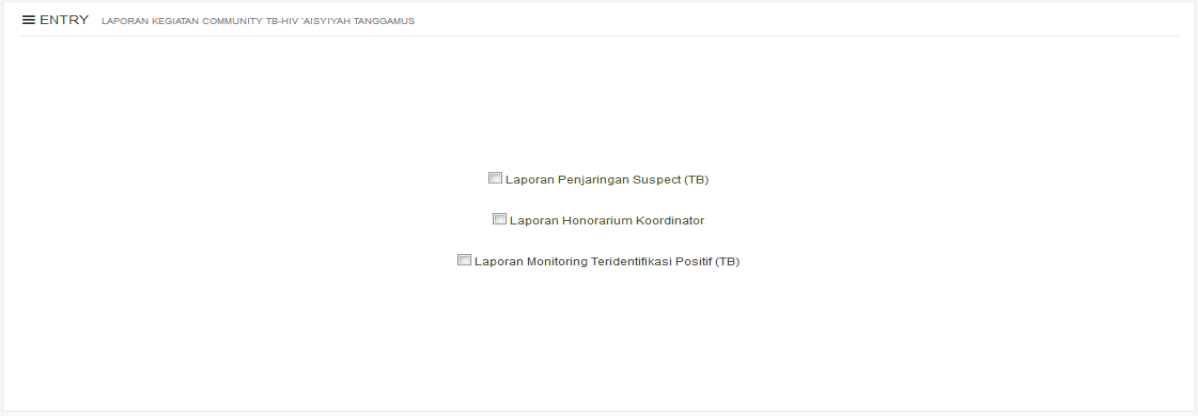
BAB V
IMPLEMENTASI DAN PENGUJIAN SISTEM

5.1 Implementasi Sistem

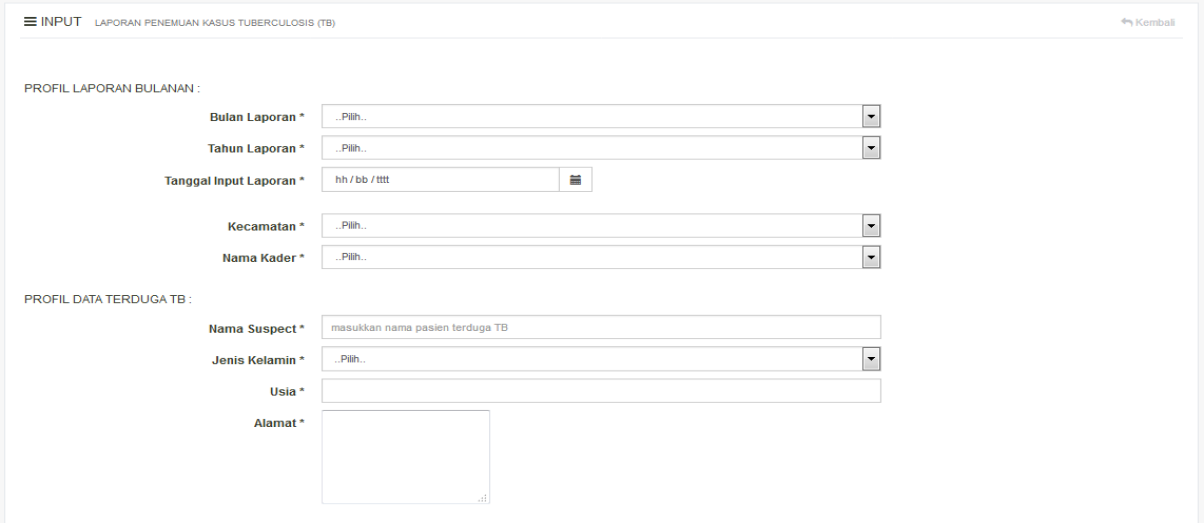
Implementasi sistem merupakan tahapan yang dilakukan untuk merepresentasikan hasil yang diperoleh dari tahap perancangan yang dikerjakan sebelumnya. Dalam implementasinya pengguna sistem informasi reporting adalah admin atau staff dc SSR Tanggamus, koordinator, dan kepala SSR Tanggamus yang sudah terdaftar dalam sistem. Sistem informasi reporting ini dibangun menggunakan Framework Codeignitter versi 3.1.8, Bootstrap, dan basis data MysQL. Adapun implementasi pada Sistem Informasi Reporting Community TB-HIV ‘Asiyiyah Tanggamus, adalah sebagai berikut.

5.1.1 Implementasi UC-01

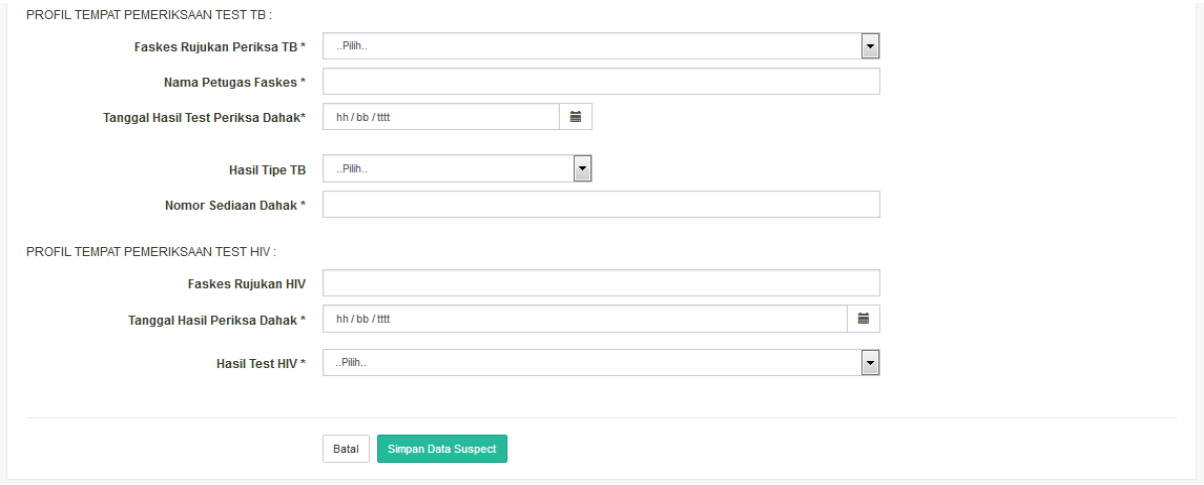
Halaman mengelola laporan penjaringan data suspect merupakan halaman yang digunakan oleh koordinator pada masing-masing kecamatan yang sudah terdaftar. Halaman ini bertujuan untuk menginputkan data pasien terduga (*suspect*) penyakit Tuberculosis (TB) dan menyimpan data terduga penyakit TB tersebut oleh koordinator. Disamping itu staff dc maupun koordinator kader dilapangan juga dapat mengelola laporan data suspect seperti merubah dan menghapus data apabila diperlukan. Adapun implementasi pada halaman memasukkan laporan penjaringan data suspect dapat dilihat pada Gambar 5.1 yang menunjukan halaman home untuk entry laporan kegiatan yang nantinya koordinator memilih menu untuk memasukkan laporan penjaringan suspect. Kemudian pada Gambar 5.2 dan 5.3 menunjukan halaman *form* untuk memasukkan data suspect TB. Dan Gambar 5.4 menunjukkan halam pencarian data suspect.



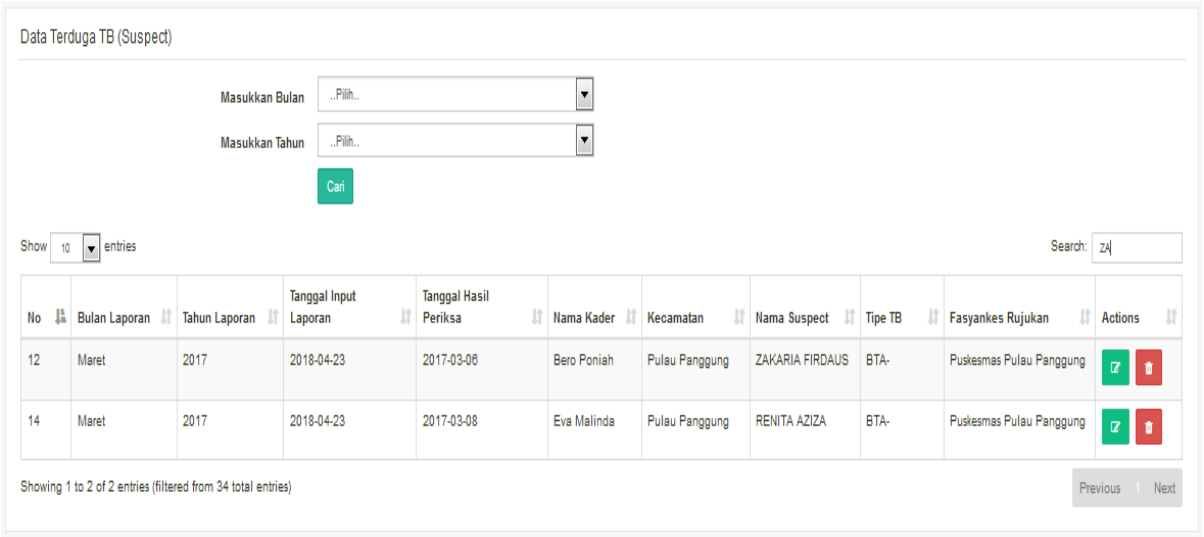
Gambar 5.1 Halaman Home Entry Laporan



Gambar 5.2 Form Tambah Data Suspect TB



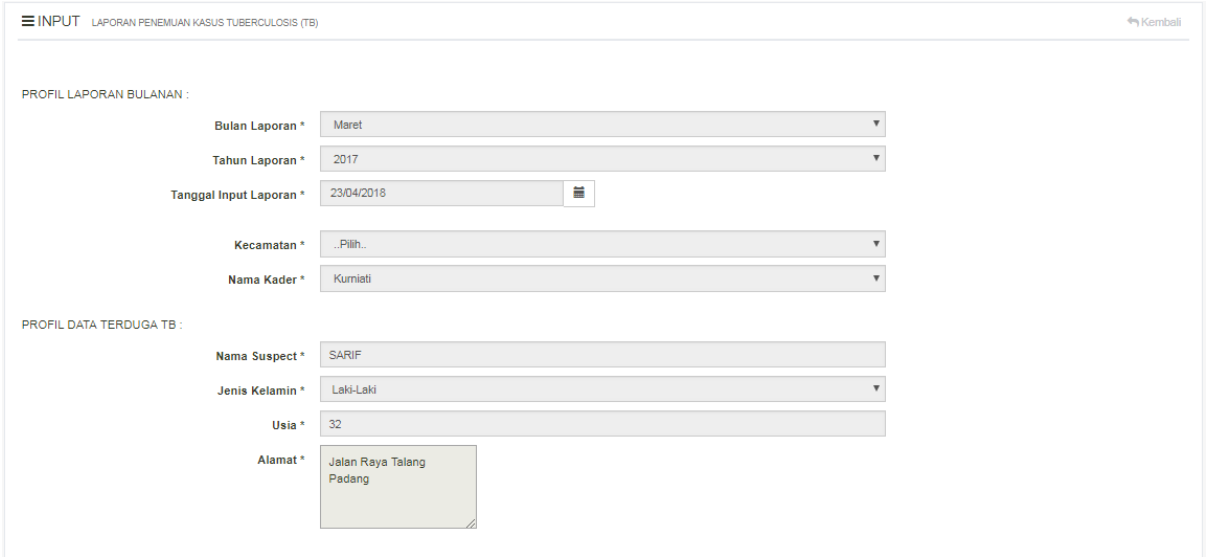
Gambar 5.3 Form Tambah Data Suspect TB



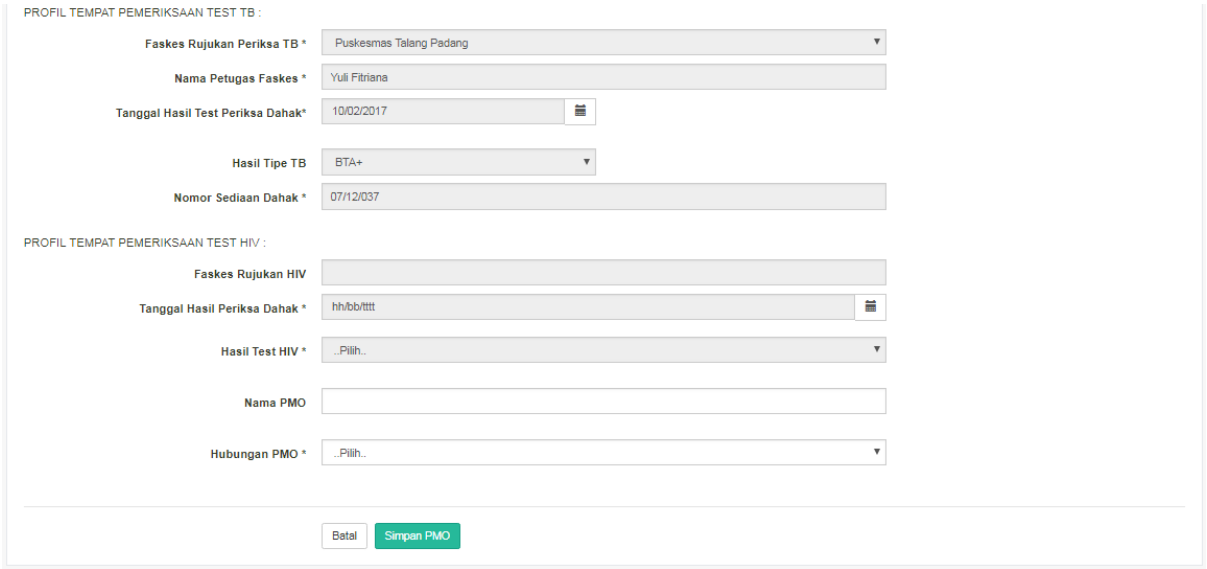
Gambar 5.4 Halaman Pencarian Suspect TB

5.1.2 Implementasi UC-02

Halaman mengelola data teridentifikasi TB-HIV digunakan oleh staff dc yang bertujuan untuk menambah data baru yaitu data pmo (pengawas menelan obat). Adapun implementasi pada halaman melakukan pencarian data suspect dapat dilihat pada Gambar 5.5 dan 5.6.



Gambar 5.5 Halaman form tambah pmo

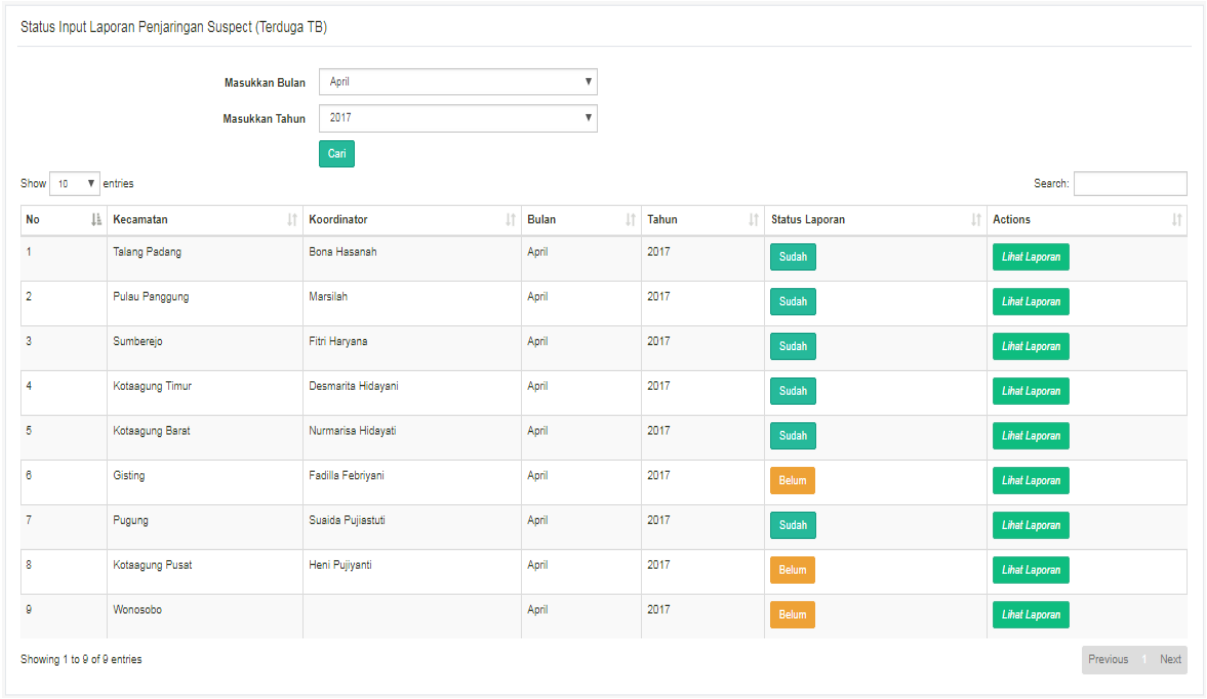


Gambar 5.6 Halaman form tambah pmo

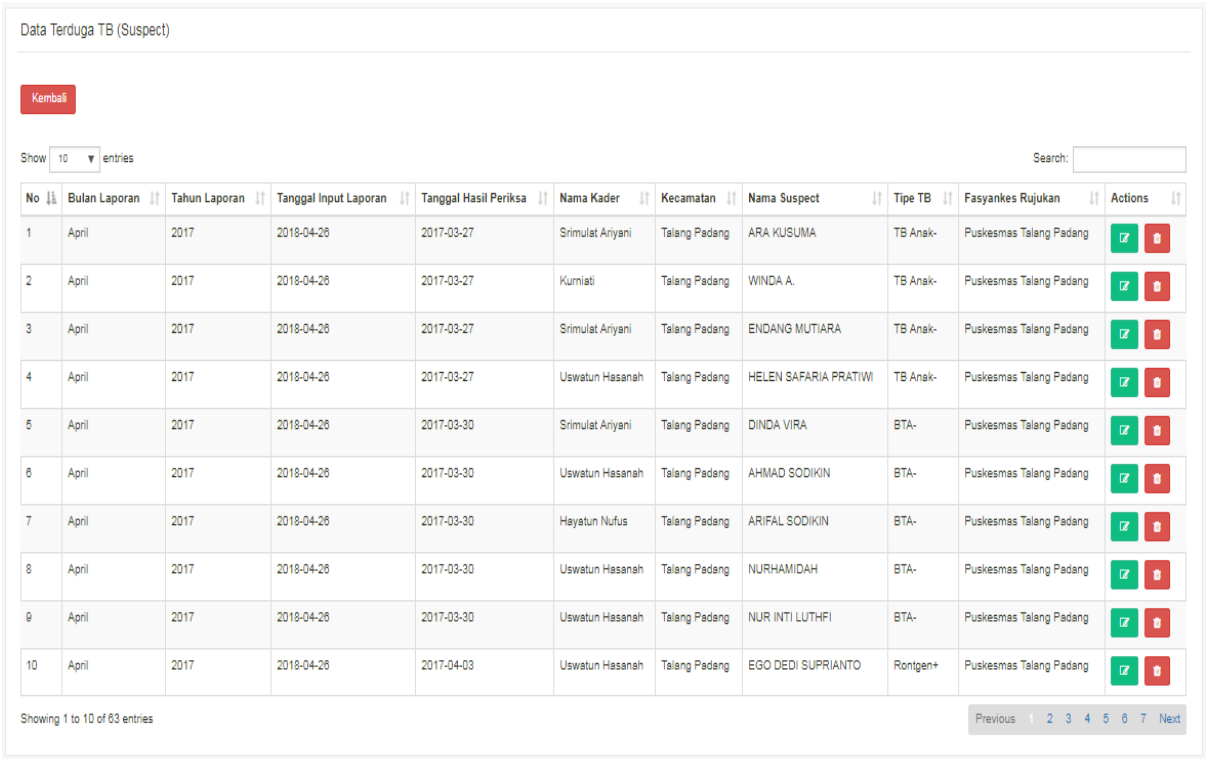
5.1.3 Implementasi UC-03

Halaman melihat status laporan masuk penjaringan *suspect* digunakan oleh Staff dc, merupakan halaman yang bertujuan untuk melakukan pengecekan terhadap status laporan masuk penjaringan suspect berdasarkan *entry* laporan penjaringan *suspect* oleh Koordinator.

Adapun implementasi sistem pada halaman melihat status laporan masuk dapat dilihat pada Gambar 5.7 dan 5.8 yang menunjukan beberapa hasil entry laporan data laporan penjaringan *suspect*.



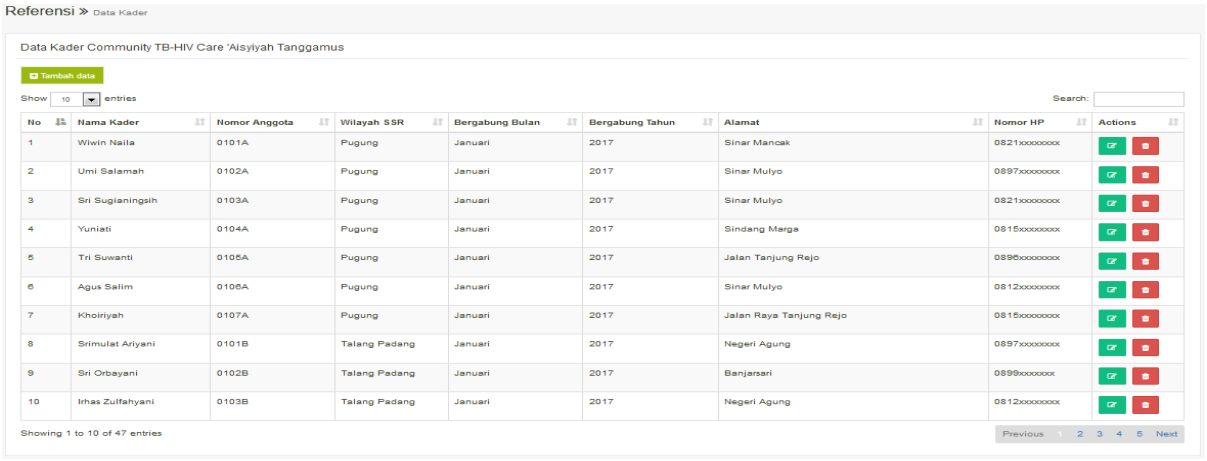
Gambar 5.7 Halaman Status *Entry* Laporan Penjaringa *Suspect*



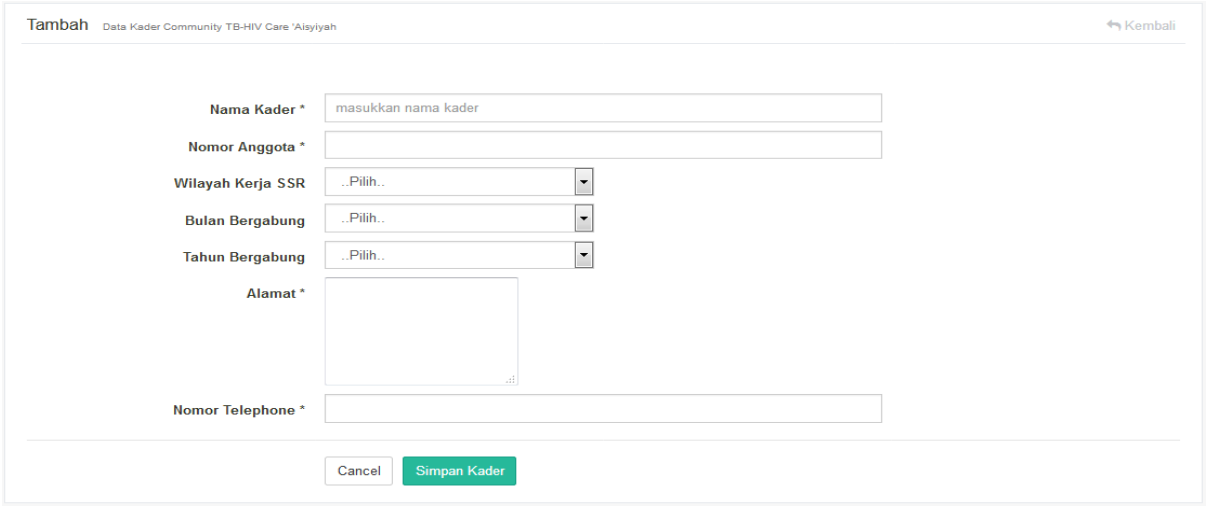
Gambar 5.8 Halaman Data *Suspect* TB

5.1.4 Implementasi UC-04

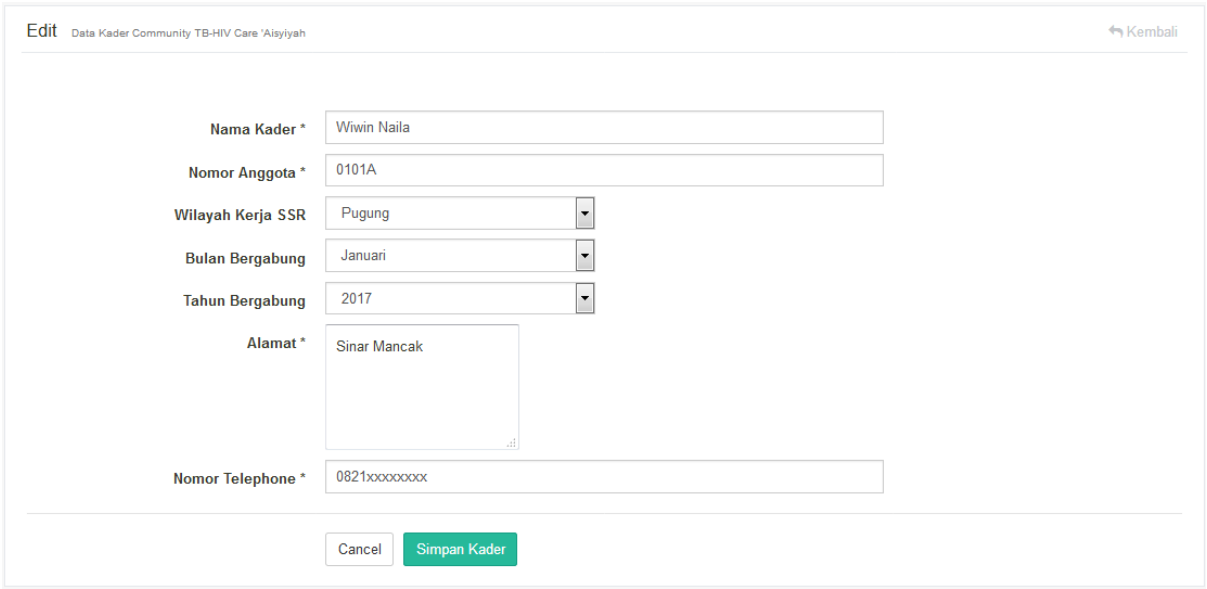
Halaman mengelola data referensi merupakan halaman yang digunakan oleh Admin atau *Staff* Data *Collection* SSR Tanggamus. Halaman data referensi terdiri dari 4 kateori yaitu data kecamatan, data faskes, data kader tb, dan data tipe tb. Namun dalam pembahasan ini dipilih salah satu yaitu mengelola data referensi (kader tb). Halaman mengelola data refrensi kader tb merupakan halaman yang bertujuan untuk menyimpan data 48 kader 'Aisyiyah yang tergabung dalam program kerja atau community TB-HIV 'Aisyiyah Tanggamus. Disamping itu mengelola data referensi kader TB seperti tambah, merubah, dan hapus data apabila diperlukan. Adapun implementasi sistem pada mengelola data referensi dapat dilihat pada Gambar 5.9 menunjukan halaman daftar data kader, Gambar 5.10 menunjukkan halaman form tambah kader, pada gambar 5.11 menunjukkan halaman form edit data kader, dan gambar 5.12 menunjukkan halaman konfirmasi untuk menghapus data kader.



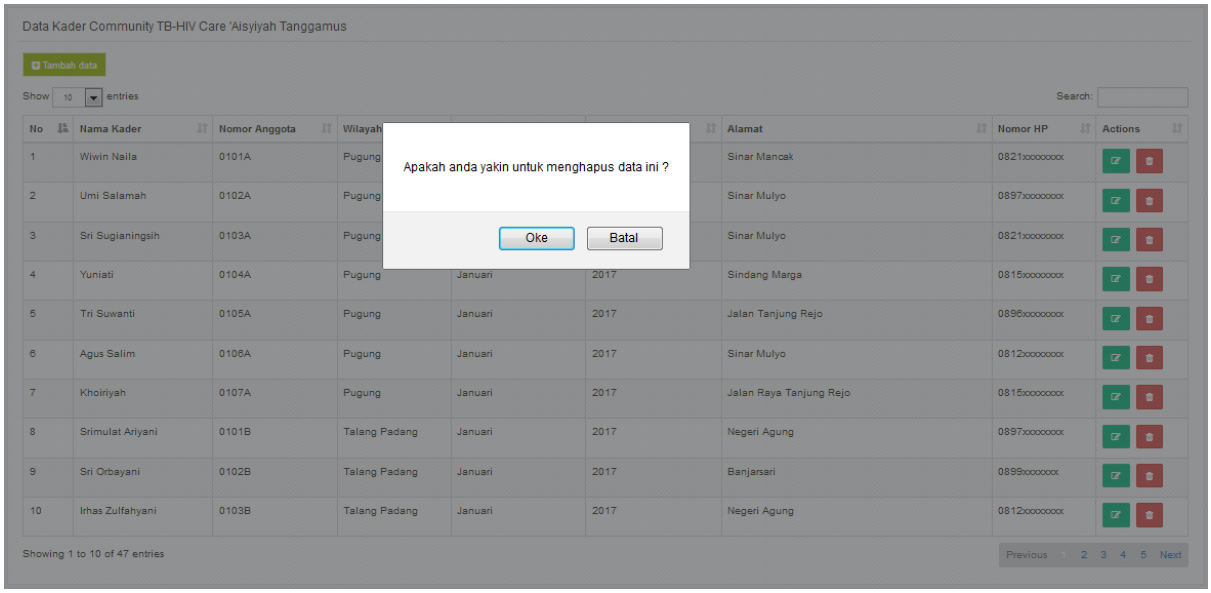
Gambar 5.9 Halaman Daftar Data Kader TB



Gambar 5.10 Halaman Form Tambah Data Kader



Gambar 5.11 Halaman Form Edit Data Kader

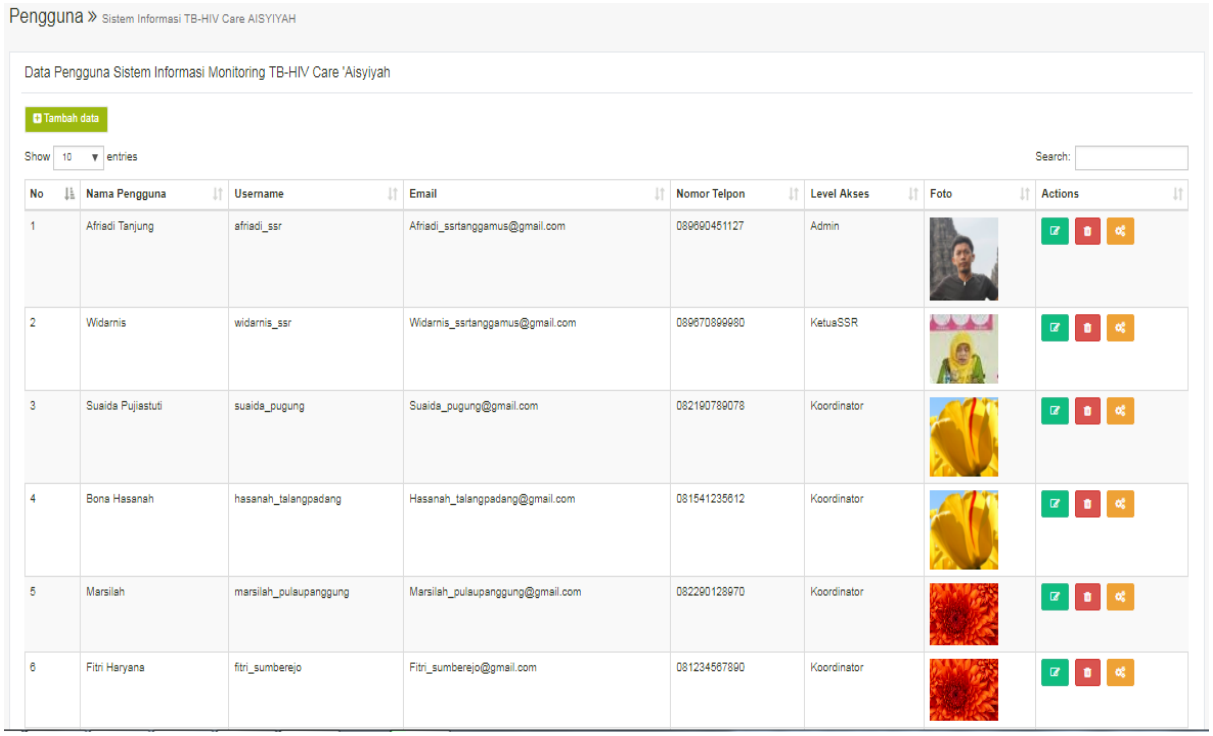


Gambar 5.12 Halaman Konfirmasi Hapus Data Kader

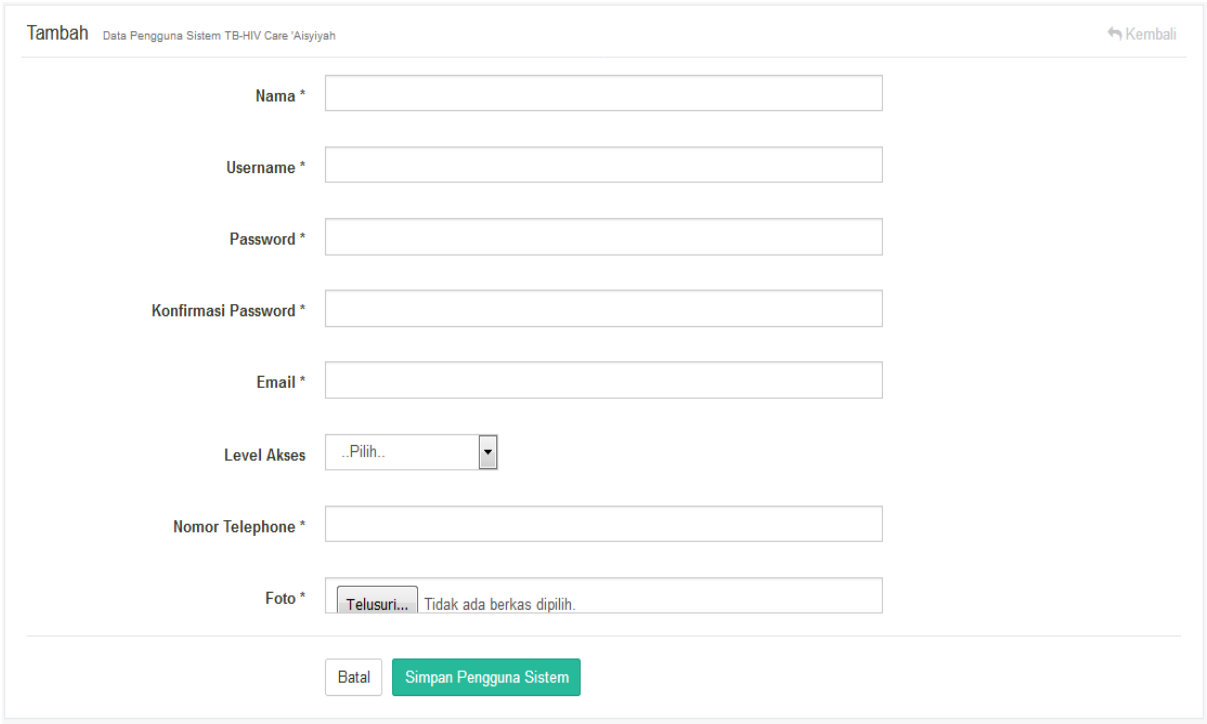
5.1.5 Implementasi UC-05

Halaman mengelola data pengguna sistem merupakan halaman yang digunakan oleh admin atau staff dc SSR Tanggamus. Halaman ini bertujuan untuk menambahkan data pengguna sistem yang mendapat akses ke dalam sistem informasi reporting community tb-hiv ‘Aisyiyah sesuai dengan level akses nya. Disamping itu mengelola data pengguna sistem seperti tambah, merubah data pengguna, dan menghapus data pengguna apabila diperlukan. Adapun implementasi mengelola data pengguna sistem dapat dilihat pada Gambar 5.13 yang menunjukkan halaman daftar pengguna sistem informasi reporting, Gambar 5.14 menunjukkan

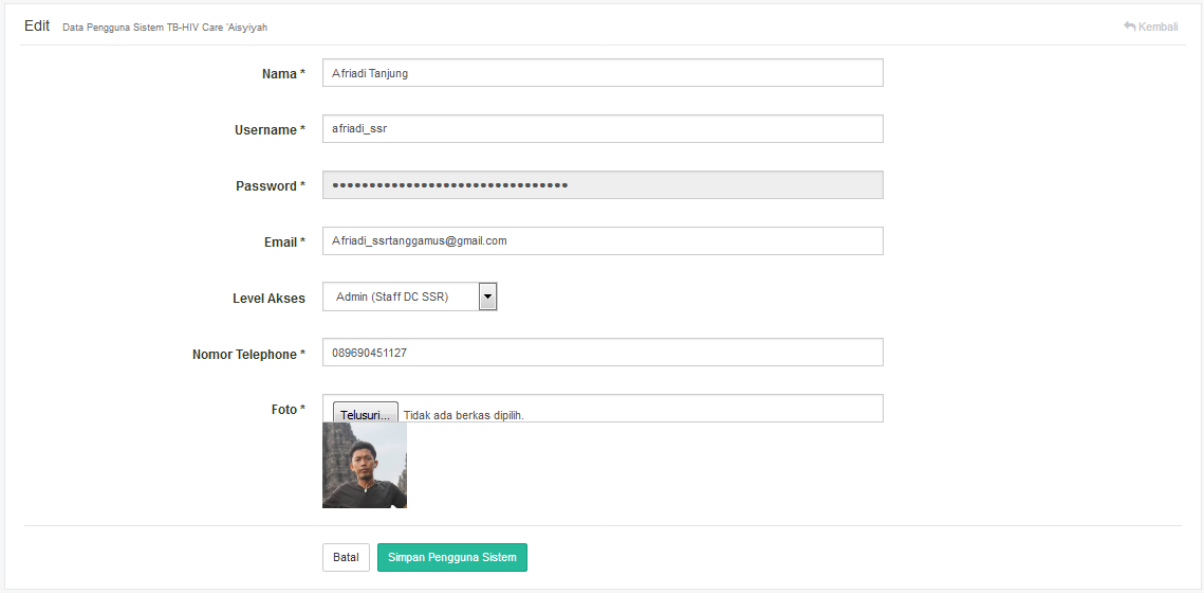
halaman form tambah pengguna sistem, Gambar 5.15 merupakan halaman merubah data pengguna sistem, dan Gambar 5.16 merupakan halaman konfirmasi hapus data pengguna.



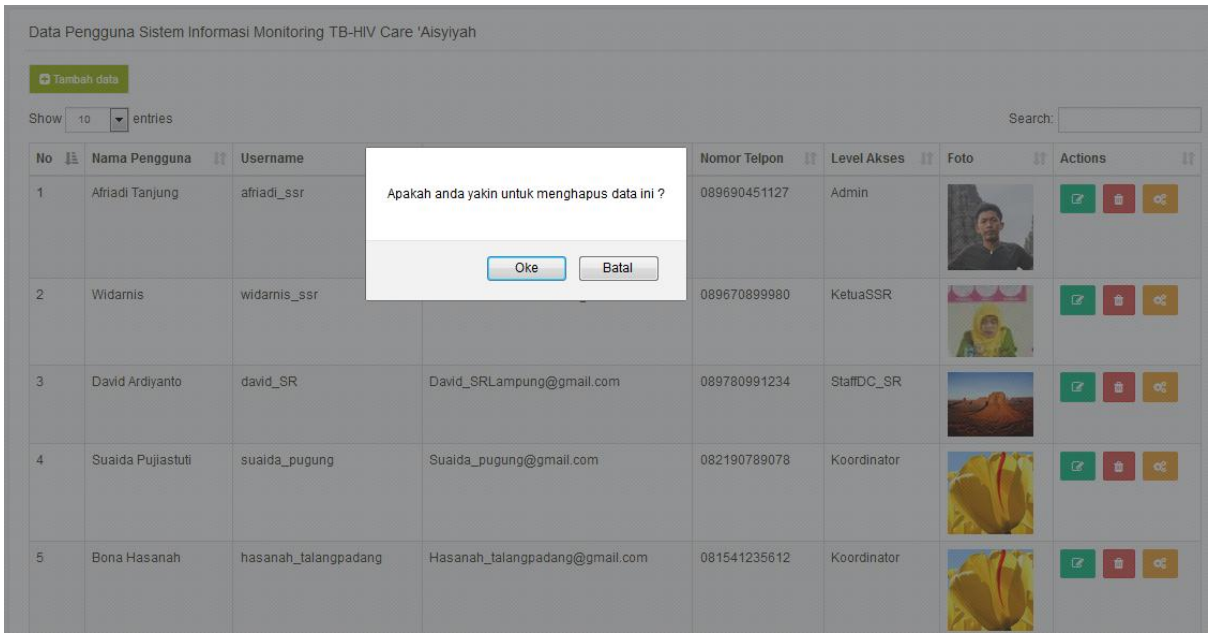
Gambar 5.13 Halaman Daftar Data Pengguna



Gambar 5.14 Halaman Form Tambah Pengguna



Gambar 5.15 Halaman Form Edit Pengguna

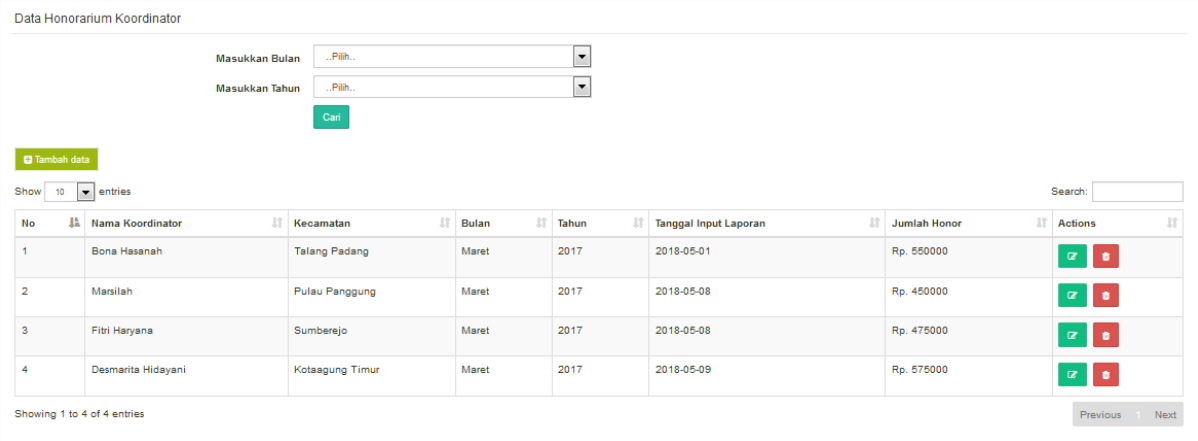


Gambar 5.16 Halaman Konfirmasi Data Pengguna

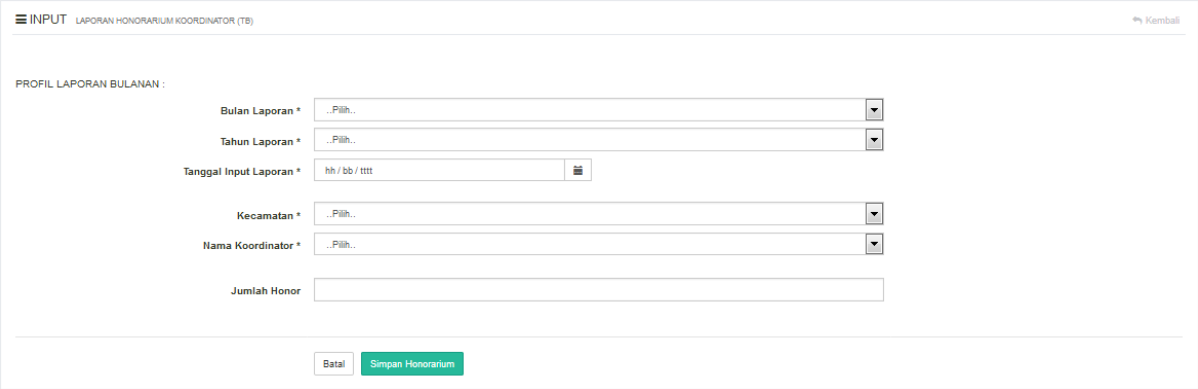
5.1.6 Implementasi UC-06

Halaman mengelola laporan honorarium koordinator merupakan halaman yang diakses oleh staff dc SSR. Halaman ini bertujuan untuk menampilkan data honorarium koordinator. Pada halaman ini staff dc ssr dapat mengelola data rekapitulasi honorarium koordinator seperti tambah, mengubah data, dan hapus. Adapun implementasi pada mengelola rekapitulasi honorarium koordinator dapat dilihat pada Gambar 5.17 yang menunjukkan data honorarium koordinator, Gambar 5.18 menunjukkan halaman form tambah data laporan honorarium

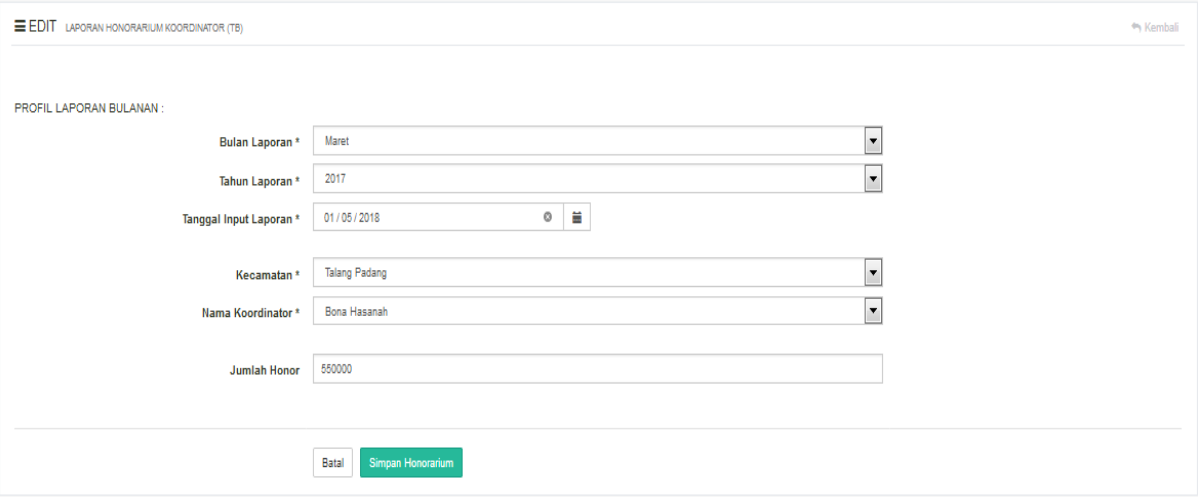
koordinator, Gambar 5.19 menunjukkan halaman form edit data laporan honorarium koordinator, dan Gambar 5.20 halaman konfirmasi hapus data honorarium koordinator.



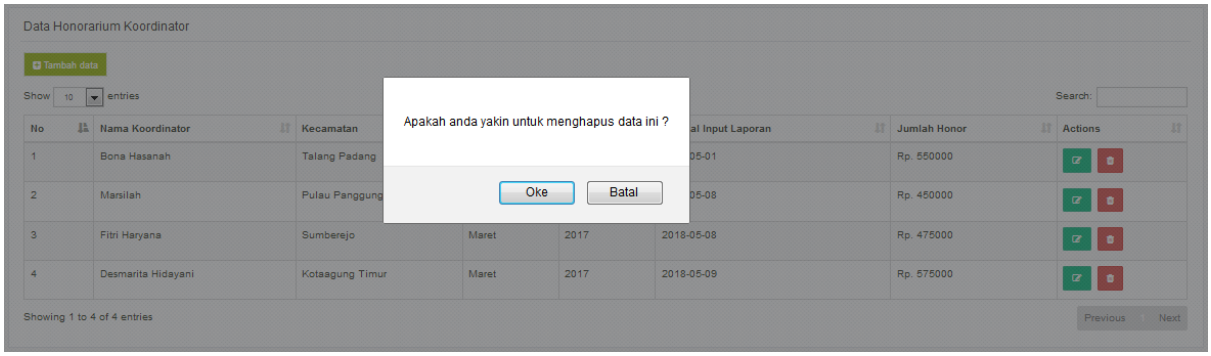
Gambar 5.17 Halaman Data Honorarium Koordinator



Gambar 5.18 Halaman Form Tambah Honorarium Koordinator



Gambar 5.19 Halaman Form Edit Honorarium Koordinator



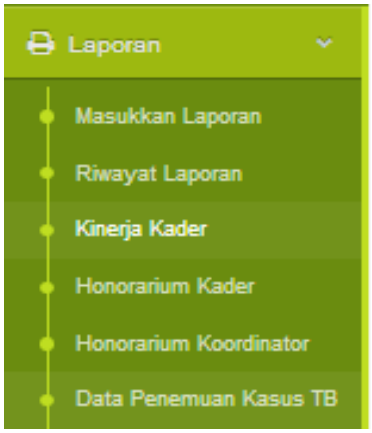
Gambar 5.20 Halaman Konfirmasi Hapus Data

5.1.7 Implementasi UC-07

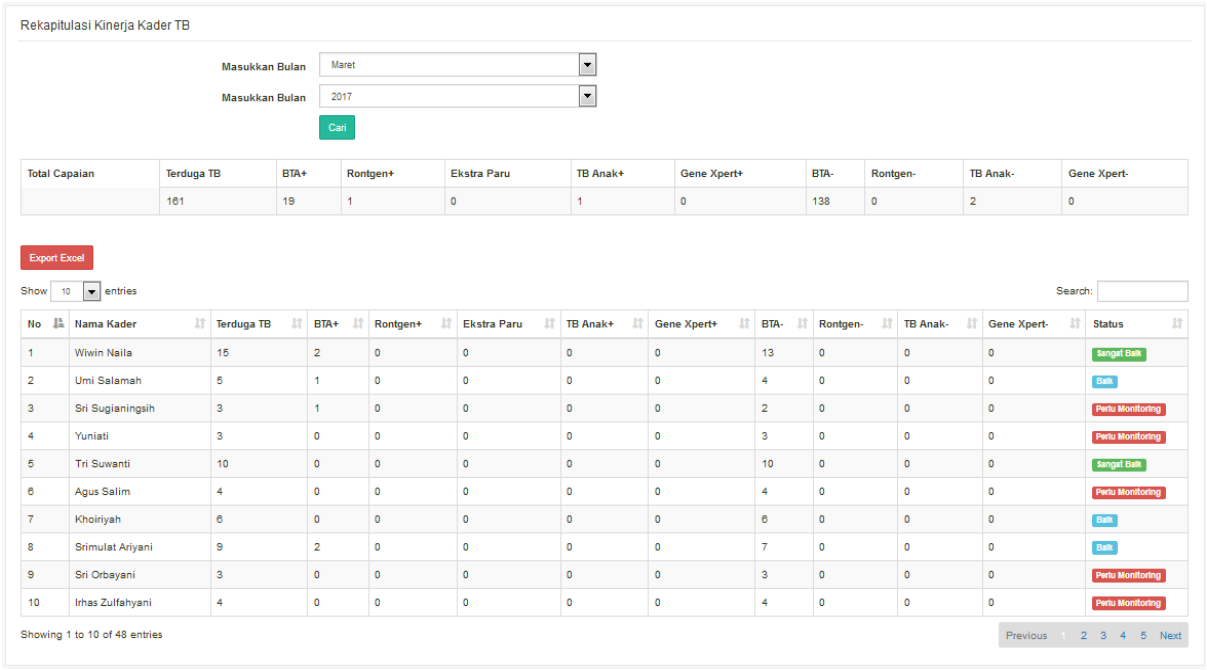
Halaman melihat rekapitulasi laporan bulanan merupakan halaman yang digunakan oleh staff dc dan kepala SSR Tanggamus. Beberapa rekapitulasi laporan bulanan meliputi rekapitulasi kinerja kader, rekapitulasi laporan honorarium kader, honorarium koordinator, dan rekapitulasi data penemuan kasus TB-HIV. Halaman ini bertujuan untuk menampilkan seluruh data rekapitulasi laporan, Adapun penjelasan mengenai implementasi sistem melihat rekapitulasi laporan bulanan adalah sebagai berikut:

a. Halaman rekapitulai kinerja kader

Halaman ini bertujuan untuk mengetahui berapa capaian yang diperoleh oleh masing-masing kader dengan menilai seberapa aktif kader pada kegiatan community tb-hiv ‘Aisyiyah SSR Tanggamus. Adapun beberapa menu dalam mengelola rekapitulasi laporan dapat dilihat pada Gambar 5.21 dan implementasi sistem halaman rekapitulasi kinerja kader dapat dilihat pada Gambar 5.22 yang menunjukkan data rekapitulasi pada kinerja kader community tb-hiv care ‘Aisyiyah Tanggamus.



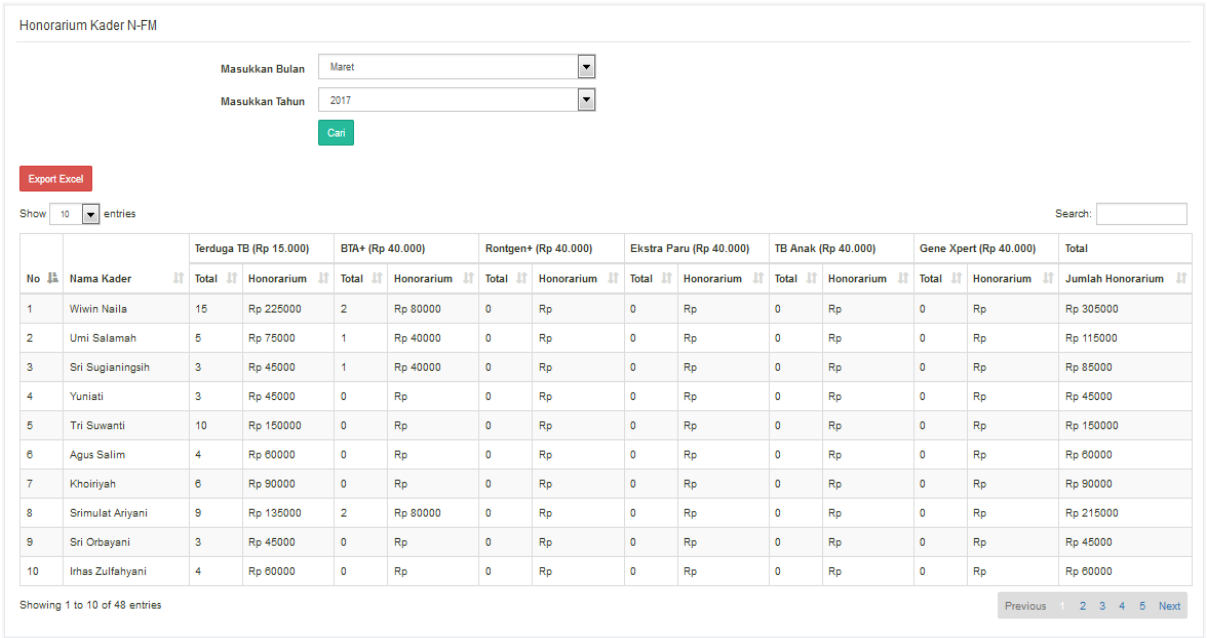
Gambar 5.21 Menu Laporan



Gambar 5.22 Halaman Rekapitulasi Kinerja Kader

b. Halaman rekapitulasi honorarium kader

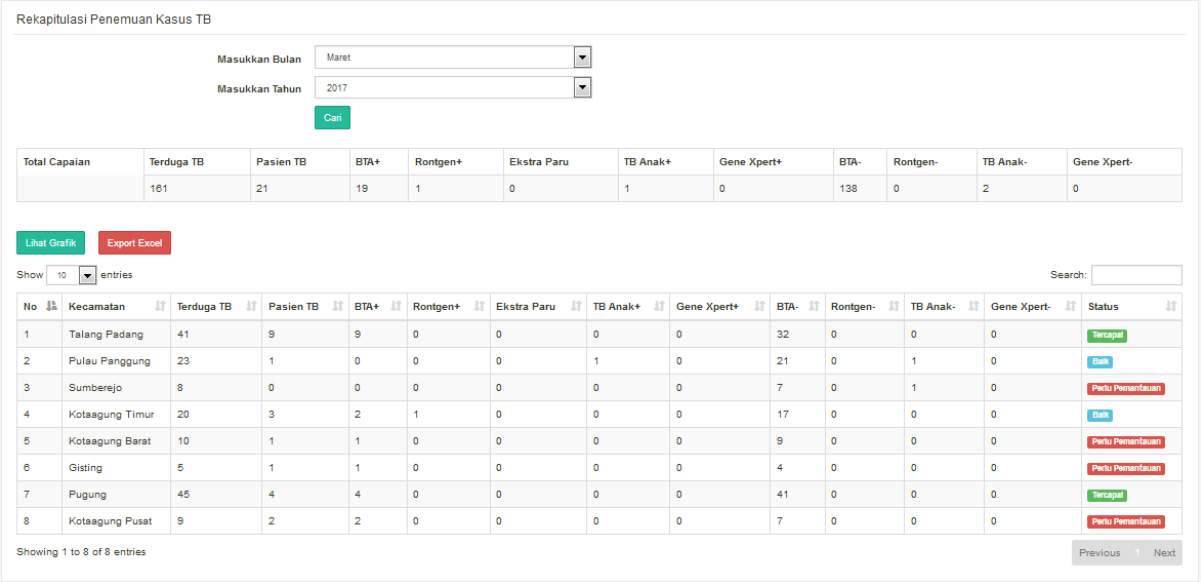
Halaman rekapitulasi honorarium kader merupakan halaman yang diakses oleh staff dc ssr dan kepala ssr. Halaman ini bertujuan untuk mengetahui honorarium yang diperolah oleh masing-masing kader community tb-hiv care ‘Aisyiyah Tanggamus. Adapun implementasi sistem rekapitulasi honorarium kader dapat dilihat pada Gambar 5.23.



Gambar 5.23 Halaman Rekapitulasi Honorarium Kader

c. Halaman rekapitulasi penemuan kasus TB

Halaman rekapitulasi penemuan kasus TB merupakan halaman yang bertujuan untuk menampilkan data rekapitulasi penemuan kasus TB pada setiap daerah atau kecamatan di kabupaten Tanggamus yang berasal dari laporan penjaringan *suspect*. Adapun implementasi sistem halaman rekapitulasi penemuan kasus TB dapat dilihat pada Gambar 5.24.

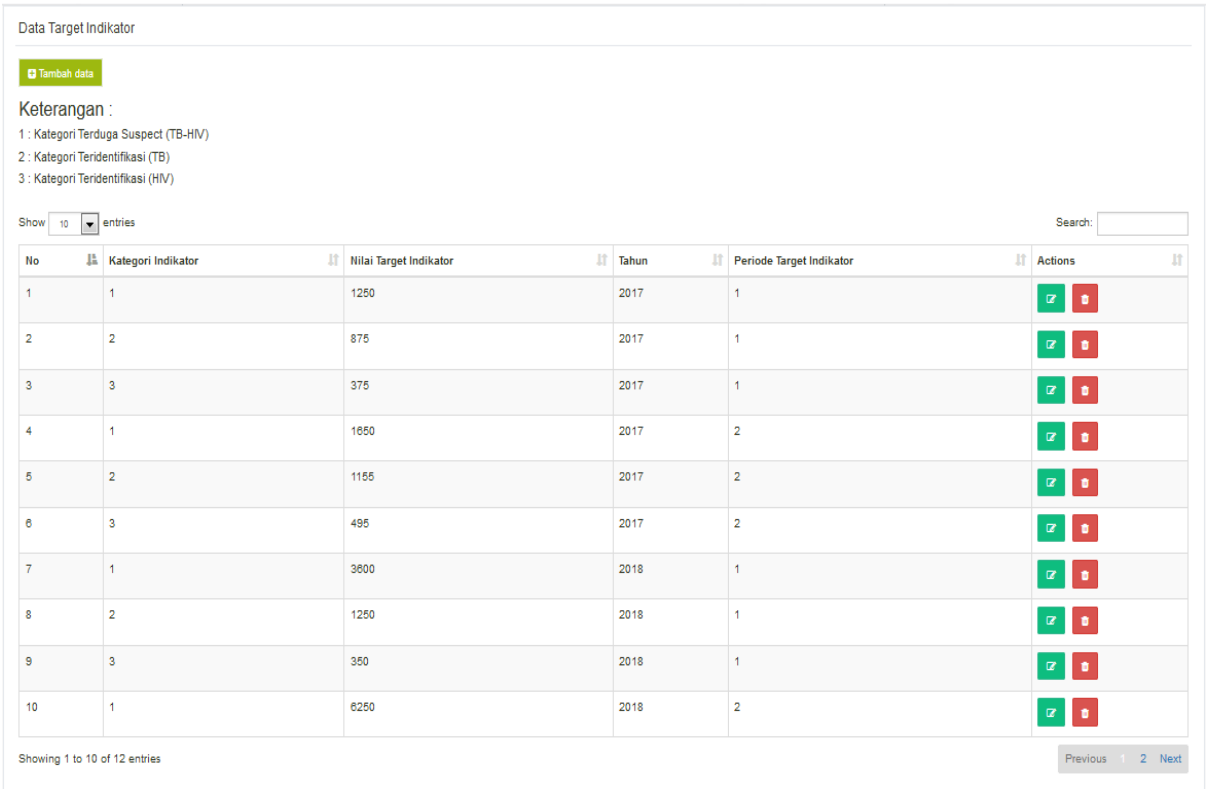


Gambar 5.24 Halaman Rekapitulasi Penemuan Kasus TB

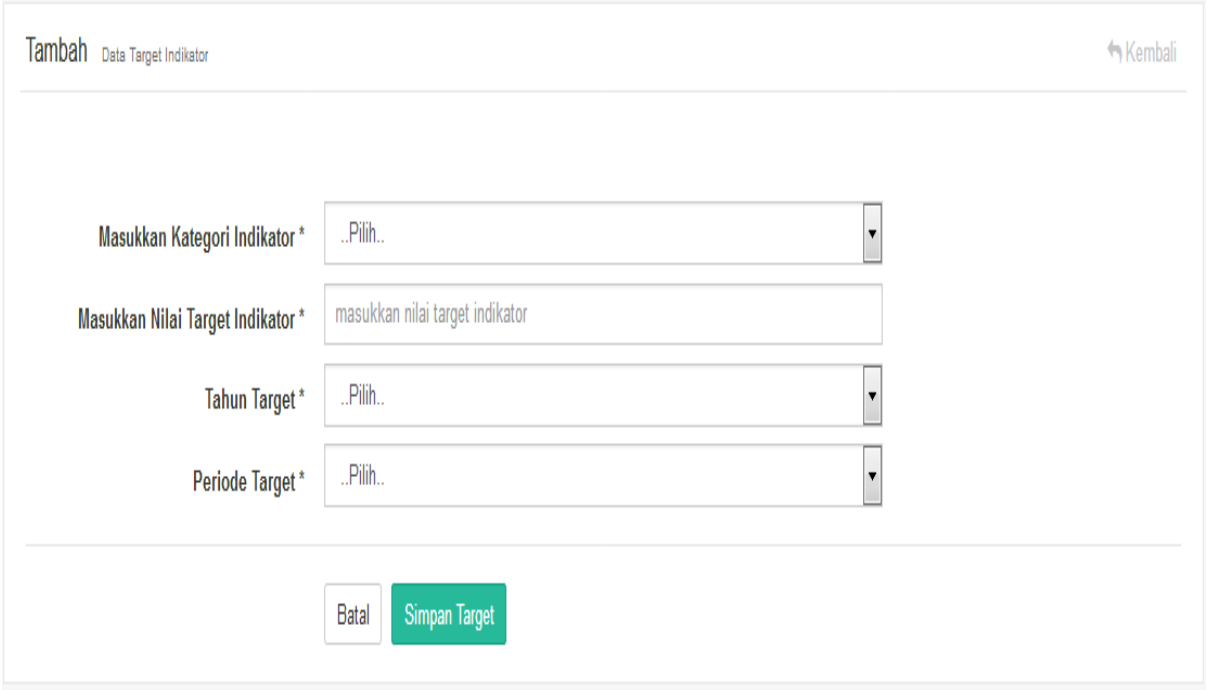
5.1.8 Implementasi UC-08

Halaman mengelola target indikator dan capaian indikator merupakan halaman yang digunakan oleh staff dc SSR Tanggamus. Halaman ini bertujuan untuk mengelola data target indikator, seperti tambah target, merubah target dan menghapus apabila diperlukan. Disamping itu tujuan halaman manajemen target untuk mengetahui detail pengukuran target indikator dan capaian yang diperoleh pada setiap indikator dari target yang ditentukan sebelumnya. Adapun implementasi sistem mengelola target indikator dapat dilihat pada Gambar 5.25 menunjukkan halaman data target indikator pada setiap kategori indikator seperti jumlah target yang ditentukan pada setiap semester untuk penjaringan suspect tb, teridentifikasi tb, dan teridentifikasi hiv. Pada Gambar 5.26 menunjukkan halaman form tambah target indikator dan Gambar 5.27 merupakan halaman yang menunjukkan detail pengukuran target. pada Gambar 5.28 merupakan halaman yang menunjukkan data capaian yang diperoleh pada setiap kecamatan untuk pertriwulan dan Gambar 5.29 merupakan halaman yang menunjukkan data capaian untuk seluruh kategori indikator dalam 1 tahun. Sedangkan pada Gambar 5.30

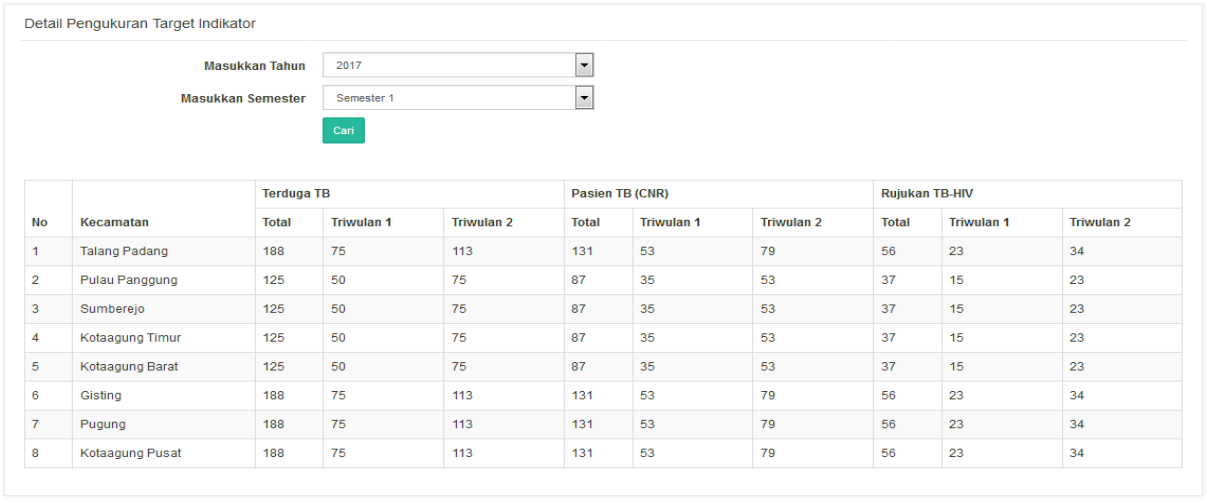
merupakan halaman yang menunjukkan grafik capaian yang diperoleh pada setiap kecamatan, dan Gambar 5.31 menunjukkan grafik capaian dari setiap tipe tb.



Gambar 5.25 Halaman Data Target Indikator



Gambar 5.26 Halaman Form Tambah Target



Detail Pengukuran Target Indikator

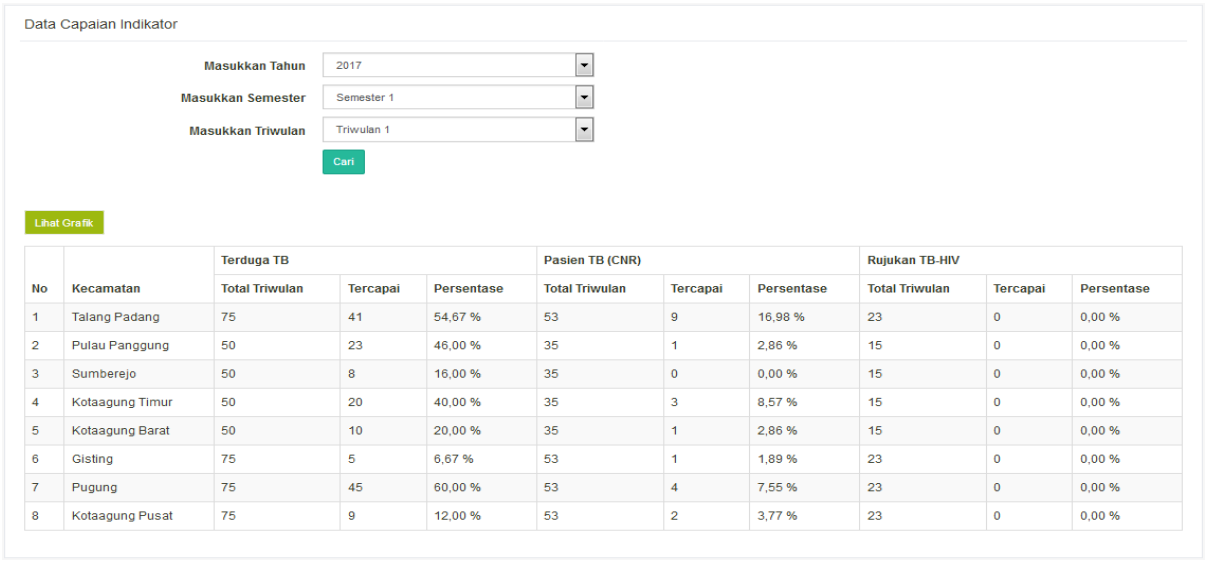
Masukkan Tahun: 2017

Masukkan Semester: Semester 1

Cari

No	Kecamatan	Terduga TB			Pasien TB (CNR)			Rujukan TB-HIV		
		Total	Triwulan 1	Triwulan 2	Total	Triwulan 1	Triwulan 2	Total	Triwulan 1	Triwulan 2
1	Talang Padang	188	75	113	131	53	79	56	23	34
2	Pulau Panggung	125	50	75	87	35	53	37	15	23
3	Sumberejo	125	50	75	87	35	53	37	15	23
4	Kotaagung Timur	125	50	75	87	35	53	37	15	23
5	Kotaagung Barat	125	50	75	87	35	53	37	15	23
6	Gisting	188	75	113	131	53	79	56	23	34
7	Pugung	188	75	113	131	53	79	56	23	34
8	Kotaagung Pusat	188	75	113	131	53	79	56	23	34

Gambar 5.27 Halaman Detail Pengukuran Target



Data Capaian Indikator

Masukkan Tahun: 2017

Masukkan Semester: Semester 1

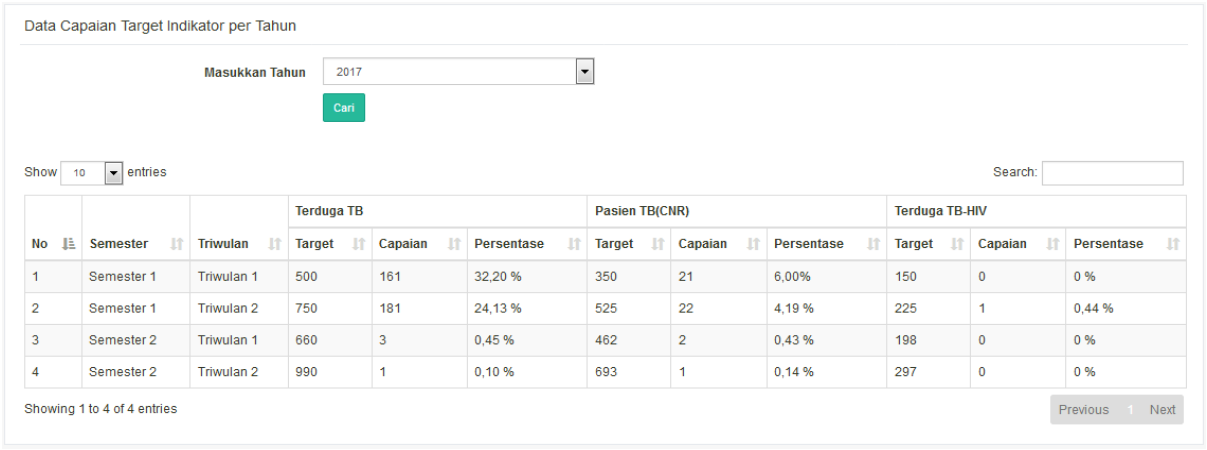
Masukkan Triwulan: Triwulan 1

Cari

Lihat Grafik

No	Kecamatan	Terduga TB			Pasien TB (CNR)			Rujukan TB-HIV		
		Total Triwulan	Tercapai	Persentase	Total Triwulan	Tercapai	Persentase	Total Triwulan	Tercapai	Persentase
1	Talang Padang	75	41	54,67 %	53	9	16,98 %	23	0	0,00 %
2	Pulau Panggung	50	23	46,00 %	35	1	2,86 %	15	0	0,00 %
3	Sumberejo	50	8	16,00 %	35	0	0,00 %	15	0	0,00 %
4	Kotaagung Timur	50	20	40,00 %	35	3	8,57 %	15	0	0,00 %
5	Kotaagung Barat	50	10	20,00 %	35	1	2,86 %	15	0	0,00 %
6	Gisting	75	5	6,67 %	53	1	1,89 %	23	0	0,00 %
7	Pugung	75	45	60,00 %	53	4	7,55 %	23	0	0,00 %
8	Kotaagung Pusat	75	9	12,00 %	53	2	3,77 %	23	0	0,00 %

Gambar 5.28 Halaman Capaian Indikator Kecamatan



Data Capaian Target Indikator per Tahun

Masukkan Tahun: 2017

Cari

Show 10 entries

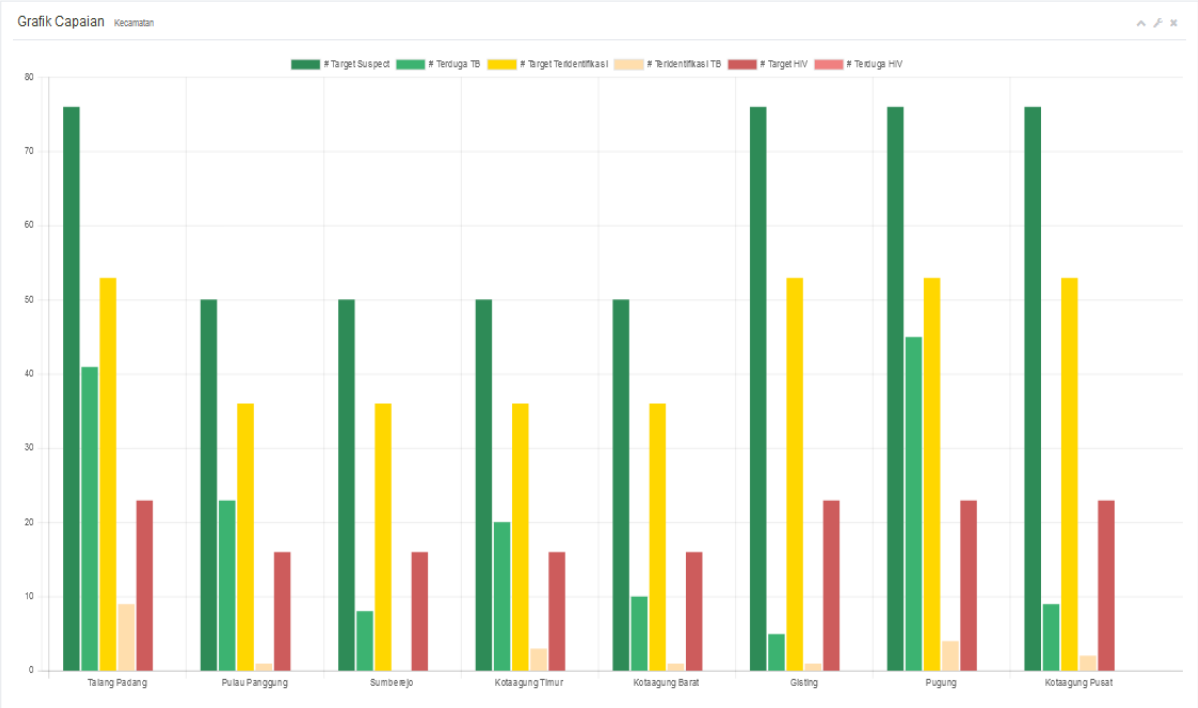
Search:

No	Semester	Triwulan	Terduga TB			Pasien TB(CNR)			Terduga TB-HIV		
			Target	Capaian	Persentase	Target	Capaian	Persentase	Target	Capaian	Persentase
1	Semester 1	Triwulan 1	500	161	32,20 %	350	21	6,00%	150	0	0 %
2	Semester 1	Triwulan 2	750	181	24,13 %	525	22	4,19 %	225	1	0,44 %
3	Semester 2	Triwulan 1	660	3	0,45 %	462	2	0,43 %	198	0	0 %
4	Semester 2	Triwulan 2	990	1	0,10 %	693	1	0,14 %	297	0	0 %

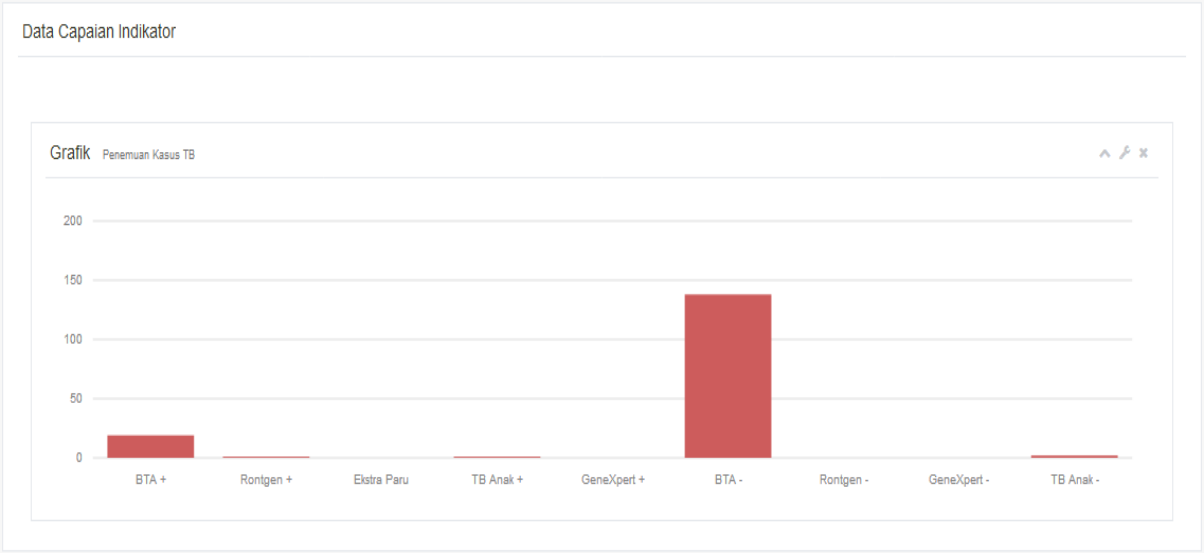
Showing 1 to 4 of 4 entries

Previous 1 Next

Gambar 5.29 Halaman Capaian Indikator



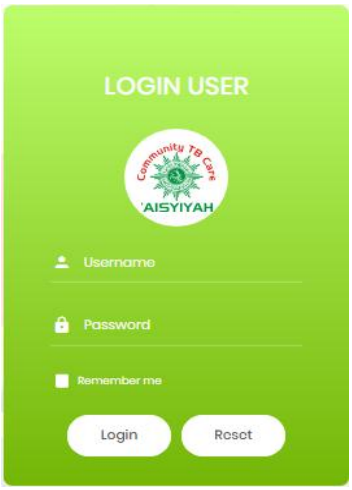
Gambar 5.30 Grafik Capaian Kecamatan



Gambar 5.31 Grafik Capaian Tipe TB

5.1.9 Implementasi KNF-01 (Login)

Halaman login merupakan halaman masuk untuk pengguna sistem yang akan menggunakan sistem informasi reporting tb-hiv care. Baik admin, koordinator, dan kepala SSR akan memberikan masukkan username dan password yang sudah terdaftar dalam sistem. Halaman login dapat dilihat pada Gambar 5.32.



Gambar 5.32 Halaman *Login*

5.2 Pengujian Sistem Informasi Reporting

Pengujian sistem meruapakan tahapan terakhir dalam suatu pengembangan perangkat lunak. Pengujian merupakan bagian terpenting dalam suatu perangkat lunak, pengujian dilakukan untuk mendeteksi kesalahan-kesalahan yang mungkin ditemukan pada implementasi kode program yang dibuat. Disamping itu pengujian dilakukan dengan tujuan untuk memastikan apakah semua fungsi yang ada pada sistem informasi reporting dapat berjalan baik sebagaimana mestinya dan sudah memenuhi kriteria yang sesuai dengan tujuan perancangan.

5.2.1 Rencana Pengujian Sistem

Dalam penelitian ini akan dilakukan pengujian dengan menggunakan metode *white box testing* yang didasarkan pada pengujian dengan melihat *source code* pada perangkat lunak yang dibuat. Pengujian sistem menggunakan *white box testing* dilakukan untuk meyakinkan semua *statement* (baris kode) program dan semua kondisi logis dieksekusi secara minimal. Rencana pengujian yang dilakukan yaitu dengan menentukan data-data yang akan digunakan untuk keperluan pengujian dan menentukan method pada masing-masing use case pada sistem. Adapun beberapa daftar *usecase* dan method *function* yang terdapat dalam setiap *class* dapat dilihat pada Tabel 5.1.

Tabel 5.1 Tabel Daftar Method

No	Nama File Controller	Nama Method	Jumlah *Statement*	Jumlah Kondisi
UC-01	Datasuspect	+ index()	12	0
		+ tampil_datasuspect()	24	3
		+ edit_suspect()	60	1

		+ simpan	3	1
		+ delete	10	1
UC-02	Datasuspect	+ tampil_teridentifikasi_tb()	24	3
		+ tampil_teridentifikasi_hiv()	24	3
		+ edit_pmo()	62	1
		+ simpan_pmo	17	1
		+ delete	10	1
UC-03	Home	+ index()	22	2
UC-04	Kecamatan	+ index()	8	0
		+ tambah()	7	0
		+ edit()	20	1
		+ simpan()	15	1
		+ delete()	10	1
	Faskes	+ index()	8	0
		+ tambah()	8	0
		+ edit()	30	1
		+ simpan()	20	1
		+ delete()	10	1
	Kader	+ index()	8	0
		+ tambah()	8	0
		+ edit()	35	1
		+ simpan()	22	1
		+ delete()	10	1
	Tipetb	+ index()	8	0
		+ tambah()	7	0
		+ edit()	22	1
		+ simpan()	16	1
		+ delete()	10	1
UC-05	Datauser	+ index()	8	0
		+ tambah()	7	0
		+ edit_user()	36	1
		+ editpassword()	29	1
		+ simpan()	36	1
		+ simpan_password()	9	0
		+ delete()	10	1
UC-06	Laporan_honorarium	+ index()	10	0
		+ tampil_honorkoordinator()	22	1
		+ edit()	35	1
		+ simpan()	23	1
		+ delete()	10	1
UC-07	Laporan_honorarium	+ tampil_honorariumkader()	18	3
		+ tampil_kinerjakader()	50	3

		+ excel_honorariumkader($bulan_laporan, $tahun_laporan)	14	1
		+ excel_kinerjakader($bulan_laporan, $tahun_laporan)	44	1
	Laporan_kasus_tb	+ index()	54	4
		+ excel_penemuan_kasustb($bulan_laporan, $tahun_laporan)	50	1
UC-08	Target	+ index()	11	0
		+ tambah()	8	0
		+ tampil_target()	23	1
		+ edit_target()	28	3
		+ simpan()	19	1
		+ delete()	10	1
	Capaian	+ index()	45	5
		+ tampil_capaian_kec()	52	5
	Grafik	+ tampil_grafik_kecamatan()	57	5
KNF-01	Login	+ index()	4	0
		+ getlogin()	14	2
		+ logout()	4	0

5.2.2 Pengujian *White Box Testing*

Pada tahap ini pengujian dilakukan baik dari segi logika kode program maupun fungsi-fungsi yang pada sistem. Pengujian *white box testing* dilakukan dengan cara melihat kedalam modul untuk melihat *source code* dari perangkat lunak dan melakukan analisa apakah ada kesalahan atau tidak pada logikanya. Adapun teknik dari *white box testing* yang digunakan yaitu *statement coverage* dan *branch coverage*, untuk memastikan bahwa setiap pernyataan baris kode dan setiap pernyataan *if* telah dieksekusi dengan tepat sehingga menghasilkan keluaran nilai yang benar dan valid. Uji coba sistem mengacu pada 8 *usecase* dan 1 KNF (Kebutuhan non-fungsionalitas) yang ada. Semua *source code* program pada sistem informasi reporting pada community TB-HIV care Aisyiyah telah dilakukan pengujian, namun karena keterbatasan dalam penulisan di dalam laporan, maka hanya beberapa method yang disampaikan penulis ke dalam laporan Tugas Akhir ini. Adapun tahapan-tahapan pengujian yang dilakukan dengan teknik *statement* dan *branch coverage testing* antara lain:

1. Langkah Persiapan
   a. Menentukan *source code* atau blok kode program (*function*) yang akan diuji pada file *controller*.
   b. Mengkonversikan *source code* tersebut ke dalam bentuk *flowgraph.*

c. Menghitung nilai *Cyclomatic Complexity* (cc) untuk menentukan jumlah jalur *independent path* yang terlewati pada *source code*.
d. Mengidentifikasi setiap jalur independent dengan membuat sebuah skenario uji.
e. Menentukan Rancangan data uji dengan memberikan masukkan data uji dan expected result yang diharapkan.

2. *Statement coverage*, teknik pengujian ini dilakukan dengan menjalankan data uji yang mencakup semua baris kode dijalankan atau dengan peninjauan terhadap berapa jumlah *statement* yang tercakup ketika data uji dijalankan atau *statement* mana saja yang dieksekusi ketika data uji dijalankan, jika berdasarkan pada *flowgraph* dengan melihat titik-titik (*nodes*) pada *flowgraph* yang dilalui oleh jalur-jalur dari *test case* yang dipilih. Pengujian dilakukan guna memastikan setiap *statement* pada *source code* yang dilakukan pengujian terbebas dari kesalahan logika program, kemudian menghitung persentase (nilai *coverage*) dari jumlah pernyataan (*statement*) yang telah di eksekusi.
3. *Branch coverage*, melakukan pengujian dengan peninjauan terhadap anak panah cabang atau *statement*-IF pada kode program, jika berdasarkan pada flowgraph melihat *branch edges* pada *flowgraph* yang dilalui oleh jalur-jalur dari test case yang dipilih. Pengujian dengan *branch coverage* dilakukan untuk memastikan setiap pelaksanaan keputusan atau cabang pada kondisi true dan false yang masing-masing di eksekusi setidaknya satu kali test dan menghitung persentase (nilai *coverage*) yang diperoleh berdasarkan hasil keputusan atau cabang yang dieksekusi.

5.2.2.1 Pengujian UC-01

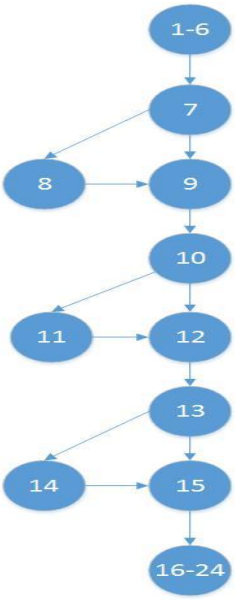
Pada halaman memasukkan laporan penjaringan data suspect (UC-01) yang dilakukan oleh koordinator kecamatan terdapat beberapa *method atau function* dalam *class* Datasuspect. Karna keterbatasan dalam penulisan laporan maka yang disampaikan dalam laporan ini yaitu pengujian pada *function* tampil_datasuspect() yang terdiri dari 3 *conditional statement* dan 24 *statement*, Pengujian dilakukan untuk memastikan bahwa strukur atau alur logika kode program pada *function* tampil_datasuspect() sudah berjalan sesuai dengan yang diharapkan dan menghasilkan keluaran nilai yang benar dan valid. Beberapa tahapan pengujian dan hasil dari pengujian pada halaman UC-01 adalah sebagai berikut:

Menentukan blok kode program yaitu *function* tampil_datasuspect(). Adapun *source code* pada *function* tampil_datasuspect() dapat dilihat pada Gambar 5.33.

```
    public function tampil_datasuspect()
    {
1       $role  = $this->role;
2       $bulan = $this->input->get('bulan_laporan');
3       $tahun = $this->input->get('tahun_laporan');

4       $_GET['bulan_laporan'] = empty($bulan) ? '' : $bulan;
5       $_GET['tahun_laporan'] = empty($tahun) ? '' : $tahun;

6       $where['1'] = 1;

7       if ($bulan != '' && $bulan != '0') {
8           $where['bulan_laporan'] = $bulan;
9       }
10      if ($tahun != '' && $tahun != '0') {
11          $where['tahun_laporan'] = $tahun;
12      }
13      if ($role != 'Admin') {
14          $where['tb_suspect.id_kecamatan']        =        $this->session-
    >userdata('id_kecamatan');
15      }

16      $this->model_squrity->getsqurity();
17      $isi['content']    = 'suspect/tampil_datasuspect';
18      $isi['title']      =   'SIMONEV   COMMUNITY   TB-HIV   CARE   AISYIYAH
    TANGGAMUS';
19      $isi['sub_title'] = 'TB-HIV CARE';
20      $isi['judul']      = 'Home';
21      $isi['sub_judul'] = 'Data Suspect';
22      $isi['data']       = $this->model_datasuspect->listdata($where);
23      $this->load->view('backend/tampilan_home',$isi);
24   }
```

Gambar 5.33 *Source Code Function* tampil_datasuspect()

Pengubahan *source code* menjadi *flowgraph* pada *function* tampil_datasuspect() dapat dilihat pada Gambar 5.34.



Gambar 5.34 *Flowgraph* tampil_datasuspect()

Berdasarkan *flowgraph* pada Gambar 5.34 terdapat total 24 *statement*, 6 *branch*, 11 node (N), dan 13 edge (E). Dari *flowgarph* pada Gambar 5.34 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 13 – 11 + 2 = 2 + 2 = 4

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.34 adalah 4. Berdasarkan perhitungan *cyclomatic complexity* tersebut, maka akan ditentukan *independent* path (jalur) yang mana jalur independen dari *function* tampil_datasuspect() ada 4 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-011 = 1-6,7,9,10,12,13,15,16-24.
   Keterangan admin login ke sistem informasi reporting, kemudian tidak menginputkan bulan laporan dan tahun laporan yang dipilih.
2) Jalur P-012 = 1-6,7,8,9,10,12,13,15,16-24
   Keterangan admin login ke sistem informasi reporting, menginputkan bulan laporan yaitu bulan maret, tetapi tidak menginputkan tahun laporan.
3) Jalur P-013 = 1-6,7,8,9,10,11,12,13,15,16-24
   Keterangan admin login ke sistem informasi reporting, menginputkan bulan laporan yaitu bulan maret dan tahun laporan yaitu tahun 2017.
4) Jalur P-014 = 1-6,7,8,9,10,11,12,13,14,15,16-24
   Keterangan salah satu koordinator kecamatan login ke sistem informasi reporting, menginputkan bulan laporan yaitu bulan maret dan tahun laporan yaitu tahun 2017

Tabel 5.2 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity* .

Tabel 5.2 Tabel Rancangan data uji *function* tampil_datasuspect()

No Jalur	*Statement* (baris kode)	Jumlah *Branch*	Masukkan	Keluaran yang di harapkan
P-011	$where['bulan_laporan'] = ''; $where['tahun_laporan'] = ''; $isi['data'] = $this->model_datasuspect->listdata_tb($where);	3	Role = admin "bulan_laporan" => NULL "tahun_laporan" => NULL	Data suspect ditampilkan adalah keseluruhan kecamatan, bulan dan tahun. ["bulan_laporan"]=> string(0) = "semua" ["tahun_laporan"]=> string(0) = "semua"
P-012	if ($bulan != '' && $bulan != '0') { $where['bulan_laporan'] = $bulan;	1	Role = admin “bulan_laporan” => Maret “tahun_laporan” => NULL	Data suspect yang ditampilkan adalah keseluruhan kecamatan berdasarkan ["bulan_laporan"]=> string(5) "Maret"

	} $isi['data'] = $this->model_datasuspect->listdata_tb($where);			["tahun_laporan"]=> string(0) "semua"

P-013	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] = $bulan;
	}
	
	if ($tahun != '' && $tahun != '0') {
	$where['tahun_laporan'] = $tahun;
	}
	
	$isi['data'] = $this->model_datasuspect->listdata_tb($where);
		2	Role = admin
			“bulan_laporan” => Maret
			“tahun_laporan” => 2017
				Data suspect yang ditampilkan adalah keseluruhan kecamatan
				["bulan_laporan"]=>
				string(5) "Maret"
				["tahun_laporan"]=>
				string(4) "2017"

P-014	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] = $bulan;
	}
	if ($tahun != '' && $tahun != '0') {
	$where['tahun_laporan'] = $tahun;
	}
	if ($role != 'Admin')
	{
	$where['tb_suspect.id_kecamatan'] = $this->session->userdata('id_kecamatan');
	}
	
	$isi['data'] = $this->model_datasuspect->listdata_tb($where);
		3	Role = Koordinator Kecamatan Talang Padang
			“bulan_laporan” => Maret
			“tahun_laporan” => 2017
				Data suspect yang ditampilkan adalah kecamatan Talang Padang
				["bulan_laporan"]=>
				string(5) "Maret"
				["tahun_laporan"]=>
				string(4) "2017"

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 4 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*). Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* ditunjukkan pada Tabel 5.3.

Tabel 5.3 Tabel Pengujian *Statement coverage* UC-01

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal *Statement* Tereksekusi	Nilai *Coverage*

TC01-1	P-014	Mengambil data suspect suspect ["bulan_laporan"]=> string(5) "Maret" ["tahun_laporan"]=> string(4) "2017" tb_suspect.id_kecamatan = 1 dan menampilkan data suspect kecamatan Talang Padang berdasarkan bulan Maret dan Tahun 2017	Berhasil	24	24/24 = 100%

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah satu *test case*. Karena pada jalur P-014 sudah mencakup semua *statement* pada program atau dengan kata lain sudah melewati semua node yang ada pada *flowgraph*. Dengan demikian pada TC01-1 sudah memastikan bahwa semua *statement* pada program telah di eksekusi dengan sekali uji dan berhasil dilakukan pengujian pada *function* tampil_datasuspect().

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya yaitu melakukan pengujian dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan (*true* dan *false*) dieksekusi dengan tepat. Adapun kasus uji (*test case*) yang mungkin mencakup nilai 100% *branch coverage* ditunjukkan pada Tabel 5.4.

Tabel 5.4 Tabel Pengujian *Branch coverage* UC-01

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Coverage*
TC01-1	P-011	Mengambil data suspect suspect ["bulan_laporan"] => semua ["tahun_laporan"] => semua dan menampilkan data suspect seluruh kecamatan, seluruh bulan dan seluruh tahun	Berhasil	3	3/6 = 50 %
TC01-2	P-014	Mengambil data suspect suspect ["bulan_laporan"]=> string(5) "Maret" ["tahun_laporan"]=> string(4) "2017" tb_suspect.id_kecamatan = 1	Berhasil	6	6/6 = 100 %

		dan menampilkan data suspect kecamatan Talang Padang berdasarkan bulan Maret dan Tahun 2017			

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah dua *test case*. Pengujian dilakukan dengan mengidentifikasi terhadap keputusan atau cabang yang ada program seperti pada TC01-1 dan TC01-2. Dengan demikian pengujian pada ke dua *test case* tersebut memastikan bahwa setiap keputusan atau cabang yang ada pada program sudah dilakukan eksekusi minimal dengan sekali test dan pengujian pada *function* tampil_datasuspect() telah berhasil dilakukan menggunakan teknik *branch coverage*.

5.2.2.2 Pengujian UC-02

Pada halaman indikator terdapat fungsi untuk data teridentifikasi TB ataupun HIV yang digunakan baik oleh koordinator kecamatan ataupun *staff dc* untuk melihat data pasien, disamping itu dapat menambah data Pengawas Menelan Obat (UC-02) yang hanya bisa dilakukan oleh *staff dc*, fungsi untuk teridentifikasi tb ataupun hiv terdapat dalam *class* Datasuspect juga. Pada UC-02 ini yang akan disampaikan di dalam laporan yaitu pengujian pada *function* tampil_teridentifikasi_tb() yang terdiri dari 3 *conditional statement* dan 24 *statement*. Pengujian dilakukan untuk memastikan bahwa struktur atau logika kode program pada *function* tampil_teridentifikasi_tb() sudah berjalan sesuai dengan yang diharapkan dan menghasilkan *output* yang tepat. Beberapa tahapan pengujian dan hasil dari pengujian pada halaman UC-02 adalah sebagai berikut:

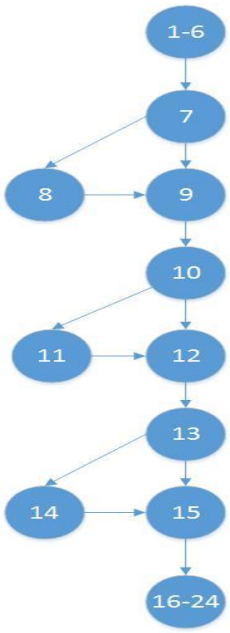
Menentukan blok kode program yaitu *function* tampil_teridentifikasi_tb(). Adapun *source code* pada *function* tampil_teridentifikasi_tb() dapat dilihat pada Gambar 5.35.

```
      public function tampil_teridentifikasi_tb()
       {
1.        $role  = $this->role;
2.        $bulan = $this->input->get('bulan_laporan');
3.        $tahun = $this->input->get('tahun_laporan');

4.        $_GET['bulan_laporan'] = empty($bulan) ? '' : $bulan;
5.        $_GET['tahun_laporan'] = empty($tahun) ? '' : $tahun;

6.        $where[1] = '1';

7.        if ($bulan != '' && $bulan != '0') {
8.            $where['bulan_laporan'] = $bulan;
9.        }
10.       if ($tahun != '' && $tahun != '0') {
11.           $where['tahun_laporan'] = $tahun;
```

```
12.     }
13.     if ($role != 'Admin') {
14.         $where['tb_suspect.id_kecamatan']        =        $this->session-
    >userdata('id_kecamatan');
15.     }

16.     $this->model_squrity->getsqurity();
17.     $isi['content']    = 'suspect/tampil_teridentifikasi_tb';
18.     $isi['title']      =   'SIMONEV   COMMUNITY   TB-HIV   CARE   AISYIYAH
    TANGGAMUS';
19.     $isi['sub_title'] = 'TB-HIV CARE';
20.     $isi['judul']      = 'Home';
21.     $isi['sub_judul'] = 'Data Teridentifikasi TB';
22.     $isi['data']       = $this->model_datasuspect->listdata_tb($where);
23.     $this->load->view('backend/tampilan_home',$isi);
24. }
```

Gambar 5.35 *Source Code Function* tampil_teridentifikasi_tb()

Pengubahan *source code* menjadi *flowgraph* pada *function* tampil_teridentifikasi_tb() dapat dilihat pada Gambar 5.36.



Gambar 5.36 *Flowgraph* tampil_teridentifikasi_tb()

Berdasarkan *flowgraph* pada *function* tampil_teridentifikasi_tb() terdapat total 24 *statement*, 6 *branch*, 11 *node* (N), dan 13 *edge* (E). Dari *flowgarph* pada Gambar 5.36 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 13 – 11 + 2 = 2 + 2 = 4

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.36 adalah 4. Berdasarkan perhitungan *cyclomatic complexity* tersebut, maka akan ditentukan

independent path (jalur) dari *function* tampil_teridentifikasi_tb() berdasarkan perolehan cc yaitu ada 4 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-021 = 1-6,7,9,10,12,13,15,16-24.
   Keterangan pada kasus ini ketika admin login ke sistem informasi reporting untuk melihat data teridentifikasi tb (pasien tb), kemudian tidak menginputkan bulan laporan dan tahun laporan yang dipilih.
2) Jalur P-022 = 1-6,7,8,9,10,12,13,15,16-24
   Keterangan admin login ke sistem informasi reporting, menginputkan bulan laporan yaitu bulan maret, tetapi tidak menginputkan tahun laporan.
3) Jalur P-023 = 1-6,7,8,9,10,11,12,13,15,16-24
   Keterangan admin login ke sistem informasi reporting, menginputkan bulan laporan yaitu bulan maret dan tahun laporan yaitu tahun 2017.
4) Jalur P-024 = 1-6,7,8,9,10,11,12,13,14,15,16-24
   Keterangan salah satu koordinator kecamatan login ke sistem informasi reporting, menginputkan bulan laporan yaitu bulan maret dan tahun laporan yaitu tahun 2017.

Tabel 5.5 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity* .

Tabel 5.5 Tabel rancangan uji function tampil_teridentifikasi_tb()

No Jalur	***Statement* (baris kode)**	**Jumlah *Branch***	**Masukkan**	**Keluaran yang di harapkan**
P-021	$where['bulan_laporan'] = ' '; $where['tahun_laporan'] = ' '; $isi['data'] = $this->model_datasuspect->listdata_tb($where);	3	Role = admin "bulan_laporan" => NULL "tahun_laporan" => NULL	Data teridentifikasi tb yang ditampilkan adalah keseluruhan kecamatan, bulan dan tahun. ["bulan_laporan"]=> string(0) = "semua" ["tahun_laporan"]=> string(0) = "semua"

P-022	if ($bulan != '' && $bulan != '0')
	{
	$where['bulan_laporan'] = $bulan;
	}
	
	$isi['data'] = $this->model_datasuspect->listdata_tb($where);
		1	Role = admin
			“bulan_laporan” => Maret
			“tahun_laporan” => " "
				Data pasien teridentifikasi tb yang ditampilkan adalah keseluruhan kecamatan ["bulan_laporan"]=>
				string(5) "Maret"
				["tahun_laporan"]=>
				string(0) "semua"

P-023	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] = $bulan;
	}
	
	if ($tahun != '' && $tahun != '0') {
	$where['tahun_laporan'] = $tahun;
	}
	
	$isi['data'] = $this->model_datasuspect->listdata_tb($where);
		2	Role = admin
			“bulan_laporan” => Maret
			“tahun_laporan” => 2017
				Data pasien teridentifikasi tb yang ditampilkan adalah keseluruhan kecamatan
				["bulan_laporan"]=>
				string(5) "Maret"
				["tahun_laporan"]=>
				string(4) "2017"

P-024	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] = $bulan;
	}
	if ($tahun != '' && $tahun != '0') {
	$where['tahun_laporan'] = $tahun;
	}
	if ($role != 'Admin')
	{
	$where['tb_suspect.id_kecamatan'] = $this->session->userdata('id_kecamatan');
	}
	
	$isi['data'] = $this->model_datasuspect->listdata_tb($where);
		3	Role = Koordinator Kecamatan Talang Padang
			“bulan_laporan” => Maret
			“tahun_laporan” => 2017
				Data pasien teridentifikasi tb yang ditampilkan adalah kecamatan Talang Padang
				["bulan_laporan"]=>
				string(5) "Maret"
				["tahun_laporan"]=>
				string(4) "2017"

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 4 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*), Adapun minimal test yang diperlukan untuk mencakup nilai 100% *statement coverage* ditunjukkan pada Tabel 5.6.

Tabel 5.6 Tabel Pengujian *Statement coverage* UC-02

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal *Statement* Tereksekusi	Nilai *Coverage*
TC02-1	P-024	Mengambil data teridentifikasi tb ["bulan_laporan"]=> string(5) "Maret" ["tahun_laporan"]=> string(4) "2017" tb_suspect.id_kecamatan = 1	Berhasil	24	24/24 = 100%

		dan menampilkan data pasien teridentifikasi tb kecamatan Talang Padang berdasarkan bulan Maret dan Tahun 2017			

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah satu *test case*. Karena pada jalur P-024 sudah mencakup semua *statement* pada program. Dengan demikian pada TC02-1 sudah memastikan bahwa semua *statement* pada program telah di eksekusi dengan sekali uji dan berhasil dilakukan pengujian pada *function* tampil_teridentifikasi_tb().

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya yaitu melakukan pengujian dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan (*true* dan *false*) dieksekusi dengan tepat. Adapun kasus uji (*test case*) yang mungkin mencakup nilai 100% *branch coverage* ditunjukkan pada Tabel 5.7.

Tabel 5.7 Tabel Pengujian *Branch coverage* UC-02

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Coverage*
TC02-1	P-021	Mengambil data teridentifikasi tb berdasarkan ["bulan_laporan"] => semua ["tahun_laporan"] => semua dan menampilkan data pasien teridentifikasi tb seluruh kecamatan, seluruh bulan dan seluruh tahun	Berhasil	3	3/6 = 50 %
TC02-2	P-024	Mengambil data teridentifikasi tb ["bulan_laporan"]=> string(5) "Maret" ["tahun_laporan"]=> string(4) "2017" tb_suspect.id_kecamatan = 1 dan menampilkan data pasien teridentifikasi tb kecamatan Talang Padang berdasarkan bulan Maret dan tahun 2017	Berhasil	6	6/6 = 100 %

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah dua *test case*. Pengujian dilakukan dengan

mengidentifikasi terhadap keputusan atau cabang yang ada program seperti pada TC02-1 dan TC02-2. Dengan demikian pengujian pada ke dua *test case* tersebut memastikan bahwa setiap keputusan atau cabang yang ada pada program sudah dilakukan eksekusi minimal dengan sekali test dan pengujian pada *function* tampil_teridentifikasi_tb() telah berhasil dilakukan.

5.2.2.3 Pengujian UC-03

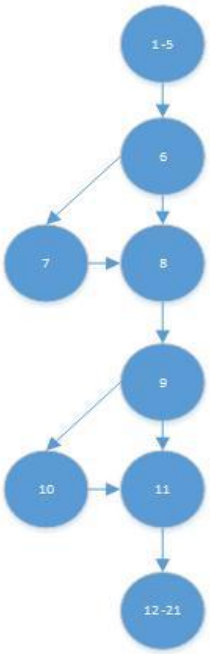
Pada halaman melihat staus laporan penjaringan data suspect (UC-03) hanya terdapat *function* index() untuk menampilkan halaman status laporan penjaringan data suspect yang sudah masuk atau data laporan yang ada. *Function* index() terdiri dari 2 *conditional statement* dan 22 *statement*, pengujian dilakukan untuk memastikan bahwa stukru atau alur logika kode program pada *function* index() sudah berjalan sesuai dengan yang diharapkan. Beberapa tahapan pengujian dan hasil dari pengujian pada function index() untuk menampilkan status laporan penjaringan suspecct pada UC-03 adalah sebagai berikut:

Menentukan blok kode program yaitu *function* index(). Adapun *source code* pada *function* index() dapat dilihat pada Gambar 5.37.

```
Public function index()
{
1       $bulan = $this->input->get('bulan_laporan');
2       $tahun = $this->input->get('tahun_laporan');
3       $_GET['bulan_laporan'] = empty($bulan) ? '' : $bulan;
4       $_GET['tahun_laporan'] = empty($tahun) ? '' : $tahun;
5       $where['1'] = 1;

6       if ($bulan != '' && $bulan != '0') {
7           $where['bulan_laporan'] = $bulan;
8       }
9       if ($tahun != '' && $tahun != '0') {
10          $where['tahun_laporan'] = $tahun;
11      }

12      $this->model_squrity->getsqurity();
13      $isi['content']    = 'backend/tampilan_content';
14      $isi['title']      = 'SIMONEV COMMUNITY TB-HIV CARE AISYIYAH TANGGAMUS';
15      $isi['sub_title']  = 'TB-HIV CARE AISYIYAH';
16      $isi['data']       = $this->model_kecamatan->status_laporan($where);
17      $isi['bulan']      = $bulan;
18      $isi['tahun']       = $tahun;
19      $isi['datasuspect'] = $this->db->get('tb_suspect');
20      $this->load->view('backend/tampilan_home',$isi);
21  }
```

Gambar 5.37 *Source Code Function* index() status laporan

Pengubahan *source code* menjadi *flowgraph* pada *function* index() status laporan dapat dilihat pada Gambar 5.38.



Gambar 5.38 *Flowgraph function* index() status laporan

Berdasarkan *flowgraph* pada *function* index() status laporan penjaringan suspect total 21 *statement*, 4 branch, 8 node (N), dan 9 edge (E). Dari *flowgarph* pada Gambar 5.36 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 9 – 8 + 2 = 1 + 2 = 3

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.36 adalah 3. Berdasarkan perhitungan *cyclomatic complexity* tersebut, maka akan ditentukan *independent path* (jalur) dari *function* index() berdasarkan perolehan cc yaitu ada 3 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-031 = 1-5,6,8,9,11,12-21
   Keterangan admin tidak menginputkan bulan laporan (kosong) dan tahun laporan (kosong).
2) Jalur P-032 = 1-5,6,7,8,9,11,12-21
   Keterangan admin menginputkan bulan laporan yaitu bulan April, tetapi tidak menginputkan tahun laporan.
3) Jalur P-033 = 1-5,6,7,8,9,10,11,12-21
   Keterangan admin menginputkan bulan laporan yaitu bulan April dan tahun laporan yaitu tahun 2017.

Tabel 5.8 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity* .

Tabel 5.8 Tabel Rancangan data uji pada function index() status laporan

No Jalur	***Statement* (baris kode)**	**Jumlah Branch**	**Masukkan data**	**Keluaran yang diharapkan**
P-031	$where['bulan_laporan'] = ' '; $where['tahun_laporan'] = ' '; $isi['data'] = $this->model_kecamatan->status_laporan($where);	2	"bulan" => " " "tahun" => " "	Menampilkan data status laporan penjaringan suspect pada 8 kecamatan berdasarkan ["bulan_laporan"] => string(0) "Semua" ["tahun_laporan"] => string(0) "Semua"

P-032	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] = $bulan;
	}
	
	$isi['data'] = $this->model_kecamatan->status_laporan($where);
		1	"bulan" => April
			"tahun" => " "
				Menampilkan data status laporan penjaringan suspect pada 8 kecamatan berdasarkan
				["bulan"]=> string(5) "April"
				["tahun"]=> string(0) ""

P-033	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] = $bulan;
	}
	if ($tahun != '' && $tahun != '0') {
	$where['tahun_laporan'] = $tahun;
	}
	$isi['data'] = $this->model_kecamatan->status_laporan($where);
		2	"bulan" => April
			"tahun" => 2017
				Menampilkan data status laporan penjaringan suspect pada 8 kecamatan berdasarkan
				["bulan"]=> string(5) "April"
				["tahun"]=> string(4) "2017"

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 3 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*), Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* pada *function* index() status laporan ditunjukkan pada Tabel 5.9.

Tabel 5.9 Tabel Pengujian *Statement coverage* UC-03

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal *Statement* Tereksekusi	Nilai *Coverage*

TC03-1	P-033	Mengambil jumlah entry laporan penjaringan suspect yang masuk berdasarkan pada
		`["result_object"]=>`
		`array(8)`
		`["bulan"]=>`
		`string(5) "April"`
		`["tahun"]=>`
		`string(4) "2017"`
		
		dan ditampilkan status entry laporan penjaringan suspect pada 8 kecamatan pada bulan April dan tahun 2017
			Berhasil	21	21/21 = 100%

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah satu *test case*. Karena pada jalur P-033 sudah mencakup semua *statement* pada program. Dengan demikian pada TC03-1 sudah memastikan bahwa semua *statement* pada program telah di eksekusi dengan sekali uji dan berhasil dilakukan pengujian pada *function* index() status laporan.

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya yaitu melakukan pengujian dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan (*true* dan *false*) dieksekusi dengan tepat. Adapun kasus uji (*test case*) yang mungkin mencakup nilai 100% *branch coverage* ditunjukkan pada Tabel 5.10.

Tabel 5.10 Tabel Pengujian *Branch coverage* UC-03

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Coverage*

TC03-1	P-031	Mengambil jumlah entry laporan penjaringan suspect yang masuk berdasarkan pada
		`["result_object"]=>`
		`array(8)`
		`["bulan"]=>`
		`string(0) "Semua"`
		`["tahun"]=>`
		`string(0) "Semua"`
		
		dan ditampilkan status entry laporan penjaringan suspect pada 8 kecamatan pada semua bulan dan tahun
			Berhasil	2	2/4 = 50 %

TC03-2	P-033	Mengambil jumlah entry laporan penjaringan suspect yang masuk berdasarkan pada ["result_object"]=> array(8) ["bulan"]=> string(5) "April" ["tahun"]=> string(4) "2017" dan ditampilkan status entry laporan penjaringan suspect pada 8 kecamatan pada bulan April dan tahun 2017	Berhasil	4	4/4 = 100 %

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah dua *test case*. Pengujian dilakukan dengan mengidentifikasi terhadap keputusan atau cabang yang ada program seperti pada TC03-1 dan TC03-2. Dengan demikian pengujian pada ke dua *test case* tersebut memastikan bahwa setiap keputusan atau cabang yang ada pada program sudah dilakukan eksekusi minimal dengan sekali test dan pengujian pada *function* index() status laporan telah berhasil dilakukan.

5.2.2.4 Pengujian UC-04

Pada halaman mengelola data referensi kader (UC-04) terdapat beberapa kategori, pada UC-04 dipilih salah satu kategori untuk disampaikan dalam laporan yaitu mengelola data referensi kader. Dalam mengelola referensi kader terdiri dari beberapa method pada *class* Kader_tb. Pengujian yang dilakukan pada *function* edit() yang terdapat dalam class Kader_tb terdiri 1 *conditional statement* dan 33 *statement*, pengujian dilakukan untuk memastikan bahwa struktur atau alur logika kode program pada *function* edit() kader sudah berjalan sesuai dengan yang diharapkan. Beberapa tahapan pengujian dan hasil dari pengujian pada halaman UC-04 adalah sebagai berikut:

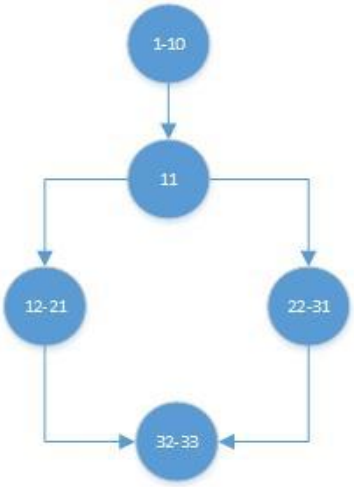
Menentukan blok kode program yaitu *function* edit(). Adapun *source code* pada *function* edit() dapat dilihat pada Gambar 5.39.

```
    public function edit()
    {
1       $this->model_squrity->getsqurity();
2       $isi['content']       = 'referensi/form_editkader';
3       $isi['title']         = 'SI_Reporting COMMUNITY TB-HIV CARE AISYIYAH
    TANGGAMUS';
4       $isi['sub_title']     = 'TB-HIV CARE';
5       $isi['judul']         = 'Referensi';
6       $isi['sub_judul']     = 'Edit Data Kader';
7       $isi['datakecamatan'] = $this->db->get('tb_kecamatan');
```

```
8       $key = $this->uri->segment(4);
9       $this->db->where('id_kader',$key);
10      $query = $this->db->get('tb_kader');
11      if($query->num_rows()>0){
12          $row = $query->row();
13              $isi['id_kader']         = $row->id_kader;
14              $isi['namakader']        = $row->nama_kader;
15              $isi['no_anggota']       = $row->nomor_anggota;
16              $isi['id_kecamatan']     = $row->id_kecamatan;
17              $isi['bergabung_bulan'] = $row->bulan_bergabung;
18              $isi['bergabung_tahun'] = $row->tahun_bergabung;
19              $isi['alamat']           = $row->alamat;
20              $isi['no_telp']          = $row->nomor_telp;
21      }
22        else {
23          $isi['id_kader']         = '';
24          $isi['namakader']        = '';
25          $isi['no_anggota']       = '';
26          $isi['id_kecamatan']     = '';
27          $isi['bergabung_bulan'] = '';
28          $isi['bergabung_tahun'] = '';
29          $isi['alamat']           = '';
30          $isi['no_telp']          = '';
31      }
32      $this->load->view('backend/tampilan_home',$isi);
33  }
```

Gambar 5.39 *Source Code Function* edit()

Pengubahan *source code* menjadi *flowgraph* pada *function* edit() kader dapat dilihat pada Gambar 5.40.



Gambar 5.40 *Flowgraph* edit() kader

Berdasarkan *flowgraph* pada *function* edit() untuk data referensi data kader terdapat total 33 *statement*, 2 *branch*, 5 node (N), dan 5 edge (E). Dari *flowgarph* pada Gambar 5.40 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 5 – 5 + 2 = 0 + 2 = 2

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.40 adalah 2. Berdasarkan perhitungan *cyclomatic complexity* tersebut, maka akan ditentukan *independent path* (jalur) dari *function* edit() kader tb berdasarkan perolehan cc yaitu ada 2 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-041 = 1-10,11,12-21,32-33

   Keterangan, admin akan melakukan pembaharuan atau edit untuk data kader tb, jika datanya ada dalam database maka akan ditampilkan nilai pada setiap variabel berdasarkan inputan data sebelumnya.

2) Jalur P-042 = 1-10,11,22-31,32-33

   Keterangan, admin akan melakukan pembaharuan untuk data kader tb, jika data tidak ada maka nilai dari setiap variabel adalah kosong.

Tabel 5.11 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity* .

Tabel 5.11 Tabel Rancangan data uji pada *function* edit() kader

No Jalur	*Statement*	Jumlah Branch	Masukkan data	Keluaran yang diharapkan
P-041	$isi['id_kader'] = $row->id_kader; $isi['namakader'] = $row->nama_kader; $isi['no_anggota'] = $row->nomor_anggota; $isi['id_kecamatan'] = $row->id_kecamatan; $isi['bergabung_bulan'] = $row->bulan_bergabung; $isi['bergabung_tahun'] = $row->tahun_bergabung; $isi['alamat'] = $row->alamat; $isi['no_telp'] = $row->nomor_telp;	1	"id_kader" => 1 "namakader" => Wiwin Naila "no_anggota" => 0101A "id_kecamatan" => 7 "bergabung_bulan" => Januari "bergabung_tahun" => 2017 "alamat" => Sinar Muncak "no_telp" => 0819xxxxxxxxx	Menampilkan data kader tb yang akan di update atau diedit. Berdasarkan pada variabel ["id_kader"]=> string(1) "1" ["namakader"]=> string(11) "Wiwin Naila" ["no_anggota"]=> string(5) "0101A" ["id_kecamatan"]=> string(1) "7" ["bergabung_bulan"]=> string(7) "Januari" ["bergabung_tahun"]=> string(4) "2017" ["alamat"]=> string(12) "Sinar Muncak" ["no_telp"]=> string(13) "0819xxxxxxxxx"
P-042	$isi['id_kader'] = ' '; $isi['namakader'] = ' '; $isi['no_anggota'] = ' ';	1	"id_kader" => "" "namakader" => "" "no_anggota" => "" "id_kecamatan" => "" "bergabung_bulan" => "" "bergabung_tahun" => ""	Tidak menampilkan nilai variabel ["id_kader"]=> string(0) "" ["namakader"]=>

	$isi['id_kecamatan'] = ' '; $isi['bergabung_bulan'] = ' '; $isi['bergabung_tahun'] = ' '; $isi['alamat'] = ' '; $isi['no_telp'] = ' ';		"alamat" => "" "no_telp" =>""	string(0) "" ["no_anggota"]=> string(0) "" ["id_kecamatan"]=> string(0) "" ["bergabung_bulan"]=> string(0) "" ["bergabung_tahun"]=> string(0) "" ["alamat"]=> string(0) "" ["no_telp"]=> string(0) ""

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 2 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*). Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* ditunjukkan pada Tabel 5.12.

Tabel 5.12 Tabel Pengujian *Statement coverage* UC-04

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Coverage*
TC04-1	P-041	Mengambil data kader tb pada database dan ditampilkan data kader tb yang akan di update atau diedit. Berdasarkan pada variabel ["id_kader"]=> string(1) "1" ["namakader"]=> string(11) "Wiwin Naila" ["no_anggota"]=> string(5) "0101A" ["id_kecamatan"]=> string(1) "7" ["bergabung_bulan"]=> string(7) "Januari" ["bergabung_tahun"]=> string(4) "2017" ["alamat"]=> string(12) "Sinar Muncak" ["no_telp"]=> string(13) "0819xxxxxxxxx"	Berhasil	23	23/33 = 69,69 %

TC04-2	P-042	Tidak ditampilkan nilai variabel, karna hasil dari query datanya tidak ada. ["id_kader"]=> string(0) "" ["namakader"]=> string(0) "" ["no_anggota"]=> string(0) "" ["id_kecamatan"]=> string(0) "" ["bergabung_bulan"]=> string(0) "" ["bergabung_tahun"]=> string(0) "" ["alamat"]=> string(0) "" ["no_telp"]=> string(0) ""	Berhasil	10	33/33 = 100 %

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang diperoleh untuk mencakup semua *statement* dan memperoleh nilai *coverage* 100% dilakukan *test* pada ke dua jalur yang sudah ditetapkan sebelumnya. Dimana pada jalur P-041 memperoleh nilai sebesar 69,69%, hal tersebut menyatakan bahwa ada *statement* yang belum di eksekusi, maka akan dilakukan pengujian pada jalur berikutnya yaitu pada jalur P-042 untuk mencakup semua *statement* pada program. Dengan demikian pengujian dengan TC04-1 dan TC04-2, bahwa setiap *statement* telah di eksekusi dengan minimal sekali test pada setiap jalur yang sudah ditetapkan sebelumnya. Dan pengujian pada *function* edit() kader telah berhasil dilakukan dan semua *statement* (baris kode) pada *function* tersebut telah di eksekusi dengan menggunakan dua *test case* berdasarkan *independent path* yang diperoleh dari perhitungan *Cyclomatic Complexity* .

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya yaitu melakukan pengujian dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan (*true* dan *false*) dieksekusi dengan tepat. Adapun kasus uji (*test case*) yang mungkin mencakup nilai 100% *branch coverage* ditunjukkan pada Tabel 5.13.

Tabel 5.13 Tabel Pengujian *Branch coverage* UC-04

Test Case Id	Jalur	*Statement* If	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Coverage*

TC04-1	P-041	if($query->num_rows()>0)	Mengambil data kader tb pada database dan ditampilkan data kader tb yang akan di update atau diedit. Berdasarkan pada variabel ["id_kader"]=> string(1) "1" ["namakader"]=> string(11) "Wiwin Naila" ["no_anggota"]=> string(5) "0101A" ["id_kecamatan"]=> string(1) "7" ["bergabung_bulan"]=> string(7) "Januari" ["bergabung_tahun"]=> string(4) "2017" ["alamat"]=> string(12) "Sinar Muncak" ["no_telp"]=> string(13) "0819xxxxxxxxx"	Berhasil	1	1/2 = 50 %
TC04-2	P-042	if($query->num_rows()<0)	Tidak ditampilkan nilai variabel, karna hasil dari query datanya tidak ada. ["id_kader"]=> string(0) "" ["namakader"]=> string(0) "" ["no_anggota"]=> string(0) "" ["id_kecamatan"]=> string(0) "" ["bergabung_bulan"]=> string(0) "" ["bergabung_tahun"]=> string(0) "" ["alamat"]=> string(0) "" ["no_telp"]=> string(0) ""	Berhasil	2	2/2 = 100 %

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah dua *test case*. Pengujian dilakukan dengan mengidentifikasi terhadap keputusan atau cabang yang ada program seperti pada TC04-1 dan TC04-2. Dengan demikian pengujian pada ke dua *test case* tersebut memastikan bahwa setiap keputusan atau cabang yang ada pada program sudah dilakukan eksekusi minimal dengan sekali test dan pengujian pada *function* edit() kader telah berhasil dilakukan. Berdasarkan pengujian dengan 2 jalur diatas nilai *coverage* yang diperoleh yaitu 100%, dimana pengujian

pada kondisi percabangan sisi *true* pada jalur 1 diperoleh nilai *coverage* sebesar 50% dan dalam mencapai nilai *branch coverage* 100% maka dilakukan ujicoba pada sisi *false* juga sehingga setiap masing-masing kondisi keputusan atau cabang telah tervalidasi atau telah dieksekusi.

5.2.2.5 Pengujian UC-05

Pada halaman mengelola pengguna sistem (UC-05) terdapat beberapa method dalam *class* Datauser. Pada UC-05 yang disampaikan dalam penulisan laporan yaitu pengujian yang dilakukan pada *function* edit_user() yang terdiri dari 1 *conditional statement* dan 32 *statement*, pengujian dilakukan untuk memastikan bahwa struktur dan alur logika kode program pada *function* edit_user() sesuai dengan yang diharapkan dan menghasilkan output nilai yang benar dan valid. Pada function edit_user() ini strukut kode programnya tidak jauh berbeda dari function edit() untuk data kader sebelumnya. Beberapa tahapan pengujian dan hasil pengujian pada halaman UC-05 adalah sebagai berikut:

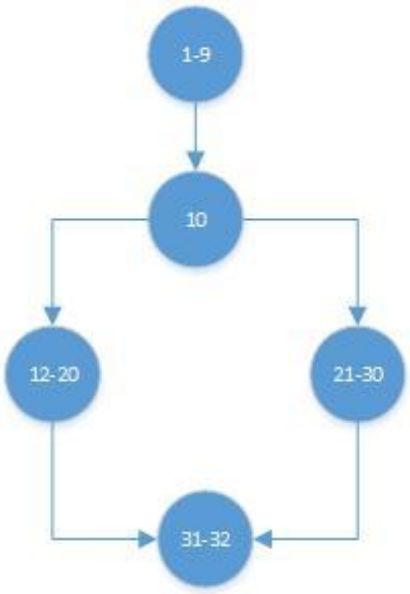
Menentukan blok kode program yaitu *function* edit_user(). Adapun *source code* pada *function* edit_user() dapat dilihat pada Gambar 5.41.

```
    public function edit_user()
    {
1.      $this->model_squrity->getsqurity();
2.      $isi['content']       = 'pengguna_sistem/form_edituser';
3.      $isi['title']         =  'SIMONEV  COMMUNITY  TB-HIV  CARE  AISYIYAH
    TANGGAMUS';
4.      $isi['sub_title']     = 'TB-HIV CARE';
5.      $isi['judul']         = 'Pengguna';
6.      $isi['sub_judul']     = 'Sistem Informasi TB-HIV Care AISYIYAH';

7.      $key = $this->uri->segment(4);
8.      $this->db->where('id_pengguna',$key);
9.      $query = $this->db->get('tb_pengguna');

10.     if($query->num_rows()>0){
11.         $row = $query->$row
12.             $isi['id_pengguna']  = $row->id_pengguna;
13.             $isi['nama_lengkap'] = $row->nama_lengkap;
14.             $isi['username']    = $row->username;
15.             $isi['password']    = $row->password;
16.             $isi['email']       = $row->email;
17.             $isi['no_telp']     = $row->nomer_telp;
18.             $isi['foto']        = $row->foto;
19.             $isi['jabatan']     = $row->jabatan;
20.     }
21.      else {
22.             $isi['id_pengguna']   = "";
23.             $isi['nama_lengkap']  = "";
24.             $isi['username']      = "";
25.             $isi['password']      = "";
26.             $isi['email']         = "";
27.             $isi['no_telp']       = "";
28.             $isi['foto']          = "";
29.             $isi['jabatan']       = "";
30.     }
```

```
31.     $this->load->view('backend/tampilan_home',$isi);
32. }
```

Gambar 5.41 *Source Code Function* edit_user

Pengubahan *source code* menjadi *flowgraph* pada *function* edit_user() dapat dilihat pada Gambar 5.42.



Gambar 5.42 *Flowgraph* edit_user()

Berdasarkan *flowgraph* pada *function* edit_user() terdapat total 32 *statement*, 2 *branch*, 5 node (N), dan 5 edge (E). Dari *flowgarph* pada Gambar 5.42 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 5 – 5 + 2 = 0 + 2 = 2

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.42 adalah 2. Berdasarkan perhitungan *cyclomatic complexity* tersebut, maka akan ditentukan *independent path* (jalur) dari *function* edit_user() berdasarkan perolehan cc yaitu ada 2 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-051 = 1-9,10,12-20,31-32

   Keterangan, fungsi ini dilakukan oleh admin untuk melakukan pembaharuan data pengguna sistem informasi reporting communtiy TB-HIV care ‘Aisyiyah. Admin akan melakukan pembaharuan data pengguna sistem, jika datanya ada dalam database maka akan ditampilkan nilai pada setiap variabel berdasarkan inputan data sebelumnya.

2) Jalur P-052 = 1-9,10,21-30,31-32

   Keterangan, admin akan melakukan pembaharuan atau edit data pengguna sistem, jika data tidak ada maka nilai pada setiap variabel nya kosong.

Tabel 5.14 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity* .

Tabel 5.14 Tabel Rancangan data uji function edit_user()

No Jalur	*Statement*	Jumlah Branch	Masukkan data	Keluaran yang diharapkan
P-051	$isi['id_pengguna'] = $row->id_pengguna; $isi['nama_lengkap'] = $row->nama_lengkap; $isi['username'] = $row->username; $isi['password'] = $row->password; $isi['email'] = $row->email; $isi['no_telp'] = $row->nomer_telp; $isi['foto'] = $row->foto; $isi['jabatan'] = $row->jabatan;	1	"id_pengguna" => 1 "nama_lengkap" => Afriadi Tanjung "username" => afriadi_ssr "password" => 123456789 "email" => Afriadi_ssrtanggamus@gmail.com "no_telp" => 089690451127 "foto" => uncu3.jpg "jabatan" => Admin	Menampilkan data pengguna yang akan di update atau diedit. Berdasarkan pada variabel ["id_pengguna"]=> string(1) "1" ["nama_lengkap"]=> string(15) "Afriadi Tanjung" ["username"]=> string(11) "afriadi_ssr" ["password"]=> string(32) "25d55ad283aa400af464c76d713c07ad" ["email"]=> string(30) "Afriadi_ssrtanggamus@gmail.com" ["no_telp"]=> string(12) "089690451127" ["foto"]=> string(9) "uncu3.jpg" ["jabatan"]=> string(5) "Admin"
P-052	$isi['id_pengguna'] = ""; $isi['nama_lengkap'] = ""; $isi['username'] = ""; $isi['password'] = ""; $isi['email'] = ""; $isi['no_telp'] = ""; $isi['foto'] = ""; $isi['jabatan'] = "";	1	"id_pengguna" => "" "nama_lengkap" => "" "username" => "" "password" => "" "email" => "" "no_telp" => "" "foto" => "" "jabatan" =>""	Tidak menampilkan nilai dari setiap variabel ["id_pengguna"]=> string(0) " " ["nama_lengkap"]=> string(0) " " ["username"]=> string(0) " " ["password"]=> string(0) " " ["email"]=> string(0) " " ["no_telp"]=> string(0) " " ["foto"]=> string(0) " " ["jabatan"]=>

				string(0) " "

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 2 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*). Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* ditunjukkan pada Tabel 5.15.

Tabel 5.15 Tabel Pengujian *Statement coverage* UC-05

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Statement*
TC05-1	P-051	Mengambil data pengguna sistem pada database dan ditampilkan data pengguna yang akan di update atau diedit. Berdasarkan pada variabel ["id_pengguna"]=> string(1) "1" ["nama_lengkap"]=> string(15) "Afriadi Tanjung" ["username"]=> string(11) "afriadi_ssr" ["password"]=> string(32) "25d55ad283aa400af464c76d713c07ad" ["email"]=> string(30) "Afriadi_ssrtanggamus@gmail.com" ["no_telp"]=> string(12) "089690451127" ["foto"]=> string(9) "uncu3.jpg" ["jabatan"]=> string(5) "Admin"	Berhasil	22	22/32 = 68,75%
TC05-2	P-052	Tidak ditampilkan nilai dari setiap variabel, karna hasil dari query datanya kosong atau tidak ada ["id_pengguna"]=> string(0) " " ["nama_lengkap"]=> string(0) " "	Berhasil	10	32/32 = 100%

		["username"]=> string(0) " " ["password"]=> string(0) " " ["email"]=> string(0) " " ["no_telp"]=> string(0) " " ["foto"]=> string(0) " " ["jabatan"]=> string(0) " "			

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang diperoleh untuk mencakup semua *statement* dan memperoleh nilai *coverage* 100% yaitu dilakukan pengujian pada ke dua jalur tersebut. Pada jalur P-051 memperoleh nilai *coverage* sebesar 68,75%, dimana pengujian yang sudah dilakukan tersebut belum mencakup seluruh *statement* (baris kode) program, sehingga perlu dilakukan pengujian dengan jalur pada jalur berikutnya untuk memenuhi nilai *coverage* 100% yaitu dengan melakukan pengujian untuk jalur independent P-052. Dengan demikian pengujian dengan TC05-1 dan TC05-2, telah memastikan bahwa setiap *statement* pada function edit_user() telah dieksekusi dan menghasilkan nilai keluaran variabel yang benar dan valid. Langkah selanjutnya yaitu melakukan pengujian dengan teknik *branch coverage*.

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya adalah dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan akan dieksekusi dengan tepat. Adapun pengujian pada *branch coverage* ditunjukkan pada Tabel 5.16.

Tabel 5.16 Tabel Pengujian *Branch coverage* UC-05

Test Case Id	Jalur	*Statement* If	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Coverage*
TC05-1	P-051	if($query->num_rows()>0)	Mengambil data pengguna sistem pada database dan ditampilkan data pengguna sistem yang akan di update atau diedit. Berdasarkan pada nilai variabel ["id_pengguna"]=> string(1) "1" ["nama_lengkap"]=>	Berhasil	1	1/2 = 50 %

			string(15) "Afriadi Tanjung" ["username"]=> string(11) "afriadi_ssr" ["password"]=> string(32) "25d55ad283aa400af464c76d713c07ad" ["email"]=> string(30) "Afriadi_ssrtanggamus@gmail.com" ["no_telp"]=> string(12) "089690451127" ["foto"]=> string(9) "uncu3.jpg" ["jabatan"]=> string(5) "Admin"			
TC05-2	P-052	if($query->num_rows()<0)	Tidak ditampilkan nilai dari setiap variabel, karna hasil dari query datanya kosong atau tidak ada ["id_pengguna"]=> string(0) " " ["nama_lengkap"]=> string(0) " " ["username"]=> string(0) " " ["password"]=> string(0) " " ["email"]=> string(0) " " ["no_telp"]=> string(0) " " ["foto"]=> string(0) " " ["jabatan"]=> string(0) " "	Berhasil	2	2/2 = 100 %

Berdasarkan pengujian dengan 2 jalur diatas maka nilai *coverage* yang diperoleh adalah 100%, pada teknik pengujian *branch coverage* dilakukan dengan menguji setiap kondisi keputusan atau cabang yang ada pada kode program. Pada kasus ini, nilai *coverage* untuk jalur 1 diperoleh yaitu sebesar 50% *coverage* dan untuk mencapai nilai 100% maka dilakukan ujicoba pada kondisi-kondisi kemungkinan yang lainnya dengan menguji pada jalur berikutnya, sehingga mendapatkan nilai 100% setelah melakukan uji coba pada jalur 2. Setidaknya pada *branch coverage* diperlukan 2 uji data dalam sekali pengujian untuk memastikan masing-masing kondisi cabang pada kondisi true maupun false.

5.2.2.6 Pengujian UC-06

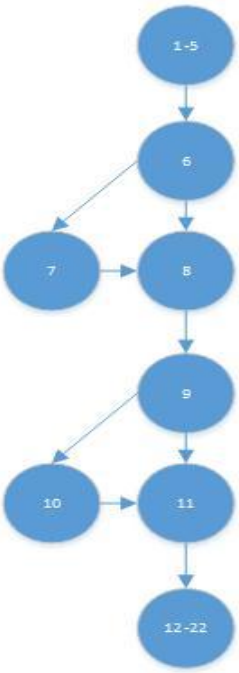
Pada halaman mengelola laporan honorarium koordinator (UC-06) terdapat beberapa method dalam *class* Laporan_honorarium. Dari beberapa *function* tersebut dilakukan pengujian terhadap *function* tampil_honorkoordinator() yang terdiri dari 2 *conditional statement* dan 22 *statement*, Pengujian dilakukan untuk memastikan bahwa struktur dan alu logika program pada *function* tampil_honorkoordinator() sudah berjalan dengan yang diharapkan. Beberapa tahapan pengujian dan hasil dari pengujian pada halaman UC-06 adalah sebagai berikut:

Menentukan blok kode program yaitu *function* tampil_honorkoordinator(). Adapun *source code* pada *function* tampil_honorkoordinator() dapat dilihat pada Gambar 5.43.

```
    public function tampil_honorkoordinator()
     {
1.      $bulan = $this->input->get('bulan');
2.      $tahun = $this->input->get('tahun');
3.      $_GET['bulan'] = empty($bulan) ? '' : $bulan;
4.      $_GET['tahun'] = empty($tahun) ? '' : $tahun;
5.      $where['1'] = 1;

6.      if ($bulan != '' && $bulan != '0') {
7.          $where['bulan'] = $bulan;
8.      }
9.      if ($tahun != '' && $tahun != '0') {
10.         $where['tahun'] = $tahun;
11.     }
12.     $this->model_squrity->getsqurity();
13.     $isi['content']       = 'laporan/tampil_honorkoordinator';
14.     $isi['title']         =  'SI_Reporting COMMUNITY TB-HIV CARE AISYIYAH
    TANGGAMUS';
15.     $isi['sub_title']     = 'TB-HIV CARE';
16.     $isi['judul']         = 'Home';
17.     $isi['sub_judul']     = 'Laporan Honor Koordinator';
18.     $isi['datakecamatan'] = $this->db->get('tb_kecamatan');
19.     $isi['datapengguna']  = $this->db->get('tb_pengguna');
20.     $isi['data']          =                   $this->model_datahonorarium-
    >listdatakoordinator($where);

21.     $this->load->view('backend/tampilan_home',$isi);
22. }
```

Gambar 5.43 *Source Code Function* tampil_honorkoordinator()

Pengubahan *source code* menjadi *flowgraph* pada *function* tampil_honorkoordinator() dapat dilihat pada Gambar 5.44.



Gambar 5.44 *Flowgraph* tampil_honorkoordinator()

Berdasarkan *flowgraph* pada *function* tampil_honorkoordinator() terdiri dari 22 *statement*, 4 branch, 8 node (N), dan 9 edge (E). Dari *flowgarph* pada Gambar 5.44 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 9 – 8 + 2 = 1 + 2 = 3

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.44 adalah 3. Berdasarkan perhitungan *Cyclomatic Complexity* tersebut, maka akan ditentukan *independent path* (jalur) dari *function* tampil_honorkoordinator() berdasarkan perolehan cc yaitu ada 3 jalur. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-061 = 1-5,6,8,9,11,12-22
   Keterangan admin tidak menginputkan bulan laporan (kosong atau nol) dan tahun laporan (kosong atau nol) yaitu yang ditampilkan data keseluruhan bulan dan tahun.
2) Jalur P-062 = 1-5,6,7,8,9,11,12-22
   Keterangan admin menginputkan bulan laporan yaitu bulan Juni, tetapi tidak menginputkan tahun laporan.
3) Jalur P-063 = 1-5,6,7,8,9,10,11,12-22
   Keterangan admin menginputkan bulan laporan yaitu bulan Juni dan tahun laporan yaitu tahun 2017.

Tabel 5.17 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity* .

Tabel 5.17 Tabel Rancang data uji pada *function* tampil_honorkoordinator()

No Jalur	*Statement* (baris kode)	Jumlah Branch	Masukkan data	Keluaran yang diharapkan
P-061	$where['bulan_laporan'] = ' '; $where['tahun_laporan'] = ' '; $isi['data'] = $this-> model_datahonorarium- >listdatakoordinator ($where);	2	"bulan" => " " "tahun" => " "	Menampilkan data laporan honorarium koordinator dari masing-masing kecamatan untuk seluruh bulan dan seluruh tahun ["bulan_laporan"] => string(0) " " ["tahun_laporan"] => string(0) ""

P-062	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] =
	$bulan;
	}
	
	$isi['data'] = $this->
	model_datahonorarium-
	>listdatakoordinator
	($where);
		1	"bulan" => Juni
			"tahun" => " "
				Menampilkan data laporan honorarium koordinator dari masing-masing kecamatan ditampilkan berdasarkan pada
				
				["bulan"]=>
				string(5) "Juni"
				["tahun"]=>
				string(0) ""

P-063	if ($bulan != '' && $bulan != '0') {
	$where['bulan_laporan'] =
	$bulan;
	}
	if ($tahun != '' && $tahun != '0') {
	$where['tahun_laporan'] =
	$tahun;
	}
	$isi['data'] = $this->
	model_datahonorarium-
	>listdatakoordinator
	($where);
		2	"bulan" => Juni
			"tahun" => 2017
				Menampilkan data laporan honorarium koordinator dari masing-masing kecamatan berdasarkan pada
				
				["bulan"]=>
				string(5) "Juni"
				["tahun"]=>
				string(4) "2017"

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 3 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*). Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* atau data uji yang dapat mencakup semua baris kode program ditunjukkan pada Tabel 5.18.

Tabel 5.18 Tabel Pengujian *Statement coverage* UC-06

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal *Statement* Tereksekusi	Nilai *Coverage*
TC06-1	P-063	Mengambil data laporan honorarium koordinator ["bulan_laporan"]=> string(5) "Juni" ["tahun_laporan"]=> string(4) "2017" dan ditampilkan data laporan honorarium koordinator berdasarkan bulan Juni dan Tahun 2017	Berhasil	22	22/22 = 100%

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah satu *test case*. Karena pada jalur P-063 sudah mencakup semua *statement* pada program. Dengan demikian pada TC06-1 sudah memastikan bahwa semua *statement* pada program telah di eksekusi dengan sekali uji dan berhasil dilakukan pengujian pada *function* tampil_honorkoordinator().

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya yaitu melakukan pengujian dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan (*true* dan *false*) dieksekusi dengan tepat. Adapun kasus uji (*test case*) yang mungkin mencakup nilai 100% *branch coverage* ditunjukkan pada Tabel 5.19.

Tabel 5.19 Tabel Pengujian *Branch coverage* UC-06

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Coverage*
TC06-1	P-061	Mengambil data laporan honorarium koordinator berdasarkan ["bulan_laporan"]=> string(0) " " ["tahun_laporan"]=> string(0) " " dan ditampilkan data laporan honorarium koordinator berdasarkan seluruh bulan dan seluruh tahun	Berhasil	2	2/4 = 50 %

TC06-2	P-063	Mengambil data laporan honorarium koordinator berdasarkan ["bulan_laporan"]=> string(5) "Juni" ["tahun_laporan"]=> string(4) "2017" dan ditampilkan data laporan honorarium koordinator berdasarkan bulan Juni dan Tahun 2017	Berhasil	4	4/4 = 100 %

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah dua *test case*. Pengujian dilakukan dengan mengidentifikasi pada setiap hasil keputusan atau cabang yang ada program seperti pada TC06-1 dan TC06-2. Dengan demikian pengujian pada ke dua *test case* tersebut memastikan bahwa setiap keputusan atau cabang yang ada pada program sudah dilakukan eksekusi masing-masing baik pada kondisi true maupun false, dengan minimal dengan sekali test pada masing-masing cabang dan pengujian pada *function* tampil_honorkoordinator() telah berhasil dilakukan.

5.2.2.7 Pengujian UC-07

Pada halaman melihat informasi rekapitulasi laporan bulanan (UC-07) terdiri dari beberapa rekapitulasi laporan, dengan keterbatasan waktu dan penulisan laporan maka pada UC-07 ini yang disampaikan dalam laporan yaitu pengujian pada halaman rekapitulasi laporan kinerja kader. Informasi rekapitulasi kinerja kader terdapat dalam class Laporan_honorarium yang disampaikan dalam *function* tampil_kinerjakader() yang terdiri dari 2 *conditional statement,* 1 *conditional looping* (perulangan) dan 52 *statement*. Pengujian dilakukan untuk memastikan bahwa struktur atau alur logika kode program pada *function* tampil_kinerjakader() sudah berjalan sesuai dengan yang diharapkan. Beberapa tahapan pengujian dan hasil dari pengujian pada halaman UC-07 adalah sebagai berikut:

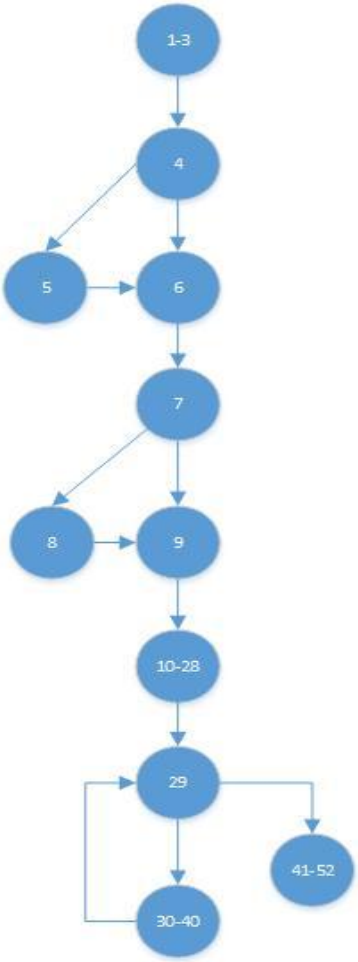
Menentukan blok kode program yaitu *function* tampil_kinerjakader(). Adapun *source code* pada *function* tampil_kinerjakader() dapat dilihat pada Gambar 5.45.

```
    public function tampil_kinerjakader(){

1.      $arr_bulan = array(1 => 'Januari', 'Februari', 'Maret', 'April',
    'Mei', 'Juni', 'Juli', 'Agustus', 'September', 'Oktober', 'November',
    'Desember');

2.      $bulan_laporan = $arr_bulan[(int) date('m')];
3.      $tahun_laporan = date('Y');

4.      if ($this->input->post('bulan_laporan') != "") {
```

```
5.          $bulan_laporan = $this->input->post("bulan_laporan");
6.      }
7.      if ($this->input->post('tahun_laporan') != "") {
8.          $tahun_laporan = $this->input->post("tahun_laporan");
9.      }

10.     $this->model_squrity->getsqurity();
11.     $isi['content']    = 'laporan/tampil_laporankinerjakd';
12.     $isi['judul']      = 'Laporan';
13.     $isi['sub_judul']  = 'Data Capaian Kinerja Kader TB';
14.     $isi['title']      =   'SIMONEV   COMMUNITY   TB-HIV   CARE   AISYIYAH
    TANGGAMUS';
15.     $isi['sub_title']  = 'COMMUNITY TB-HIV CARE AISYIYAH TANGGAMUS';
16.     $isi['bulan_laporan'] = $bulan_laporan;
17.     $isi['tahun_laporan'] = $tahun_laporan;
18.     $isi['data'] = $this->model_datahonorarium->listdata($bulan_laporan,
    $tahun_laporan);

19.     $tot_terduga_tb     = 0;
20.     $tot_bta_plus       = 0;
21.     $tot_rontgen_plus   = 0;
22.     $tot_ekstra_paru    = 0;
23.     $tot_tbanak_plus    = 0;
24.     $tot_geneXpert_plus = 0;
25.     $tot_bta_minus      = 0;
26.     $tot_rontgen_minus  = 0;
27.     $tot_tbanak_minus   = 0;
28.     $tot_geneXpert_minus = 0;

    // Menghitung total
29.     foreach ($isi['data']->result() as $row) {
30.         $tot_terduga_tb     += $row->jmlterduga_tb;
31.         $tot_bta_plus       += $row->jmlbta_plus;
32.         $tot_rontgen_plus   += $row->jmlrontgen_plus;
33.         $tot_ekstra_paru    += $row->jmlekstra_paru;
34.         $tot_tbanak_plus    += $row->jmltbanak_plus;
35.         $tot_geneXpert_plus += $row->jmlgeneXpert_plus;
36.         $tot_bta_minus      += $row->jmlbta_minus;
37.         $tot_rontgen_minus  += $row->jmlrontgen_minus;
38.         $tot_tbanak_minus   += $row->jmltbanak_minus;
39.         $tot_geneXpert_minus += $row->jmlgeneXpert_minus;
40.     }
41.     $isi['total_terduga_tb']    = $tot_terduga_tb;
42.     $isi['total_bta_plus']      = $tot_bta_plus;
43.     $isi['total_rontgen_plus']  = $tot_rontgen_plus;
44.     $isi['total_ekstra_paru']   = $tot_ekstra_paru;
45.     $isi['total_tbanak_plus']   = $tot_tbanak_plus;
46.     $isi['total_geneXpert_plus'] = $tot_geneXpert_plus;
47.     $isi['total_bta_minus']     = $tot_bta_minus;
48.     $isi['total_rontgen_minus'] = $tot_rontgen_minus;
49.     $isi['total_tbanak_minus']  = $tot_tbanak_minus;
50.     $isi['total_geneXpert_minus'] = $tot_geneXpert_minus;
51.     $this->load->view('backend/tampilan_home',$isi);
52. }
```

Gambar 5.45 *Source Code Function* tampil_laporankinerjakd()

Pengubahan *source code* menjadi *flowgraph* pada *function* tampil_laporankinerjakd() dapat dilihat pada Gambar 5.46.



Gambar 5.46 *Flowgraph* tampil_laporankinerjakd()

Berdasarkan *flowgraph* pada *function* tampil_laporankinerjakd() terdiri dari 52 *statement*, 4 branch, 11 node (N), dan 13 edge (E). Dari *flowgarph* pada Gambar 5.46 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 13 – 11 + 2 = 2 + 1 = 4

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.46 adalah 3. Berdasarkan perhitungan *Cyclomatic Complexity* tersebut, maka akan ditentukan *independent path* (jalur) dari *function* tampil_honorkoordinator() berdasarkan perolehan cc yaitu ada 3 jalur.

Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-071 = 1-3,4,6,7,9,10-28,29,41-52

   Keterangan admin tidak menginputkan bulan laporan (kosong atau nol) dan tahun laporan (kosong atau nol) yaitu yang ditampilkan data bulan sekarang dan tahun sekarang dan tidak dihitung untuk total capaian untuk setiap tipe tb (jenis penyakit tb).

2) Jalur P-072 = 1-3,4,6,7,9,10-28,29, 30-40,41-52

   Keterangan admin tidak menginputkan bulan laporan (kosong atau nol) dan tahun laporan (kosong atau nol) yaitu yang ditampilkan data bulan sekarang dan tahun sekarang dan dihitung untuk total capaian untuk setiap tipe tb (jenis penyakit tb).

3) Jalur P-073 = 1-3,4,5,6,7,9,10-28,29,30-40,41-52

   Keterangan admin menginputkan bulan laporan yaitu bulan April, tetapi tidak menginputkan tahun laporan dan dihitung total capaian untuk setiap tipe tb (jenis penyakit tb).

4) Jalur P-074 = 1-3,4,5,6,7,8,9,10-28,29,30-40,41-52

   Keterangan admin menginputkan bulan laporan yaitu bulan April dan tahun laporan yaitu tahun 2017 dan dihitung total capaian untuk setiap tipe tb (jenis penyakit tb).

Tabel 5.20 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity.*

Tabel 5.20 Tabel Rancangan data uji pada function tampil_kinerjakd()

No Jalur	***Statement* (baris kode)**	**Jumlah Branch**	**Masukkan data**	**Keluaran yang diharapkan**
P-071	$bulan_laporan =$arr_bulan[(int) date('m')]; $tahun_laporan = date('Y'); $isi['data'] = $this-> model_datahonorarium- >listdata($bulan_laporan, $tahun_laporan);	2	"bulan" => " " "tahun" => " "	Menampilkan data laporan capaian kinerja kader tb berdasarkan pada bulan sekarang dan tahun sekarang dan hasil total capaian tidak dihitung ["bulan_laporan"]=> string(9) "Agustus" ["tahun_laporan"]=> string(4) "2018" ["total_terduga_tb"]=> int(0) ["total_bta_plus"]=> int(0) ["total_rontgen_plus"]=> int(0) ["total_ekstra_paru"]=> int(0) ["total_tbanak_plus"]=> int(0) ["total_geneXpert_plus"]=> int(0) ["total_bta_minus"]=> int(0) ["total_rontgen_minus"]=> int(0) ["total_tbanak_minus"]=> int(0) ["total_geneXpert_minus"]=> int(0)

P-072	$bulan_laporan =$arr_bulan[(int) date('m')];
	$tahun_laporan = date('Y');
	
	$isi['data'] = $this->
	model_datahonorarium-
	>listdata($bulan_laporan,
	$tahun_laporan);
	
	foreach ($isi['data']->result()
	as $row) {
	$tot_terduga_tb += $row-
	>jmlterduga_tb;
	$tot_bta_plus += $row-
	>jmlbta_plus;
	$tot_rontgen_plus += $row-
	>jmlrontgen_plus;
	$tot_ekstra_paru += $row-
	>jmlekstra_paru;
	$tot_tbanak_plus += $row-
	>jmltbanak_plus;
	$tot_geneXpert_plus +=
	$row->jmlgeneXpert_plus;
	$tot_bta_minus += $row-
	>jmlbta_minus;
	$tot_rontgen_minus +=
	$row->jmlrontgen_minus;
	$tot_tbanak_minus +=
	$row->jmltbanak_minus;
	$tot_geneXpert_minus +=
	$row->jmlgeneXpert_minus;
	}
		2	"bulan" => " "
			"tahun" => " "
				Menampilkan data laporan capaian kinerja kader tb berdasarkan pada bulan sekarang dan tahun sekarang dan hasil total capaian dihitung
				
				["bulan_laporan"]=>
				string(9) "Agustus"
				["tahun_laporan"]=>
				string(4) "2018"
				["total_terduga_tb"]=>
				int(15)
				["total_bta_plus"]=>
				int(4)
				["total_rontgen_plus"]=>
				int(1)
				["total_ekstra_paru"]=>
				int(0)
				["total_tbanak_plus"]=>
				int(0)
				["total_geneXpert_plus"]=>
				int(0)
				["total_bta_minus"]=>
				int(10)
				["total_rontgen_minus"]=>
				int(0)
				["total_tbanak_minus"]=>
				int(0)
				["total_geneXpert_minus"]=>
				int(0)

P-073	if ($this->input-
	>post('bulan_laporan') != "")
	{
	$bulan_laporan = $this-
	>input -
	>post("bulan_laporan");
	}
	
	$isi['data'] = $this->
	model_datahonorarium-
	>listdata($bulan_laporan,
	$tahun_laporan);
	
	foreach ($isi['data']->result()
	as $row) {
	$tot_terduga_tb += $row-
	>jmlterduga_tb;
	$tot_bta_plus += $row-
	>jmlbta_plus;
	$tot_rontgen_plus += $row-
	>jmlrontgen_plus;
	$tot_ekstra_paru += $row-
	>jmlekstra_paru;
	$tot_tbanak_plus += $row-
	>jmltbanak_plus;
		1	"bulan" => April
			"tahun" => " "
				Menampilkan data laporan capaian kinerja kader tb dan total capaian tipe tb berdasarkan pada
				
				["bulan_laporan"]=>
				string(5) "April"
				["tahun_laporan"]=>
				string(4) "2018"
				
				["total_terduga_tb"]=>
				int(11)
				["total_bta_plus"]=>
				int(0)
				["total_rontgen_plus"]=>
				int(1)
				["total_ekstra_paru"]=>
				int(2)
				["total_tbanak_plus"]=>
				int(0)
				["total_geneXpert_plus"]=>
				int(0)
				["total_bta_minus"]=>
				int(8)
				["total_rontgen_minus"]=>
				int(0)
				["total_tbanak_minus"]=>

	$tot_geneXpert_plus += $row->jmlgeneXpert_plus; $tot_bta_minus += $row->jmlbta_minus; $tot_rontgen_minus += $row->jmlrontgen_minus; $tot_tbanak_minus += $row->jmltbanak_minus; $tot_geneXpert_minus += $row->jmlgeneXpert_minus; }			int(0) ["total_geneXpert_minus"]=> int(0)

P-074	if ($this->input->post('bulan_laporan') != "")
	{
	$bulan_laporan = $this->input->post("bulan_laporan");
	}
	
	if ($this->input->post('tahun_laporan') != "")
	{
	$tahun_laporan = $this->input->post("tahun_laporan");
	}
	
	$isi['data'] = $this->model_datahonorarium->listdata($bulan_laporan, $tahun_laporan);
	
	foreach ($isi['data']->result() as $row) {
	$tot_terduga_tb += $row->jmlterduga_tb;
	$tot_bta_plus += $row->jmlbta_plus;
	$tot_rontgen_plus += $row->jmlrontgen_plus;
	$tot_ekstra_paru += $row->jmlekstra_paru;
	$tot_tbanak_plus += $row->jmltbanak_plus;
	$tot_geneXpert_plus += $row->jmlgeneXpert_plus;
	$tot_bta_minus += $row->jmlbta_minus;
	$tot_rontgen_minus += $row->jmlrontgen_minus;
	$tot_tbanak_minus += $row->jmltbanak_minus;
	$tot_geneXpert_minus += $row->jmlgeneXpert_minus;
	}
		2	"bulan" => April
			"tahun" => 2017
				Menampilkan data laporan honorarium capaian kinerja kader tb dan total capaian tipe tb berdasarkan pada
				
				["bulan_laporan"]=>
				string(5) "April"
				["tahun_laporan"]=>
				string(4) "2017"
				
				["total_terduga_tb"]=>
				int(108)
				["total_bta_plus"]=>
				int(4)
				["total_rontgen_plus"]=>
				int(2)
				["total_ekstra_paru"]=>
				int(0)
				["total_tbanak_plus"]=>
				int(1)
				["total_geneXpert_plus"]=>
				int(2)
				["total_bta_minus"]=>
				int(92)
				["total_rontgen_minus"]=>
				int(0)
				["total_tbanak_minus"]=>
				int(7)
				["total_geneXpert_minus"]=>
				int(0)

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 4 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*). Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* data uji yang dapat mencakup semua baris kode program ditunjukkan pada Tabel 5.21.

Tabel 5.21 Tabel Pengujian *Statement coverage* UC-07

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Statement*
TC07-1	P-074	Mengambil data capaian kinerja kader dari penjaringan suspect dan menghitung total capaian setiap tipe tb berdasarkan pada ["bulan_laporan"]=> string(5) "April" ["tahun_laporan"]=> string(4) "2017" dan ditampilkan data laporan capaian kinerja kader dan total capaian setiap tipe tb berdasarkan bulan April dan Tahun 2017 ["total_terduga_tb"]=> int(108) ["total_bta_plus"]=> int(4) ["total_rontgen_plus"]=> int(2) ["total_ekstra_paru"]=> int(0) ["total_tbanak_plus"]=> int(1) ["total_geneXpert_plus"]=> int(2) ["total_bta_minus"]=> int(92) ["total_rontgen_minus"]=> int(0) ["total_tbanak_minus"]=> int(7) ["total_geneXpert_minus"]=> int(0)	Berhasil	52	52/52 = 100 %

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage statement coverage* 100% adalah satu *test case*. Karena pada jalur P-074 sudah mencakup semua *statement* pada program. Dengan demikian pada TC07-1

sudah membuktikan bahwa semua *statement* pada program telah di eksekusi dengan sekali uji dan berhasil dilakukan pengujian pada *function* tampil_laporankinerjakd().

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya yaitu melakukan pengujian dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan (*true* dan *false*) dieksekusi dengan tepat. Adapun kasus uji (*test case*) yang mungkin mencakup nilai 100% *branch coverage* ditunjukkan pada Tabel 5.22.

Tabel 5.22 Tabel Pengujian *Branch coverage* UC-07

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Branch*
TC07-1	P-071	Mengambil data capaian kinerja kader dari penjaringan suspect berdasarkan ["bulan_laporan"]=> string(0) " " ["tahun_laporan"]=> string(0) " " dan ditampilkan data laporan capaian kinerja kader berdasarkan bulan sekarang dan tahun sekarang	Berhasil	2	2/4 = 50 %
TC07-2	P-074	Mengambil data capaian kinerja kader dari penjaringan suspect dan menghitung total capaian setiap tipe tb berdasarkan pada ["bulan_laporan"]=> string(5) "April" ["tahun_laporan"]=> string(4) "2017" dan ditampilkan data laporan capaian kinerja kader dan total capaian setiap tipe tb berdasarkan bulan April dan Tahun 2017 ["total_terduga_tb"]=> int(108) ["total_bta_plus"]=> int(4) ["total_rontgen_plus"]=> int(2) ["total_ekstra_paru"]=> int(0) ["total_tbanak_plus"]=>	Berhasil	4	4/4 = 100 %

		int(1) ["total_geneXpert_plus"]=> int(2) ["total_bta_minus"]=> int(92) ["total_rontgen_minus"]=> int(0) ["total_tbanak_minus"]=> int(7) ["total_geneXpert_minus"]=> int(0)			

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah dua *test case*. Pengujian dilakukan dengan mengidentifikasi pada setiap hasil keputusan atau cabang yang ada program seperti pada TC07-1 dan TC07-2. Dengan demikian pengujian pada ke dua *test case* tersebut memastikan bahwa setiap keputusan atau cabang yang ada pada program sudah dilakukan eksekusi masing-masing baik pada kondisi true maupun false, dengan minimal dengan sekali test dan pengujian pada *function* tampil_laporankinerjakd() telah berhasil dilakukan menggunakan teknik *branch coverage.*

5.2.2.8 Pengujian UC-08

Pada halaman melihat informasi capaian yang diperoleh dari setiap kecamatan (UC-08) terdiri dari beberapa kategori indikator, dengan keterbatasan waktu dan penulisan laporan maka dalam kasus ini dilakukan pengujian pada halaman capaian kecamatan. Informasi capaian terdapat dalam class Capaian pada *function* tampil_capaiankec() yang terdiri dari 3 *conditional statement* dan 45 *statement*. Pengujian dilakukan untuk memastikan bahwa alur atau logika program pada *function* tampil_capaiankec() sudah berjalan sesuai dengan yang diharapkan. Beberapa tahapan pengujian dan hasil dari pengujian pada halaman UC-08 adalah sebagai berikut:

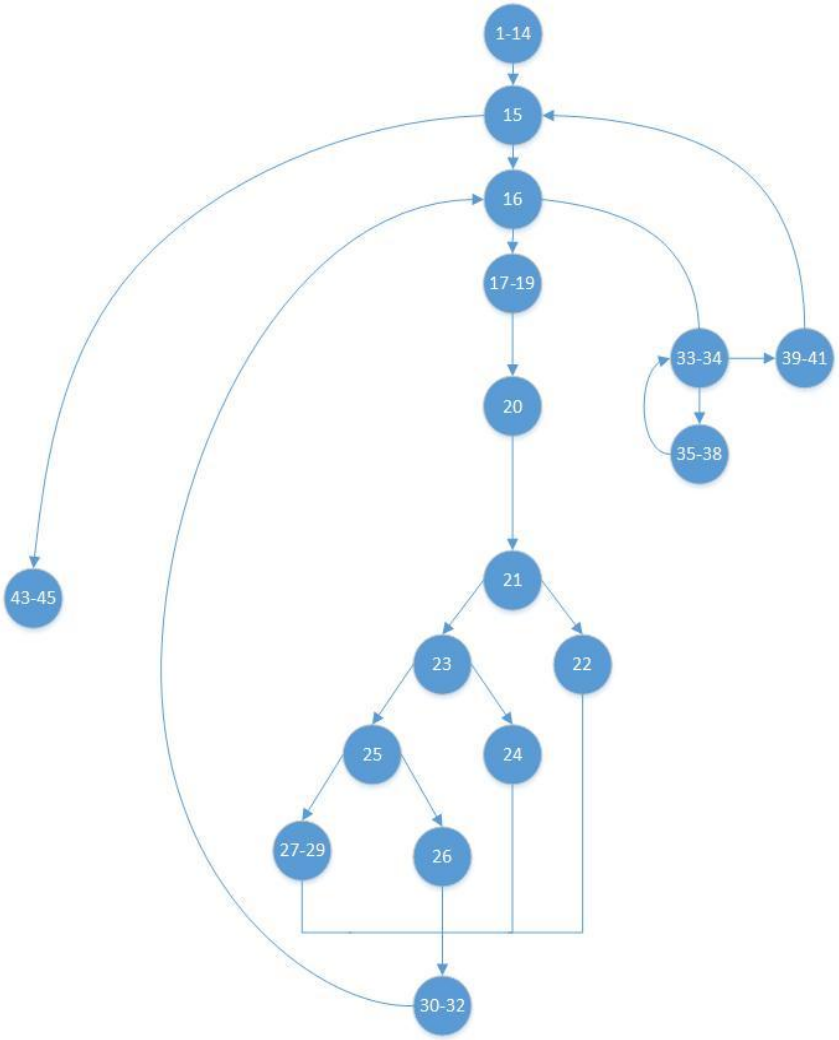
Menentukan blok kode program yaitu *function* tampil_capaiankec(). Adapun *source code* yang digunakan yaitu *function* tampil_capaiankec() dapat dilihat pada Gambar 5.47.

```
      public function tampil_capaiankec()
      {
1.      $tahun     = !empty($this->input->get('tahun')) ? $this->input-
      >get('tahun') : date('Y');
2.      $semester   =   !empty($this->input->get('semester'))   ?   $this-
      >input->get('semester') : 1;
3.      $triwulan   =   !empty($this->input->get('triwulan'))   ?   $this-
      >input->get('triwulan') : 1;
4.      $isi['tahun'] = $tahun;
```

```
5.     $isi['semester'] = $semester;
6.     $isi['triwulan'] = $triwulan;

7.     $this->model_squrity->getsqurity();
8.     $isi['content']  = 'target/tampil_capaiankec';
9.     $isi['title']    =  'SIMONEV  COMMUNITY  TB-HIV  CARE  AISYIYAH
   TANGGAMUS';
10.    $isi['sub_title'] = 'TB-HIV CARE';
11.    $isi['judul']    = 'Home';
12.    $isi['sub_judul'] = 'Detail Pengukuran Target Indikator';
13.    $data_kecamatan = $this->model_kecamatan->listdata()->result();
14.    $data_target =  $this->model_targetindikator->listdata(['tahun'
   => $tahun, 'semester' => $semester])->result();

15.        foreach ($data_kecamatan as $kecamatan) {
16.           for($kat=1;$kat<=3;$kat++){
17.              $tri = $triwulan;
18.              $sem = $semester;
19.              $hasiltes = "(hasil_testhiv = '' OR hasil_testhiv IS
   NULL)";
              if ($kat == 1){
                 $kats = "'2', '3', '4', '5', '6', '7', '8', '9', '10'";
              } else if ($kat == 2) {
20               $kats = "'2', '3', '4', '5', '6'";
              } else {
                 $kats = "'2', '3', '4','5','6','7'";
                 $hasiltes = "hasil_testhiv = 'Terima'";
              }
21.              if (($sem == 1) && ($tri == 1)) {
22.                 $bulans = "'Januari', 'Februari', 'Maret'";
23.              } else if (($sem == 1) && ($tri == 2)) {
24.                 $bulans = "'April', 'Mei', 'Juni'";
25.              } else if (($sem == 2) && ($tri == 1)) {
26.                 $bulans = "'Juli', 'Agustus', 'September'";
27.              } else {
28.                 $bulans = "'Oktober', 'November', 'Desember'";
29.              }

30.           $res = $this->model_datasuspect->jmldata("tahun_laporan =
   '$tahun' AND bulan_laporan IN ($bulans) AND id_tipetb IN ($kats) AND
   $hasiltes  AND  id_kecamatan  =  '".$kecamatan->id_kecamatan."'")-
   >result();
31.           $jml[$kat] = $res[0]->jml;
32.        }
33.        $append = new stdClass;
34.        foreach ($data_target as $target) {
35.           $name = 'target_'.$target->kategori_indikator;
36.           $append->$name   =   $kecamatan->target   *   $target-
   >nilai_target / 100;
37.           }
38.           $kecamatan->target = $append;
39.           $kecamatan->jml_1 = $jml[1];
40.           $kecamatan->jml_2 = $jml[2];
41.           $kecamatan->jml_3 = $jml[3];
42.        }
43.        $isi['data']     = $data_kecamatan;
44.        $this->load->view('backend/tampilan_home',$isi);
45.    }
```

Gambar 5.47 *Source Code Function* tampil_capaiankec()

Pengubahan *source code* menjadi *flowgraph* pada *function* tampil_capaiankec() dapat dilihat pada Gambar 5.48.



Gambar 5.48 *Flowgraph* tampil_capaiankec()

Berdasarkan *flowgraph* pada Gambar 5.48 terdapat total 45 *statement*, 6 *branch*, 17 *nodes*, dan 22 *edge*. Dari Gambar 5.47 dapat dhitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 22 – 17 + 2 = 5+ 2 = 7

Jadi, *Cyclomatic Complexity* yang diperoleh dari *function* tampil_capaiankec() pada Gambar 5.48 adalah 7. Berdasarkan *Cyclomatic Complexity* tersebut, maka terdapat 7 *independent path* (jalur) yang digunakan untuk mengidentifikasi jalur yang mungkin untuk dilakukan pengujian sampai semua *statement* atau cabang pada program di eksekusi.

Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1) Jalur P-081 = 1-14,15,43-45

   Keterangan admin tidak menginputkan tahun,semester, dan triwulan, menampilkan ketentuan nilai target setiap kecamatan, tidak menghitung nilai target untuk setiap indikator (terduga tb, teridentifikasi tb, dan teridentifikasi hiv) dan jumlah capaian indikator.

2) Jalur P-082 = 1-14,15,16,33-34,39-42,15,43-45

   Keterangan admin tidak menginputkan tahun,semester, dan triwulan, tidak menghitung target dan tidak dapat menampilkan jumlah capaian indikator.

3) Jalur P-083 = 1-14,15,16,33-34,35-38,39-42,15,43-45

   Keterangan admin tidak menginputkan tahun,semester, dan triwulan, di lakukan perhitungan nilai target dan tidak dapat menampilkan jumlah capaian indikator.

4) Jalur P-084 = 1-14,15,16,17-19,20,21,22,30-32,16,33-34,35-38,39-42,43-45

   Keterangan admin menginputkan tahun 2017, semester 1, dan triwulan 1, semester 1 dan triwulan 1 memiliki ketentuan yaitu bulan Januari, Februari, dan Maret

5) Jalur P-085 = 1-14,15,16,17-19,20,21,23,24,30-32,16,33-34,35-38,39-42,43-45

   Keterangan admin menginputkan tahun 2017, semester 1, dan triwulan 2, semester 1 dan triwulan 2 memiliki ketentuan yaitu bulan April, Mei, Juni.

6) Jalur P-086 = 1-14,15,16,17-19,20,21,23,25,26,30-32,16,33-34,35-38,39-42,43-45

   Keterangan admin menginputkan tahun 2017, semester 2, dan triwulan 1, semester 2 dan triwulan 1 memiliki ketentuan yaitu bulan Juli, Agustus, September.

7) Jalur P-083 = 1-14,15,16,17-19,20,21,23,25,27-29,30-32,16,33-34,35-38,39-42,43-45

   Keterangan admin menginputkan tahun 2017, semester 2, dan triwulan 1, semester 2 dan triwulan 2 memiliki ketentuan yaitu bulan Oktober, November, Desember.

Tabel 5.23 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity.*

Tabel 5.23 Tabel Rancangan data uji pada *function* tampil_capaiankec()

No Jalur	*Statement* (baris kode)	Jumlah Branch	Masukkan data	Keluaran yang diharapkan
P-081	$data_kecamatan = $this->model_kecamatan->listdata()->result(); foreach ($data_kecamatan as $kecamatan) {	0	"tahun" => " " "semester" => " " "triwulan" =>" "	Menampilkan capaian kecamatan berdasarkan pada ["tahun"]=> string(4) "2018" ["semester"]=> string(1) "1"

	} $isi['data'] = $data_kecamatan;			["triwulan"]=> string(1) "1" dan menampilkan ketentuan nilai target untuk setiap kecamatan ["nama_kecamatan"]=> string(13) "Talang Padang" ["target"]=> string(2) "15" atau sama dengan 15% ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target"]=> string(2) "10" atau sama dengan 10% ["nama_kecamatan"]=> string(9) "Sumberejo" ["target"]=> string(2) "10" atau sama dengan 10%

P-082	$data_kecamatan = $this->model_kecamatan->listdata()->result();
	
	foreach ($data_kecamatan as $kecamatan) {
	for($kat=1;$kat<=3;$kat++)
	{
	
	}
	$kecamatan->jml_1 = $jml[1];
	$kecamatan->jml_2 = $jml[2];
	$kecamatan->jml_3 = $jml[3];
	}
		0	"tahun" => " "
			"semester" => " "
			"triwulan" =>" "
				Menampilkan capaian kecamatan berdasarkan pada
				
				["tahun"]=>
				string(4) "2018"
				["semester"]=>
				string(1) "1"
				["triwulan"]=>
				string(1) "1"
				
				["nama_kecamatan"]=>
				string(13) "Talang Padang"
				["target"]=> " "
				["jml_1"]=> NULL
				["jml_2"]=> NULL
				["jml_3"]=> NULL
				
				["nama_kecamatan"]=>
				string(14) "Pulau Panggung"
				["target"]=> " "
				["jml_1"]=> NULL
				["jml_2"]=> NULL
				["jml_3"]=> NULL
				
				["nama_kecamatan"]=>
				string(9) "Sumberejo"
				["target"]=> " "
				["jml_1"]=> NULL
				["jml_2"]=> NULL
				["jml_3"]=> NULL

P-083	$data_kecamatan = $this->model_kecamatan->listdata()->result(); foreach ($data_kecamatan as $kecamatan) {	0	"tahun" => " " "semester" => " " "triwulan" =>" "	Menampilkan nilai target untuk setiap kategori indikator (terduga tb, teridentifikasi tb, teridentifikasi hiv) dengan membagi berdasarkan ketentuan nilai target pada

	for($kat=1;$kat<=3;$kat++)
	{
	
	}
	
	$append = new stdClass;
	foreach ($data_target as $target)
	{
	$name = 'target_'.$target->kategori_indikator;
	$append->$name = $kecamatan->target * $target->nilai_target / 100;
	}
	
	$kecamatan->target = $append;
	
	$kecamatan->jml_1 = $jml[1];
	$kecamatan->jml_2 = $jml[2];
	$kecamatan->jml_3 = $jml[3];
	}
				masing kecamatan sebelumnya
				
				["tahun"]=>
				string(4) "2018"
				["semester"]=>
				string(1) "1"
				["triwulan"]=>
				string(1) "1"
				
				["nama_kecamatan"]=>
				string(13) "Talang Padang"
				["target_1"]=>
				int(540)
				["target_2"]=>
				int(188)
				["target_3"]=>
				int(53)
				
				["jml_1"]=> NULL
				["jml_2"]=> NULL
				["jml_3"]=> NULL
				
				["nama_kecamatan"]=>
				string(14) "Pulau Panggung"
				["target_1"]=>
				int(360)
				["target_2"]=>
				int(125)
				["target_3"]=>
				int(35)
				
				["jml_1"]=> NULL
				["jml_2"]=> NULL
				["jml_3"]=> NULL
				
				["nama_kecamatan"]=>
				string(9) "Sumberejo"
				["target_1"]=>
				int(360)
				["target_2"]=>
				int(125)
				["target_3"]=>
				int(35)
				
				["jml_1"]=> NULL
				["jml_2"]=> NULL
				["jml_3"]=> NULL

P-084	$data_kecamatan = $this->model_kecamatan->listdata()->result();
	
	foreach ($data_kecamatan as $kecamatan) {
	for($kat=1;$kat<=3;$kat++)
	{
	if (($sem == 1) && ($tri == 1))
	{
		1	"tahun" => "2017"
			"semester" => "1"
			"triwulan" =>"1"
				Menampilkan data capaian indikator pada masing masing kecamatan berdasarkan pada
				
				["tahun"]=>
				string(4) "2017"
				["semester"]=>
				string(1) "1"
				["triwulan"]=>
				string(1) "1"
				
				["nama_kecamatan"]=>

	$bulans = "'Januari', 'Februari', 'Maret'";
	}
	
	$res = $this->model_datasuspect->jmldata("tahun_laporan = '$tahun' AND bulan_laporan IN ($bulans) AND id_tipetb IN ($kats) AND $hasiltes AND id_kecamatan = '".$kecamatan->id_kecamatan."'")->result();
	
	$jml[$kat] = $res[0]->jml;
	}
	
	$append = new stdClass;
	foreach ($data_target as $target)
	{
	$name = 'target_'.$target->kategori_indikator;
	$append->$name = $kecamatan->target * $target->nilai_target / 100;
	}
	
	$kecamatan->target = $append;
	
	$kecamatan->jml_1 = $jml[1];
	$kecamatan->jml_2 = $jml[2];
	$kecamatan->jml_3 = $jml[3];
	}
				string(13) "Talang Padang"
				["target_1"]=>
				int(188)
				["target_2"]=>
				int(132)
				["target_3"]=>
				int(56)
				
				["jml_1"]=> "42"
				["jml_2"]=> "11"
				["jml_3"]=> "1"
				
				["nama_kecamatan"]=>
				string(14) "Pulau Panggung"
				["target_1"]=>
				int(125)
				["target_2"]=>
				int(88)
				["target_3"]=>
				int(38)
				
				["jml_1"]=> "23"
				["jml_2"]=> "2"
				["jml_3"]=> "0"
				
				["nama_kecamatan"]=>
				string(9) "Sumberejo"
				["target_1"]=>
				int(125)
				["target_2"]=>
				int(88)
				["target_3"]=>
				int(38)
				
				["jml_1"]=> "8"
				["jml_2"]=> "0"
				["jml_3"]=> "0"

P-085	$data_kecamatan = $this->model_kecamatan->listdata()->result();
	
	foreach ($data_kecamatan as $kecamatan) {
	for($kat=1;$kat<=3;$kat++)
	{
	if (($sem == 1) && ($tri == 1))
	{
	$bulans = "'Januari', 'Februari', 'Maret'";
	}
	
	$res = $this->model_datasuspect->jmldata("tahun_laporan = '$tahun' AND bulan_laporan IN ($bulans) AND id_tipetb IN ($kats) AND $hasiltes
		2	"tahun" => "2017"
			"semester" => "1"
			"triwulan" =>"2"
				Menampilkan data capaian indikator pada masing masing kecamatan berdasarkan pada
				
				["tahun"]=>
				string(4) "2017"
				["semester"]=>
				string(1) "1"
				["triwulan"]=>
				string(1) "2"
				
				["nama_kecamatan"]=>
				string(13) "Talang Padang"
				["target_1"]=>
				int(188)
				["target_2"]=>
				int(132)
				["target_3"]=>
				int(56)
				
				["jml_1"]=> "90"

	AND id_kecamatan = '".$kecamatan->id_kecamatan."')->result();
	
	$jml[$kat] = $res[0]->jml;
	}
	
	$append = new stdClass;
	foreach ($data_target as $target)
	{
	$name = 'target_'.$target->kategori_indikator;
	$append->$name = $kecamatan->target * $target->nilai_target / 100;
	}
	
	$kecamatan->target = $append;
	
	$kecamatan->jml_1 = $jml[1];
	$kecamatan->jml_2 = $jml[2];
	$kecamatan->jml_3 = $jml[3];
	}
				["jml_2"]=> "10"
				["jml_3"]=> "1"
				
				["nama_kecamatan"]=>
				string(14) "Pulau Panggung"
				["target_1"]=>
				int(125)
				["target_2"]=>
				int(88)
				["target_3"]=>
				int(38)
				
				["jml_1"]=> "11"
				["jml_2"]=> "1"
				["jml_3"]=> "0"
				
				["nama_kecamatan"]=>
				string(9) "Sumberejo"
				["target_1"]=>
				int(125)
				["target_2"]=>
				int(88)
				["target_3"]=>
				int(38)
				
				["jml_1"]=> "7"
				["jml_2"]=> "1"
				["jml_3"]=> "0"

P-086	$data_kecamatan = $this->model_kecamatan->listdata()->result();
	
	foreach ($data_kecamatan as $kecamatan) {
	for($kat=1;$kat<=3;$kat++)
	{
	if (($sem == 1) && ($tri == 1))
	{
	$bulans = "'Januari', 'Februari', 'Maret'";
	}
	
	$res = $this->model_datasuspect->jmldata("tahun_laporan = '$tahun' AND bulan_laporan IN ($bulans) AND id_tipetb IN ($kats) AND $hasiltes AND id_kecamatan = '".$kecamatan->id_kecamatan."')->result();
	
	$jml[$kat] = $res[0]->jml;
	}
	
	$append = new stdClass;
		3	"tahun" => "2017"
			"semester" => "2"
			"triwulan" =>"1"
				Menampilkan data capaian indikator pada masing masing kecamatan berdasarkan pada
				
				["tahun"]=>
				string(4) "2017"
				["semester"]=>
				string(1) "2"
				["triwulan"]=>
				string(1) "1"
				
				["nama_kecamatan"]=>
				string(13) "Talang Padang"
				["target_1"]=>
				int(248)
				["target_2"]=>
				int(173)
				["target_3"]=>
				int(74)
				
				["jml_1"]=> "3"
				["jml_2"]=> "2"
				["jml_3"]=> "0"
				
				["nama_kecamatan"]=>
				string(14) "Pulau Panggung"
				["target_1"]=>
				int(165)
				["target_2"]=>
				int(115)
				["target_3"]=>

	foreach ($data_target as $target)
	{
	$name = 'target_'.$target->kategori_indikator;
	$append->$name = $kecamatan->target * $target->nilai_target / 100;
	}
	
	$kecamatan->target = $append;
	
	$kecamatan->jml_1 = $jml[1];
	$kecamatan->jml_2 = $jml[2];
	$kecamatan->jml_3 = $jml[3];
	}
				int(49)
				
				["jml_1"]=> "0"
				["jml_2"]=> "0"
				["jml_3"]=> "0"
				
				["nama_kecamatan"]=>
				string(9) "Sumberejo"
				["target_1"]=>
				int(165)
				["target_2"]=>
				int(115)
				["target_3"]=>
				int(50)
				
				["jml_1"]=> "0"
				["jml_2"]=> "0"
				["jml_3"]=> "0"

P-087	$data_kecamatan = $this->model_kecamatan->listdata()->result();
	
	foreach ($data_kecamatan as $kecamatan) {
	for($kat=1;$kat<=3;$kat++)
	{
	if (($sem == 1) && ($tri == 1))
	{
	$bulans = "'Januari', 'Februari', 'Maret'";
	}
	
	$res = $this->model_datasuspect->jmldata("tahun_laporan = '$tahun' AND bulan_laporan IN ($bulans) AND id_tipetb IN ($kats) AND $hasiltes AND id_kecamatan = '".$kecamatan->id_kecamatan."'")->result();
	
	$jml[$kat] = $res[0]->jml;
	}
	
	$append = new stdClass;
	foreach ($data_target as $target)
	{
	$name = 'target_'.$target->kategori_indikator;
	$append->$name = $kecamatan->target * $target->nilai_target / 100;
	}
		3	"tahun" => "2017"
			"semester" => "2"
			"triwulan" =>"2"
				Menampilkan data capaian indikator pada masing masing kecamatan berdasarkan pada
				
				["tahun"]=>
				string(4) "2017"
				["semester"]=>
				string(1) "2"
				["triwulan"]=>
				string(1) "2"
				
				["nama_kecamatan"]=>
				string(13) "Talang Padang"
				["target_1"]=>
				int(248)
				["target_2"]=>
				int(173)
				["target_3"]=>
				int(74)
				
				["jml_1"]=> "1"
				["jml_2"]=> "1"
				["jml_3"]=> "0"
				
				["nama_kecamatan"]=>
				string(14) "Pulau Panggung"
				["target_1"]=>
				int(165)
				["target_2"]=>
				int(115)
				["target_3"]=>
				int(49)
				
				["jml_1"]=> "0"
				["jml_2"]=> "0"
				["jml_3"]=> "0"
				
				["nama_kecamatan"]=>
				string(9) "Sumberejo"
				["target_1"]=>

	$kecamatan->target = $append; $kecamatan->jml_1 = $jml[1]; $kecamatan->jml_2 = $jml[2]; $kecamatan->jml_3 = $jml[3]; }			int(165) ["target_2"]=> int(115) ["target_3"]=> int(50) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0"

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 9 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dijalankan pengujian berdasarka kasus uji (*test case*) yang dirancang sebelumnya. Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* pada *function* tampil_capaiankec() ditunjukkan pada Tabel 5.24.

Tabel 5.24 Tabel Pengujian *Statement coverage* UC-08

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Statement*
TC08-1	P-084	Mengambil jumlah capaian indikator pada setiap kecamatan dan Ditampilkan data capaian indikator pada masing berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "1" ["triwulan"]=> string(1) "1" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(188) ["target_2"]=> int(132) ["target_3"]=> int(56) ["jml_1"]=> "42" ["jml_2"]=> "11" ["jml_3"]=> "1" ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(125)	Berhasil	38	38/45 = 84,44%

		["target_2"]=> int(88) ["target_3"]=> int(38) ["jml_1"]=> "23" ["jml_2"]=> "2" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=> int(125) ["target_2"]=> int(88) ["target_3"]=> int(38) ["jml_1"]=> "8" ["jml_2"]=> "0" ["jml_3"]=> "0"			
TC08-2	P-085	Mengambil jumlah capaian indikator pada setiap kecamatan dan Ditampilkan data capaian indikator pada masing berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "1" ["triwulan"]=> string(1) "2" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(188) ["target_2"]=> int(132) ["target_3"]=> int(56) ["jml_1"]=> "90" ["jml_2"]=> "10" ["jml_3"]=> "1" ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(125) ["target_2"]=> int(88) ["target_3"]=> int(38) ["jml_1"]=> "11" ["jml_2"]=> "1" ["jml_3"]=> "0"	Berhasil	2	40/45 = 88,88%

		["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=> int(125) ["target_2"]=> int(88) ["target_3"]=> int(38) ["jml_1"]=> "7" ["jml_2"]=> "1" ["jml_3"]=> "0"			
TC08-3	P-086	Mengambil jumlah capaian indikator pada setiap kecamatan dan Ditampilkan data capaian indikator pada masing berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "2" ["triwulan"]=> string(1) "1" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(248) ["target_2"]=> int(173) ["target_3"]=> int(74) ["jml_1"]=> "3" ["jml_2"]=> "2" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(49) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(50)	Berhasil	2	42/45 = 93,33%

		["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0"			
TC08-4	P-087	Menampilkan data capaian indikator pada masing masing kecamatan berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "2" ["triwulan"]=> string(1) "2" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(248) ["target_2"]=> int(173) ["target_3"]=> int(74) ["jml_1"]=> "1" ["jml_2"]=> "1" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(49) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(50) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0"	Berhasil	3	45/45 = 100 %

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage* 100% adalah empat *test case*. Dalam TC081, mengambil

jumlah data capaian penjaringan *suspect* untuk semester 1 dan triwulan 1 yaitu bulan Januari, Februari, dan Maret, pada tahun 2017 dengan ketentuan untuk capaian semua kategori indikator (terduga tb, teridentifikasi tb, dan teridentifikasi hiv) sehingga baris kode yang dijalankan pada *test case* ini yaitu baris 1 hingga 22 dan baris 30 hingga baris 45. Karena telah mencakup 38 pernyataan dari 45 pernyataan maka *statement coverage* yang diperoleh yaitu pada *test case* satu yaitu 84,44%. Namun pada TC08-1 belum mencakup semua pernyataan yang ada sehingga dilakukan pengujian pada jalur berikut nya sampai dengan seluruh pernyataan tercakup dan memperoleh nilai *statement coverage* 100%. Pada *function* tampil_capiankec() ini dilakukan sampai pada TC08-4 untuk memperoleh nilai *statement coverage* 100% dan pengujian berhasil dilakukan karena semua pernyataan telah di eksekusi dan menghasilkan nilai keluaran pada setiap variabel yang benar dan valid hanya saja pada perulangan dari baris kode 16 ke baris kode 39-41 tidak dapat menampilkan jumlah capaian indikator, karna belum dilakukan perulangan untuk mengambil jumlah capaian indikator (terduga tb, teridentifikasi tb, dan teridentifikasi hiv) dari baris kode 16 sampai dengan 32.

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya yaitu melakukan pengujian dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan (*true* dan *false*) dieksekusi dengan tepat. Adapun kasus uji (*test case*) yang mungkin mencakup nilai 100% *branch coverage* ditunjukkan pada Tabel 5.25.

Tabel 5.25 Tabel Pengujian *Branch coverage* UC-08

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal Eksekusi	Nilai *Branch*
TC08-1	P-087	Menampilkan data capaian indikator pada masing masing kecamatan berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "2" ["triwulan"]=> string(1) "2" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(248)	Berhasil	1	3/6 = 50%

		["target_2"]=> int(173) ["target_3"]=> int(74) ["jml_1"]=> "1" ["jml_2"]=> "1" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(49) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(50) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0"			
TC08-2	P-086	Mengambil jumlah capaian indikator pada setiap kecamatan dan Ditampilkan data capaian indikator pada masing berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "2" ["triwulan"]=> string(1) "1" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(248) ["target_2"]=> int(173) ["target_3"]=> int(74) ["jml_1"]=> "3" ["jml_2"]=> "2" ["jml_3"]=> "0"	Berhasil	4	4/6 = 66,67%

		["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(49) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=> int(165) ["target_2"]=> int(115) ["target_3"]=> int(50) ["jml_1"]=> "0" ["jml_2"]=> "0" ["jml_3"]=> "0"			
TC08-3	P-085	Mengambil jumlah capaian indikator pada setiap kecamatan dan Ditampilkan data capaian indikator pada masing berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "1" ["triwulan"]=> string(1) "2" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(188) ["target_2"]=> int(132) ["target_3"]=> int(56) ["jml_1"]=> "90" ["jml_2"]=> "10" ["jml_3"]=> "1" ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(125) ["target_2"]=> int(88) ["target_3"]=>	Berhasil	5	5/6 = 83,33%

		int(38) ["jml_1"]=> "11" ["jml_2"]=> "1" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=> int(125) ["target_2"]=> int(88) ["target_3"]=> int(38) ["jml_1"]=> "7" ["jml_2"]=> "1" ["jml_3"]=> "0"			
TC08-4	P-084	Mengambil jumlah capaian indikator pada setiap kecamatan dan Ditampilkan data capaian indikator pada masing berdasarkan pada ["tahun"]=> string(4) "2017" ["semester"]=> string(1) "1" ["triwulan"]=> string(1) "1" ["nama_kecamatan"]=> string(13) "Talang Padang" ["target_1"]=> int(188) ["target_2"]=> int(132) ["target_3"]=> int(56) ["jml_1"]=> "42" ["jml_2"]=> "11" ["jml_3"]=> "1" ["nama_kecamatan"]=> string(14) "Pulau Panggung" ["target_1"]=> int(125) ["target_2"]=> int(88) ["target_3"]=> int(38) ["jml_1"]=> "23" ["jml_2"]=> "2" ["jml_3"]=> "0" ["nama_kecamatan"]=> string(9) "Sumberejo" ["target_1"]=>	Berhasil	6	6/6 = 100%

		int(125) ["target_2"]=> int(88) ["target_3"]=> int(38) ["jml_1"]=> "8" ["jml_2"]=> "0" ["jml_3"]=> "0"			

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang didapat untuk memperoleh nilai *coverage branch coverage* 100% adalah empat *test case*. Pengujian dilakukan dengan mengidentifikasi terhadap keputusan atau cabang yang ada program seperti pada TC08-1, TC08-2, TC08-3 dan TC08-4. Dengan demikian pengujian pada ke empat *test case* tersebut memastikan bahwa setiap keputusan atau cabang yang ada pada program sudah dilakukan eksekusi minimal dengan sekali test dan pengujian pada *function* tampil_capaiankec() telah berhasil dilakukan pengujian.

5.2.2.9 Pengujian KNF-01 (Login)

Pada halaman login terdapat beberapa method dalam *class* login. Pada pengujian ini dilakukan pada *function* getlogin() yang terdiri dari 1 *conditional statement* dan 14 *statement*, pengujian dilakukan untuk memastikan bahwa alur atau logika program pada *function* getlogin() sudah berjalan sesuai dengan yang diharapkan. Beberapa tahapan pengujian dan hasil dari pengujian pada halaman login adalah sebagai berikut:

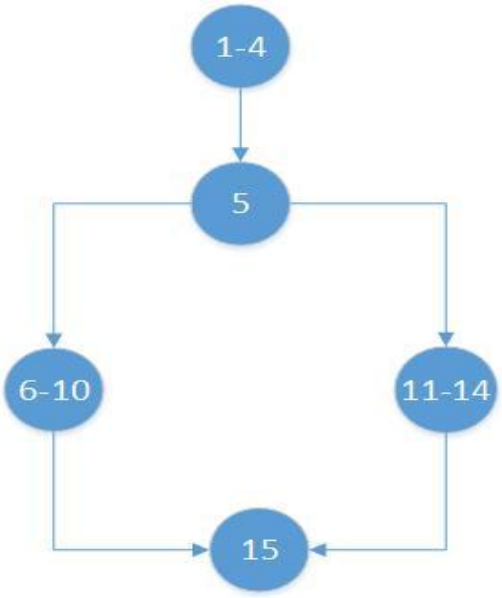
Menentukan blok kode program yaitu *function* getlogin(). Adapun *source code* pada *function* getlogin() dapat dilihat pada Gambar 5.49.

```
    public function getlogin()
        {
    1.      $u = $this->input->post('username');
    2.      $p = $this->input->post('password');
    3.      $this->load->model('model_login');
    4.      $query = $this->model_login->getlogin($u,$p);
    5.      if($query->num_rows()==1)
    6.      {
    7.          $row = $query->row();
    8.          $sess = array('username'      => $row->username,
                              'nama_lengkap'   => $row->nama_lengkap,
                              'jabatan'        => $row->jabatan,
                              'id_kecamatan'   => $row->id_kecamatan,
                              'nama_kecamatan'    => $row->nama_kecamatan,
                              'foto'           => $row->foto );
    9.          $this->session->set_userdata($sess);
    10.         redirect('backend/home');
    11.     } else {
    12.         $this->session->set_flashdata('info','maaf username atau password
        salah');
    13.         redirect('login');
```

```
14.     }
15. }
```

Gambar 5.49 *Source Code Function* getlogin()

Pengubahan *source code* menjadi *flowgraph* pada *function* getlogin() dapat dilihat pada Gambar 5.50.



Gambar 5.50 *Flowgraph* getlogin()

Berdasarkan *flowgraph* pada *function* getlogin() terdapat total 15 *statement*, 2 branch, 5 node (N), dan 5 edge (E). Dari *flowgarph* pada Gambar 5.50 maka dapat dihitung nilai *Cyclomatic Complexity* sebagai berikut:

V(G) = E – N + 2 = 5 – 5 + 2 = 0 + 2 = 2

Jadi, *Cyclomatic Complexity* yang diperoleh berdasarkan *flowgraph* pada Gambar 5.50 adalah 2. Berdasarkan *Cyclomatic Complexity* tersebut, maka terdapat 2 *independent* path (jalur) yang digunakan untuk mengidentifikasi jalur yang mungkin untuk dilakukan pengujian sampai semua *statement* atau cabang pada program di eksekusi. Adapun identifikasi jalur yang mungkin untuk dilakukan uji coba antara lain:

1. Jalur P-091 = 1-4,5,6-10,15.

   Keterangan, pengguna sistem menginputkan username afriadi_ssr dan password 12345678, login sukses.
2. Jalur P-042 = 1-4,5,11-14,15

Keterangan, pengguna sistem menginpurkan username afriadi_123 dan password 12345678, incorrect username atau password

Tabel 5.26 menunjukkan data uji yang mungkin dilakukan dalam pengujian berdasarkan dari independent path yang diukur menggunakan metriks *Cyclomatic Complexity* .

Tabel 5.26 Tabel Rancangan data uji pada *function* edit() kader

No Jalur	*Statement*	Jumlah Branch	Masukkan data	Keluaran yang diharapkan
P-KNF01	$query = $this->model_login->getlogin($u,$p); $this->session->set_userdata($sess); redirect('backend/home');	1	"username " => afriadi_ssr "password" => 12345678	Login diterima
P-KNF02	$query = $this->model_login->getlogin($u,$p); $this->session->set_flashdata('info','maaf username atau password salah'); redirect('login');	1	"username " => afriadi_123 "password" => 12345678	Login ditolak

a. Pengujian *Statement coverage*

Berdasarkan tahapan-tahapan yang telah di uraikan sebelumnya, diperoleh 2 *path* (jalur). Untuk mengetahui tingkat keberhasilan dari program maka akan dilakukan sebuah kasus uji (*test case*). Adapun minimal *test* yang diperlukan untuk mencakup nilai 100% *statement coverage* ditunjukkan pada Tabel 5.27.

Tabel 5.27 Tabel Pengujian Jalur *Statement coverage* KNF-01

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal eksekusi	Nilai *Statement*
KNF01-1	P-01	Mengambil data username dan password, Login diterima	Berhasil	10	10/15 = 66,67%
KNF01-2	P-02	Mengambil data username dan password, Login ditolak	Berhasil	5	15/15 = 100%

Berdasarkan pengujian yang sudah dilakukan, maka minimal *test case* yang diperoleh untuk mencakup semua *statement* dan memperoleh nilai *coverage* 100% dilakukan *test* pada ke dua jalur yang sudah ditetapkan sebelumnya. Dimana pada jalur P-01 memperoleh nilai sebesar 66,67%, hal tersebut menyatakan bahwa ada *statement* yang belum di eksekusi, maka akan dilakukan pengujian pada jalur berikutnya yaitu pada jalur P-02 untuk mencakup semua *statement* pada program. Dengan demikian pengujian dengan KNF01-1 dan KNF02-2, bahwa setiap *statement* telah di eksekusi dengan minimal sekali test pada setiap jalur yang sudah ditetapkan sebelumnya. Dan pengujian pada *function* getlogin() kader telah berhasil dilakukan dan semua *statement* (baris kode) pada *function* tersebut telah di eksekusi dengan menggunakan dua *test case* berdasarkan *independent path* yang diperoleh dari perhitungan *cyclomatic complexity*.

b. Pengujian *Branch coverage*

Setelah melakukan pengujian pada *statement coverage*, pengujian yang dilakukan selanjutnya adalah dengan menggunakan teknik *branch coverage* yang bertujuan untuk memastikan setiap kondisi pada percabangan dieksekusi dengan tepat. Adapun pengujian pada *branch coverage* ditunjukkan pada tabel 5.28.

Tabel 5.28 Tabel Pengujian *Branch coverage* KNF-01

Test Case Id	Jalur	*Actual Output*	Keterangan	Subtotal eksekusi	Nilai *Statement*
KNF01-1	P-01	Mengambil data username dan password, Login diterima	Berhasil	1	1/2 = 50 %
KNF01-2	P-02	Mengambil data username dan password, Login ditolak	Berhasil	2	2/2 = 100%

Berdasarkan pengujian dengan 2 jalur diatas maka nilai *coverage* diperoleh adalah 100%, pada penelitian *branch coverage* dilakukan dengan menguji pada suatu kondisi *True* dan *False*. Pada kasus ini, nilai *branch* yang didapat untuk jalur 1 memperoleh nilai sebesar 50% dan untuk mencapai nilai 100% maka dilakukan ujicoba dengan kondisi yang lainnya.

5.2.3 Hasil Pengujian Sistem *Statement* dan *Branch coverage Testing*

Pengujian menggunakan teknik *statement coverage* dan *branch coverage* dilakukan untuk mengetahui seberapa baik program dapat terhindar dari kesalahan-kesalahan seperti, kesalahan logika dan asumsi pada eksekusi jalur yang tidak benar atau tidak semestinya itu dijalankan dan juga kesalahan yang mungkin ditemukan dalam pengetikan pada *source code* program. Pada *function* tampil_capaiankec() ada bagian dari struktur pada kode program yang tidak tepat eksekusi nya sehingga menghasilkan keluaran nilai variabel yang tidak valid karena jumlah capaian indikator (terduga tb, teridentifikasi tb, dan teridentifikasi hiv) tidak dapat ditampilkan sebelum melakukan perulangan pada jalur program yang mengambil jumlah capaian kategori indikator tersebut, sehingga dilakukan perubahan pada susunan baris kode programnya dan menghasilkan keluaran nilai variabel yang benar sesuai dengan yang diharapkan.

Berdasarkan pengujian yang sudah dilakukan pada Sistem Informasi Reporting ‘Aisyiyah TB-HIV Care, diperoleh bahwa semua *function/method* dalam *class controller* di program telah berhasil dilakukan pengujian dan sistem informasi reporting berjalan dengan baik sesuai dengan yang proses bisnis yang berjalan pada sistem dan sistem siap digunakan oleh pengguna. Pengujian dilakukan dengan menguji setiap jalur pada program dengan mengeksekusi setiap *statement* (baris kode) program dan kondisi logis keputusan atau cabang pada program. Dengan *statement coverage*, setelah membuat rancangan data uji pada setiap *independent path* (jalur) pada program kemudian menentukan *test case* yang mencakup semua *statement* sehingga pengujian bisa dilakukan hanya dengan satu uji data, jika belum mencakup semua *statement*, maka dilakukan pengujian lagi pada *statement* yang belum dieksekusi. Dengan *branch coverage*, menentukan *test case* yang memastikan setiap kondisi cabang dieksekusi, baik pada sisi *true* maupun *false* masing-masing kondisi cabang tersebut setidaknya minimal satu kali dilakukan uji coba. Adapun hasil pengujian dengan teknik *statement coverage* dan *branch coverage* pada 8 *usecase* dan 1 Kebutuhan Non-Fungsionalitas dapat dilihat pada Tabel 5.29.

Tabel 5.29 Hasil pengujian *statement coverage* dan *branch coverage*

Use Case	**Method**	***Statement coverage***			***Branch coverage***		
		Jumlah	***Test Case***	***Coverage***	***Jumlah***	***Test Case***	***Coverage***
UC-01	+tampil_datasuspect()	24	1	100%	6	2	100%
UC-02	+tampil_teridentifikasi_tb()	24	1	100%	6	2	100%
UC-03	+index()	21	1	100%	4	2	100%

UC-04	+edit()	33	2	100%	2	2	100%
UC-05	+edit_user()	32	2	100%	2	2	100%
UC-06	+tampil_honorkoordinator( )	22	1	100%	4	2	100%
UC-07	+tampil_kinerjakader()	52	1	100%	4	2	100%
UC-08	+tampil_capaiankec()	45	4	100%	6	4	100%
UC-09	+getlogin()	15	2	100%	2	2	100%

Berdasarkan Tabel 5.29 diperoleh bahwa semua *method* dalam class pada program telah berhasil dilakukan pengujian dan memperoleh nilai *coverage* sebesar 100% baik pada pengujian dengan teknik *statement coverage* maupun teknik *branch coverage*. Dengan demikian pengujian pada teknik *statement* dan *branch coverage testing* telah memastikan bahwa semua *statement* dan kondisi logis pada *source code* program telah di eksekusi dengan minimal *test case* yang dijalankan dari jalur-jalur yang diperoleh dari perhitungan dengan menggunakan metriks *Cyclomatic Complexity* .

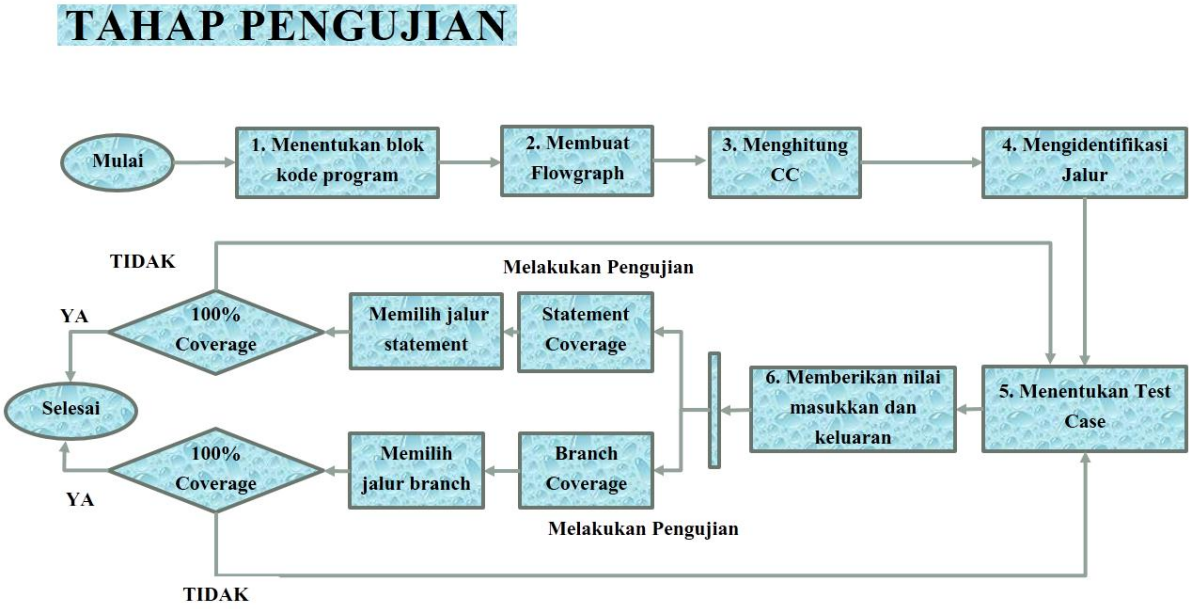
Disamping itu pengujian terhadap user atau pengguna sistem juga sudah dilakukan oleh staff data collection yaitu Bapak Afriadi Tanjung, pengujian dilakukan dengan menjalankan fitur-fitur atau menu-menu yang disediakan pada sistem, baik fitur dalam melakukan entry laporan penjaringan suspect oleh koordinator kader dilapangan, fitur untuk mengecek informasi laporan yang masuk dari 8 kecamatan, fitur memantau informasi target dan capaian indikator yang diperoleh oleh tiap kecamatan, dan fitur-fitur tambahan yang lain. Berdasarkan pengujian tersebut, yaitu bagaimana sistem dapat dikembangkan ke aplikasi android yang lebih memudahkan koordinator dalam melakukan pelaporan penjaringan suspect, sehingga nantinya masing-masing kader juga dapat langsung mengentry laporan untuk data suspect baru ke dalam aplikasi atau sistem.

5.2.4 Evaluasi Teknik Pengujian

Pada dasarnya pengujian *white box testing* dilakukan untuk memeriksa bahwa semua pernyataan yang dieksekusi pada setiap method telah dilaksanakan dan cabang dan jalur yang sesuai melalui method telah dalam cakupan. *Statement coverage* merupakan salah satu teknik dari metode pengujian *white box testing* yang memastikan setiap *executable statement* pada *source code* program yang dibuat. Perancangan *test case* pada *statement coverage testing* berdasarkan dari jalur-jalur yang mungkin terbentuk pada *flowgraph* dan jumlah jalur (*independent path*) yang didapat sebagai penentu minimal *test* yang dijalankan. Pengujian dilakukan berdasarkan kemungkinan *test case* yang memastikan semua pernyataan telah dieksekusi sekurangnya satu kali. *Statement coverage* mencapai nilai 100% *coverage* jika

seluruh titik (*nodes*) pada *flowgraph* atau setiap pernyataan pada *source code* program telah dikunjungi oleh jalur-jalur yang dilalui dari *test case* yang dijalankan.

Branch coverage merupakan salah satu teknik pengujian *white box testing* dimana pengujian dilakukan untuk mengukur hasil keputusan atau cabang yang telah di uji. Keputusan atau cabang adalah suatu pernyataan IF, pernyataan kontrol *loop* (misalnya do-while) dimana ada dua atau lebih hasil dari pernyataan keputusan atau cabang tersebut. *Branch coverage* dilakukan untuk memastikan setiap kondisi logis dari keputusan atau cabang telah tereksekusi, dimana setiap titik percabangan dalam kode diuji disetiap arahnya setidaknya sekali. *Branch coverage* mencapai 100% *coverage* jika setiap keputusan atau cabang pada program ditinjau setidaknya masing-masing satu kali test. Langkah-langkah pengujian *statement* dan *branch coverage* tidak jauh berbeda, adapun proses pengujian pada kedua teknik tersebut dapat dilihat pada Gambar 5.51.



Gambar 5.51 Proses pengujian *white box testing*

Pada Gambar 5.51 dijelaskan bahwa proses pengujian *white box* yang pertama yaitu menentukan *source code* atau blok kode program dari setiap *use case*, yang ke dua yaitu membuat *flowgraph* berdasarkan *source code* atau blok kode program yang dipilih. Proses yang ketiga yaitu menghitung *Cyclomatic Complexity*, hasil dari perhitungan *Cyclomatic Complexity* menunjukkan jumlah *independent path* atau dengan kata lain menunjukkan jumlah pengujian yang harus dijalankan. Proses yang keempat yaitu mengidentifikasi jalur yang mungkin dibentuk berdasarkan jumlah *independent path* berdasarkan pada *flowgraph*. Proses yang

kelima menentukan *test case*, kemudian proses yang keenam yaitu memberikan nilai masukan dan keluaran yang diharapkan. Keenam proses tersebut merupakan langkah persiapan dalam melakukan pengujian dengan *white box testing*, setelah melakukan proses pada keenam langkah tersebut langkah selanjutnya yaitu melakukan pengujian dengan teknik *statement coverage* dan *branch coverage*. Pada *statement coverage* dimana memilih test case yang memiliki *statement* yang paling banyak yang mencakup semua *statement* masuk dalam cakupan. Dan pada *branch coverage* memilih test case yang mencakup setiap keputusan atau cabang dapat memenuhi setiap sisi *flowgraph* pada blok kode program. Dan langkah yang terakhir yaitu menghitung persentase *coverage* dengan mengukur jumlah pernyataan dan keputusan atau cabang dalam kode sumber yang dieksekusi. Berdasarkan hasil persentase ketika sudah mencapai 100% maka pengujian selesai, tetapi jika belum mencapai 100% maka pengujian dilakukan dengan menjalankan test case yang lain.

Perbandingan antara ke dua teknik *statement coverage* dan *branch coverage* yaitu pada *statement coverage testing* berfokus bagaimana menjalankan atau mengeksekusi setiap pernyataan yang dapat mencakup semua pernyataan hanya dengan sekali test. Namun tidak menutup kemungkinan menjalankan *test* pada jalur lain ketika dalam sekali *test* belum mencakup semua pernyataan yang ada pada kode sumber program. Tetapi ketika *statement coverage* sudah mencapai 100% maka tidak menjalankan *test case* pada jalur lain, hal tersebut menandakan bahwasannya semua *statement* (baris kode) sudah tereksekusi atau sudah tercakup. Seperti pada function tampil_datasuspect(), function tampil_index() status laporan, dan beberapa function lain yang memiliki struktur yang sama dimana pada blok kode program tersebut memiliki *statement*-If kosong istilahnya, pada *conditional statement* hanya ditujukkan jika bulan laporan !=0 atau tidak sama kosong maka diambil data bulan laporan yaitu sesuai dengan bulan laporan yang dipilih, tetapi tidak ada *statement* yang menunjukkan jika bulan laporan = 0 pernyataan nya apa. Berdasarkan pada function tersebut jika dilakukan pengujian dengan teknik *statement coverage* maka tidak dipedulikan *statement* if dengan kondisi yang lain terlebih jika data uji yang dijalankan sudah mencakup semua *statement* maka pengujian dihentikan karna setiap baris kode program sudah dieksekusi. Sedangkan pada *branch coverage* sesuai dengan definisinya yaitu merupakan suatu teknik pengujian yang dilakukan untuk memastikan setiap kondisi keputusan atau cabang yang ada pada kode program untuk menghindari *statement* if mengarah pada perilaku tidak normal, sehingga pada function tersebut dijalankan kondisi dimana jika bulan laporan = 0 atau kosong, maka diambil bulan laporan dengan ketentuan dimana, jika 0 atau kosong dalam artian data nya diambil itu semua bulan dan semua tahun yang ada pada sistem. Dan pada *branch coverage* minimal setidaknya

diperlukan 2 uji data untuk menguji pada kondisi true dan false masing-masing satu kali dalam memvalisi semua cabang yang ada di dalam kode program dan memastikan bahwa tidak ada percabangan yang mengarah ke perilaku tidak normal pada sistem. Dengan kata lain bahwa pada *branch coverage* akan melakukan pengujian yang tidak dapat dicakup ketika dilakukan pengujian dengan teknik *statement coverage*, hal tersebut menunjukkan bahwasannya 100% *statement coverage* tidak menyiratkan 100% *branch coverage*, tetapi sebaliknya 100% *branch coverage* sudah pasti menyiratkan 100% *statement coverage* karna semua baris disetiap kode program sudah dieksekusi. Terdapat beberapa perbandingan dari kedua teknik tersebut, dapat dilihat pada Tabel 5.30.

Tabel 5.30 Perbandingan *Statement coverage* dan *Branch coverage*

No	*Statement coverage*	*Branch coverage*
1	Mengeksekusi semua pernyataan yang ada pada *source code* (kode sumber) program setidaknya sekali test. *Statement coverage* akan mencapai 100% ketika semua baris kode telah tereksekusi	Mengeksekusi setiap keputusan atau cabang. Masing-masing pada kondisi setiap cabang di uji coba dengan sekali test. Dan *branch coverage* dapat mencapai 100% ketika setiap kondisi keputusan masing-masing sudah di eksekusi.
2	Jika ada *statement*-If kosong setidaknya hanya diperlukan 1 uji data, jika data uji tersebut sudah mencakup semua baris kode program	Melakukan pengujian yang tidak dicapai dengan *statement coverage* ketika pada blok kode program terdapat *statement*-If, dan setidaknya diperlukan 2 uji data dalam melakukan pengujian untuk setiap masing-masing kondisi keputusan atau cabang dari sisi true maupun false
3	100% *statement coverage* tidak menyiratkan 100% *branch coverage* testing	100% *branch coverage* testing menyiratkan *statement coverage* testing 100%

Pada dasarnya pengujian dengan *statement coverage* testing adalah teknik pengujian yang paling mudah dipahami dan digunakan. Dan juga memberikan informasi dengan cepat baris kode program yang belum di eksekusi atau dijalankan uji coba, sehingga perlu menjalankan test yang lain. Hal tersebut menjelaskan bagaimana suatu *statement coverage* bisa 100% dengan sekali test jika semua pernyataan sudah tercakup. Namun yang menjadi permasalahan pada *statement* yaitu ketika melalui potongan kode cabang tertentu terkadang tidak dapat menjalankan *statement* tersebut sehingga masih ada sisi (*edge*) pada *flowgraph* yang belum terjangkau, maka dari itu dilakukan pengujian dengan *branch coverage* yang memastikan semua kondisi pernyataan tercakup. Dengan kata lain pengujian dengan *statement coverage* yaitu sebagai dasar pengujian dalam memastikan bahwa baris kode program sudah benar. Dan sangat dianjurkan untuk menargetkan *statement coverage* 100% sebelum menggunakan teknik pengujian lain.