JURNAL ILMU KOMPUTER KHARISMA .TECH

Jurnal KHARISMA Tech | ISSN:1907-2317 | e-ISSN: 2810-0344
https://jurnal.kharisma.ac.id/kharismatech/
published by: Pusat Penelitian STMIK KHARISMA Makassar

Volume: 17 no. 02 – September 2022- hlm. 45-57

PENGUJIAN *WHITE BOX TESTING* TERHADAP *WEBSITE* ROOM MENGGUNAKAN TEKNIK *BASIS PATH*

Oleh:

Judith Bryan L Sie[1], Izmy Alwiah Musdar[2*], Syamsul Bahri [3]
[1,2,3]Teknik Informatika, STMIK Kharisma Makassar
e-mail: [1]judithbryan_19@kharisma.ac.id [2] izmyalwiah@kharisma.ac.id
[3] syamsulbahri@kharisma.ac.id

Abstrak:*.Penelitian ini bertujuan untuk melakukan pengujian perangkat lunak terhadap website Room, untuk menemukan adanya cacat atau error. Metode pengujian perangkat lunak yang digunakan yaitu white box testing dengan teknik basis path sebagai metode perancangan test case. Teknik basis path terdiri dari pembuatan flow graph, perhitungan cyclomatic complexity, graph matrix, penentuan independent path, dan pembuatan test case. Lalu ditambah juga dengan perhitungan tingkat resiko berdasarkan jumlah cyclomatic complexity. Hasil pengujian menunjukkan pengujian white box dengan teknik basis path dapat digunakan untuk menemukan cacat atau error pada perangkat lunak. Terdapat 68 skenario pengujian dengan tingkat resiko rendah terhadap cacat sebesar 94% dan tingkat resiko menengah sebesar 6%. Lalu dari 352 jalur pengujian terdapat 192 jalur pass, 5 jalur fail, 114 jalur menangkap error, dan 41 jalur dengan kondisi yang salah.*

Kata kunci: *White box testing, Pengujian perangkat lunak, Website, Basis path.*

Abstract: *This study aims to conduct software testing on the Room website, to find any defects or errors. The software testing method used is white box testing with the basis path technique as a test case design method. The basis path technique consists of making flow graphs, calculating cyclomatic complexity, graph matrix, determining independent paths, and making test cases. Then added with the calculation of the level of risk based on the amount of cyclomatic complexity. The test results show that white box testing with the basis path technique can be used to find defects or errors in the software. There are 68 test scenarios with a low risk level of 94% for defects and a medium risk level of 6%. Then from the 352 test paths there are 192 pass lines, 5 fail paths, 114 error catch paths, and 41 paths with wrong conditions.*

Keywords: *White box testing, Software testing, Website, Basis path.*

1. PENDAHULUAN

Perangkat lunak pada masa ini sudah sangat mudah untuk dimiliki semua orang, mulai dari perangkat lunak yang berbayar sampai perangkat lunak yang gratis. Perangkat lunak hadir dengan memberikan berbagai macam jenis layanan yang berbeda, layanan yang memberikan hiburan, layanan yang memudahkan aktivitas kita, atau layanan yang membantu pekerjaan kita. Sering kali kita menjadi tergantung pada perangkat lunak. Namun tidak semua perangkat lunak yang dikembangkan memiliki kualitas yang baik, dikarenakan tidak melalui proses pengujian perangkat lunak, sehingga terdapat banyak *bug* dan *error* pada suatu perangkat lunak. Seperti yang terjadi pada *Therac-25*, yang merupakan alat pengolah radiasi yang berfungsi untuk pengobatan pasien kanker, kemudian ternyata pada alat ini terdapat *bug* yang

* Corresponding author : Izmy Alwia Musdar (izmyalwiah@kharisma.ac.id)

Diterima : *Juli, 2022* ***Disetujui :*** *Juli, 2022* ***Dipublikasikan :*** *September, 2022*

serius, dimana alat ini gagal berfungsi dengan seharusnya dan menyebabkan dosis radiasi 10 kali lebih tinggi dari seharusnya, yang menyebabkan pasien keracunan radiasi dan bahkan kehilangan nyawa [1]

Room (https://room-idn.com/) merupakan sebuah layanan berbasis *website* aplikasi yang memiliki tujuan untuk membantu dan memudahkan pengguna dalam melakukan pembuatan dan pengolahan untuk buku tamu dan juga event. Untuk mengetahui apakah *website* Room sudah dibangun dengan baik, benar dan juga terbebas dari *error* yaitu dengan melakukan *testing* atau pengujian perangkat lunak. Pengujian perangkat lunak merupakan metode untuk melakukan pengujian apakah perangkat lunak sudah memenuhi kebutuhan yang diharapkan dan juga untuk memeriksa apakah produk perangkat lunak bebas cacat atau *error*. Pengujian merepresentasikan ketidaknormalan yang terjadi pada pengembangan perangkat lunak [2]. Pengujian perangkat lunak merupakan serangkaian proses yang dirancang untuk memastikan kode program sudah melakukan sesuai dengan apa yang telah dirancang [3]. Dalam melakukan pengujian perangkat lunak ada 2 metode yang biasanya digunakan yaitu *white box* dan *black box testing*.

Black box testing merupakan metode yang menguji fungsionalitas suatu perangkat lunak tanpa pengetahuan tentang rincian implementasi dan kode program perangkat lunak tersebut [4]. Sedangkan *white box testing* merupakan metode yang menguji struktur internal perangkat lunak, rancangan dan kode program perangkat lunak terkait [4]. *White Box* dapat mengungkapkan kesalahan dalam implementasi dari sebuah perangkat lunak [5]. Penguji yang menggunakan metode *white box* dalam pengujian perangkat lunak harus memiliki pengetahuan atau pemahaman penuh mengenai sumber kode perangkat lunak. Dalam penelitian ini metode yang akan digunakan yaitu *white box testing*.

White box testing memiliki beberapa teknik dalam melakukan pengujian perangkat lunak diantaranya yaitu, *loop testing* [6] yang berfokus kepada pengujian validasi struktur sebuah perulangan [7], *data flow testing* [8] yang melihat bagaimana data bergerak dalam suatu program [7], *control flow testing* [9] yang menggunakan aliran kontrol program sebagai model dalam acuan untuk membuat *test case* [7], *branch testing* yang berfokus pada pengujian percabangan dalam program, dan *basis path testing* [10]–[14] yang merupakan teknik yang akan melakukan pengujian pada semua pernyataan atau *statement* setidaknya sekali [7].

Teknik *basis path testing* lebih cocok digunakan dibandingkan dengan teknik lainnya, karena *basis path testing* akan menghasilkan jumlah *test case* dengan cakupan *test* yang lebih menyeluruh dibandingkan teknik lainnya [15]. Teknik ini memungkinan perancang *test case* untuk menghasilkan pengukuran kompleksitas logika dari perancangan prosedural dan menggunakan pengukuran ini sebagai perkiraan untuk menguraikan jalur dasar eksekusi [4]. Teknik *basis path* terdiri dari *flow graph notation* yang merupakan notasi sederhana yang menggambarkan alur kontrol program [7], *cyclomatic complexity* yang merupakan perhitungan untuk menentukan jumlah dari jalur pengujian [7], *independent path* yang merupakan penentuan jalur pengujian yang dilewati setidaknya sekali [7], *graph matrix* yang merupakan matriks 2-dimensional yang bertujuan untuk membantu menghitung *cyclomatic complexity* [7],

test case yang merupakan penentuan alur pengujian berdasarkan jalur *independent path* yang sudah ditentukan [7]. Pada setiap *test case* kita perlu mendefinisikan *output* yang kita harapkan [16]. Berdasarkan hal tersebut maka dalam penelitian ini teknik *white box testing* yang akan digunakan adalah *basis path testing*. Perbedaan dengan beberapa penelitian yang menggunakan Teknik basis path testing adalah dilakukan juga perhitungan tingkat resiko terhadap cacat berdasarkan jumlah *cyclomatic complexity.*

Oleh karena itu, pada penelitian dilakukan pengujian perangkat lunak *white box testing* terhadap *website* Room menggunakan teknik *basis path*, untuk mengetahui apakah *website* Room sudah dibagun dengan baik dan untuk menemukan *error*, kesalahan, dan ketidaksesuaian pada struktur internal perangkat lunak dan kode program perangkat lunak. Sehingga diharapkan *website* Room dapat memberikan kepuasan kepada pengguna dengan layanan produk yang berkualitas dan terbebas dari *error* ataupun *bug*.

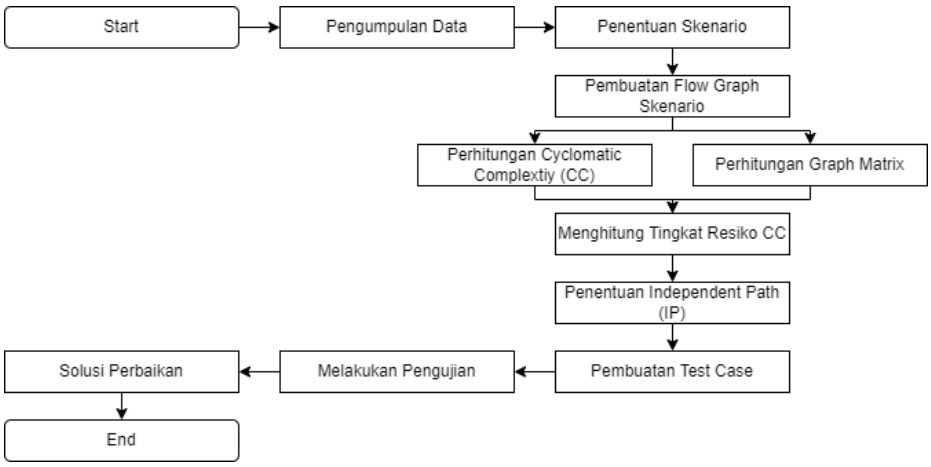
2. METODE PENELITIAN

2.1 Jenis Data & Sumber Data

Jenis data yang digunakan pada peneleitian ini yaitu data kualitatif, karena data yang akan digunakan dalam pada penelitian ini yaitu kode program dari *website* Room. Data kualitatif merupakan data yang disajikan dalam bentuk berupa kata – kata dan bukan berupa angka. Pada umumnya jenis data ini diperoleh melalui observasi, wawancara dan juga analisis dokumen. Sehingga kode program *website* Room termasuk dalam kategori jenis dokumen.

Kemudian sumber data pada penelitian ini yaitu sumber data primer. Data Primer merupakan data yang dikumpulkan secara langsung oleh peneliti-nya sendiri atau dari sumber pertamanya, perorangan maupun kelompok dan tidak melalui perantara. Sumber data primer dalam penelitian ini adalah kode program dari *website* Room.

2.2 Tahapan Penelitian



Gambar 1. Alur Tahapan Penelitian

Dalam pelaksanaan penelitian ini alur tahapan penelitiannya dapat dilihat pada Gambar 1, dimulai dengan tahapan pengumpulan data yang merupakan kode program *file* app.js dari *website* Room, kemudian berdasarkan kode program tersebut akan dilakukan penentuan seleksi yang bertujuan untuk menentukan kode program mana yang akan dilakukan pengujian dengan 2 ketentuan syarat yaitu berdasarkan fungsi kode program dan kompleksitas kode program. Kemudian berdasarkan skenario yang telah ditentukan akan dibuatkan *flow graph* untuk tiap skenario. Lalu dilanjutkan dengan perhitungan *cyclomatic complexity* (CC) dan juga *graph matrix*. Setelah mendapatkan hasil perhitungan CC maka akan berikutnya akan dilanjutkan dengan penentuan jalur *independent path* dan juga menghitung hubungan tingkat resiko berdasarkan CC, yang akan diikuti dengan pembuat *test case* berdasarkan jalur tersebut, hingga pada tahap akhir yaitu melakukan pengujian berdasarkan *test case* yang telah dibuat. Dan jika terdapat kesalahan pada jalur yang diuji maka akan lakukan perbaikan pada kode program.

Pada penelitian ini juga akan dilakukan perhitungan tingkat resiko dari suatu skenario berdasarkan dari jumlah *cyclomatic complexity*-nya, perhitungan ini bukan merupakan bagian dari teknik *basis path*, melainkan berdasarkan dari salah satu penelitian terkait yang dirujuk [14]. Ini akan menunjukan hubungan dari jumlah CC terhadap tingkat resiko akan cacat atau error [14] yang dapat dilihat pada Tabel 1.

Tabel 1: Hubungan *Cyclomatic Complexity* Dengan Resiko [14]

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

3. HASIL DAN PEMBAHASAN

3.1 Deskripsi Data

Data yang digunakan dalam penelitian ini yaitu kode program *website* Room, *file* kode program yang dimaksud yaitu *file* app.js. *File* app.js ini merupakan bagian utama dari kode program *website* Room, yang berfungsi untuk mengatur semua proses yang terjadi pada *website* Room. Termasuk juga meng-*handle route request*, menampilkan halaman yang sesuai, mengolah dan menyimpan data ke dalam *database*.

Data tersebut kemudian yang akan dijadikan sebagai acuan dalam melakukan pengujian *white box testing* dengan teknik *basis path*. *File* kode program app.js totalnya terdiri dari 3240 baris kode program. Tapi tidak semua kode program akan dilakukan pengujian. Kode program akan dibedakan berdasarkan *route*-nya, lalu kemudian kode program juga akan dilihat berdasarkan fungsi dan kompleksitasnya. Jika fungsi dari *route* tersebut merupakan fungsi utama atau penting pada *website* Room, maka *route* tersebut akan dilakukan pengujian, sedangkan jika berdasarkan kompleksitasnya yaitu, kode

program yang minimal memiliki dua percabangan. Namun jika sebuah *route* hanya memenuhi salah satu dari dua ketentuan yang ada maka *route* tersebut tetap akan dilakukan pengujian. Sehingga berdasarkan ketentuan tersebut *route* yang akan diambil adalah seperti pada Tabel 2.

Tabel 2: Data *Route* Pengujian

SC	*Route*	Keterangan
1	app.get("/")	*Landing Page*
2	app.post("/masuk")	Memproses Masuk Akun
3	app.post("/daftar",)	Memproses Daftar Akun
4	app.get("/verifikasi/:token")	Memproses Verifikasi Akun
5	app.post("/lupa-password")	Memproses Lupa *Password*
6	app.post("/lupa-password-1/:userId/:token")	Memproses perubahan lupa password
7	app.get("/home")	Halaman *Home*
...	...	...
...	...	...
68	app.get("/verifikasi/:langgananId/:userId")	Memproses konfirmasi pembayaran langganan

Berdasarkan data pada Tabel 2, yang lengkapnnya dapat dillihat pada lampiran, maka akan menghasilkan total 68 skenario pengujian yang mana tiap skenario akan memiliki *test case*-nya masing – masing. Dalam pembuatan *test case* keterangan hasil yang pada umumnya digunakan adalah 2 yaitu *pass* (sukses) dan *fail* (gagal). Tetapi pada penelitian ini keterangan hasil *test case* akan dibagi menjadi 4 dengan keterangan sebagai berikut:

1) *Pass*, ketika melakukan pengujian pada suatu *test case* berdasarkan *independent path* yang sudah ditentukan lalu hasil dari *expected result* menunjukan hal yang sama dengan *actual result.*
2) *Fail*, ketika melakukan pengujian pada suatu *test case* berdasarkan *independent path* yang sudah ditentukan lalu hasil dari *expected result* menunjukan hal yang berbeda dengan *actual result.*
3) Tangkap *Error*, kondisi ini merupakan kondisi yang dibuat dikarenakan pada kode program terdapat suatu fungsi yang bertujuan untuk menangkap *error* atau kesalahan yang secara tiba – tiba terjadi pada suatu fungsi, seperti pada fungsi mencari data dalam *database* atau menyimpan data dalam *database*, tetapi kondisi tersebut tidak dapat secara sengaja direkayasa, sehingga kondisi tersebut tidak dapat dilakukan pengujian.
4) Salah Kondisi, kondisi ini merupakan kondisi yang dibuat dikarenakan pada kode program terdapat kelebihan atau kesalahan implementasi kode. Kesalahan yang dimaksud ini adalah kebanyakan terjadi pada kasus *IF ELSE*, dimana kode program yang seharusnya hanya memiliki 2 kondisi tetapi karena kesalahan implementasi

kode program sehingga terdapat 3 kondisi, dimana 1 kondisi lainnya merupakan kondisi yang tidak mungkin terjadi atau secara tidak sengaja terbuat

3.2 Pembahasan

Penelitian kemudian akan dilakukan sesuai dengan alur tahapan penelitian pada Gambar 1. Untuk melakukan pengujian akan dijalankan pada *local environment* dengan aplikasi *code editor visual studio code*, terminal *hyper*, dan *studio 3T*. Sehingga tidak mengganggu aktivitas dari *website* Room.

1) *Route Form Edit* Buku amu (SC17)

Route ini memiliki fungsi untuk menampilkan halaman untuk mengedit *form* buku tamu yang sudah ada. Berikut kode lengkapnya pada Tabel 3.

Tabel 3: Kode Program *Route Form Edit* Buku tamu

Kode Program	Node
app.get("/form-edit-Bukutamu", function(req,res){ if (req.isAuthenticated()) {	1
if (_.isEmpty(infoRoom)) {	2
res.redirect("/home");	9
} else { if (!edit) {	3
MyRoom.find({_id: infoRoom.RoomId}, function (err, foundRoom) { if (err) {	4
res.render("error1");	11
} else { if (foundRoom) {	5

foundRoom[0].form.forEach(function (foundForm) {
listForm.push(foundForm);
});
edit=true;
res.render("form-edit-Bukutamu", {ListForm: listForm, Rooms: foundRoom});
}
}
});
	6

} else {
MyRoom.find({_id: infoRoom.RoomId}, function (err, foundRoom) {
if (err) {
	10

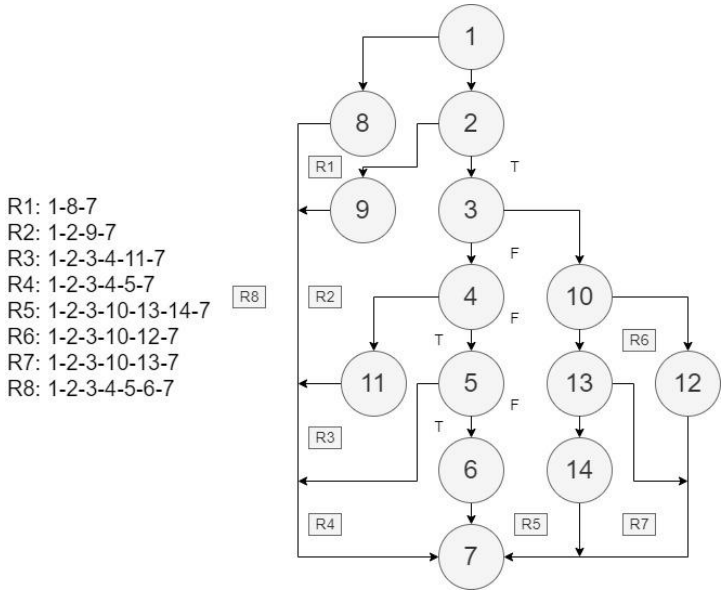
res.render("error1")	12
} else if (foundRoom) {	13

res.render("form-edit-Bukutamu", {ListForm: listForm, Rooms: foundRoom});
}
});
}
}
	14

} else {
res.redirect("/masuk");
}
	8

});	7

Kemudian untuk tahap selanjutnya akan dibuat *flow graph* berdasarkan kode program *route form edit* buku tamu seperti pada Gambar 2.



Gambar 2. *Flow Graph Route Form Edit* Buku tamu

Berdasarkan *flow graph* pada Gambar 2 diketahui bahwa jumlah *edge* (E) = 20 yang merupakan garis yang menghubungkan *node*, jumlah *node* (N) = 14 yang merupkan linggkaran yang menggambarkan suatu aktivitas, jumlah *predicate* (P) = 7 yang merupakan *node* bercabang, dan jumlah *region* (R) = 8 yang menandakan suatu area dalam *flow graph,* yang dapat dilihat dengan simbol R1 hingga R8 pada Gambar 2, sehingga jika dimasukkan ke dalam rumus perhitungan *cyclomatic complexity* dan juga bantuan *graph matrix* pada Tabel 4 dan Tabel 5. maka akan menghasilkan sebagai berikut.

Tabel 4: *Cyclomatic Complexity Route Form Edit* Buku tamu

$V(G) = E - N + 2$	$V(G) = P + 1$	$V(G) = R$
$V(G) = 14 - 20 + 2$	$V(G) = 7 + 1$	$V(G) = 8$
$V(G) = 8$	$V(G) = 8$	

Tabel 5: *Graph Matrix Route Form Edit* Buku tamu

X17	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Hasil
1		1						1							1
2			1						1						1
3				1						1					1
4					1						1				1
5						1	1								1
6							1								0
7															
8							1								0
9							1								0
10												1	1		1
11							1								0
12							1								0
13							1							1	1
14							1								0
Jumlah															8

Berdasarkan dari hasil perhitungan *cyclomatic complexity* pada Tabel 4 dan *graph matrix* pada Tabel 5, didapatkan jumlah hasil *independent path* untuk *route home* yaitu 8 dengan jalur *independent path* sebagai berikut:

- Jalur 1: 1-2-3-4-5-6-7
- Jalur 2: 1-2-3-4-11-7
- Jalur 3: 1-2-3-4-5-7
- Jalur 4: 1-2-9-7
- Jalur 5: 1-2-3-10-12-7
- Jalur 6: 1-2-3-10-13-14-7
- Jalur 7: 1-2-3-10-13-7
- Jalur 8: 1-8-7

Berdasarkan Tabel 1, diketahui bahwa *route form edit* buku tamu dengan jumlah CC 8 memiliki tingkat resiko yang rendah dan prosedur yang terstruktur dengan baik dan stabil. Setelah menentukan jalur *independent path* maka Langkah selanjutnya yaitu membuat dan *test case* dan melaksanakan pengujian. Hasil pengujian dapat dilihat pada Tabel 6.

Tabel 6: *Test Case Route Form Edit* Buku tamu

No	Rute IP	Keterangan	*Expected Result*	*Actual Result*	*Pass/Fail*
SC17-01	1-2-3-4-5-6-7	1 = *True* 2 = *False* 3 = *True* 4 = *False* 5 = *True*	Mengubah status edit menjadi "*True*" dan menampilkan halaman *form edit* buku tamu	Berhasil Mengubah status edit menjadi "*True*" dan menampilkan halaman *form edit* buku tamu	*Pass*
SC17-02	1-2-3-4-11-7	1 = *True* 2 = *False* 3 = *True* 4 = *True*	Menampilkan halaman *error*, karena terdapat kesalahan pada saat ingin mencari data Room pada *database*	Kondisi untuk menangkap *error* yang secara tiba - tiba terjadi, tidak bisa dilakukan *test*	-
SC17-03	1-2-3-4-5-7	1 = *True* 2 = *False* 3 = *True* 4 = *False* 5 = *False*	Kondisi *else* tidak memungkinkan untuk terjadi	Kondisi *else* tidak memungkinkan untuk terjadi	-
SC17-04	1-2-9-7	1 = *True* 2 = *True*	Menampilkan halaman *home*, karena infoRoom ternyata masih kosong	Berhasil Menampilkan halaman *home*, karena infoRoom ternyata masih kosong	*Pass*
SC17-05	1-2-3-10-12-7	1 = *True* 2 = *False* 3 = *False* 10 = *True*	Menampilkan halaman *error*, karena terdapat kesalahan pada saat ingin mencari data Room pada *database*	Kondisi untuk menangkap *error* yang secara tiba - tiba terjadi, tidak bisa dilakukan *test*	-
SC17-06	1-2-3-10-13-14-7	1 = *True* 2 = *False* 3 = *False* 10 = *False* 13 = *True*	Menampilkan halaman *form edit* buku tamu	Berhasil Menampilkan halaman *form edit* buku tamu	*Pass*

No	Rute IP	Keterangan	*Expected Result*	*Actual Result*	*Pass/Fail*
SC17-07	1-2-3-10-13-7	1 = *True* 2 = *False* 3 = *False* 10 = *False* 13 = *False*	Kondisi *else* tidak memungkinkan untuk terjadi	Kondisi *else* tidak memungkinkan untuk terjadi	-
SC17-08	1-8-7	1 = *False*	Menampilkan halaman masuk, karena session habis atau mengakses halaman tanpa login	Berhasil menampilkan halaman masuk, karena session habis atau mengakses halaman tanpa login	*Pass*

Berdasarkan hasil pengujian yang telah dilakukan didapatkan hasil total dari 8 jalur yang ada terdapat 4 jalur yang *pass*, 2 jalur yang merupakan kondisi untuk menangkap *error*, 2 jalur dengan kondisi yang salah, dan 0 jalur yang *fail*. Perbaikan yang dilakukan untuk 2 jalur yang berfungsi untuk menangkap *error* dapat dilakukan dengan memperbaiki kode program seperti pada Gambar 3.

```
app.get("/form-edit-Bukutamu", function(req,res){
  if (req.isAuthenticated()) {
    if (_.isEmpty(infoRoom)) {
      res.redirect("/home");
    } else {
      if (ledit) {
        MyRoom.find({_id: infoRoom.RoomId}, function (err, foundRoom) {
          if (err) {
            res.render("error1");
          } else {
            foundRoom[0].form.forEach(function (foundForm) {
              listForm.push(foundForm);
            });
            edit=true;
            res.render("form-edit-Bukutamu", {ListForm: listForm, Rooms: foundRoom});
          }
        });
      } else {
        MyRoom.find({_id: infoRoom.RoomId}, function (err, foundRoom) {
          if (err) {
            res.render("error1");
          } else {
            res.render("form-edit-Bukutamu", {ListForm: listForm, Rooms: foundRoom});
          }
        });
      }
    }
  } else {
    res.redirect("/masuk");
  }
});
```

Gambar 3. Perbaikan Kode Program *Route Form Edit* Buku tamu

Setelah melakukan hal yang sama juga pada semua skenario yang ada maka dapat dilihat pada Tabel 7 merupakan ringkasan dari hasil pengujian yang telah dilakukan pada seluruh skenario yang telah ditentukan. yang lengkapnnya dapat dillihat pada lampiran.

Dengan keterangan *node* (N), *edge* (E), *region*s (R), *predicate* (P), *cyclomatic complexity* (CC), tangkap *error* (TE) dan salah kondisi (SK).

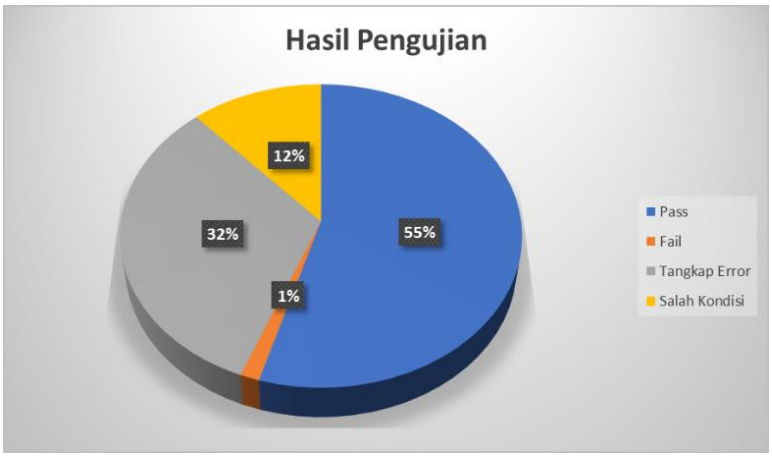
Tabel 7: Ringkasan Hasil Pengujian Skenario

SC	Jumlah								
	N	E	R	P	CC	Pass	Fail	TE	SK
1	14	20	8	7	8	**3**	**0**	**4**	**1**
2	10	13	5	4	5	**3**	**0**	**2**	**0**
3	7	8	3	2	3	**1**	**1**	**1**	**0**
4	7	8	3	2	3	**1**	**0**	**2**	**0**
5	6	7	3	2	3	**1**	**1**	**1**	**0**
6	8	10	4	3	4	**1**	**0**	**3**	**0**
7	19	27	10	9	10	**4**	**0**	**6**	**0**
…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
68	10	13	5	4	5	**1**	**0**	**4**	**0**

Berdasarkan keseluruhan hasil dari pengujian yang telah dilakukan pada Tabel 7, terdapat total 352 jalur yang dihasilkan dari 68 skenario yang telah ditentukan. Sehingga menghasilkan hasil seperti pada Tabel 8.

Tabel 8 Keterangan Hasil Pengujian

Keterangan	Jumlah
Pass	192
Fail	5
Tangkap *Error*	114
Salah Kondisi	41



Gambar 4. Hasil Pengujian

Dari Tabel 8 dan Gambar 4 dapat dilihat hasil dari pengujian yang telah dilakukan. Terdapat total 352 jalur pengujian, persentase jalur yang *pass* sebesar 55% atau 192 jalur, 1% untuk jalur yang *fail* atau sebanyak 5 jalur, 32% untuk jalur yang berfungsi untuk

menangkap *error* atau 114 jalur, dan 12% untuk jalur yang merupakan kesalahan kondisi atau 41 jalur. Jalur tangkap error yang sebelumnya tidak dapat dilakukan pengujian, terbukti ada 4 jalur yang dapat berjalan sesuai fungsinya. Jalur salah kondisi dapat dihilangkan dengan memperbaiki penulisan kode program sehingga tidak ada kesalahan implementasi pemrograman, misalnya menghapus percabangan yang berlebihan.

Pada jalur *fail* ada 4 jalur yang *fail* disebabkan oleh perubahan pengaturan privasi dari Google, 4 jalur ini berfungsi untuk mengirimkan email verifikasi secara otomatis kepada para *user* dan juga admin. Perbaikan untuk 4 jalur ini dilakukan sedikit perubahan pada kode program dengan menggunakan *tools* yang disediakan pada *cpanel* yaitu *simple mail transfer protocol* (SMTP). Kemudian 1 jalur yang *fail* merupakan kode program yang terjadi dikarenakan kode program yang tidak lengkap, untuk memperbaiki jalur ini perlu untuk melengkapi jalur kode program untuk menangkap kondisi tersebut.

Tabel 9: Keterangan Hubungan CC Terhadap Tingkat Resiko

File	Resiko rendah dengan prosedur sederhana maupun terstruktur dengan baik dan stabil	Resiko menengah dengan prosedur yang lebih kompleks	Resiko tinggi dengan prosedur yang kompleks dan krisis
App.js	64	4	0

Gambar 5. Hubungan CC Terhadap Tingkat Resiko

Berdasarkan Tabel 9 dan Gambar 5 bisa dilihat hasil dari 68 *route* yang sudah ditentukan terdapat 64 *route* kategori rendah dengan prosedur sederhana maupun terstruktur dengan baik dan stabil, lalu ada 4 *route* kategori menengah dengan prosedur yang lebih kompleks, dan 0 *route* kategori tinggi dengan prosedur yang kompleks dan krisis. Maka persentase pada semua *route* yang telah diuji tingkat resiko yang rendah terhadap cacat atau *error* sebesar 94%, lalu tingkat resiko menengah sebesar 6% dan tingkat resiko tinggi sebesar 0%.

4. KESIMPULAN

Setelah melakukan penelitian didapatkan kesimpulan bahwa pengujian dengan teknik *basis path testing* dapat menghasilkan *test case* yang mampu menemukan *error*, kesalahan pada kode program *website* Room. Terdapat total 68 skenario pengujian dengan 94% skenario memiliki tingkat resiko error rendah dan 6% tingkat resiko menengah. Total 352 jalur pengujian, dimana 192 jalur *pass*, 5 jalur *fail*, 114 jalur tangkap *error*, dan 41 jalur kondisi yang salah. Oleh karena itu dapat dikatakan bahwa *website* Room memiliki tingkat resiko yang rendah terhadap *error* atau cacat. Namun, *website* Room masih membutuhkan perbaikan karena setelah mengadakan pengujian masih terdapat *error* atau *bug*, dan juga kesalahan, dan ketidaksesuaian implementasi kebutuhan dalam sistem *website* Room.

DAFTAR PUSTAKA

[1] Tachta Citra Elfira, “Tragedi! 4 Peristiwa Kesalahan Software yang Disebabkan oleh Bug,” Aug. 04, 2017. https://techno.okezone.com/read/2017/08/04/207/1750186/tragedi-4-peristiwa-kesalahan-software-yang-disebabkan-oleh-bug (accessed Jun. 17, 2022).

[2] D. Susun, O. : Linda, L.-41813120100 Dosen, P. : Wahyu, H. Haji, and M. Kom, “UNIVERSITAS MERCU BUANA 2015 | REKAYASA PERANGKAT LUNAK PENGUJIAN PERANGKAT LUNAK (SOFTWARE TESTING),” 2015.

[3] M. E. Khan, “Different Forms of Software Testing Techniques for Finding Errors,” *IJCSI International Journal of Computer Science Issues*, vol. 7, no. 1, 2010.

[4] M. E. Khan and F. Khan, “A Comparative Study of White Box, Black Box and Grey Box Testing Techniques,” 2012. [Online]. Available: www.ijacsa.thesai.org

[5] M. Farhan Londjo, “IMPLEMENTASI WHITE BOX TESTING DENGAN TEKNIK BASIS PATH PADA PENGUJIAN FORM LOGIN,” vol. 7, no. 2, p. 2021.

[6] M. Y. Rafi, I. Yusuf Arifin, D. Safutri, D. Fadilah, and J. Riyanto, “Pengujian White Box Testing Menggunakan Teknik Loop Testing pada Aplikasi Sistem Informasi Perpustakaan (Studi Kasus SMKN 3 Kota Tangerang Selatan),” 2021. [Online]. Available: http://pijarpemikiran.com/index.php/Scientia

[7] M. E. Khan, “Different Approaches to White Box Testing Technique for Finding Errors,” *International Journal of Software Engineering and Its Applications*, vol. 5, no. 3, 2011.

[8] B. D. Saputra, M. H. Subagja, M. Aldiansyah, W. Setiawan, Y. Jovanka, and J. Riyanto, “Pengujian White Box berbasis Data Flow Testing pada Program Penghitungan Luas Segitiga,” 2021. [Online]. Available: http://pijarpemikiran.com/index.php/Scientia

[9] A. Pamuji, “Strategi Perbaikan Uji Coba Struktural Perangkat Lunak Pada Metode White-Box,” *JURNAL INFORMATIKA*, vol. 5, no. 1, 2018.

[10] C. T. Pratala, E. M. Asyer, I. Prayudi, and A. Saifudin, “Pengujian White Box pada Aplikasi Cash Flow Berbasis Android Menggunakan Teknik Basis Path,” *Jurnal Informatika Universitas Pamulang*, vol. 5, no. 2, p. 111, Jun. 2020, doi: 10.32493/informatika.v5i2.4713.

[11] R. Tamara Aldisa, “Aplikasi Pengolahan Data Penjualan Pembangkit Listrik Tenaga Surya Menggunakan Model View Controler Berbasis Framework CodeIgniter Dan White Box Testing,” 2021.

[12] C. Pamela, C. Munaiseche, and G. C. Rorimpandey, “Penerapan Metode Basis Path Analysis dalam Pengujian White Box Sistem Pakar,” 2021.

[13] J. Puspitek and K. Tangerang Selatan, “ANALISIS WHITE BOX TESTING PADA APLIKASI WEB PEMESANAN SABLON KAOS,” 2021.

[14] R. Subagia, R. Alit, and A. Akbar, “PENGUJIAN WHITE BOX PADA SISTEM INFORMASI MONITORING SKRIPSI PROGRAM STUDI INFORMATIKA,” 2020.

[15] D. Madhavi, “A White Box Testing Technique in Software Testing: Basis Path Testing”, [Online]. Available: www.journalforresearch.org

[16] S. Nidhra, “Black Box and White Box Testing Techniques - A Literature Review,” *International Journal of Embedded Systems and Applications*, vol. 2, no. 2, pp. 29–50, Jun. 2012, doi: 10.5121/ijesa.2012.2204.

LAMPIRAN

Tabel Data Route Pengujian

Tabel Data Ringkasan Hasil Pengujian Skenario