PENGUJIAN WHITE BOX PADA SISTEM INFORMASI MONITORING SKRIPSI PROGRAM STUDI INFORMATIKA

Ronny Subagia[1)], Ronggo Alit[2)], Fawwaz Ali Akbar[3)]
E-mail : [1)]ronnysubagia@gmail.com, [2)]ronggoa@gmail.com,
[3)]fawwaz.a.akbar@gmail.com
[1,2,3)]Program Studi Informatika, Fakultas Ilmu Komputer, UPN "Veteran" Jawa Timur

Abstrak

Testing atau pengujian perangkat lunak adalah bagian dari *Software Development Life Cycle (SDLC)* yang digunakan untuk mengidentifikasi kesalahan dari kebutuhan secara fungsional ataupun non – fungsional. Pengujian perangkat lunak secara garis besar dibagi menjadi dua yakni *white box* dan *black box testing*. *White box testing* bisa disebut juga sebagai pengujian kotak kaca atau pengujian struktural dimana pengujian yang dikembangkan berdasarkan pada kode program. Sistem Informasi Monitoring Skripsi Program Studi Informatika UPNVJT merupakan suatu sistem pengawasan terhadap mahasiswa yang mengampu skripsi dimana dengan adanya aplikasi ini diharapkan sistem pengawasan skripsi mahasiswa Program Studi Informatika menjadi semakin efisien maupun efektif dalam pelaksanaannya. Hasil yang didapat dari pengujian white box menghasilkan 69 fungsi masuk kategori tingkat resiko rendah terhadap cacat atau error yang memiliki tipe prosedur yang sederhana dan terstruktur dengan baik serta stabil dengan persentase 94,5 %, 2 fungsi masuk kategori tingkat resiko menengah terhadap cacat atau error dengan tipe prosedur lebih kompleks yang memiliki persentase 2,75 %, dan 2 fungsi masuk kategori tingkat resiko tinggi terhadap cacat atau error yang memiliki tipe prosedur kompleks dan kritis dengan persentase 2,75 %. Setiap source code dari semua fungsi yang diuji memiliki hubungan dengan setiap menu pada aplikasi. Persentase dari pengujian yang dapat dilakukan sejumlah 221 pengujian dari yang seharusnya 285 pengujian berdasarkan jalur independen sebesar 77,5 %. Dari total 73 fungsi diperoleh hasil yang sama untuk 3 jenis perhitungan menggunakan cyclomatic complexity sehingga bisa dikatakan script code program adalah relevan serta dari 221 skenario uji diperoleh hasil yang valid tanpa error.

Kata kunci: *Testing atau Pengujian Perangkat Lunak, White Box, Sistem Informasi, Skripsi*

1. PENDAHULUAN

Testing atau pengujian perangkat lunak adalah bagian dari *Software Development Life Cycle* (SDLC) yang digunakan untuk mengidentifikasi kesalahan dari kebutuhan secara fungsional ataupun non – fungsional. Meskipun semua kesalahan tidak dapat diidentifikasi namun setidaknya dapat meminimalisir terjadinya banyak kesalahan dalam seluruh fungsi yang ada pada sistem [2]. Pengujian perangkat lunak secara garis besar dibagi menjadi dua yakni *white box dan black box testing*. *Black box testing* bisa disebut juga sebagai pengujian fungsional yang dibuat berdasarkan spesifikasi dari klien dan penguji dalam *black box testing* tidak memiliki akses ke dalam kode program. *White box testing* bisa disebut juga sebagai pengujian kotak kaca atau pengujian struktural dimana pengujian yang dikembangkan berdasarkan pada kode program. Penguji dalam *white box testing* memiliki pengetahuan tentang kode dan penulisan kasus uji dengan parameter yang sesuai. Hal ini terutama menyangkut dengan aliran kontrol dan aliran data suatu program [3]. *White Box* sendiri mempunyai beberapa teknik di dalam pengujiannya, seperti : *Data Flow Testing, Control Flow Testing, Basic Path / Path Testing,* dan *Loop Testing* [4].

Sistem Informasi Monitoring Skripsi Program Studi Informatika Universitas Pembangunan Nasional Veteran Jawa Timur merupakan suatu sistem pengawasan terhadap mahasiswa Universitas Pembangunan Nasional Veteran Jawa Timur khususnya Program Studi Informatika Fakultas Ilmu Komputer yang dimana dengan adanya aplikasi ini diharapkan sistem pengawasan skripsi mahasiswa Program Studi Informatika yang

sebelumnya masih manual bisa menjadi *paperless* dan digital serta semakin efisien maupun efektif dalam pelaksanaannya [6].

Dari penjelasan aplikasi Sistem Informasi Monitoring Skripsi Program Studi Informatika Universitas Pembangunan Nasional Veteran Jawa Timur, juga tentang pengujian aplikasi perangkat lunak untuk mengetahui apakah ada kesalahan atau tidak pada aplikasi yang dibuat, dan pengujian *black box* yang pernah dilakukan oleh peneliti sebelumnya serta belum pernah diuji dengan pengujian *white box*. Maka disini penulis memutuskan untuk melakukan penelitian mengenai pengujian *white box* pada Sistem Informasi Monitoring Skripsi Program Studi Informatika Universitas Pembangunan Nasional Jawa Timur menggunakan *Basic Path Testing* dengan metode *Flowgraph notation, Cyclomatic Complexity,* dan *Deriving Test Case* [4].

2. METODOLOGI

Ada beberapa tahapan dalam proses penelitian ini yaitu langkah – langkah penelitian yang terdiri dari : analisis data, proses pengerjaan, dan prosedur cara kerja sistem kemudian skenario pengujian sistem

2.1 Langkah – Langkah Penelitian

Di dalam langkah – langkah penelitian ini terdiri dari proses bagaimana berjalannya pengujian yang akan dipaparkan seperti di bawah ini :

2.1.1 Analisis Data

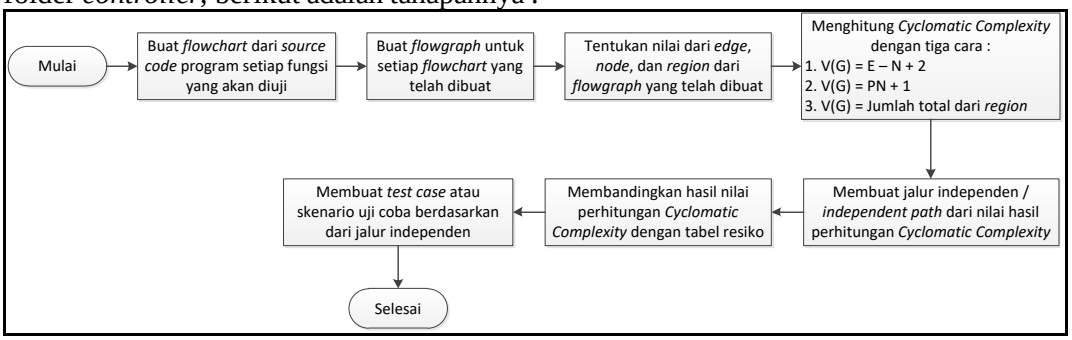
Data yang digunakan dalam penelitian ini adalah *file* berekstensi *.php* yang letaknya terdapat pada folder *htdocs / skripsi / application / controller* dan di dalamnya terdapat nama – nama *file ekstensi .php* yang akan dilakukan untuk pengujian, seperti di bawah ini :

a. Bima.php
b. Pimpinan.php
c. Auth.php
d. Dosen.php
e. Mahasiswa.php
f. Admin.php

Untuk pengujian berdasarkan teknik dan metode yang diambil maka yang diambil di dalam kode program di dalam setiap *file* berekstensi *.php* yang memiliki *class* bernama *function* di dalamnya.

2.1.2 Proses Pengerjaan

Langkah dalam proses pengerjaan ini akan dijelaskan bagaimana melakukan pengujian setelah dilakukan analis data berupa *file* berekstensi *.php* yang terdapat dalam folder *controller*, berikut adalah tahapannya :



Gambar 1 *Flowchart* Proses Pengerjaan

Gambar 3.1 adalah *flowchart* yang dibuat untuk menjelaskan bagaimana alur dari proses pengerjaan dari pengujian *white box* menggunakan *basic path* dengan teknik *flowgraph notation, Cyclomatic Complexity,* dan *Deriving Test Case*.

2.1.3 Prosedur Cara Kerja Sistem

Cara kerja dari sistem yang dilakukan peneliti untuk melakukan pengujian didapatkan dengan panduan dari pembuat program aplikasi tersebut secara langsung.

2.2 Skenario Pengujian Sistem

Dalam skenario pengujian sistem ini akan dilakukan sesuai dengan hasil jalur independen yang didapatkan dari *cyclomatic complexity* dengan format tabel yang terdiri dari : nama skenario, kegiatan, hasil yang diharapkan, hasil, dan keterangan (berisi kondisi valid dan tidak valid), berikut adalah model dari tabelnya :

Tabel 1 Contoh Tabel Pengujian Untuk *Deriving Test Case* [5]

No.	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-

Tabel 1 adalah contoh dari tabel yang akan digunakan untuk membuat skenario pengujian yang dilakukan terhadap setiap fungsi yang ada pada setiap *controller* di dalam aplikasi yang akan di uji. Tabel di atas terdiri dari beberapa kolom yang diantaranya adalah sebagai berikut :

a. Kolom "**No.**" adalah kolom untuk penomoran secara urut terhadap pengujian yang dilakukan dalam skenario pengujian sistem.

b. Kolom "**Nama Skenario**" adalah kolom untuk nama dari skenario pengujian sistem.

c. Kolom "**Kegiatan**" adalah kolom yang digunakan untuk menjelaskan apa yang dilakukan penguji dalam melakukan pengujian dari fungsi yang ada di di dalam *controller* dengan fitur yang ada pada aplikasi yang saling berkaitan.

d. Kolom "**Hasil yang Diharapkan**" adalah kolom yang berisi tentang hasil yang seharusnya diperoleh ketika melakukan pengujian berdasarkan kolom "**Kegiatan**".

e. Kolom "**Hasil**" adalah kolom yang berisi hasil yang didapatkan dari sistem setelah melakukan kolom "**Kegiatan**".

f. Kolom "**Keterangan**" adalah kolom yang hanya diisi dengan kata *Valid* dan *Invalid* yang dimana *valid* adalah sesuai dan *invalid* adalah tidak sesuai. Keterangan *valid* didapatkan jika kolom "**Hasil yang Diharapkan**" dan "**Hasil**" hasilnya sama kemudian keterangan *invalid* didapatkan jika kolom "**Hasil yang Diharapkan**" dan "**Hasil**" hasilnya tidak sama.

3. HASIL DAN PEMBAHASAN

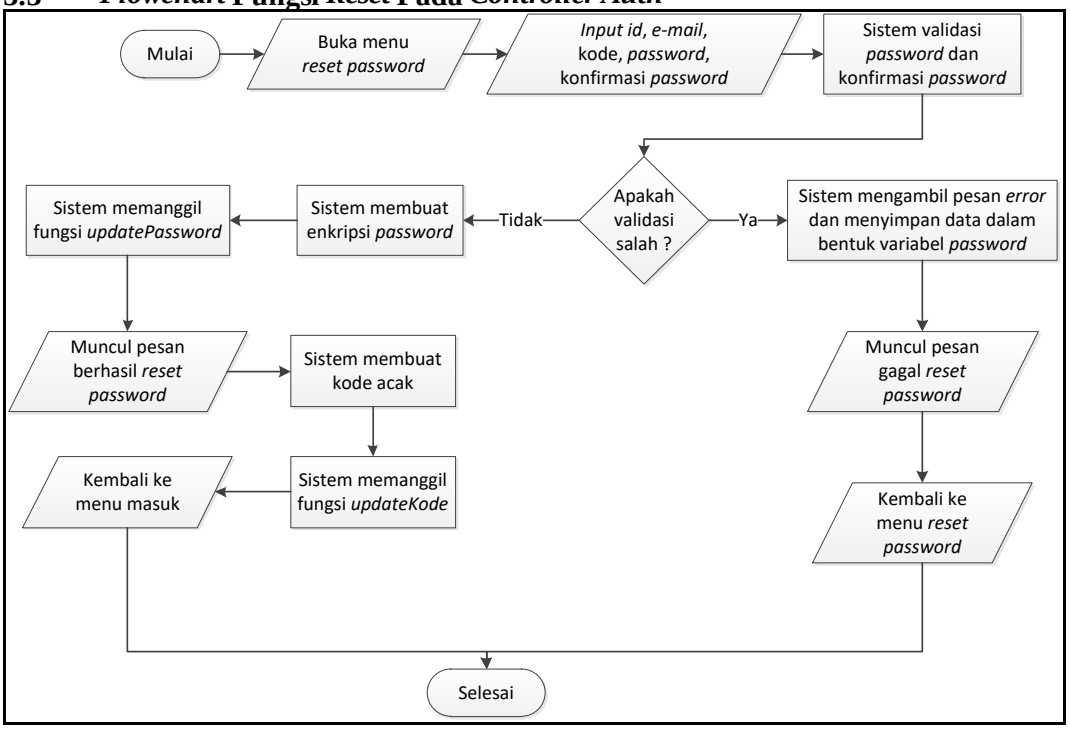
Pada penelitian ini akan dijelaskan mengenai contoh salah satu penerapan pengujian *white box* pada salah satung fungsi dari sekian banyak fungsi yang tidak bisa dijelaskan semua dengan tahapan – tahapan sebagai berikut :

3.1 *Source Code* Program Fungsi *Reset* Pada *Controller Auth*

```
function reset(){
$id       = $this->input->post('id');
$email    = $this->input->post('email');
$kode     = $this->input->post('kode');
$password = $this->input->post('password');
$konfirmasi = $this->input->post('konfirmasi');
$this->form_validation->set_rules('password', 'Password',
'required|trim|xss_clean|min_length[8]|matches[konfirmasi]');
$this->form_validation->set_rules('konfirmasi', 'Konfirmasi password',
'required|trim|xss_clean');
// jika tidak lolos validasi
if ($this->form_validation->run() == false) {
```

```
        $pesan = form_error('password', '<small class="text-danger">','</small>');
        $this->session->set_flashdata('password', $pesan);
        $this->setPesan('password','mereset','err');
                redirect("bima/reset/$email/$kode");
        }
        else{
        $password       = password_hash($password,PASSWORD_DEFAULT);
        $status  = $this->m_user->updatePassword($id,$password,'tbl_user_mahasiswa');
        $this->setPesan('password','merubah',$status);
        $kode   = rand(100000,999999);
        $this->m_user->updateKode($email,$kode);
                redirect('bima/masuk');
        }
}
```

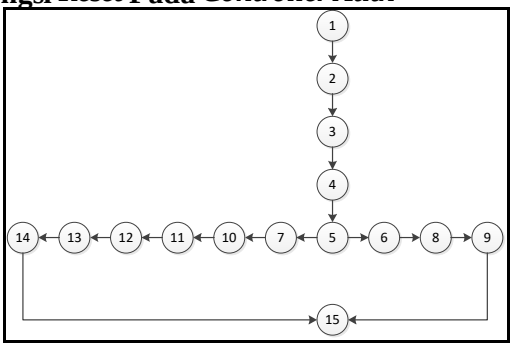
3.3 *Flowchart* Fungsi *Reset* Pada *Controller Auth*



Gambar 2 *Flowchart* Fungsi *Reset*

Gambar 2 adalah *flowchart* yang dibuat berdasarkan dari *source code* program dari fungsi *reset* yang ada pada *controller auth*.

3.4 *Flowgraph* Fungsi *Reset* Pada *Controller Auth*



Gambar 3 *Flowgraph* Fungsi *Reset*

Gambar 3 adalah *flowgraph* yang dibuat berdasarkan dari *flowchart* fungsi *reset* gambar 2. Diketahui *Node* dari *flowgraph* fungsi *reset* berjumlah 15, *edge* berjumlah 15, *region* berjumlah 2, dan *predicate node* berjumlah 1. Setelah diketahui jumlah masing – masing, maka berikut adalah perhitungannya :

a. V (G) = E – N + 2
   V (G) = 15 – 15 + 2
   V (G) = 2
b. V (G) = PN + 1
   V (G) = 1 + 1
   V (G) = 2
c. Diketahui jumlah *Region* = 2

Maka jalur independen pada fungsi *reset* di *controller auth* berjumlah 2, antara lain :

i. 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 15 (*If* (5) = *true*)
ii. 1 – 2 – 3 – 4 – 5 – 7 – 10 – 11 – 12 – 13 – 14 – 15 (*If* (5) = *false*)

Setelah diketahui jumlah jalur independennya, maka akan dilakukan perbandingan menggunakan tabel hubungan antara *cyclomatic compelexity* dan resiko pada *slide* berikutnya :

Tabel 2 Hubungan Tabel 2 Hubungan *Cyclomatic Complexity* Dengan Resiko [1]

Nilai CC	Tipe Prosedur	Tingkat Resiko
1 – 4	Prosedur Sederhana	Rendah
5 – 10	Prosedur yang terstruktur dengan baik dan stabil	Rendah
11 – 20	Prosedur yang lebih kompleks	Menengah
21 – 50	Prosedur yang kompleks dan kritis	Tinggi
>50	Rentan kesalahan, sangat mengganggu, prosedur tidak dapat diuji	Sangat Tinggi

Jadi menurut tabel di atas untuk fungsi *reset* ini memiliki tingkat resiko yang rendah dengan tingkat prosedur yang sederhana karena memiliki jalur independen berjumlah 2. Lalu, setelah diketahui jalur independennya maka langkah selanjutnya adalah *deriving test case* (membuat kasus uji), seperti tabel di bawah ini :

Tabel 3 *Deriving Test Case* Fungsi *Reset* [5]

No.	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
1	UjiBima18	Daftar akun tanpa konfirmasi lalu coba lakukan lupa *password*, buka *email* dan klik tombol *reset* untuk menuju ke halaman *reset password*	Sistem menampilkan halaman *login* / masuk	Sistem menampilkan halaman *login* / masuk	*Valid*
2	UjiBima19	Daftar akun lalu konfirmasi, lakukan lupa *password*, dan	Sistem menampilkan halaman *reset password*	Sistem menampilkan halaman *reset password*	*Valid*

		buka *email* setelahnya klik tombol *reset* untuk menuju ke halaman *reset password*			

Tabel 3 adalah *Deriving Test Case* yang dibuat berdasarkan dari jalur independen yang telah dibuat dan didapatkan hasil *valid* untuk dua pengujian yang dilakukan sehingga tidak ditemukan *error*.

3.5 Analisa Hasil Dan Pembahasan

Pada sub bab 3.5 akan dijelaskan mengenai hasil analisa penulis terhadap hasil dan pengujian terhadap keseluruhan fungsi yang diuji oleh penulis yang dibagi menjadi 3, yaitu sebagai berikut :

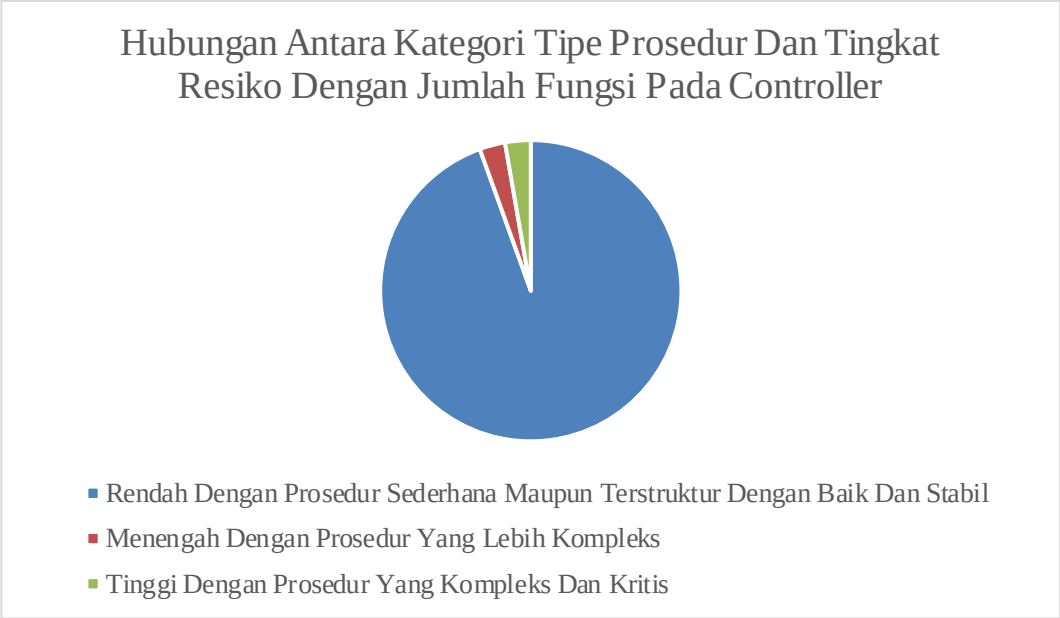
3.5.1 Analisa Dari Hubungan Jumlah Fungsi Terhadap Tipe Prosedur Dan Tingkat Resiko

Pada sub bab ini akan ditampilkan dalam bentuk tabel tentang ringkasan dari seluruh hasil pengujian dari 6 *controller* dengan total fungsi berjumlah 73 :

Tabel 4 Hubungan Kategori Tipe Prosedur Dan Tingkat Resiko Dengan Jumlah Fungsi Pada Controller

Kategori / Nama Controller	Rendah Dengan Prosedur Sederhana Maupun Terstruktur Dengan Baik Dan Stabil	Menengah Dengan Prosedur Yang Lebih Kompleks	Tinggi Dengan Prosedur Yang Kompleks Dan Kritis
Controller Bima	13	-	-
Controller Auth	10	-	-
Controller Pimpinan	7	-	-
Controller Dosen	7	1	-
Controller Mahasiswa	13	-	-
Controller Admin	19	1	2
Jumlah	73		

Dari tabel 4 di atas dari 73 fungsi yang memiliki tipe prosedur dan tingkat resiko dengan kategori rendah dengan prosedur sederhana maupun terstruktur dengan baik dan stabil sebanyak 69 fungsi, 2 fungsi kategori menengah dengan prosedur yang lebih kompleks, dan 2 fungsi kategori tinggi dengan prosedur yang kompleks dan kritis. Maka persentase terhadap fungsi yang ada pada semua *controller* di aplikasi yang diuji tingkat resiko yang rendah terhadap cacat atau *error* dengan persentase sebesar 94,5 %, lalu yang memiliki resiko tingkat menengah terhadap cacat atau *error* sebesar 2,75%, dan memiliki resiko tingkat tinggi terhadap cacat atau *error* sebesar 2,75%. Selanjutnya adalah tampilan grafik dari hubungan antara kategori tipe prosedur dan tingkat resiko dengan jumlah fungsi pada *controller* :



Gambar 4 Grafik Pie Hubungan Antara Kategori Tipe Prosedur Dan Tingkat Resiko Dengan Jumlah Fungsi Pada *Controller*

Dari tabel 4 dan gambar 4 kita bisa mengetahui seberapa besar *source code* program yang ada pada *controller* aplikasi ini memiliki persentase terhadap tingkat resiko terhadap cacat atau *error*, sehingga dapat membantu dalam mengetahui dimana letak dari fungsi – fungsi yang memiliki tingkat resiko baik dari rendah, menengah, dan tinggi sehingga mempermudah dalam mendeteksi kemungkinan besar jika terjadi cacat atau *error* dalam program aplikasi tersebut.

3.5.2 Analisa *Source Code* Pada Setiap Fungsi Dengan Fitur Aplikasi

Untuk *source code* program pada setiap fungsi yang diuji oleh penulis memiliki hubungan dengan setiap fitur yang ada pada aplikasi namun terdapat fungsi yang dimana untuk mengetahui fungsi tersebut berjalan sebagaimana mestinya perlu menjalankan fungsi lain yang men – *trigger* fungsi tersebut agar bisa diuji apakah fungsi itu berjalan benar atau terjadi *error*.

3.5.3 Analisa Tentang Jumlah *Deriving Test Case* Atau Skenario Pengujian Sistem

Dari keseluruhan pengujian yang dilakukan penulis, total dari pengujian yang dilakukan sebanyak 221 uji dari yang seharusnya 285 uji sehingga *missing* 64 uji karena alasan tertentu dengan rincian sebagai berikut :

a. *Controller* Bima

Pada fungsi yang ada pada *controller* Bima pengujian yang dilakukan jumlahnya sesuai dengan jalur independen sesuai perhitungan *Cyclomatic Complexity* karena setiap fungsi yang diuji memiliki kondisi if yang bisa diuji dengan kondisi *inputan* benar dan salah.

b. *Controller Auth*

Pada fungsi yang ada pada *controller Auth* terdapat fungsi yang dimana tidak dapat dilakukan pengujian sesuai dengan jumlah jalur independennya sebanyak 2 pengujian, yaitu :

i. Fungsi yang kurang 1 pengujian.
ii. Fungsi daftar kurang 1 pengujian.

c. *Controller* Pimpinan

Pada fungsi yang ada pada controller Pimpinan terdapat fungsi yang dimana tidak dapat dilakukan pengujian sesuai dengan jumlah jalur independennya sebanyak 8 pengujian, yaitu :

i. Fungsi dashboard kurang 4 pengujian.

ii. Fungsi mahasiswa kurang 2 pengujian.
iii. Fungsi cekAlur kurang 2 pengujian.

d. *Controller* Dosen

Pada fungsi yang ada pada *controller* Dosen terdapat fungsi yang dimana tidak dapat dilakukan pengujian sesuai dengan jumlah jalur independennya sebanyak 9 pengujian, yaitu :

i. Fungsi *dashboard* kurang 2 pengujian.
ii. Fungsi mahasiswa kurang 5 pengujian.
iii. Fungsi *cekAlur* kurang 2 pengujian.

e. *Controller* Mahasiswa

Pada fungsi yang ada pada *controller* Mahasiswa terdapat fungsi yang dimana tidak dapat dilakukan pengujian sesuai dengan jumlah jalur independennya sebanyak 9 pengujian, yaitu :

i. Fungsi topik kurang 4 pengujian.
ii. Fungsi dokumen kurang 1 pengujian.
iii. Fungsi bimbingan kurang 2 pengujian.
iv. Fungsi kuota kurang 2 pengujian.

f. *Controller* Admin

Pada fungsi yang ada pada *controller* Admin terdapat fungsi yang dimana tidak dapat dilakukan pengujian sesuai dengan jumlah jalur independennya sebanyak 36 pengujian, yaitu :

i. Fungsi *dashboard* kurang 3 pengujian.
ii. Fungsi pengajuan kurang 1 pengujian.
iii. Fungsi *SAW* kurang 19 pengujian.
iv. Fungsi akun kurang 3 pengujian.
v. Fungsi berita kurang 6 pengujian.
vi. Fungsi histori kurang 1 pengujian.
vii. Fungsi kontak kurang 1 pengujian.
viii. Fungsi *cekAlur* kurang 2 pengujian

Dari penjabaran di atas bisa dikatakan secara garis besar jika pengujian yang tidak bisa dilakukan karena terdapat kondisi *if* yang digunakan untuk mengecek fungsi berjalan benar atau salah sehingga jika berjalan benar tidak bisa melakukan pengujian untuk kondisi salah karena harus dengan sengaja mematikan fungsi yang dicek untuk memperoleh kondisi salah dan terdapat kondisi *if* yang *inputan* berasal dari sistem guna untuk memproses data yang menghasilkan satu *output*. Dan dari 221 jumlah pengujian yang dilakukan terhadap 73 fungsi seluruhnya *valid* tidak ditemukan cacat atau *error*, maka nilai persentase untuk pengujian yang bisa dilakukan dengan jumlah pengujian yang harusnya dicapai berdasarkan jumlah jalur independen sebanyak 285, yakni 221 / 285 x 100 = 77,5 %.

4. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Setelah melakukan penelitian pengujian *white box* pada Sistem Informasi *Monitoring* Skripsi Program Studi Informatika UPNVJT, didapatkan beberapa kesimpulan yang diantaranya sebagai berikut :

a) Dari pengujian yang dilakukan dapat disimpulkan langkah – langkah dalam pengujian *white box* menggunakan *basic path* dengan teknik *flowgraph notation*, *cyclomatic complexity*, dan *deriving test case* yang dilakukan adalah membuat *flowchart* dari *source code* program yang akan diuji; membuat *flowgraph* dari *flowchart* yang telah dibuat; menentukan nilai dari notasi *flowgraph* yang terdiri dari : *node*, *edge*, *predicate node* dan *region*; melakukan perhitungan *cyclomatic complexity* dengan 3 cara, yakni : a.V(G) = E – N + 2, b.V(G) = PN + 1, c.V(G) = Jumlah *Region*; menentukan jalur independen dari hasil perhitungan *cyclomatic complexity*; membandingkan hasil perhitungan *cyclomatic*

complexity dengan tabel resiko; membuat *deriving test case* dan melakukan pengujian terhadap fungsi yang berhubungan dengan fitur program berdasarkan tabel *deriving test case* yang dibuat.
b) Dari total fungsi berjumlah 73 menurut tabel hubungan *cyclomatic complexity* dengan resiko didapatkan 69 fungsi memiliki tingkat resiko yang rendah, 2 fungsi dengan tingkat resiko menengah dan 2 fungsi dengan tingkat resiko tinggi. Maka diperoleh persentase sebesar 94,5 % dengan tingkat resiko rendah, 2,75 % tingkat resiko menengah, dan 2,75 % tingkat resiko tinggi terhadap cacat atau *error* pada seluruh fungsi yang akan diuji.
c) Dengan diketahuinya nilai dari tingkat resiko dan tipe prosedur maka dapat diketahui darimana kemungkinan besar terjadinya cacat atau *error* terhadap *source code* fungsi program dari aplikasi yang diuji serta setiap fungsi yang diuji mewakili setiap menu pada aplikasi.
d) Diketahui jumlah uji coba yang berjumlah 221 dari yang seharusnya 285 menurut teori yang dimana setiap jalur independen harus bisa dieksekusi namun untuk kondisi pengujian yang dilakukan peneliti terdapat fungsi yang dimana kondisi *if* – nya untuk memproses data *inputan* dari sistem dan kondisi *if* yang digunakan untuk mengecek apakah fungsi berjalan dengan baik atau tidak. Lalu diperoleh persentase dari pengujian yang dapat dilakukan terhadap pengujian yang harusnya dicapai sebesar 77,5 % dan tidak ditemukan *error* dalam pengujian yang dilakukan.

4.2 Saran

Berdasarkan hasil yang diperoleh pada penelitian ini, penulis memberikan saran sebagai berikut :
a) Diperlukan untuk melakukan pengujian *white box* menggunakan teknik selain dari *basic path* untuk memperoleh hasil yang lebih baik.
b) Pengujian *white box* menggunakan teknik *basic path* dengan metode *flowgraph notation* dan *cyclomatic complexity* cocok untuk menemukan tingkat resiko dan tipe prosedur dari *source code* program yang akan diuji namun untuk metode *deriving test case* tidak dianjurkan dilakukan jika di dalam *script code* program terdapat kondisi *if* untuk proses data yang *inputan* dari sistem dan kondisi *if* untuk mengecek fungsi berjalan dengan benar atau salah.

5. DAFTAR RUJUKAN

[1] Achmad, Ganif, Puguh, 2018. Jurnal Pendidikan : Teori, Penelitian, dan Pengembangan. Nilai Cyclomatic Complexity Konflik Kerja terhadap Pengaruh Pimpinan dan Beban Kerja Karyawan dengan Menggunakan Model Reflektif PLS SEM, pp.655-648.
[2] Danang, Defri, Yani, 2018. SIMETRIS. TEKNIK PENGUJIAN PERANGKAT LUNAK DALAM EVALUASI SISTEM LAYANAN MANDIRI PEMANTAUAN HAJI PADA KEMENTERIAN AGAMA PROVINSI JAWA TENGAH, pp.746-731.
[3] Rahul, Nitish, Dr Manjula, 2016. IJRASET. Survey of Software Testing Techniques, pp.929-924.
[4] Vijay, Mr. Sarvesh, 2019. GJRA-GLOBAL JOURNAL FOR RESEARCH ANALYSIS. WHITE-BOX TESTING TECHNIQUE FOR FINDING DEFECTS, pp.85-83.
[5] Nuris, M., 2015. WHITE BOX TESTING PADA SISTEM PENILAIAN PEMBELAJARAN. S.Kom. Malang: UIN Maulana Malik Ibrahim Malang.
[6] Shobirin, Muhammad Agung, 2020. RANCANG BANGUN SISTEM INFORMASI MONITORING TUGAS AKHIR ATAU SKRIPSI PROGRAM STUDI INFORMATIKA UNIVERSITAS PEMBANGUNAN NASIONAL VETERAN JAWA TIMUR. S.Kom. Surabaya: UPN "Veteran" Jawa Timur.