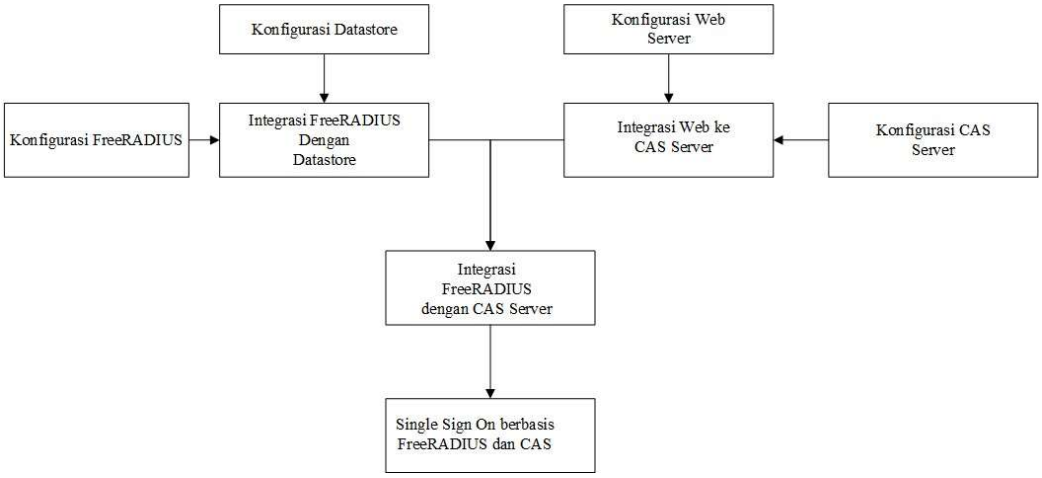


Gambar 8. Diagram Rancangan Penelitian.

Berdasarkan Gambar 8 berikut merupakan penjelasan langkah-langkah dari rancangan penelitian :

1. Melakukan pembuatan studi kepustakaan dibuat berdasarkan jurnal mengenai penelitian terkait dan juga teori pendukung mengenai integrasi *central authentication service* (CAS) dengan FreeRADIUS sebagai *Single Sign-On* (SSO).
2. Melakukan perancangan sistem termasuk diantaranya topologi jaringan sistem secara umum, serta diagram alir proses-proses pada sistem *single Sign-On*.
3. Melakukan implementasi sesuai dengan perancangan. Pada tahap ini akan dilakukan integrasi antara FreeRADIUS dengan *datastore* sebagai *authentication Server* yang nantinya akan diintegrasikan dengan CAS *Server* sebagai SSO *server*.
4. Setelah tahap implementasi berhasil dilakukan untuk memastikan keberjalan sistem sesuai dengan tujuan dan rancangan dari penelitian, maka dilakukan pengujian terhadap sistem sesuai dengan rancangan pengujian.

3.2 Perancangan Sistem



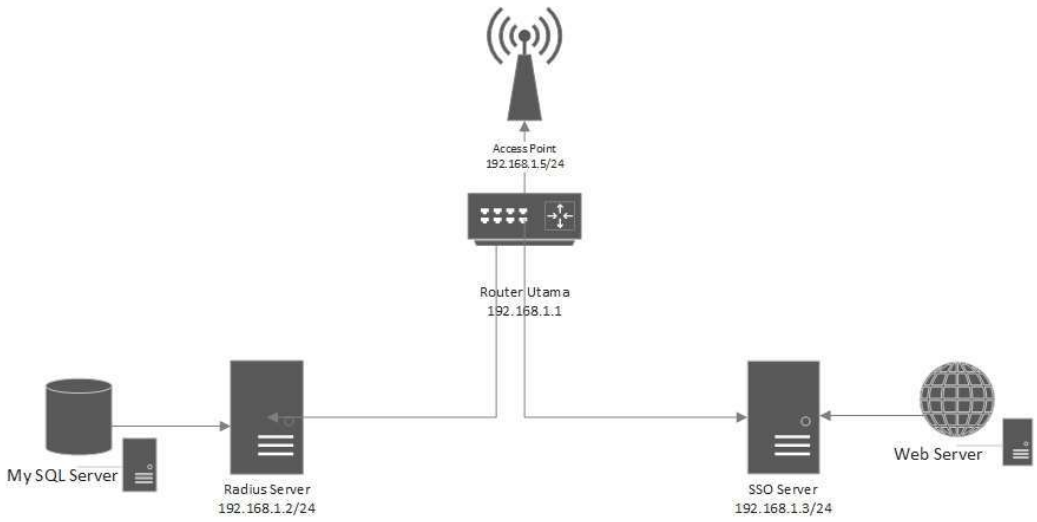
Gambar 9. Rancangan Pengembangan Sistem.

Gambar 9 menjelaskan tentang alur dari pengembangan sistem. Sebelum melakukan integrasi antara *Server* FreeRADIUS dengan *Server* CAS. Masing-masing *server* perlu dikonfigurasi dan diintegrasikan terlebih dahulu. *Server* FreeRADIUS akan diintegrasikan dengan *datastore* yang nantinya akan dijadikan sebagai *authentication Server.* Sementara itu untuk menghubungkan setiap aplikasi web agar dapat diakses dengan sistem *login* tunggal dilakukan integrasi antara *Server* CAS dengan *Server* web. Integrasi ini termasuk mengkonfigurasi aplikasi web untuk didaftarkan sebagai klien CAS.

3.2.1 Perancangan Sistem Secara Umum

Gambar 3.3 menampilkan rancangan sistem dan konektivitas antar perangkat yang digunakan. Sistem yang dibuat pada penelitian ini menggunakan 3 buah *Server*, masing-masing *Server* yaitu *Server* web, *Server* CAS dan *Server* FreeRADIUS. Setiap *Server* akan saling terhubung dan berkomunikasi. *Server* web digunakan sebagai *Server* yang mengandung semua aplikasi web pada perusahaan X, semua layanan aplikasi web berada pada *Server* ini. *Server* CAS merupakan *Server* tersendiri yang bertindak sebagai SSO *Server*. Pada *Server* CAS administrator dapat

melakukan konfigurasi untuk menambahkan dan mengurangi aplikasi yang menggunakan layanan CAS. Untuk dapat mengakses setiap aplikasi *user* akan diarahkan ke *Server* CAS dahulu untuk melakukan proses *login*. *Server* CAS akan menampilkan portal halaman *login* yang nantinya data kredensial pengguna tersebut akan dikirimkan untuk dilakukan autentikasi oleh *Server* FreeRADIUS. *Server* FreeRADIUS akan melakukan autentikasi dan pencocokan data pada *datastore.* Setiap aplikasi *server* tersebut akan dipasangkan dalam 2 komputer sebagai komputer *server*. Untuk mengetahui topologi dari sistem Gambar 10 merupakan topologi sistem :



Gambar 10. Topologi Sistem.

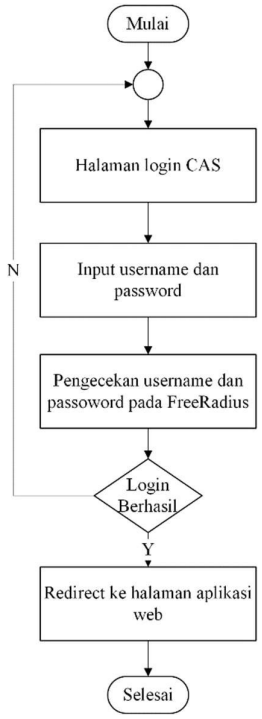
Pengguna yang berhasil melakukan autentikasi akan langsung diarahkan ke halaman web yang diakses. Pengguna diberikan tiket yang didapatkan dari *Server* CAS sebagai tiket masuk ke aplikasi web dan aplikasi web akan mencocokkan tiket yang dimiliki pengguna ke *Server* CAS. Apabila data yang dicocokkan benar, maka CAS akan memberikan informasi *username* pengguna pada web tersebut. CAS juga akan mengirimkan *cookie* ke browser pengguna yang membuat pengguna tidak perlu melakukan pengulangan autentikasi untuk dapat mengakses aplikasi lain.

Mekanisme *logout* yang dilakukan *user* pada aplikasi web akan diteruskan diteruskan ke *Server* CAS. Setelah berhasil maka *Server* CAS akan meneruskan ke halaman *logout* aplikasi tersebut. *Session* dan *cookie* lokal pada aplikasi web tersebut akan dimusnahkan. Pemusnahan ini akan membuat pengguna secara otomatis keluar dari setiap aplikasi yang ada.

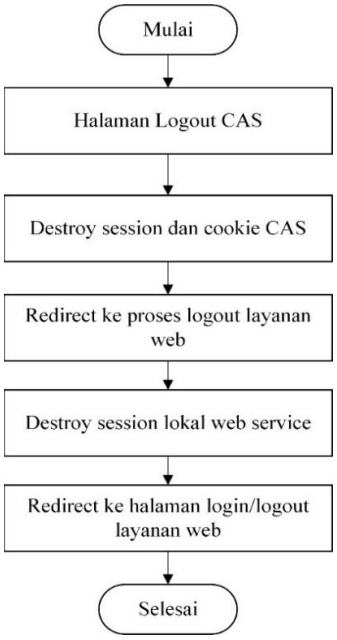
3.2.2 Perancangan *Server* CAS

Server CAS dan *Server* web akan dipasangkan pada satu komputer *server* yang sama, hal ini dilakukan dikarenakan keterbatasan sumber daya perangkat yang dimiliki pada saat pengembangan sistem. Pada penelitian ini *Server* CAS diintegrasikan dengan FreeRADIUS *Server* dalam hal autentikasi pengguna, sehingga peran utama dari CAS adalah untuk membuat pengguna dapat mengakses setiap aplikasi yang ada pada perusahaan dengan sekali *login*. Untuk dapat memahami cara kerja *Server* CAS diagram alir proses *login* dan *logout* SSO menggunakan CAS dapat dilihat pada Gambar 11 dan Gambar 12.

Rancangan *Server* CAS pada penelitian ini menggunakan sistem operasi Linux distro ubuntu *Server* yang merupakan turunan Debian dan akan dipasangkan beberapa fitur seperti JAVA, Apache Tomcat, Gradle dan aplikasi yang diperlukan oleh *Server* CAS dan *web server*. Penggunaan Gradle dibutuhkan untuk melakukan instalasi aplikasi CAS *Server* sebelum dapat digunakan. Apache Tomcat digunakan sebagai *web machine* yang berguna untuk menjalankan aplikasi *Server* CAS.



Gambar 11. Diagram Alir Proses *Login* Pada *Server* CAS.

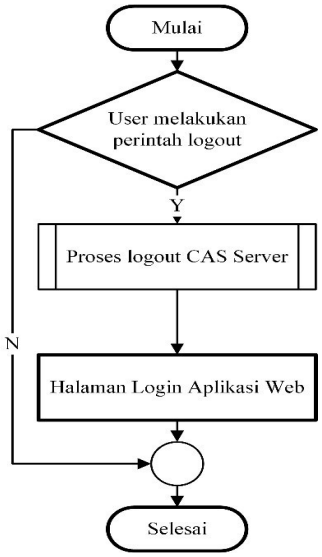


Gambar 12. Proses *Logout* Pada *Server* CAS.

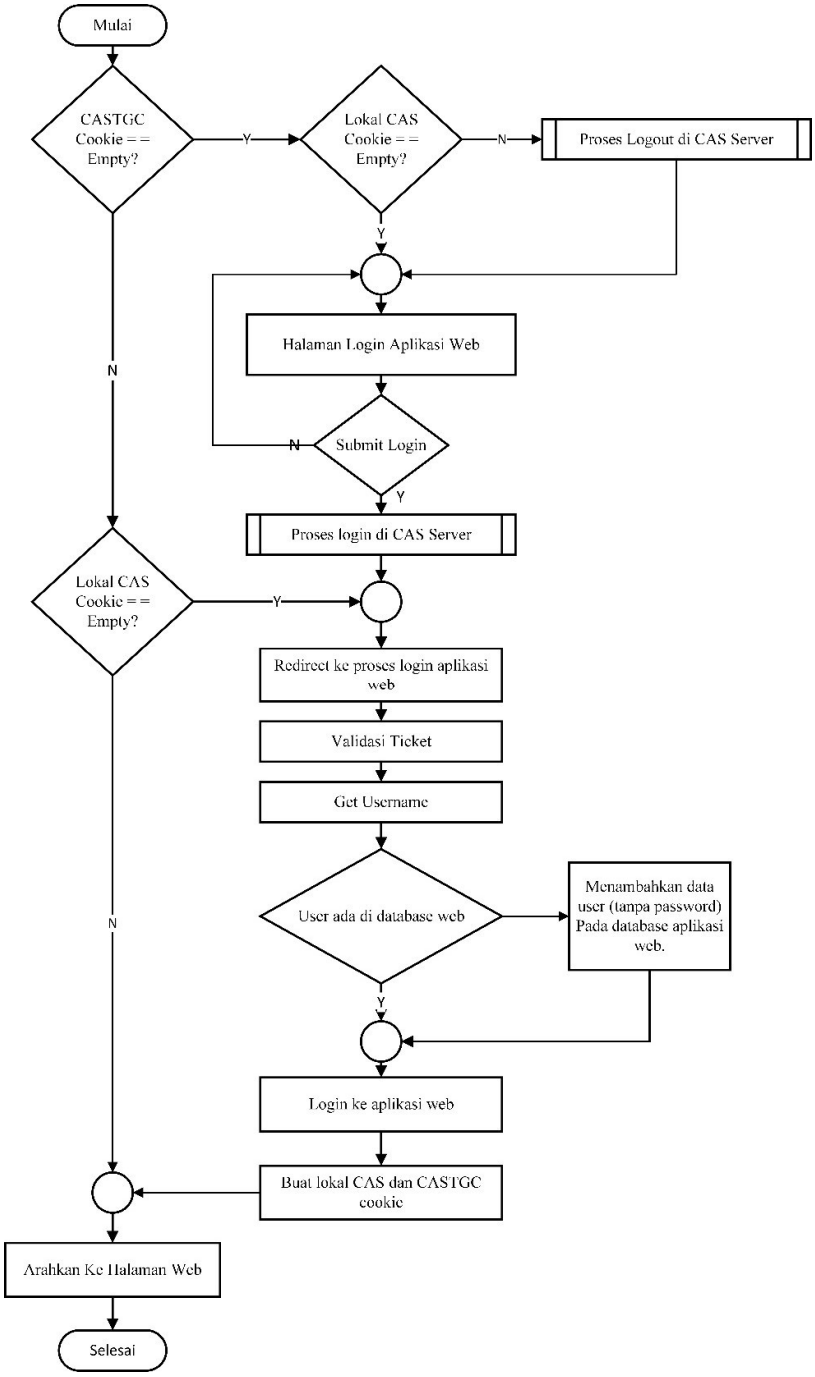
3.2.3 Perancangan Sistem Klien CAS

Klien CAS pada penelitian ini merupakan *prototype* aplikasi-aplikasi web menggunakan bahasa program php dan HTML. Untuk dapat menjalankan aplikasi web yang ada maka diperlukan Apache *Server*, PHP untuk dipasangkan pada *Server* ini. Pada setiap sistem *login* yang dimiliki oleh masing-masing aplikasi web nantinya perlu diarahkan dan diintegrasikan ke proses *login* CAS. Gambar proses *login* dari aplikasi web dapat dilihat pada Gambar 13.

Gambar 13 Menjelaskan bagaimana proses apabila *user* mencoba melakukan *login* pada salah satu aplikasi. Dapat dilihat pada proses tersebut klien CAS akan melakukan pengecekan ketersediaan *cookie* TGC (*Ticket Granting Cookie*) untuk mengetahui apakah pengguna sudah melakukan autentikasi. Kemudian dilanjutkan dengan melakukan pengecekan ketersediaan *cookie* lokal CAS yang dimiliki oleh pengguna pada web browsernya. Apabila pengguna tidak memiliki kedua *cookie* tersebut, maka akan langsung diarahkan ke proses *login* CAS dan akan dilakukan pengecekan. Sedangkan pada proses *logout* juga akan diarahkan ke *Server* CAS. Diagram alir proses logout pada CAS dapat dilihat pada Gambar 13 berikut.



Gambar 13. Diagram alir proses *Logout* pada aplikasi web.

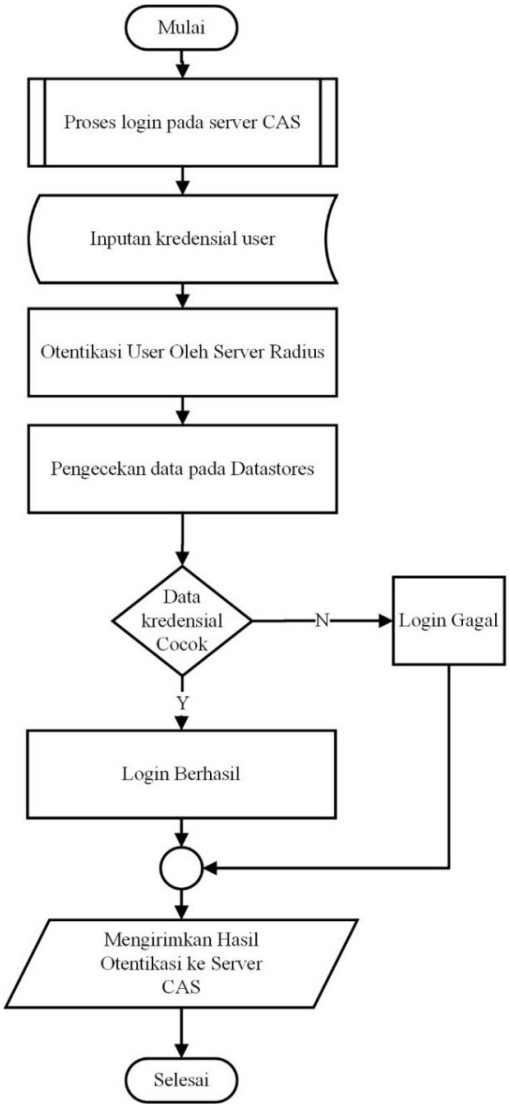


Gambar 14. Diagram alir proses *Login* pada aplikasi web.

3.2.4 Perancangan Sistem FreeRADIUS

Server FreeRADIUS akan digunakan sebagai *authentication Server* pada penelitian ini. FreeRADIUS akan menindaklanjuti inputan data kredensial yang telah diinputkan pengguna pada halam *login Server* CAS. *Server* CAS akan mengirimkan data *username* dan *password* yang telah dimasukkan oleh *user*. Kemudian data tersebut akan dicocokkan dengan data *user* yang ada pada *datastore* yang telah terintegrasi dengan FreeRADIUS. Hasil autentikasi yang telah diproses kemudian akan dikirimkan ke *Server* CAS untuk memutuskan apakah *user* akan diberikan *session* dan juga *cookie* CAS. Untuk dapat memahami proses autentikasi pada *Server* FreeRADIUS dapat dilihat pada Gambar 14.

Gambar 14 menjelaskan alur dari proses autentikasi pada sisi *Server* FreeRADIUS. *Server* ini bekerja setelah user menuju ke halaman *login* SSO oleh CAS *Server* dan pengguna telah melakukan *submit* data yang telah dimasukkan. FreeRADIUS selanjutnya akan melakukan fungsi autentikasi dan melakukan pengecekan ke *datastore* yang menampung seluruh data kredensial pengguna secara terpusat. Hasil dari proses autentikasi pengguna akan dikirimkan oleh *server* FreeRADIUS ke *server* CAS untuk dilakukan tindak lanjut dari proses autentikasi tersebut. Apabila data kredensial pengguna berhasil dicocokkan, maka informasi akan dikirimkan ke *Server* CAS dan kemudian *Server* CAS tersebut akan membuat dan memberikan tiket layanan yang berisi data dan juga hak akses dari pengguna yang berupa *cookie.* Apabila *login* gagal FreeRADIUS akan mengirimkan hasil autentikasi tersebut dan *Server* CAS akan memberikan pesan gagal ke pengguna untuk membuat pengguna melakukan autentikasi ulang.



Gambar 15. Proses Autentikasi *Server* FreeRADIUS.

3.3 Metode Pengujian Sistem

Pengujian sistem dilakukan untuk menentukan apakah sistem yang dikembangkan dapat berjalan dengan baik sesuai dengan rencana dan tujuan dari penelitian ini. Untuk itu pengujian pada penelitian ini diuji terhadap 4 aspek, yaitu pengujian *basic path,* pengujian fungsional, pengujian *response time* dan pengujian *load test*. Berikut merupakan penjabaran dari metode pengujian pada penelitian ini :

3.3.1 Pengujian *Basic Path*

Pengujian *Basic Path* adalah pengujian struktur program yang digunakan untuk efisiensi dan efektivitas pengujian *white box*, karena didalam pengujian *white box* tidak mungkin seluruh jalur dieksekusi. Pengujian ini memungkinkan perancangan *test case* yang diturunkan dari *cyclomatic complexity* struktur program. Ukuran dari *cyclomatic complexity* akan menjadi panduan dalam menentukan *basic path. Test case* yang dapat akan menjadi *test case* yang digunakan untuk membuktikan bahwa setiap *statement* dari program akan dieksekusi minimal sekali[12].

Cyclomatic complexity atau *V(G)* adalah sebuah besaran perangkat lunak yang menyatakan ukuran tingkat kompleksitas sebuah program. Angka ini menentukan jumlah jalur dasar yang harus diuji minimal sekali dari sebuah program. Secara matematis, *V(G)* dapat ditentukan menggunakan pendekatan berikut[12] :

$$\boldsymbol{V(G) = E - N + 2} \qquad (1)$$

V(G) : *cyclomatic complexity*

E : total jumlah *edge*

N : total jumlah *node*

Studi yang dibuat oleh sejumlah industri, semakin besar nilai *V(G)* maka semakin besar probabilitas terjadinya kesalahan program[12].

Pengujian *basic path* yang digunakan pada penelitian ini difokuskan pada proses *login* SSO. Pengujian ini bertujuan untuk melihat *path-path* yang dilalui saat program dijalankan dan untuk mengetahui kompleksitas atau kerumitan dari proses *login* sistem SSO.

3.3.2 Pengujian Fungsional

Pengujian fungsional merupakan metode pengujian perangkat lunak yang digunakan untuk menguji perangkat lunak tanpa mengetahui struktur internal kode atau program. Dalam pengujian ini, *tester* menyadari apa yang harus dilakukan oleh program tetapi tidak memiliki pengetahuan tentang bagaimana melakukannya.

Pengujian fungsional yang digunakan pada penelitian ini adalah pengujian *output* dan *input* dari sistem *Single Sign-On.* Pengujian ini ditujukan untuk mengetahui kebenaran *output* dari perintah yang dimasukkan. Tabel *scenario testing* yang akan diujicobakan dapat dilihat pada Tabel 4 :

Tabel 4. *Scenario Testing*

Test Case	***Description***		***Pre Requisite***	***Step***	***Expect Result***
Login Awal Tanpa Memiliki *Ticket Granting*	*Valid*	*User* memasukkan Data yang Benar	1. Data *user* sudah terdaftar ke dalam *database* 2. *User* mengakses salah satu aplikasi web dan diarahkan ke portal SSO/mengakses portal SSO	1.*User* memasukkan *username* yang benar 2.*User* memasukkan password yang benar 3. User menekan tombol submit	*User* berhasil melakukan *login* dan langsung diarahkan ke layanan web sebagai *user* yang berhasil *login*
	Invalid	*User* memasukkan data kredensial yang tidak benar atau belum terdaftar dalam *database*	*User* mengakses salah satu aplikasi web dan diarahkan ke portal SSO/mengakses portal SSO	1.*User* memasukkan *username* yang benar 2. *User* memasukkan *password* yang benar 3. *User* menekan tombol submit	Sistem menampilkan peringatan *username* dan *password* salah
User memiliki *Ticket Granting* mengakses aplikasi	*User* mencoba mengakses aplikasi/aplikasi lain setelah memiliki *ticket granting*		*User* telah berhasil melakukan proses *login*	*User* mengakses aplikasi/aplikasi lain	Sistem langsung mengarah pada halaman utama web aplikasi tanpa diarahkan ke halaman *login*
User yang memiliki *ticket granting* yang sudah kadaluarsa mengakses aplikasi	*User* mencoba mengakses aplikasi setelah *session* tiket berakhir		*User* telah berhasil melakukan proses *login*	*User* mengakses salah satu aplikasi web	*User* akan langsung diarahkan ke portal SSO untuk melakukan login

3.3.3 Pengujian *Response Time*

Response Time Testing adalah pengujian waktu respon yang mengacu pada waktu yang diperlukan untuk satu *server* untuk menanggapi permintaan dari yang lain. Waktu respon dimulai ketika pengguna mengirim permintaan dan berakhir saat waktu aplikasi menyatakan bahwa permintaan tersebut telah selesai. Dalam pengujian ini perlu dipastikan bahwa pengguna situs/aplikasi mendapatkan respon dalam waktu yang dapat diterima[13].

Pengujian ini dilakukan untuk mengetahui waktu yang dibutuhkan oleh *Server* SSO dalam melayani pengguna untuk melakukan autentikasi. Pengujian *response time* dilakukan dengan melakukan proses autentikasi pada *Server* SSO. Komputer penguji (pengguna) dan *Server* SSO yang akan digunakan berada pada suatu *network* yang sama dan jumlah pengguna yang melakukan proses autentikasi secara bersamaan adalah satu pengguna, dua pengguna dan empat pengguna.

Loadtestingtool.com merupakan suatu website penyedia layanan *Web Application Performance Tool* (WAPT). Website ini menjelaskan terdapat 3 point penting dalam penilaian *response time testing* yaitu[13] :

- 0,1 detik adalah response time yang paling disukai. Pengguna akan merasakan bahwa aplikasi atau sistem merespon secara instan, dan tidak merasakan gangguan apapun.
- 1,0 detik adalah batas waktu respon yang diterima. Pengguna mungkin tidak merasakan gangguan apapun meskipun mereka merasakan beberapa penundaan. Waktu respon yang lebih dari 1 detik dapat mengganggu pengalaman pengguna.
- 10 Detik Ini adalah batas dimana *response time* menjadi tidak dapat diterima. Selain itu, studi terbaru menunjukkan bahwa jika waktu respons melebihi 8 detik, pengalaman pengguna akan sangat terganggu dan sebagian besar pengguna akan meninggalkan situs atau sistem.

Umumnya, waktu respon harus secepat mungkin yaitu dalam interval 0,1 – 1 detik. Namun, orang dapat beradaptasi dengan waktu respon yang lebih lambat, tetapi mereka tidak akan pernah senang dengan waktu tanggapan lebih dari 2 detik[13].

Maka dari itu pada pengujian *response time* target waktu yang akan diraih dari penelitian ini adalah dibawah 2 detik atau 2000 milidetik.

3.3.4 Pengujian *Load Test*

Load testing software adalah alat evaluasi untuk menentukan bagaimana suatu aplikasi akan bekerja ketika tingkat kerja mendekati batas spesifikasi aplikasi. Tujuan utama *Load testing* (pengujian beban) adalah untuk menentukan jumlah maksimum pekerjaan yang dapat ditangani sistem tanpa penurunan kerja secara signifikan. Pengujian ini dilakukan untuk mengetahui kinerja *Server* SSO dalam menangani autentikasi dalam jumlah pengguna yang banyak dalam waktu bersamaan. Pengujian *Load Test* dilakukan dengan melakukan proses autentikasi pada *Server* SSO. Komputer penguji (pengguna) dan *Server* SSO yang akan digunakan berada pada suatu *network* yang sama. Pengujian *load test* dilakukan dengan melakukan proses autentikasi pada *Server* SSO dengan jumlah 50,100 dan 200 *user* pada saat bersamaan. Target dari pengujian ini adalah 0 jumlah *error request* pada percobaan hingga 200 pengguna.