Jurnal JAMIKA Manajemen Informatika

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

Optimasi Sistem Informasi Konsultasi Hukum melalui Pendekatan Pengujian Kombinasi *White-box* dan *Black-box*

Optimization of Legal Consultation Information System through Combination White-box and Black-box Testing Approach

Muhammad Agy Nurwicaksono[1], **Irma Noor Lisa**[2], **Astia Rahma Tiara**[3], **Rangga Sidik**[4*]
[1,4]Program Studi Sistem Informasi, Universitas Komputer Indonesia, Bandung, Indonesia
[2,3]Program Studi Sistem Informasi, Universitas Sari Mulia, Banjarmasin, Indonesia
*E-mail: rangga.sidik@email.unikom.ac.id

Abstrak

Penelitian ini bertujuan untuk menguji sistem informasi konsultasi hukum "Advogo" dengan harapan dapat menemukan kekurangan dalam sistem yang telah dibangun, dan membuka peluang perbaikan serta pengembangan sistem lebih lanjut. Untuk mendapatkan sebuah sistem informasi konsultasi hukum yang secara fungsional memenuhi kebutuhan harus dapat divalidasi dan diverifikasi melalui pengujian. Tidak dilaksanakannya validasi dan verifikasi dapat menyebabkan kecacatan pada sistem yang dibangun. Oleh karena itu dalam penelitian ini, untuk memvalidasi dan memverikasi, pengujian dilakukan dengan menggabungkan dua metode, white-box dan black-box testing. Metode white-box yang digunakan pada penelitian ini adalah branch dan basis path testing, sementara metode black-box yang digunakan adalah cause-effect graph. Batasan pengujian white-box pada penelitian ini adalah file SessionController.php dengan fungsi Index, Create, Store, Show dan Update. Sementara batasan black-box-nya adalah pembuatan sesi konsultasi, autentikasi serta booking sesi. Hasil dari penelitian ini adalah terbentuknya 10 test case untuk white-box testing dengan hasil 100% valid, serta 14 test case untuk black-box testing dengan hasil 92% valid, dimana terdapat 1 test case yang invalid sehingga perlu diadakannya perbaikan agar sistem dapat berfungsi sesuai dengan yang diharapkan.

Kata kunci: *Pengujian Sistem Informasi; Pengujian Black-box; Cause Effect Graph; Pengujian White-box; Basis Path.*

Abstract

This study aims to test the legal consulting information system "Advogo" in the hope of finding deficiencies in the system that has been built, and opening up opportunities for improvement and further development of the system. Testing is done by 2 methods, white-box, and black-box testing. The white-box method used in this study is branch and basis path testing, while the black-box method used is the cause-effect graph. The limitation of white-box testing in this study is the SessionController.php file with Index, Create, Store, Show, and Update functions. While the limitations of the black-box are the creation of consultation sessions, authentication, and session bookings. The results of this study are the formation of 10 test cases for white-box testing with 100% valid results, and 14 test cases for black-box testing with 92% valid results, where there is 1 invalid test case so it is necessary to make improvements so that the system can function according to which are expected.

Keywords: *Information System Testing; Black-box Testing; Cause Effect Graph; White-box Testing; Basis Path.*

Naskah diterima 21 Jun 2023; direvisi 14 Okt. 2023; dipublikasikan 1 Apr. 2024.

I. PENDAHULUAN

Perkembangan teknologi saat ini mendorong integrasi teknologi ke berbagai sektor, termasuk sektor hukum, melalui pembuatan sistem informasi terkomputerisasi. Kata “Sistem Informasi’ terdiri dari 2 kata yang memiliki artinya masing-masing, sistem dan informasi. Menurut Frisdayanti, Sistem merupakan kumpulan dari elemen atau hal yang saling bersangkut paut serta saling mempengaruhi satu sama lain untuk mencapai sebuah tujuan [1]. Sementara itu, informasi merupakan data yang telah diproses dalam berbagai bentuk dan berguna bagi penerimanya [2]. Sehingga, Sallaby dan Kanedi mendefinisikan Sistem Informasi sebagai sistem yang mengintegrasikan pengolahan transaksi harian dan kegiatan manajerial dalam sebuah organisasi, menyediakan informasi penting melalui laporan bagi pihak eksternal [3].

Penelitian ini membahas mengenai sistem informasi *platform* konsultasi hukum dengan advokat bernama “Advogo”. Advokat, berasal dari bahasa Latin "advocatus", berarti pembela ahli dalam hukum. Mereka adalah ahli yang membantu dalam isu-isu hukum dan dianggap sebagai profesi mulia, "Officium nobile", berdasarkan kepercayaan klien untuk memperjuangkan hak-haknya [4]. Layanan konsultasi hukum

oleh advokat adalah kegiatan pribadi antara klien dan penasihat yang memberikan pendapat dan saran untuk memenuhi kebutuhan klien [5]. Black's Law Dictionary mendefinisikannya sebagai aktivitas perundingan antara klien dan penasihat hukum [6]. Manuhutu dan Wattimena melihat konsultasi sebagai layanan yang membantu memperoleh pemahaman, wawasan, dan cara-cara menangani masalah klien [7]. Keseluruhannya, layanan konsultasi adalah proses kerjasama dalam mendapatkan informasi atau memecahkan masalah bagi mereka yang membutuhkannya [8]. Sistem Advogo dirancang berdasarkan kebutuhan para advokat akan adanya aplikasi konsultasi hukum yang terintegrasi dengan fitur *video call*. Sistem ini memungkinkan konsultasi hukum dengan advokat dapat dilakukan secara cepat, efisien dan mudah diakses tanpa harus datang ke kantor advokat tersebut.

Dalam kegiatan pengembangan sisten informasi ini, terdapat beberapa tahapan pengembangan yang disebut dengan *Software Development Life Cycle* atau SDLC. SDLC merupakan sebuah siklus pengembangan perangkat lunak yang terdiri dari tahapan-tahapan dan keberadaannya sangat penting terutama dari segi pengembangan sebuah sistem informasi [9]. Tahapan utama yang ada pada SDLC ini mencakup analisis, perancangan, implementasi, pengujian dan *maintenance* [10]. Tetapi, seperti setiap perangkat lunak lainnya, penting untuk memastikan bahwa platform ini bekerja dengan baik dan sesuai dengan kebutuhannya.

Tahap pengujian atau *testing* dalam SDLC memainkan peran vital dalam menjamin kualitas dan keandalan sistem. Menurut Uddin A dan Anand A, pengujian penting dilakukan mengingat potensi kesalahan intrinsik dalam proses pengembangan oleh manusia, verifikasi dan validasi perangkat lunak menjadi imperatif guna memastikan bahwa perangkat lunak bersifat kokoh atau *robust*, sesuai dengan kebutuhan pengguna, dan diimplementasikan sesuai spesifikasi yang ditentukan [11]. Kesalahan, walaupun tampak trivial, dapat berujung pada konsekuensi serius yang mengakibatkan dampak negatif pada pengguna dan dapat merusak integritas reputasi pengembang. Pengujian ini dilakukan dengan menjalankan sebuah sistem dengan harapan menemukan sebuah masalah [12] dalam sistem yang memungkinkan dalam menyebabkan proses bisnis tidak terpenuhi. Dengan demikian, implementasi pengujian *software* yang komprehensif adalah krusial dalam menjamin bahwa *output* yang dihasilkan oleh proses pengembangan adalah perangkat lunak yang memiliki standar kualitas yang tinggi, kohesif, dan konsisten dengan ekspektasi serta kebutuhan *stakeholder* [13]. Secara umum, ada dua jenis pengujian yang sering dilakukan, yaitu *White-box* dan *Black-box* [14]. *White-box* testing mengkaji aspek internal dari perangkat lunak [15], dengan beberapa metode populer di antaranya adalah basis *path, branch coverage, condition coverage, loop testing, multiple condition coverage, statement coverage* dan metode lainnya. Sementara itu, *Black-box* testing fokus pada input dan output tanpa perlu mengetahui proses internal [16], dengan beberapa metode seperti *cause-effect diagram, all pair testing, boundary value analysis, equivalance partitioning, fuzzing*, dan metode lainnya.

Walaupun sistem informasi Advogo ini telah rampung dikembangkan, sistem ini belum pernah dilakukan pengujian secara *black-box* maupun *white-box* untuk memeriksa kinerja dan kualitas dari sistem tersebut. Oleh karena itu, diperlukan sebuah pengujian terhadap sistem informasi Advogo ini sebagai bahan pertimbangan dalam perawatan, perbaikan hingga pengembangan sistem lebih lanjut agar dapat digunakan sebagaimana mestinya. Terdapat beberapa penelitian terdahulu terkait pengujian sistem informasi. Seperti yang dilakukan oleh Gunawan et. al. yang membahas tentang pengujian *white-box* pada pengembangan sistem informasi kemajuan akademik dengan menggunakan *basis path testing* [17]. Namun, penelitian ini hanya menjabarkan hasil dari penelusuran *path* dan perhitungan kompleksitasnya saja, tidak menampilkan *test case* yang justru merupakan kunci utama dari pengujian sistem. Kemudian, Wijaya et. al. dalam penelitiannya yang berjudul "Pengujian *Black-box* Sistem Informasi Penilaian Kinerja Karyawan PT Inka (Persero) Berbasis Equivalence Partitions" membahas pengujian *black-box* pada sistem informasi penilaian kinerja karyawan menggunakan *Equivalance Partition* atau EV [18]. Namun, penggunaan metode EV hanya berpatokan pada input dari setiap form dan menguji batas-batas partisinya saja, dan cenderung hanya mencakup pengujian berbasis angka dan pengujian sederhana saja sehingga dinilai tidak efektif dan tidak komplit dalam pengujiannya.

Berdasarkan penelitian terdahulu di atas, rata-rata pengujian sistem yang dilakukan hanya berfokus pada satu metode saja dan belum ditemukan penelitian yang menggabungkan antara metode *white-box* dan *black-box* dalam menguji sistem informasi konsultasi hukum secara spesifik. Penelitian sebelumnyapun belum menerapkan metode pengujiannya secara maksimal. Dalam penelitian ini akan menggabungkan kedua metode tersebut dalam menguji satu sistem informasi konsultasi hukum. Penggabungan dua metode pengujian ini dilakukan agar validasi dan verifikasi perangkat lunak dapat tercapai. Sistem informasi konsultasi hukum dapat menjamin kesesuaian fungsional dari sisi pengkodean dan juga pola jalur fungsi dari sisi pengguna. Penelitian ini diharapkan dapat memberikan kontribusi signifikan dengan menyediakan kerangka kerja pengujian yang komprehensif bagi sistem informasi konsultasi hukum. Melalui pendekatan kombinasi, penelitian ini berupaya

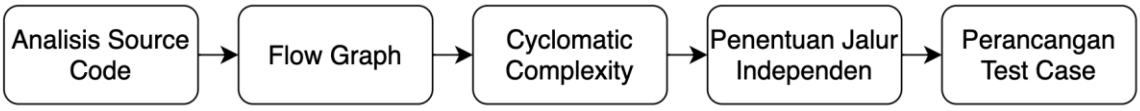
Jurnal JAMIKA Manajemen Informatika
Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

untuk memvalidasi dan memverifikasi perangkat lunak dari sisi teknis dan fungsional, menjamin kesesuaian fungsional dari sisi pengkodean dan juga pola jalur fungsi dari sisi pengguna.

II. METODE PENELITIAN

White-box testing dan *black-box testing* adalah metodologi pengujian yang digunakan dalam penelitian ini. *White-box testing* merupakan pendekatan pengujian yang berfokus pada sistem internal, khususnya program kode sumber atau *source code*. Teknik *white-box testing* memiliki tujuan utama, yaitu sebagai alat untuk merakit program atau kode. Untuk *developer* atau pemrogram, melakukan *white-box testing* adalah langkah penting dalam menentukan tingkat kerumitan kode tertentu. Pengujian *white-box* ini-pun sangat mampu dipergunakan dalam memeriksa apakah kode sumber mengikuti desain yang telah dibuat, apakah memenuhi persyaratan fungsional yang telah dibuat, dan apakah terdapat kesalahan atau *bug* [19].

Desain perangkat lunak, struktur internal, dan kode dievaluasi untuk memastikan keakuratan *input-output* dan meningkatkan kualitas dan kegunaannya. Karena kode sumber dapat dilihat oleh pengembang atau penguji selama *white-box testing*, istilah pengujian kotak kosong atau *clear box testing* sering digunakan. Pengujian *White-box* adalah komponen penting dari proses pembangunan otomatis yang dilakukan dalam *pipeline Continuous Integration (CI)* atau *Continuous Delivery (CD)* [20]. *Branch Testing* dan *Basis Path Testing* adalah metode *white-box testing* yang digunakan. Berikut prosedur pengujian *white-box* yang digunakan dalam penelitian ini seperti yang ditampilkan pada Gambar 1.



Gambar 1. Perencanaan Pengujian *White-box*

Berikut penjelasan secara detail mengenai rancangan prosedur *white-box testing* di atas:

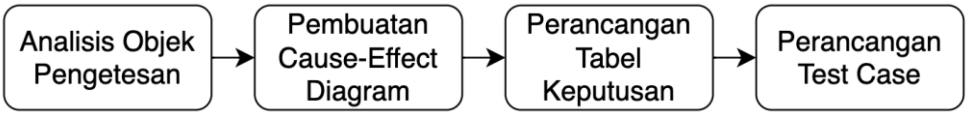
1. Analisis *Source Code*
   Kode sumber (*source code*) dari aplikasi akan dianalisis terlebih dahulu, mulai dari menentukan *file* dan fungsi (*function*) apa saja yang harus di uji, struktur kodingan hingga pembagian baris kodingan ke dalam grup penomoran untuk selanjutnya dibuat *flow graph*. *File* dan fungsi yang akan di tes adalah *SessionController.php* dengan fungsi *Index, Create, Store, Show* dan *Update*.
2. Flow Graph
   Flow graph merupakan notasi sederhana untuk merepresentasi *control flow*. Pada tahap ini, kodingan yang sudah dibuat ke dalam grup penomoran akan direpresentasikan ke dalam bentuk grafis yang saling tersambung dengan anak panah. Setiap percabangan dan perulangan akan dinotasikan dengan grafis agar lebih mudah dalam menelusuri *control flow*-nya.
3. Cyclomatic Complexity
   Cyclomatic complexity (VG) atau kompleksitas siklomatik digunakan untuk menentukan jumlah jalur yang diperlukan. Definisi kompleksitas siklomatik menentukan jumlah minimum kasus uji yang harus diterapkan untuk memastikan bahwa setiap pernyataan telah diterapkan setidaknya sekali. Ini juga memberikan jumlah pernyataan independen yang harus ada di setiap *file* dasar program. Untuk mendapatkan Kompleksitas siklomatik ditunjukkan pada persamaan (1) atau persamaan (2)
   $$V(G) = E - N + 2 \quad (1)$$
   Atau
   $$V(G) = P + 1 \quad (2)$$
   Dimana:
   E = Total Jumlah *Edge*
   N = Total Jumlah *Node*
   P = Jumlah Predikat (*Node* yang memiliki cabang)
4. Jalur Independen / *Independent Path*
   Jalur independen merupakan jalur dalam diagram alir yang tidak bersilang satu sama lain dan memiliki awal dan akhir yang jelas. Dalam *basis path testing*, jalur independen memiliki tingkat kompleksitas yang berbeda dan harus diuji secara terpisah. Setiap jalur independen harus memiliki setidaknya satu tes kasus yang mencakup semua kondisi yang mungkin terjadi pada jalur tersebut. Dengan menguji setiap jalur independen secara terpisah, metode *basis path testing* memastikan bahwa semua jalur dalam kode sumber telah diuji dan meminimalisir kemungkinan adanya *bug* yang tidak terdeteksi.

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

5. Perancangan *Test Case*
   Setelah ditentukan jalur independen dari *flow graph* di atas, maka langkah selanjutnya adalah membuat *test case* minimal sejumlah jalur independen yang telah dibuat, memastikan bahwa setiap kemungkinan jalur yang akan dilewati dibuat *test case*-nya.

Sementara itu, *black-box testing* merupakan pengujian tanpa sepengetahuan dari cara kerja internal aplikasi yang sedang diuji atau *Application Under Test* (AUT). Ini juga dikenal sebagai pengujian fungsional atau pengujian yang berorientasi kepada *input driven*. Teknik pengujian *software* dimana cara kerja dari aplikasi internal dari item yang diuji tidak diketahui oleh penguji sistem. Pengujian secara *black-box* ini dapat dilaksanakan dengan menggunakan 5 cara, yaitu *Equivalence Partitioning*, *Boundary Value Analysis*, *Cause Effect Graphing, Fuzzy Testing*, dan *Model Based Testing* [19].

Pada *black-box testing*, pengujian dilakukan berdasar detail aplikasi seperti, tampilan aplikasi, fungsi-fungsi yang ada pada aplikasi, dan kesesuaiannya terhadap alur fungsi dengan bisnis proses yang diinginkan oleh para *stackholder* atau klien. Pengujian *black-box* ini tidak melihat dan menguji kode sumber dari aplikasi. *Black-box testing* ini bekerja dengan mengabaikan struktur *control* dari program sehingga perhatiannya ter fokuskan kepada informasi domain [21]. Metode *black-box testing* yang digunakan pada penelitian ini adalah *Cause-Effect Graph* dengan langkah-langkah yang ditampilkan pada Gambar 2 di bawah ini.



Gambar 2. Langkah Pengujian *Black-box*

1. Analisis Objek Pengetesan
   Tahap pertama dalam pengujian *black-box* ini adalah mendefinisikan objek atau fitur apa saja yang akan di tes. Pada penelitian ini, objek yang di tes adalah formulir-formulir pada sistem advogo, diantaranya adalah; *Login*, *Register* Akun, dan *Booking* Sesi.
2. Pembuatan *Cause-Effect Diagram*
   Dari spesifikasi yang telah ditentukan penguji dapat mengidentifikasi *cause* (penyebab) dan *effect* (akibat). *Cause* merupakan kondisi dari suatu *input*, sedangkan *Effect* merupakan kondisi *output* atau transformasi dari sebuah sistem. Setelah semua *cause* dan *effect* teridentifikasi, maka harus digambarkan hubungan antar *cause* dan *effect*-nya menggunakan grafik *cause-effect*, mengidentifikasi *constraint* untuk menjelaskan kombinasi antar *cause* dan antar *effect* yang tidak mungkin.
3. Tabel Keputusan
   Setelah diagram *Cause-Effect* terbuat, maka diagram tersebut diubah menjadi dalam bentuk *decision table*/tabel keputusan. Tabel ini mencakup daftar *input* dan *output* dan menunjukkan bagaimana setiap *input* mempengaruhi *output*. Tabel Keputusan digunakan untuk membantu identifikasi faktor-faktor yang mempengaruhi hasil akhir dan memperkuat analisis hubungan sebab-akibat.
4. Perancangan *Test Case*
   Tahap terakhir dalam pengujian *black-box* dengan *Cause-Effect* diagram adalah perancangan *test case*. *Test Case* ini dirancang atas dasar tabel keputusan yang sudah dibuat sebelumnya. Minimal jumlah *test case* yang dibuat mengikuti jumlah kolom dalam tabel keputusan, bisa lebih banyak, tapi tidak boleh lebih sedikit.

III. HASIL DAN PEMBAHASAN

Dalam penelitian ini, hasil dan pembahasan dari pengujian dibagi menjadi 2 bagian, *white-box* dan *black-box testing*. Sesuai dengan metodologi dan prosedur penelitian yang telah dirancang sebelumnya, tahap pertama dari pengujian yang akan dilakukan adalah pengujian *white-box* dengan metode *branch* dan *basis path testing*. Proses pengujian ini dilakukan dengan modul-modul berbentuk fungsi program yang telah ditentukan sebagai objek pengujian, yaitu *SessionController.php* dengan fungsi *Index, Create, Store, Show*, dan *Update*.

Pengujian White-box

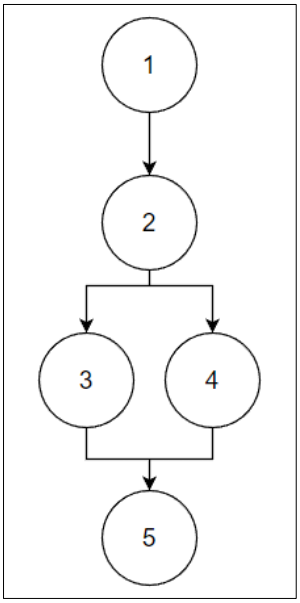
Pada pengujian *white-box* yang dilakukan hanya berfokus pada fungsi *Index, Create, Store, Show*, dan *Update* yang berada pada file *SessionController.php*.

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

Source Code Index()

```
    public function index()
1   {
        $user = Auth::user();
2       if($user->user_type == "advocate"){
            $sessions = Session::with('invoice')
            ->whereHas('invoice', function($q) {
                $q->where('status', 'lunas');
            })
            ->orderBy('status', 'asc')
3           ->where('advocate_id', $user->advocate->id)-
    >paginate(20);

            return view('session.index_advocate', [
                'sessions' => $sessions
            ]);
        } else {
            $sessions = Session::where('client_id', $user-
    >client->id)->orderBy('created_at', 'desc')
                        ->paginate(5);
4
            return view('session.index', [
                'sessions' => $sessions
            ]);
        }
5   }
```



Gambar 3. *Flow Graph* Fungsi *Index* pada *SessionController.php*

Dari *flow graph* yang telah terbentuk maka didapatkan *Cyclomatic Complexity* dan jalur independen sebanyak 2. Sehingga *test case* yang terbentuk dapat dilihat pada tabel 1 di bawah ini.

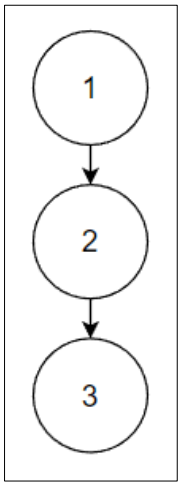
TABEL 1
TEST CASE FUNGSI INDEX PADA SESSIONCONTROLLER.PHP

Nama Skenario	Kegiatan	Hasil yang Diharapkan	Hasil	Keterangan
Sesi Konsultasi View List_Advocate	Melihat daftar konsultasi klien jika user_type adalah "advocate"	Masuk ke halaman "Daftar Konsultasi Client"	Masuk ke halaman "Daftar Konsultasi Client"	**VALID**
Sesi Konsultasi View List_Client	Melihat riwayat konsultasi dengan advokat jika user_type adalah "client"	Masuk ke halaman "Riwayat Konsultasi"	Masuk ke halaman "Riwayat Konsultasi"	**VALID**

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

Source Code Create()

```
1   public function create(Advocate $advocate)
    {
        return view('session.create', [
2           'advocate' => $advocate
        ]);
3   }
```



Gambar 4. *Flow Graph* Fungsi *Create* pada *SessionController.php*

Cyclomatic Complexity yang terbentuk adalah 1 sehingga jalur independen pun sebanyak 1. *Test case* pada fungsi *create* dapat dilihat pada tabel 2 di bawah ini.

TABEL 2
TEST CASE FUNGSI CREATE PADA SESSIONCONTROLLER.PHP

Nama Skenario	Kegiatan	Hasil yang Diharapkan	Hasil	Keterangan
Booking Sesi Konsultasi View Client	Menampilkan halaman booking sesi konsultasi di user_type client	Masuk ke halaman "Booking Sesi Konsultasi"	Masuk ke halaman "Booking Sesi Konsultasi"	**VALID**

Source Code Store()

```
1       public function store(Advocate $advocate, StoreSessionRequest $request)
        {
            \Midtrans\Config::$serverKey = env('MIDTRANS_SERVER_KEY');
            \Midtrans\Config::$isProduction = false;
            \Midtrans\Config::$isSanitized = true;
2           \Midtrans\Config::$is3ds = true;

            $user = Auth::user();
            $no_invoice = 'INV-' . Carbon::now()->format('dmyhis');
3           if($request->input('durasi_konsultasi') == '30'){
4               $total = 150000;
            }else{
5               $total = 300000;
            }
            $params = array(
                'transaction_details' => array(
                    'order_id' => $no_invoice,
                    'gross_amount' => $total,
6               ),
                'item_details' => array(
                    array(
                        'id' => "1",
                        'price' => $total,
```

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

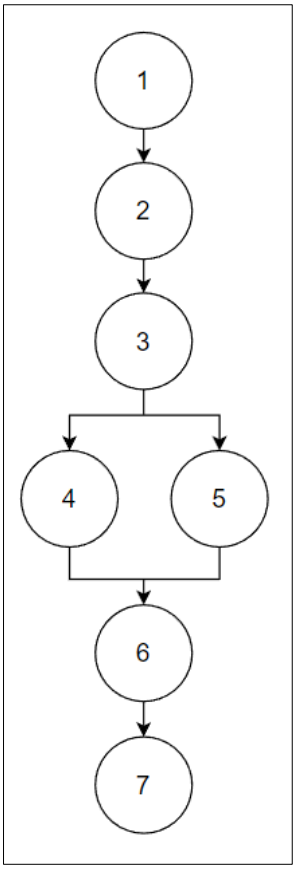
```
                    'quantity' => 1,
                    'name' => 'Sesi Konsultasi dengan ' . $advocate->user->name . '(' . $request->input('durasi_konsultasi') . '
menit' . ')'
                ]
            ],
            'customer_details' => array(
                'first_name' => $user->name,
                'last_name' => '',
                'email' => $user->email
            ),
        );

        $snapToken = \Midtrans\Snap::getSnapToken($params);

        $invoice = Invoice::create([
            'no_invoice' => $no_invoice,
            'status' => 'belum_lunas',
            'midtrans_snap_token' => $snapToken,
            'total' => $total
        ]);

        $session = Session::create([
            'invoice_id' => $invoice->id,
            'advocate_id' => $advocate->id,
            'client_id' => $user->client->id,
            'judul' => $request->input('bidang_permasalahan'),
            'deskripsi_permasalahan' => $request->input('deskripsi_permasalahan'),
            'session' => $request->input('jadwal_konsultasi_jam'),
            'session_date' => $request->input('jadwal_konsultasi_tanggal'),
            'session_key' => Str::random(20),
            'status' => 'pending'
        ]);

        return redirect(route('riwayat-konsultasi.detail-riwayat-konsultasi', $session->id));
7   }
```



Gambar 5. *Flow Graph* Fungsi *Store* pada *SessionController.php*

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

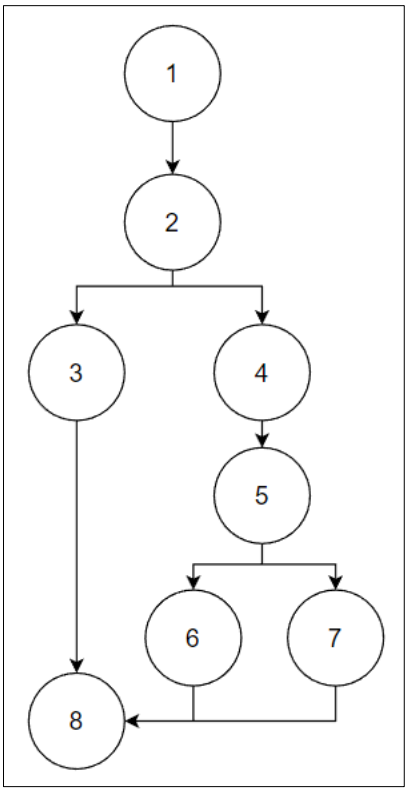
Cyclomatic Complexity dan jalur independen yang terbentuk berdasarkan *flow graph* pada gambar 5 adalah 2 sehingga *test case* dapat terlihat pada tabel 3 berikut ini.

TABEL 3
TEST CASE FUNGSI STORE PADA SESSIONCONTROLLER.PHP

Nama Skenario	Kegiatan	Hasil yang Diharapkan	Hasil	Keterangan
Sesi Konsultasi View List_Advocate	Melihat daftar konsultasi klien jika user_type adalah "advocate"	Masuk ke halaman "Daftar Konsultasi Client"	Masuk ke halaman "Daftar Konsultasi Client"	**VALID**
Sesi Konsultasi View List_Client	Melihat riwayat konsultasi dengan advokat jika user_type adalah "client"	Masuk ke halaman "Riwayat Konsultasi"	Masuk ke halaman "Riwayat Konsultasi"	**VALID**

Source Code Show()

```
1   public function show(Session $session)
    {
2       if(Auth::user()->user_type == "advocate"){
            return view('session.show_advocate', [
                'session' => $session,
3               'advocate' => $session->advocate
            ]);
4       } else {
5           if ($session->client->user->id != Auth::user()->id) {
                return redirect('/');
6           }
            return view('session.show', [
                'session' => $session,
7               'advocate' => $session->advocate
            ]);
        }
8   }
```



Gambar 6. *Flow Graph* Fungsi *Show* pada *SessionController.php*

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

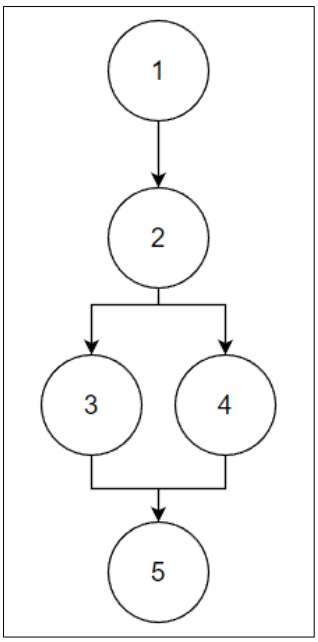
Dari *Flow graph* pada gambar 6 maka didapatkan *Cyclomatic Complexity* sebanyak 3 yang artinya dapat dideskripsikan dengan 3 jalur Independen masing-masing. Yaitu: *Path 1* = 1-2-3-8, *Path 2* = 1-2-4-5-6-8, dan *Path 3* = 1-2-4-5-7-8. Adapun *test case* untuk kasus ini adalah terlihat pada tabel 4 berikut ini.

TABEL 4
TEST CASE FUNGSI SHOW PADA SESSIONCONTROLLER.PHP

Nama Skenario	Kegiatan	Hasil yang Diharapkan	Hasil	Keterangan
Detail Konsultasi User Advokat	Melihat detail konsultasi advokat dengan klien	Masuk ke halaman "Detail Konsultasi" sesuai dengan data sesi klien yang dipilih	Masuk ke halaman "Detail Konsultasi" sesuai dengan data sesi klien yang dipilih	**VALID**
Detail Konsultasi User Klien_1	Me-redirect klien jika ID sesi konsultasi tidak sesuai	Redirect ke home (route '/')	Redirect ke home (route '/')	**VALID**
Detail Konsultasi User Klien_2	Melihat detail sesi konsultasi klien dengan advokat	Masuk ke halaman "Sesi Konsultasi" sesuai dengan sesi konsultan yang dipilih	Masuk ke halaman "Sesi Konsultasi" sesuai dengan sesi konsultan yang dipilih	**VALID**

Source Code Update()

```
1   public function update(UpdateSessionRequest $request, Session $session)
    {
2       if($request->has('status')){
3           $session->status = $request->input('status');
        }else{
            $session->session_start_at = Carbon::parse($session->session_date . ' ' . $request->input('jam_konsultasi') .
    ':00')->format('Y-m-d H:i:s');
4           $session->session_end_at = Carbon::parse($session->session_date . ' ' . $request->input('jam_konsultasi') .
    ':00')->addMinutes( ($session->invoice->total / 150000) * 30 )->format('Y-m-d H:i:s');
        }
        $session->save();
5       return redirect()->back();
    }
```



Gambar 7. *Flow Graph* Fungsi *Update* pada *SessionController.php*

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

Berdasarkan penggambaran *flow graph* maka didapatkan bahwa *cyclomatic Complexity* sebanyak 2. Sehingga didapatkan jalur Independen yang terbentuk sebanyak 2, yaitu Path 1 = 1-2-3-5 dan Path 2 = 1-2-4-5. Langkah selanjutnya adalah mengubah jalur independen menjadi *test case* seperti terlihat pada tabel 5 di bawah ini.

TABEL 5
TEST CASE FUNGSI UPDATE PADA SESSIONCONTROLLER.PHP

Nama Skenario	Kegiatan	Hasil yang Diharapkan	Hasil	Keterangan
Sesi Konsultasi Update_Status	Melakukan update status konsultasi sesuai dengan input yang dipilih	Status konsultasi ter-update sesuai dengan input yang dipilih	Status konsultasi ter-update sesuai dengan input yang dipilih	**VALID**
Sesi Konsultasi Update_Jadwal	Melakukan update jadwal jam mulai dan selesai konsultasi sesuai	Jam mulai dan jam selesai konsultasi ter-update sesuai dengan input jam yang dimasukkan	Jam mulai dan jam selesai konsultasi ter-update sesuai dengan input jam yang dimasukkan	**VALID**

Tahap kedua dari pengujian yang akan dilakukan adalah pengujian *black-box* dengan metode *Cause-Effect Diagram.* Proses pengujian dilakukan terhadap modul *register* dan *booking* sesi konsultasi.

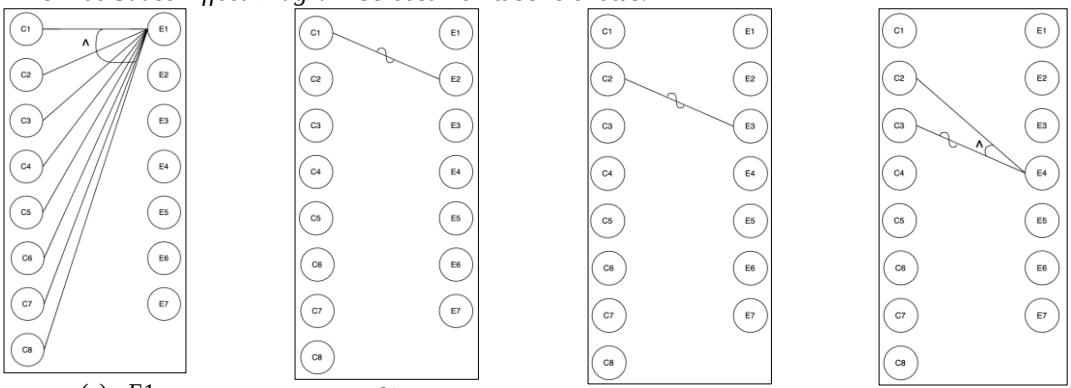
Black-box testing

Untuk pengujian *black-box* dilakukan pada fitur *Register* dan *Booking* Sesi Konsultasi, langkah pertama adalah menentukan sebab-akibat dari fitur *register* seperti disajikan pada tabel 6 berikut ini. Langkah selanjutnya adalah membuat diagram *cause-effect* seperti terlihat pada gambar 8.

TABEL 6
CAUSE-EFFECT PADA FITUR REGISTER

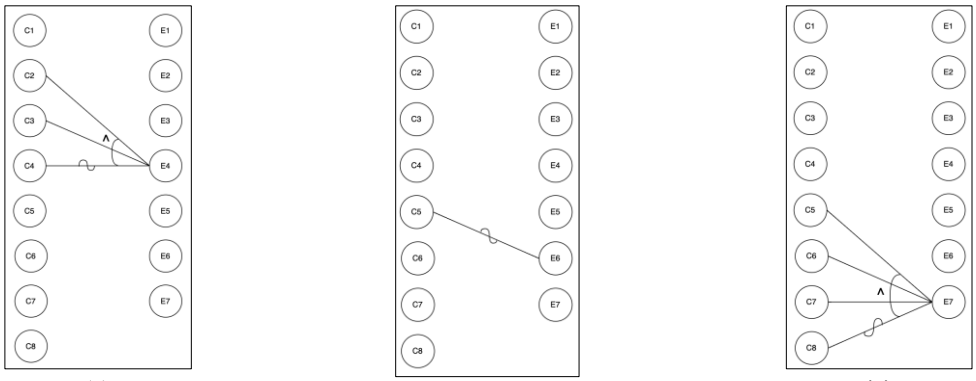
Cause	Effect
C1 = Input “Nama Lengkap” tidak kosong C2 = Input “Email” tidak kosong C3 = Input “Email” menggunakan format email C4 = Input “Email” unique (belum terdaftar dalam database) C5 = Input “Password” tidak kosong C6 = Input Password berisi minimal 8 karakter C7 = Input “Ulangi Password” tidak kosong C8 = Input “Ulangi Password” sama dengan input “Password”	E1 = Register Sukses E2 = Alert “Nama Lengkap Harus Diisi” E3 = Alert “Email Harus Diisi” E4 = Alert “Email Harus berupa format email ‘@’” E5 = Alert “Email telah terdaftar sebelumnya” E6 = Alert “Password Harus Diisi” E7 = Alert “Konfirmasi Password harus sama dengan Password”

Berikut *Cause-Effect Diagram* berdasarkan tabel 6 di atas:



(a) E1 (b) E2 (c) E3 (d) E4

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika



(e) E5 (f) E6 (g) E7

Gambar 8. *Cause-Effect Diagram*

Untuk mendapatkan *test case* yang sesuai maka terlebih dahulu harus dibentuk tabel keputusan. Pada tabel 7 merupakan tabel keputusan yang menjadi dasar pembuatan *test case*.

TABEL 7
TABEL KEPUTUSAN

Actions	TC1	TC2	TC3	TC4	TC5	TC6	TC7
C1	1	0	0	0	0	0	0
C2	1	0	0	1	1	0	0
C3	1	0	0	0	1	0	0
C4	1	0	0	0	0	0	0
C5	1	0	0	0	0	0	1
C6	1	0	0	0	0	0	1
C7	1	0	0	0	0	0	1
C8	1	0	0	0	0	0	0
E1	1	0	0	0	0	0	0
E2	0	1	0	0	0	0	0
E3	0	0	1	0	0	0	0
E4	0	0	0	1	0	0	0
E5	0	0	0	0	1	0	0
E6	0	0	0	0	0	1	0
E7	0	0	0	0	0	0	1

Adapun *test case* untuk fitur *register* tersebut adalah dapat dilihat pada tabel 8 di bawah ini. Dari *test case* yang dilakukan, sebanyak 6 *test case* bisa menunjukkan hasil yang sesuai dengan yang diharapkan. Sedangkan 1 *test case* tidak sesuai dengan yang diharapkan.

TABEL 8
TEST CASE FITUR REGISTER *BLACK-BOX* TESTING

TC Name	*Description*	*Expected Results*
TC1	Memvalidasi bahwa sistem dapat melakukan register jika semua field input dimasukkan dengan benar.	Akun berhasil terbuat dan otomatis Login ke dalam sistem. **(VALID)**
TC2	Memvalidasi bahwa jika nama lengkap tidak diisi, akan menampilkan error "Nama Lengkap Harus Diisi".	Menampilkan error "Nama Lengkap Harus Diisi". **(VALID)**
TC3	Memvalidasi bahwa jika email tidak diisi, akan menampilkan error "Email Harus Diisi".	Menampilkan error "Email Harus Diisi". **(VALID)**
TC4	Memvalidasi bahwa jika email mengandung format yang benar (mengandung simbol '@' dan top level domain/TLD), atau akan menampilkan error "Email Harus berupa format email '@'".	Menampilkan error "Email Harus berupa format '@'". **(VALID)**
TC5	Memvalidasi bahwa jika email telah terdaftar sebelumnya, akan menampilkan error "Email telah terdaftar sebelumnya".	Menampilkan error "validation.unique". **(INVALID)**

Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

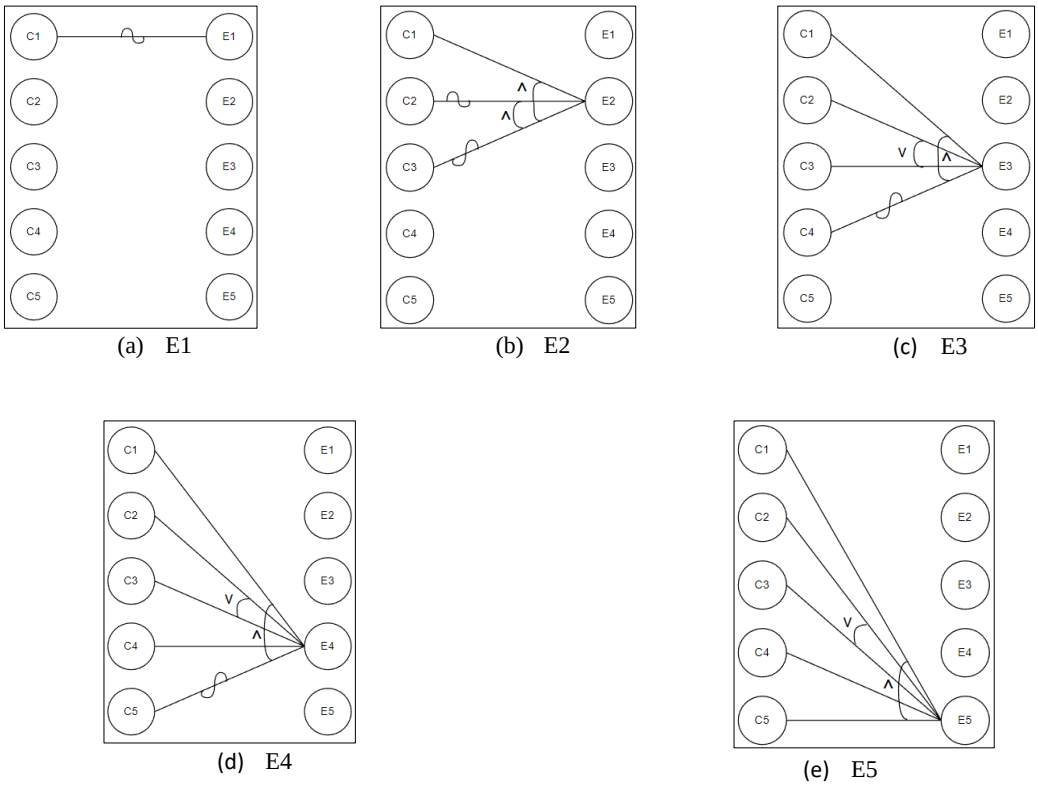
TC Name	*Description*	*Expected Results*
TC6	Memvalidasi bahwa jika password tidak diisi, maka akan menampilkan error "Password Harus Diisi".	Menampilkan error "Password Harus Diisi". **(VALID)**
TC7	Memvalidasi bahwa jika konfirmasi password tidak sama dengan password, maka akan menampilkan error "Konfirmasi password harus sama dengan password"	Memampilkan error "Konfirmasi Password harus sama dengan password". **(VALID)**

Untuk fitur *booking* sesi konsultan, sebab-akibat dari fitur tersebut dapat dilihat pada tabel 9 berikut ini. terdapat 5 sebab dan 5 akibat yang masing-masing mempunyai relasi yang berbeda seperti terlihat pada diagram *cause-effect* pada gambar 9.

TABEL 9
CAUSE-EFFECT FITUR BOOKING

Cause	Effect
C1 = Select "Pilih Bidang Permasalahan" tidak null	E1 = Alert "Bidang Permasalahan Harus Diisi"
C2 = Value "Durasi Konsultasi" adalah 30 Menit	E2 = Alert "Jam Konsultasi harus antara 30 atau 60 menit"
C3 = Value "Durasi Konsultasi" adalah 60 Menit	E3 = Alert "Tanggal Konsultasi harus lebih besar Hari H"
C4 = Value "Hari Konsultasi" diisi H+1	E4 = Alert "Tanggal Konsultasi harus lebih kecil dari Hari H+4"
C5 = Value "Hari Konsultasi" diisi H-4	E5 = Booking Sesi Sukses

Adapun *Cause-Effect Diagram* berdasarkan tabel 9 *Cause* dan *Effect* di atas, yaitu terlihat pada gambar 9:



(a) E1 (b) E2 (c) E3

(d) E4 (e) E5

Gambar 9. *Cause-Effect Diagram* Fitur *Booking*

Untuk mendapatkan kesesuaian *test case*, merujuk pada tabel keputusan pada tabel 10. Pada tabel tersebut ditunjukkan pada 1 *Cause* dan *effect* dapat dibuktikan dengan 1 atau lebih *test case*.

TABEL 10
TABEL KEPUTUSAN FITUR BOOKING

Actions	TC1	TC2	TC3	TC4	TC5	TC6	TC7
C1	0	1	1	1	1	1	1

Jurnal JAMIKA Manajemen Informatika
Jurnal Manajemen Informatika (JAMIKA)
Volume 14 Nomor 1 Edisi April 2024
E ISSN: 2655-6960 | P ISSN: 2088-4125
OJS: https://ojs.unikom.ac.id/index.php/jamika

Actions	TC1	TC2	TC3	TC4	TC5	TC6	TC7
C2	0	0	1	0	1	0	1
C3	0	0	0	1	0	1	1
C4	0	0	0	0	1	1	1
C5	0	0	0	0	0	0	1
E1	1	0	0	0	0	0	0
E2	0	1	0	0	0	0	0
E3	0	0	1	1	0	0	0
E4	0	0	0	0	1	1	0
E5	0	0	0	0	0	0	1

Adapun pelaksanaan *test case* yang menjadi acuan pengujian disajikan pada tabel 11. Berdasarkan tabel tersebut semua *test case* dapat berjalan sesuai dengan hasil yang diharapkan.

TABEL 11
TEST CASE FITUR BOOKING SESI *BLACK-BOX* TESTING

TC Name	*Description*	*Expected Results*
TC1	Memvalidasi bahwa jika bidang permasalahan tidak dipilih, akan menampilkan error "Bidang Permasalahan Harus Diisi".	Menampilkan error "Bidang Permasalahan Harus Diisi". **(VALID)**
TC2	Memvalidasi bahwa jam konsultasi harus pilihan antara 30 atau 60 menit, atau akan menampilkan error "Jam Konsultasi harus antara 30 atau 60 menit".	Menampilkan error "Jam Konsultasi harus antara 30 atau 60 menit". **(VALID)**
TC3	Memvalidasi bahwa Tanggal Konsultasi harus lebih besar dari hari H ketika semua value di-input dan Jam Konsultasi adalah 30 Menit, atau akan menampilkan error "Tanggal Konsultasi harus lebih besar dari hari H".	Menampilkan error "Tanggal Konsultasi harus lebih besar dari hari H". **(VALID)**
TC4	Memvalidasi bahwa Tanggal Konsultasi harus lebih besar dari hari H ketika semua value di-input dan Jam Konsultasi adalah 60 Menit, atau akan menampilkan error "Tanggal Konsultasi harus lebih besar dari hari H".	Menampilkan error "Tanggal Konsultasi harus lebih besar dari hari H". **(VALID)**
TC5	Memvalidasi bahwa Tanggal Konsultasi harus lebih kecil dari hari H+3 ketika semua value di-input dan Jam Konsultasi adalah 30 Menit, atau akan menampilkan error "Tanggal Konsultasi harus lebih kecil dari Hari H+3".	Menampilkan error "Tanggal Konsultasi harus lebih kecil dari Hari H+3". **(VALID)**
TC6	Memvalidasi bahwa Tanggal Konsultasi harus lebih kecil dari hari H+3 ketika semua value di-input dan Jam Konsultasi adalah 60 Menit, atau akan menampilkan error "Tanggal Konsultasi harus lebih kecil dari Hari H+3".	Menampilkan error "Tanggal Konsultasi harus lebih kecil dari Hari H+3". **(VALID)**
TC7	Memvalidasi bahwa sistem dapat melakukan booking sesi dengan konsultan jika semua input field dimasukkan dengan benar.	Booking Sesi berhasil dibuat. **(VALID)**

Dalam proses pengujian yang dilaksanakan, baik melalui metode *white-box* maupun *black-box*, telah dilakukan analisis mendalam mengenai kesesuaian prosedur dan penerapan logika. Hasil pengujian untuk setiap metode dan *test case* menunjukkan bahwa pengujian *white-box* telah dilaksanakan sesuai dengan logika dan prosedur yang telah ditetapkan. Hal ini ditandai dengan seluruh kemungkinan dalam fungsi program telah berhasil dijalani tanpa ditemukannya *bug* atau *error* pada *test case* yang ditentukan. Sebaliknya, meskipun pengujian *black-box* telah dilakukan sesuai dengan prosedur yang seharusnya, ada satu ketidaksesuaian yang ditemukan pada fitur register. Ketika suatu email yang telah terdaftar dalam *database* dicoba untuk didaftarkan kembali, program gagal menampilkan pesan *error* yang seharusnya muncul. Hal ini berpotensi menyebabkan kebingungan bagi pengguna layanan Advogo. Maka dari itu, langkah selanjutnya yang perlu dilakukan oleh pengembang adalah menambahkan logika untuk menampilkan *error message* tersebut.

IV. KESIMPULAN

Setelah melaksanakan serangkaian pengujian pada Sistem Informasi Jasa Konsultasi Hukum Advogo menggunakan metode *white-box*, khususnya *Branch Testing* dan *Basis Path Testing*, serta metode *black-box* melalui *Cause-Effect Diagram*, ditemukan hasil yang spesifik, yaitu pengujian menunjukkan bahwa mayoritas fungsi berjalan sesuai dengan spesifikasi dan kebutuhan yang ditentukan. Namun, ditemukan anomali pada fitur register, dimana terdapat *'bug'* yang mempengaruhi validitas pengujian pada fitur tersebut. Setelah ditelusuri *Bug* tersebut dapat terjadi karena terdapat beberapa baris kode yang belum dimasukkan, sehingga *test case* tersebut menjadi Invalid. Meskipun begitu, keseluruhan sistem telah dikonstruksi dengan baik dan hanya membutuhkan sedikit perbaikan. Dari hasil penelitian tentang pengetesan sistem ini, dapat ditarik kesimpulan bahwa disarankan untuk menggunakan metode *white-box* pada tahap unit testing yang menargetkan bagian terkecil dari kode. Sebaliknya, metode *black-box* lebih sesuai untuk pengujian integrasi antar sub-sistem dan lebih optimal jika diterapkan pada tahap pengujian di luar unit testing.

DAFTAR PUSTAKA

[1] A. Frisdayanti, "PERANAN BRAINWARE DALAM SISTEM INFORMASI MANAJEMEN," *Jurnal Ekonomi Manajemen Sistem Informasi*, vol. 1, no. 1, pp. 60–69, Oct. 2019, doi: 10.31933/jemsi.v1i1.47.

[2] A. Jimi, "Rancang Bangun Sistem Informasi Desa Berbasis Website (Studi Kasus Desa Netpala)," *Jurnal Pendidikan Teknologi Informasi (JUKANTI)*, vol. 2, no. 1, pp. 1–7, May 2019, doi: 10.37792/jukanti.v2i1.17.

[3] A. F. Sallaby and I. Kanedi, "Perancangan Sistem Informasi Jadwal Dokter Menggunakan Framework Codeigniter," *JURNAL MEDIA INFOTAMA*, vol. 16, no. 1, Aug. 2020, doi: 10.37676/jmi.v16i1.1121.

[4] M. A. Chairani, "HAK IMUNITAS ADVOKAT TERKAIT MELECEHKAN AHLI," *JUSTITIA JURNAL HUKUM*, vol. 2, no. 1, Apr. 2018, doi: 10.30651/justitia.v2i1.1236.

[5] M. Ilyas, "Tinjauan Hukum Islam terhadap Musyawarah dalam Penyelesaian Sengketa Ekonomi Syariah," *Jurnal Al-Qadau: Peradilan dan Hukum Keluarga Islam*, vol. 5, no. 2, p. 227, Dec. 2018, doi: 10.24252/al-qadau.v5i2.7104.

[6] N. Muna, M. K. Anam, and K. F. Hana, "Teknik Penyelesaian Sengketa Melalui Non-Litigasi Di Perbankan Syariah," *Sahmiyya: Jurnal Ekonomi dan Bisnis*, vol. 1, no. 2, pp. 212–220, Nov. 2022.

[7] M. Manuhutu and J. Wattimena, "Perancangan Sistem Informasi Konsultasi Akademik Berbasis Website," *JURNAL SISTEM INFORMASI BISNIS*, vol. 9, no. 2, p. 149, Nov. 2019, doi: 10.21456/vol9iss2pp149-156.

[8] U. I. Kurnia, F. Rizki, and R. Y. Pratama, "Konsultasi Pendidikan Terkait Perbedaan Pendidikan Teknologi Informasi dan Teknik Informatika," *Jurnal Pengabdian Kepada Masyarakat Ungu (ABDI KE UNGU)*, vol. 4, no. 2, pp. 64–68, Aug. 2022.

[9] L. Siregar, "Review Pengujian Keamanan Perangkat Lunak dalam Software Development Life Cycle (SDLC)," *Journal of Applied Sciences, Electrical Engineering and Computer Technology*, vol. 1, no. 3, pp. 1–11, Dec. 2020.

[10] A. A. Wahid, "Analisis metode waterfall untuk pengembangan sistem informasi," *Jurnal Ilmu-ilmu Informatika dan Manajemen STMIK*, pp. 1–5, 2020.

[11] A. Uddin and A. Anand, "Importance of Software Testing in the Process of Software Development," *International Journal for Scientfic Research and Development*, vol. 6, no. 12, pp. 141–145, 2019.

[12] A. R Laisouw, S. Lutfi, and F. Tempola, "SISTEM PENDUKUNG KEPUTUSAN PEMBERIAN BANTUAN PROGRAM KELUARGA HARAPAN (PKH) PADA ORANG MISKIN DI KOTA TERNATE MENGGUNAKAN METODE AHP," *JIKO (Jurnal Informatika dan Komputer)*, vol. 2, no. 1, pp. 34–60, Apr. 2019, doi: 10.33387/jiko.v2i1.973.

[13] M. E. Khan and F. Khan, "Importance of software testing in software development life cycle," *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 2, pp. 120–123, Mar. 2014.

[14] D. Wintana, D. Pribadi, and M. Y. Nurhadi, "Analisis Perbandingan Efektifitas *White-box* Testing dan *Black-box* Testing," *Jurnal Larik: Ladang Artikel Ilmu Komputer*, vol. 2, no. 1, pp. 8–16, Jul. 2022, doi: 10.31294/larik.v2i1.1382.

[15] M. F. Londjo, "Implementasi *White-box* Testing Dengan Teknik Basis Path Pada Pengujian Form Login," *Jurnal Siliwangi Seri Sains dan Teknologi*, vol. 7, no. 2, pp. 35–40, Dec. 2021.

[16] R. Parlika, T. A. Nisaa, S. M. Ningrum, and B. A. Haque, "Studi Literatur Kekurangan dan Kelebihan Pengujian *Black-box*," *Teknomatika*, vol. 10, no. 2, pp. 131–140, Oct. 2020.

[17] I. M. A. O. Gunawan, G. Indrawan, and S. Sariyasa, "PENGEMBANGAN SISTEM INFORMASI KEMAJUAN AKADEMIK MENGGUNAKAN MODEL INCREMENTAL BERBASIS EVALUASI USABILITY DAN *WHITE-BOX* TESTING," *SINTECH (Science and Information Technology) Journal*, vol. 4, no. 1, pp. 67–78, Apr. 2021, doi: 10.31598/sintechjournal.v4i1.661.

[18] Y. D. Wijaya and M. W. Astuti, "PENGUJIAN *BLACK-BOX* SISTEM INFORMASI PENILAIAN KINERJA KARYAWAN PT INKA (PERSERO) BERBASIS EQUIVALENCE PARTITIONS," *Jurnal Digital Teknologi Informasi*, vol. 4, no. 1, p. 22, Mar. 2021, doi: 10.32502/digital.v4i1.3163.

[19] I. R. Dhaifullah, M. Muttanifudin H, A. Ananda Salsabila, and M. Ainul Yaqin, "Survei Teknik Pengujian Software," *Journal Automation Computer Information System*, vol. 2, no. 1, pp. 31–38, Jun. 2022, doi: 10.47134/jacis.v2i1.42.

[20] N. R. Dewi, "Mengenal *white-box* testing: Jenis dan 6 teknik yang digunakan," Ekrut. Accessed: Jan. 29, 2023. [Online]. Available: https://www.ekrut.com/media/*white-box*-testing-adalah

[21] A. O. Nurshanty, A. Saputra, F. R. Hardjanto, M. B. Franklyn, and D. Yudanegara, "Teknik Dalam *White-box* dan *Black-box* Testing," Binus University. Accessed: Jan. 29, 2023. [Online]. Available: https://socs.binus.ac.id/2020/07/02/teknik-dalam-*white-box*-dan-*black-box*-testing/