White Box Testing Menggunakan Teknik Basis Path pada Tiga Modul Sistem SMART-SIM PDAM Kalijati

Muthiah Wahyuliana[*1]

[1]Program Studi Sistem Informasi, Politeknik Negeri Subang

E-mail: muthiahwahyuliana@gmail.com[1]

Abstrak. Website memiliki peranan penting dalam mendukung aktivitas sehari-hari sehingga diperlukan pengujian untuk memastikan kualitas dan keandalannya. Salah satu metode yang digunakan adalah *White Box Testing* dengan pendekatan *Basis Path*, yang mampu menelusuri logika internal sistem serta mengevaluasi jalur eksekusi program. Penelitian ini menerapkan metode tersebut pada website SMART-SIM PDAM Cabang Kalijati yang berfungsi sebagai media pengaduan pelanggan, pengelolaan keluhan, serta penyampaian pengumuman. Tahapan pengujian meliputi pembuatan *flowgraph,* perhitungan *Cyclomatic Complexity*, identifikasi jalur *independen,* dan pelaksanaan *test case*. Hasil penelitian menunjukkan bahwa setiap jalur *independen* pada modul Tambah Pengaduan, Update Status Pengaduan, dan Tambah Pengumuman berhasil diuji sesuai skenario. Hal ini membuktikan bahwa sistem berjalan sesuai harapan, mampu menangani berbagai kondisi input, dan memiliki tingkat keandalan yang baik.

Kata kunci: Sistem Informasi; Sistem berbasis *Website;* PDAM; Pengaduan; pengujian

Abstract. *Websites play an important role in supporting daily activities, making testing essential to ensure their quality and reliability. One method applied is White Box Testing using the Basis Path approach, which traces the internal logic of the system and evaluates program execution paths. This study applies the method to the SMART-SIM PDAM Kalijati website, which serves as a platform for customer complaints, complaint management, and announcements. The testing stages include constructing flowgraphs, calculating Cyclomatic Complexity, identifying independent paths, and executing test cases. The results show that all independent paths in the Add Complaint, Update Complaint Status, and Add Announcement modules were successfully tested according to scenarios. This indicates that the system performs as expected, can handle various input conditions, and demonstrates good reliability.*

Keywords: *Information System; Website-based system; PDAM; Complaints; testing*

1. Pendahuluan

Perangkat lunak kini mudah diakses, baik berbayar maupun gratis, dengan berbagai layanan yang memudahkan aktivitas sehari-hari. Namun, tidak semua perangkat lunak memiliki kualitas baik karena kurangnya proses pengujian, sehingga masih sering ditemukan bug dan error [1]. Sebelum suatu aplikasi dapat diakses oleh pengguna, seorang *programmer* atau pengembang aplikasi harus melewati tahap rangkaian pengujian. Pengujian aplikasi bertujuan untuk memastikan tidak hanya kesesuaian dengan kebutuhan pengguna, tetapi juga kelancaran operasioal tanpa kendala yang mengganggu pengalaman pengguna. Salah satu metode pengujian yang dapat digunakan Adalah *white box testing* [2].

White box testing Adalah salah satu metode pengujian yang dikembangkan berdasarkan kode pada program dan syarat untuk pengujian yang akan menggunakan metode *white box testing* harus memiliki

pengetahuan akan kode serta penulisan kasus uji dengan parameter yang sesuai[3]. Pada white box testing terdapat beberapa pengujian yakni data flow testing, control flow testing, basis path, dan loop testing [4]. Ada beberapa penamaan lain dari *white box* yakni clear box, glass box, maupun open box. Namun, pada penelitian ini metode yang akan dipakai untuk white box testing adalah basis path [5].

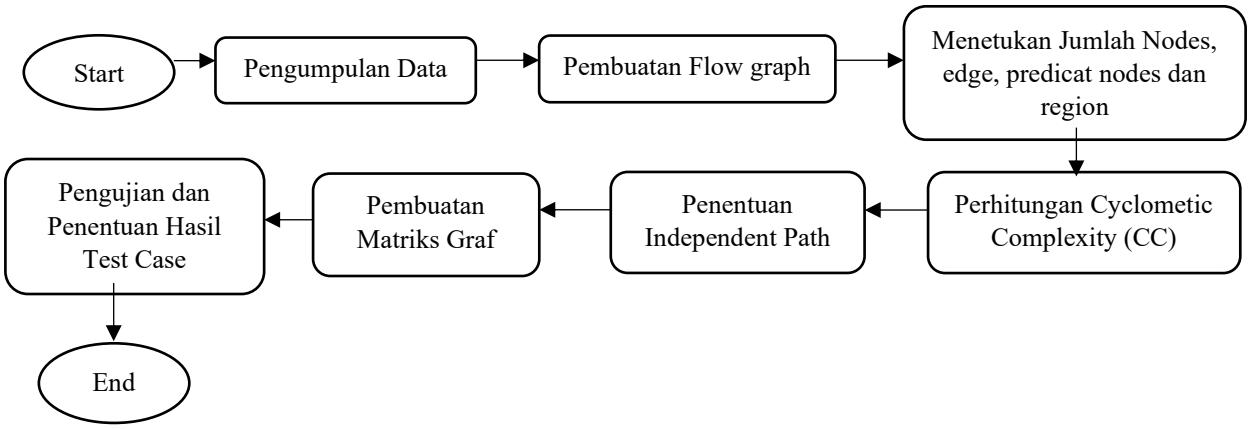
Teknik basis path adalah pendekatan pengujian yang berkonsentrasi pada pengujian rute eksekusi program[6]. Untuk memastikan bahwa setiapkomponen perangkat lunak diuji secara menyeluruh, penguji dapat mengembangkan skenario pengujian menyeluruh dengan mengetahui rute-rute ini [7]. Teknik basis path terdiri dari *flowgraph notation* yang merupakan gambaran dari alur kontrol program, *cyclomatic complexity* yang merupakan perhitungan untuk menentukan jumlah dari jalur pengujian, dan independent pathmerupakan penentuan jalur pengujian yang hanya boleh dilewati sekali [8].

Cyclomatic Complexity merupakan perangkat lunak metricyang menyediakan ukuran kuantitatif terhadap kompleksitas logis suatu program[9]. *Cyclomatic Complexity* membantu penguji untuk mengetahui seberapa rumit atau kompleks logika yang diterapkan pada suatu program. Semakin tinggi hasil perhitungan CC, maka semakin kompleks dan semakin rumit suatu kode untuk dilakukan pengujian. *Cyclomatic Complexity* dapat menentukan minimal test case pada jalur independen yang dihasilkan untuk dapat diuji[10]. Dalam perhitungan *Cyclomatic Complexity (CC)* dibutuhkan sebuah node yakni sebuah lingkaran yang berada pada flowgraph yang berfungsi untuk menggambarkan statement secara berurutan, edge atau notasi garis yakni sebuah panah yang menghubungkan node selanjutnya sehingga menggambarkan sebuah aliran kontrol pada program yang kita analisis, dan region yang merupakan sebuah area yang dibatasi oleh edge dan node [11].

PDAM atau Perusahaan Daerah Air Minum merupakan salah satu unit usaha milik daerah[12]. PDAM bergerak dalam distribusi air bersih bagi kepentingan masyarakat umum [13]. PDAM Cabang Kalijati Adalah salah satu cabang dari PDAM yang berpusat di Kota Subang. Perusahaan Daerah Air Minum Tirta Rangga Subang merupakan salah satu perusahaan yang bergerak di bidang penyediaan makan minum dan dikenal dengan Badan Pengelola Air Minum (PDAM). Badan Pengelola Air Minum (PDAM) yang dibentuk berdasarkan SK. Direktur Jendral Cipta Karya No. 126/KPTS/CK/1980, tertanggal 12 Desember. Jumlah karyawan PDAM Tirta Rangga Rangga Subang pada tahun 2020 berjumlah 161 karyawan [14].

2. Metode

Pengujian awal SMART-SIM PDAM dimulai dengan analisis kode program, kemudian dibuat flowgraph untuk memvisualisasikan alur sistem dan mengidentifikasi jalur *independen*. Selanjutnya dihitung *Cyclomatic Complexity* untuk memastikan seluruh jalur potensial tercakup, lalu ditentukan *independent path* sebagai dasar penyusunan *test case*[15]. Pengujian dijalankan sesuai skenario yang ditetapkan, hasilnya dievaluasi, dan disusun kesimpulan serta saran perbaikan guna meningkatkan kualitas sistem.



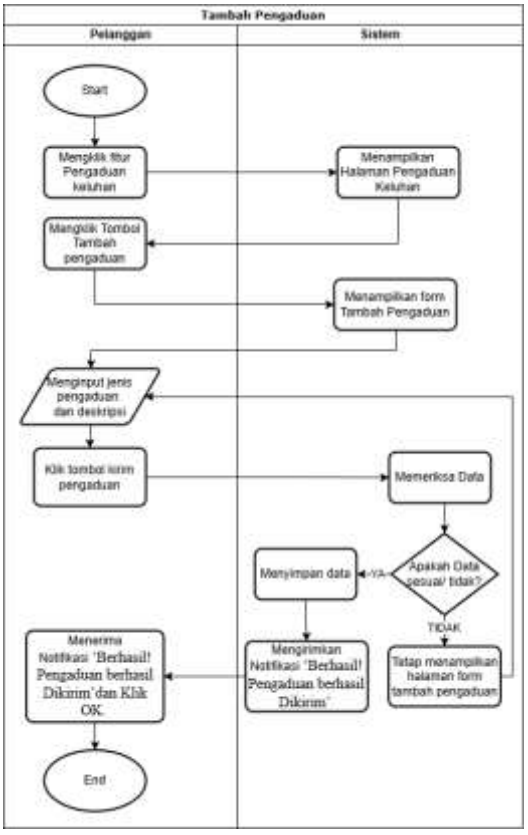
Gambar 1. Tahap Penelitian

2.1 Pengumpulan Data

Tahapan awal penelitian adalah pengumpulan data berupa *source code* dari sistem SMART-SIM PDAM Cabang Kalijati. Fokus pengujian dilakukan pada tiga modul utama, yaitu Tambah Pengaduan, Update Status Pengaduan, dan Tambah Pengumuman. Kode program dari masing-masing modul dianalisis untuk memetakan alur logika program dalam bentuk *flowgraph*. Selanjutnya dilakukan perhitungan *Cyclomatic Complexity* untuk menentukan jumlah jalur independen *(independent path)* yang akan diuji. Setiap jalur independen kemudian digunakan sebagai dasar penyusunan skenario *test case* dalam proses pengujian dalam metode *White Box Testing*.

2.2 Alur Tambah Pengaduann

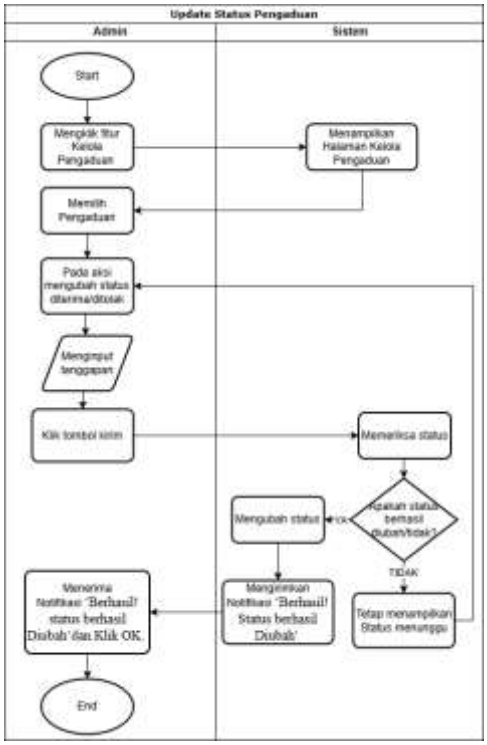
Salah satu form yang diuji pada website SMART SIM PDAM Kalijati adalah Tambah Pengaduan. Alurnya dimulai ketika pelanggan membuka fitur pengaduan keluhan, sistem menampilkan form, lalu pelanggan mengisi dan mengirim data. Sistem memeriksa validasi input: jika tidak sesuai, form ditampilkan kembali tanpa menyimpan data; jika sesuai, data disimpan ke database dan muncul notifikasi "Berhasil! Pengaduan berhasil dikirim". Secara keseluruhan, proses ini memastikan bahwa pengaduan hanya tersimpan apabila data valid, sehingga sistem bekerja secara aman dan akurat. *Flowchart* Tambah Pengaduan ditunjukkan pada Gambar 2.



Gambar 2. *Flowchart* Alur Tambah Pengaduan

2.3 Alur Update Status Pengaduan

Modul Update Status Pengaduan diuji dengan alur dimulai saat admin membuka fitur Kelola Pengaduan, memilih data yang akan diubah, lalu menentukan status baru (diterima atau ditolak) dan menambahkan tanggapan. Setelah admin mengirim perubahan, sistem menyimpan update status ke database. Secara keseluruhan, proses ini memastikan bahwa perubahan status pengaduan tercatat dengan benar dan aman. *Flowchart* Update Status Pengaduan ditunjukkan pada Gambar 3.



Gambar 3. *Flowchart* Alur Update Status Pengaduan

2.4 Alur Tambah Pengumuman

Modul Tambah Pengumuman diuji dengan alur ketika admin membuka fitur Kelola Pengumuman, memilih Tambah Pengumuman, lalu mengisi judul, deskripsi, dan gambar. Jika data valid, sistem menyimpan pengumuman dan menampilkan notifikasi sukses; jika tidak, form tetap ditampilkan tanpa menyimpan data. Proses ini memastikan hanya data valid yang tersimpan sehingga sistem bekerja aman dan akurat. *Flowchart* Tambah Pengumuman ditunjukkan pada Gambar 4.

Gambar 4. *Flowchart* Alur Tambah Pengumuman

2.5 Pembuatan Flowgraph
Pada tahap ini, peneliti membangun *flowgraph* berdasarkan alur kerja tambah pengaduan, update status pengaduan, tambah pengumuman. Flowgraph memberikan visualisasi terkait alur *sequence, if-else*, hingga *looping* yang terdapat pada sistem SMART-SIM PDAM Kalijati.

2.6 Penentuan jumlah nodes, edge, predicat nodes dan region
Pada tahap ini peneliti menentukan *jumlah nodes, edge, predicat nodes dan region* berdasarkan alur flowgraph setiap modul. Dimana tahap ini diperlukan saat perhitungan *Cyclomatic Complexity* (CC).

2.7 Perhitungan Cyclomatic Complexity (CC)
Metode yang digunakan untuk penentuan *independent path* dan jumlah region yaitu perhitungan *Cyclometic Complexity*. Perhitungan ini digunakan untuk memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. *Cyclometic Complexity* dapat dihitung dengan tiga cara, yaitu :

a. $V(G) =$ Jumlah Region
b. $V(G) = E$ (edges)- N (node) $+2$
c. $V(G) = P$ (Predicate Node) $+1$

2.8 Penentuan Independent Path
Tahapan ini dilakukan setelah mendapatkan hasil perhitungan *Cyclometic Complexity*. Jumlah yang dihasilkan dari perhitungan *Cyclometic Complexity* akan digunakan untuk menentukan *independent path* yang akan diuji. Penting untuk memastikan bahwa setiap *independent path* telah dilewati minimal 1 kali.

2.9 Pembuatan Matriks Graf
Matriks graf dalam pengujian digunakan untuk merepresentasikan hubungan antar node dalam sebuah *flowgraph* ke dalam bentuk matriks. Setiap baris dan kolom pada matriks menggambarkan *node*, sedangkan nilai elemen di dalamnya menunjukkan adanya hubungan *(edge)* dari satu *node* ke *node* lain.

2.10 Pengujian dan Penentuan Hasil Test Case
Pengujian pada modul tambah pengaduan, update status pengaduan, Tambah Pengumuman diuji sesuai dengan *independent path* yang telah ditentukan. Pengujian dilakukan dengan menyusun skenario pengujian dalam bentuk tabel yang mencakup ID, Rute, Rincian Pengujian, Hasil Pengujian, dan Keterangan.

Tabel 1. Template scenario pengujian

ID	Rute	Rincian Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Keterangan
…	…	…	…	…	…

Setiap ID dalam tabel menandakan identitas masing-masing *independent path*, sedangkan Rute mencerminkan *independent path* yang diuji pada modul tambah pengaduan, update status pengaduan, Tambah Pengumuman. Hasil yang diharapkan menandakan output yang seharusnya diperoleh dari setiap pengujian, sementara Hasil Pengujian mencatat hasil aktual dari pelaksanaan pengujian. Keterangan berisikan hasil dari pengujian dengan kategori “Berhasil” dan “Tidak Berhasil”. Dengan pendekatan ini, peneliti dapat melakukan evaluasi setiap modul yang diuji berdasarkan hasil pengujian yang telah dilakukan.

3. Hasil dan Pembahasan

3.1 Source Code
Pada bagian ini diketahui terdapat bagian-bagian *source code* dari modul Tambah Pengadun, Update Status Pengaduan, Tambah pengumuman sebagai berikut:
3.1.1 Source Code Tambah Pengaduan

Pada Tabel 2 di bawah ini diketahui bagian-bagian *source code function store PengaduanController.php* pada Tambah pengaduan.

Tabel 2. *Source Code* Tambah Pengaduan

Source Code

public function store(Request $request)
{
(1) Start - form disubmit
(2) Validasi Input
$request->validate([
'jenis_pengaduan' => 'required|string',
'deskripsi' => 'required|string',
]);
// Jika validasi gagal → otomatis kembali ke form dengan pesan error
(3) Input Tidak Valid
(4) Input Valid → simpan data
Pengaduan::create([
'user_id' => Auth::id(),
'tgl_pengaduan' => now(),
'jenis_pengaduan' => $request->jenis_pengaduan,
'deskripsi' => $request->deskripsi,
]);
(5) Redirect Sukses
Return redirect ()
->route('pengaduan.index')
->with ('success', 'Pengaduan berhasil dikirim.');
(6) End
}

3.1.2 Source Code Update Status Pengaduan

Pada Tabel 3 di bawah ini diketahui bagian-bagian *source code function store PengaduanController.php* pada Tambah pengaduan.

Tabel 3. *Source Code* Update Status Pengaduan

Source Code

public function respond(Request $request, $id)
{
(1) Start - User submit update status
(2) Validasi input
$request->validate([
'status' => 'required|string',
'tanggapan' => 'nullable|string'
]);
(3) Ambil data pengaduan berdasarkan ID
$pengaduan = Pengaduan::where('id_pengaduan', $id)->firstOrFail();
(4) Cek apakah status sudah final (Selesai/Ditolak)?
if (in_array($pengaduan->status, ['Selesai', 'Ditolak'])) {
(5) Status final → kembalikan error
return back()->with('error', 'Status sudah final dan tidak bisa diubah.');
}
(6) Status masih bisa diubah → update data
$pengaduan->update([
'status' => $request->status,
'tanggapan' => $request->tanggapan
]);
(7) Redirect sukses
return back()->with('success', 'Tanggapan berhasil dikirim.');
(8) End
}

3.1.3 Source Code Tambah Pengumuman

Pada Tabel 4 di bawah ini diketahui bagian-bagian *source code function store PengumumanController.php* pada Tambah pengumuman.

Tabel 4. *Source Code* Tambah Pengumuman

Source Code

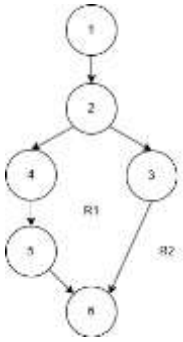
```
public function store (Request $request)
{
    (1) Start - User submit form tambah pengumuman
    (2) Validasi input (judul wajib, isi wajib, gambar opsional)
   $request->validate([
      'judul' => 'required',
      'isi' => 'required',
      'gambar' => 'nullable|image|mimes:jpg,jpeg,png,gif|max:2048',
   ]);
    (3) Ambil data admin berdasarkan user_id
   $admin = Admin::where('user_id', Auth::id())->first();
    (4) Cek apakah admin ditemukan?
   if (!$admin) {
      (5) Jika admin tidak ada → error dan kembali ke form
      return back()->withErrors(['error' => 'Akun admin tidak terdaftar.'])->withInput();
   }
   (6) Cek apakah ada file gambar?
   if ($request->hasFile('gambar')) {
      (7) Jika ada → simpan file gambar
      $gambar = $request->file('gambar');
      $nama_gambar = time().'_'.$gambar->getClientOriginalName();
      $gambar->move(public_path('gambar-pengumuman'), $nama_gambar);
   } else {
      (8) Jika tidak ada gambar → set null
      $nama_gambar = null;
   }
   (9) Simpan data pengumuman ke database
   Pengumuman::create([
      'judul' => $request->judul,
      'isi' => $request->isi,
      'gambar' => $nama_gambar,
      'id_admin' => $admin->id_admin,
   ]);
   (10) Redirect sukses
   return redirect()->route('pengumuman.index')->with ('success', 'Pengumuman berhasil disimpan!');
   (11) End
}
```

3.2 Flowgraph

Berdasarkan *source code* modul Tambah Pengaduan (*store* pada *PengaduanController.php*), Update Status Pengaduan (*respond* pada *PengaduanController.php*), dan Tambah Pengumuman (*store* pada *PengumumanController.php*), dibuat *flowgraph* untuk menggambarkan alur logika dari masing-masing fungsi tersebut.

3.2.1 Flowgraph Tambah Pengaduan

Dari *flowgraph* Gambar 5, terdapat jumlah *edge* (E) = 6, yang merupakan garis untuk menghubungkan node. Jumlah *node* (N) = 6 yang berbentuk lingkatan menggambarkan aktivitas, jumlah *predicate* (P)= 1 yang merupakan node bercabang, dari jumlah *region* (R) = 2 yang menandakan suratu area dalam *flowgraph*, yang dapat dilihat dengan simbol R1 dan R2 pada Gambar 5.



Gambar 5. *Flowgraph* Tambah Pengaduan

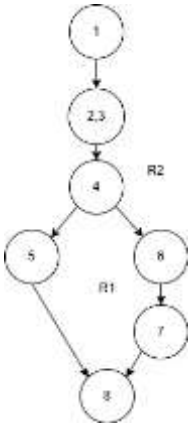
Pada Tabel 5 terdapat rincian *action* dari masing-masing *node*, yaitu:

Tabel 5. Deskripsi *Node*.

Node	Action
Node 1	Start - form disubmit
Node 2	Validasi input
Node 3	Input Tidak Valid
Node 4	Input Valid → simpan data
Node 5	Redirect Sukses
Node 6	End

3.2.2 Flowgraph Update Status Pengaduan

Dari *flowgraph* Gambar 6, terdapat jumlah *edge* (E) = 7, yang merupakan garis untuk menghubungkan node. Jumlah *node* (N) = 7 yang berbentuk lingkatan menggambarkan aktivitas, jumlah *predicate* (P)= 1 yang merupakan node bercabang, dari jumlah *region* (R) = 2 yang menandakan suratu area dalam *flowgraph*, yang dapat dilihat dengan simbol R1 hingga R2 pada Gambar 6.



Gambar 6. *Flowgraph* Update status pengaduan

Pada Tabel 6 terdapat rincian *action* dari masing-masing *node*, yaitu:

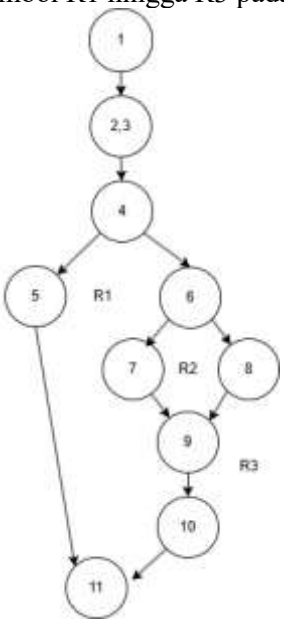
Tabel 6. Deskripsi *Node*.

Node	Action
Node 1 (1)	Start - User submit update status
Node 2 (2,3)	Validasi input → Ambil data pengaduan berdasarkan ID
Node 3 (4)	Cek apakah status sudah final (Selesai/Ditolak)?
Node 4 (5)	Status final → kembalikan error
Node 5 (6)	Status masih bisa diubah → update data

Node 6 (7)	Redirect sukses
Node 7 (8)	End

3.2.3 Flowgraph Tambah Pengumuman

Dari *flowgraph* Gambar 7, terdapat jumlah *edge* (E) = 11, yang merupakan garis untuk menghubungkan node. Jumlah *node* (N) = 10 yang berbentuk lingkatan menggambarkan aktivitas, jumlah *predicate* (P)= 2 yang merupakan node bercabang, dari jumlah *region* (R) = 3 yang menandakan suratu area dalam *flowgraph*, yang dapat dilihat dengan simbol R1 hingga R3 pada Gambar 7.



Gambar 7. *Flowgraph* Tambah Pengumuman

Pada Tabel 7 terdapat rincian *action* dari masing-masing *node*, yaitu:

Tabel 7. Deskripsi *Node*.

Node	Action
Node 1 (1)	Start - User submit form tambah pengumuman
Node 2 (2,3)	Validasi input (judul wajib, isi wajib, gambar opsional) → Ambil data admin berdasarkan user_id
Node 3 (4)	Cek apakah admin ditemukan?
Node 4 (5)	Jika admin tidak ada → error dan kembali ke form
Node 5 (6)	Cek apakah ada file gambar?
Node 6 (7)	Jika ada → simpan file gambar
Node 7 (8)	Jika tidak ada gambar → set null
Node 8 (9)	Simpan data pengumuman ke database
Node 9 (10)	Redirect sukses
Node 10 (11)	End

3.3 Cyclomatic Complexity

3.3.1 Cyclomatic Complexity Tambah Pengaduan

Merujuk pada Gambar 5 telah diketahui jumlah *edge*, *node*, *predicate*, dan *region*, maka selanjutnya yaitu perhitungan *Complexity Cyclomatic* yang menghasilkan perhitungan seperti berikut:

Tabel 8. Perhitungan *Complexity Cyclomatic*

Keterangan	Jumlah
Node	6

Edge	6
Predikat	1
V(G): E – N + 2	6 – 6 = 0 +2 = 2
V(G): P + 1	1 +1 = 2

Tabel 8 menyajikan perhitungan *Complexity Cyclomatic* berdasarkan jumlah *node, edge,* dan predikat dalam suatu program. Dengan 6 node, 6 edge, dan 1 predikat, *Complexity Cyclomatic* dihitung menggunakan rumus V(G) = E - N + 2, yang menghasilkan nilai 2. Selanjutnya, nilai ini juga dapat dihitung dengan rumus alternatif V(G) = P + 1, dan hasilnya adalah 2.

3.3.2 Cyclomatic Complexity Update status pengaduan

Merujuk pada Gambar 6 telah diketahui jumlah *edge*, *node*, *predicate*, dan *region*, maka selanjutnya yaitu perhitungan *Complexity Cyclomatic* yang menghasilkan perhitungan seperti berikut:

Tabel 9. Perhitungan *Complexity Cyclomatic*

Keterangan	Jumlah
Node	7
Edge	7
Predikat	1
V(G): E – N + 2	7 – 7 = 0 +2 = 2
V(G): P + 1	1 +1 = 2

Tabel 9 menyajikan perhitungan *Complexity Cyclomatic* berdasarkan jumlah *node, edge*, dan predikat dalam suatu program. Dengan 7 node, 7 edge, dan 1 predikat, *Complexity Cyclomatic* dihitung menggunakan rumus V(G) = E - N + 2, yang menghasilkan nilai 2. Selanjutnya, nilai ini juga dapat dihitung dengan rumus alternatif V(G) = P + 1, dan hasilnya adalah 2.

3.3.3 Cyclomatic Complexity Tambah Pengumuman

Merujuk pada Gambar 7 telah diketahui jumlah *edge*, *node*, *predicate*, dan *region*, maka selanjutnya yaitu perhitungan *Complexity Cyclomatic* yang menghasilkan perhitungan seperti berikut:

Tabel 10. Perhitungan *Complexity Cyclomatic*

Keterangan	Jumlah
Node	10
Edge	11
Predikat	2
V(G): E – N + 2	11– 10 = 1 +2 = 3
V(G): P + 1	2 +1 = 3

Tabel 9 menyajikan perhitungan *Complexity Cyclomatic* berdasarkan jumlah *node, edge*, dan predikat dalam suatu program. Dengan 10 node, 11 edge, dan 2 predikat, *Complexity Cyclomatic* dihitung menggunakan rumus V(G) = E - N + 2, yang menghasilkan nilai 3. Selanjutnya, nilai ini juga dapat dihitung dengan rumus alternatif V(G) = P + 1, dan hasilnya adalah 3.

3.4 Independent Path

3.4.1 Independent Path Tambah Pengaduan

Berdasarkan perhitungan *Complexity Cyclomatic* didapatkan hasil *independent path* untuk modul tambah pengaduan yaitu sebanyak 2 dengan jalur independent path, seperti pada Tabel 11 berikut:

Tabel 11. Penentuan *Independent Path*

Path	Flow
Path 1	1-2-3-6
Path 2	1-2-4-5-6

Pada *Path* 1, ketika pelanggan mengisi form dengan data valid, sistem melewati validasi, menyimpan data ke database, lalu menampilkan pesan sukses. Sedangkan pada *Path* 2, jika form tidak diisi dengan benar, sistem mendeteksi *error* pada validasi, menolak penyimpanan, menampilkan kembali form dengan pesan *error*, dan alur berakhir tanpa perubahan pada database.

3.4.2 Independent Path Update Status Pengaduan

Berdasarkan perhitungan *Complexity Cyclomatic* didapatkan hasil *independent path* untuk modul tambah pengaduan yaitu sebanyak 2 dengan jalur *independent path*, seperti pada Tabel 12 berikut:

Tabel 12. Penentuan *Independent Path*

Path	Flow
Path 1	1-2-3-4-6-7-8
Path 2	1-2-3-4-5-8

Pada *Path* 1, menggambarkan kondisi normal ketika admin atau teknisi mengirimkan update status valid pada pengaduan yang belum final, sehingga sistem memproses perubahan status sesuai input dan menampilkan pesan sukses. Sedangkan pada *Path* 2, jika pengaduan sudah berstatus final ("Selesai" atau "Ditolak"), sistem menolak perubahan, tidak melakukan update pada database, dan mengembalikan pesan *error*.

3.4.3 Independent Path Tambah Pengumuman

Berdasarkan perhitungan *Complexity Cyclomatic* didapatkan hasil *independent path* untuk modul tambah pengaduan yaitu sebanyak 3 dengan jalur *independent path*, seperti pada Tabel 12 berikut:

Tabel 11. Penentuan *Independent Path*

Path	Flow
Path 1	1-2-3-4-6-8-9-10-11
Path 2	1-2-3-4-6-7-9-10-11
Path 3	1-2-3-4-5-11

Pada *Path* 1, jika input valid dan admin terdaftar namun tanpa gambar, sistem menyimpan pengumuman dengan gambar kosong dan menampilkan pesan sukses. *Path* 2 terjadi saat input valid, admin terdaftar, serta ada gambar yang diunggah, sehingga file disimpan di folder publik, data lengkap tersimpan ke database, dan sistem menampilkan pesan sukses. Sementara itu, *Path* 3 terjadi ketika input valid tetapi admin tidak terdaftar, sehingga proses langsung dihentikan dan sistem menampilkan pesan *error*.

3.5 Matriks Graph

3.5.1 Matriks Graph Tambah Pengaduan

Node-node yang saling berkomunikasi sesuai dengan kode program diberikan simbol 1, dan kemudian dilakukan penjumlahan untuk setiap node edge nya dengan rumus N(E)-1, dan dijumlahkan hasilnya. Hasil dari perhitungan yaitu sebesar 2, Proses perhitungan diperlihatkan pada tabel 12.

Tabel 12. Proses perhitungan *Graph matriks* modul Tambah pengaduan

Node	Connected Node						N(E)-1
	1	2	3	4	5	6	
1		1					1-1 =0
2			1	1			2-1 =1
3						1	1-1 =0
4					1		1-1 =0
5						1	1-1 =0
6							0
			Jumlah + 1				1+1 =2

Tabel 12 menunjukkan Matriks graf modul Tambah Pengaduan adanya 6 node dan 2 jalur independen. Matriks ini dipakai untuk memverifikasi perhitungan kompleksitas logika program, memastikan bahwa seluruh jalur eksekusi (*valid* maupun *invalid*) sudah tercover dalam pengujian *White Box.*

3.5.2 Matriks Graph Update Status Pengaduan

Node-node yang saling berkomunikasi sesuai dengan kode program diberikan simbol 1, dan kemudian dilakukan penjumlahan untuk setiap *node* dan *edge* nya dengan rumus N(E)-1, dan dijumlahkan hasilnya. Hasil dari perhitungan yaitu sebesar 2, Proses perhitungan diperlihatkan pada tabel 13.

Tabel 13. Proses perhitungan *Graph matriks* modul Update Status Pengaduan

Node	Connected Node								N(E)-1
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	
1		1							1-1 =0
2			1						1-1 =0
3				1					1-1 =0
4					1	1			2-1 =1
5								1	1-1 =0
6							1		1-1=0
7								1	1-1=0
8									0
	Jumlah + 1								1+1 =2

Tabel 13 menunjukkan Matriks graf modul Update Status Pengaduan adanya 8 node dan 2 jalur independen. Matriks ini dipakai untuk memverifikasi perhitungan kompleksitas logika program, memastikan bahwa seluruh jalur eksekusi (*valid* maupun *invalid*) sudah tercover dalam pengujian *White Box.*

3.5.3 Matriks Graph Tambah Pengumuman

Node-node yang saling berkomunikasi sesuai dengan kode program diberikan simbol 1, dan kemudian dilakukan penjumlahan untuk setiap node edge nya dengan rumus N(E)-1, dan dijumlahkan hasilnya. Hasil dari perhitungan yaitu sebesar 3, Proses perhitungan diperlihatkan pada tabel 14.

Tabel 14. Proses perhitungan *Graph matriks* modul Tambah Pengumuman

Node	Connected Node											N(E)-1
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	
1		1										1-1 =0
2			1									1-1 =0
3				1								1-1 =0
4					1	1						2-1 =1
5											1	1-1 =0
6							1	1				2-1=1
7									1			1-1=0
8									1			1-1=0
9										1		1-1=0
10											1	1-1=0
11												0
	Jumlah + 1											2+1 =3

Tabel 14 menunjukkan Matriks graf modul Tambah pengaduan adanya 11 node dan 3 jalur independen. Matriks ini dipakai untuk memverifikasi perhitungan kompleksitas logika program, memastikan bahwa seluruh jalur eksekusi (*valid* maupun *invalid*) sudah tercover dalam pengujian *White Box*.

3.6 Hasil Pengujian Test Case

Setelah diketahui hasil perhitungan *Complexity Cyclomatic*, penentuan *independent path*, dan *Matriks Graph* maka selanjutnya dilakukan pengujian *test case*. Hasil yang didapatkan dari pengujian *test case* beberapa modul:

3.6.1 Hasil Pengujian Test Case Tambah Pengaduan

Tabel 15. Hasil Pengujian Tambah Pengaduan

ID	Rute	Rincian Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Keterangan
01	1-2-3-6	Pelanggan sudah mengisi form tidak lengkap dan form disubmit, sistem melakukan validasi, Input Hasil Invalid	Sistem gagal validasi, kembali ke form, tampil pesan error validasi, data tidak tersimpan	Validasi gagal, pesan error tampil, data tidak tersimpan	Berhasil
02	1-2-4-5-6	Pelanggan sudah mengisi form dengan lengkap dan form disubmit, sistem melakukan validasi, Input Valid	Sistem menyimpan data pengaduan, redirect ke index dengan pesan sukses "Pengaduan berhasil dikirim."	Data tersimpan, redirect sukses, pesan tampil sesuai.	Berhasil

Berdasarkan pengujian pada Tabel 15, seluruh 2 *independent path* pada form tambah pengaduan dinyatakan "Berhasil". Hasil ini menunjukkan sistem berjalan sesuai yang diharapkan, mampu merespons *input* dengan tepat, dan memastikan setiap jalur diuji secara efektif.

3.6.2 Hasil Pengujian Test Case Update Status Pengaduan

Tabel 16. Hasil Pengujian Update Status Pengujian

ID	Rute	Rincian Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Keterangan
01	1-2-3-4-6-7-8	Admin mengisi form dengan status baru selain final (mis. dari Menunggu → Selesai) dan menambahkan tanggapan.	Sistem melewati validasi, menemukan data, status bukan final dan update DB berhasil, redirect dengan pesan sukses "Tanggapan berhasil dikirim."	Data status & tanggapan terupdate sesuai input, pesan sukses yang sesuai.	Berhasil
02	1-2-3-4-5-8	Admin mencoba mengubah status pengaduan yang sudah final (Selesai atau Ditolak).	Sistem melewati validasi, menemukan data, status terdeteksi final → tidak update DB, kembali dengan pesan error "Status sudah final dan tidak bisa diubah."	Respon sistem Data tetap, pesan error.	Berhasil

Berdasarkan pengujian pada Tabel 16, seluruh 4 *independent path* modul update status pengaduan dinyatakan "Berhasil". Hasil ini menunjukkan sistem mampu mengelola proses dengan baik, merespons *input* secara tepat, serta memastikan setiap jalur diuji secara efektif.

3.6.3 Hasil Pengujian Test Case Tambah Pengumuman

Tabel 17. Hasil Pengujian Tambah Pengumuman

ID	Rute	Rincian Pengujian	Hasil Yang Diharapkan	Hasil Pengujian	Keterangan
01	1-2-3-4-6-8-9-10-11	Input judul & isi valid, admin terdaftar, tidak ada file gambar	Data pengumuman tersimpan di DB tanpa gambar, redirect ke index, pesan sukses "Pengumuman berhasil disimpan!"	Data tersimpan, redirect sukses, pesan sesuai	Berhasil
02	1-2-3-4-6-7-9-10-11	Input judul & isi valid, admin terdaftar, dengan file gambar valid (jpg/png)	Data pengumuman tersimpan di DB dengan nama file gambar, file tersimpan di folder, pesan sukses tampil	Data tersimpan lengkap dengan gambar, pesan sukses sesuai	Berhasil
03	1-2-3-4-5-11	Input valid tapi akun admin tidak terdaftar	Sistem mengembalikan error "Akun admin tidak terdaftar.", tidak ada data tersimpan di DB	Pesan error muncul, data tidak tersimpan	Berhasil

Berdasarkan hasil pengujian pada Tabel 17, seluruh 3 independent path pada form tambah pengumuman dinyatakan "Berhasil". Test case yang mencakup skenario *valid* maupun *invalid* menunjukkan sistem mampu mengelola proses dengan baik, memberikan respons sesuai *input*, dan memastikan setiap jalur dapat diuji secara efektif.

3.7 Analisis Hasil Pengujian

Berdasarkan hasil pengujian White Box menggunakan teknik Basis Path Testing pada tiga modul sistem SMART-SIM PDAM Kalijati, yaitu modul Tambah Pengaduan, Update Status Pengaduan, dan Tambah Pengumuman, seluruh independent path yang diperoleh dari perhitungan Cyclomatic Complexity telah berhasil diuji menggunakan test case yang telah disusun. Setiap jalur eksekusi menghasilkan output yang sesuai dengan hasil yang diharapkan sehingga tidak ditemukan kesalahan logika (bug) pada proses eksekusi program.

Selain itu, analisis coverage menunjukkan bahwa seluruh bagian logika program telah diuji. Pada statement coverage, seluruh pernyataan dalam kode program telah dieksekusi minimal satu kali selama proses pengujian. Sementara pada branch coverage, seluruh percabangan kondisi pada ketiga modul telah diuji baik pada kondisi benar maupun salah melalui skenario pengujian yang berbeda.

Dengan demikian, dapat disimpulkan bahwa seluruh jalur logika program pada modul yang diuji telah diverifikasi dengan baik dan sistem SMART-SIM PDAM Kalijati mampu menangani berbagai kondisi input tanpa menghasilkan kesalahan logika pada proses eksekusi program.

4. Kesimpulan

Berdasarkan hasil pengujian *White Box* dengan teknik *Basis Path* pada *website* SMART-SIM PDAM Cabang Kalijati, diperoleh hasil perhitungan *Cyclomatic Complexity* pada masing-masing modul. Modul Tambah Pengaduan memiliki 2 *independent path*, modul Update Status Pengaduan memiliki 2 *independent path*, dan modul Tambah Pengumuman memiliki 3 *independent path*. Setiap jalur independen tersebut diuji dengan test case yang telah disusun, dan seluruhnya menghasilkan keluaran sesuai dengan yang diharapkan (Berhasil).

Dari hasil ini dapat disimpulkan bahwa pengujian berjalan dengan baik, dan sistem SMART-SIM PDAM Cabang Kalijati tidak mengalami kesalahan logika pada ketiga modul yang diuji. Penggunaan metode Basis Path Testing terbukti efektif karena mampu memverifikasi jalur-jalur eksekusi program secara menyeluruh. Dengan demikian, sistem SMART-SIM PDAM Kalijati dapat dianggap layak digunakan dengan tingkat keandalan yang tinggi, terutama dalam mendukung layanan pengaduan dan penyampaian informasi kepada pelanggan.

5. Ucapan Terima Kasih

Penulis menyampaikan rasa terima kasih kepada Politeknik Negeri Subang, khususnya Program Studi D-III Sistem Informasi, yang telah memberikan dukungan dan bimbingan sehingga penelitian ini dapat diselesaikan dengan baik. Ucapan terima kasih juga diberikan kepada PDAM Tirta Rangga Cabang Kalijati yang telah memberikan kesempatan serta menyediakan data yang dibutuhkan dalam proses penelitian dan pengujian sistem.

Tidak lupa, penulis menghaturkan rasa hormat dan terima kasih yang mendalam kepada kedua orang tua serta keluarga tercinta atas doa, dukungan, dan motivasi yang selalu diberikan. Penulis juga berterima kasih kepada dosen pembimbing dan rekan-rekan mahasiswa/i yang turut membantu dan memberikan masukan berharga selama proses penyusunan artikel ini.

Referensi

[1] J. Sie L Bryan, I. Musdar Alwiah, and S. Bahri, "Pengujian White Box Testing Terhadap Website Room Menggunakan Teknik Basis Path," *J. KHARISMA Tech*, vol. 17, no. 2, pp. 45–57, 2022.

[2] M. F. Londjo, "Implementasi White Box Testing Dengan Teknik Basis Path Pada Pengujian Form Login," *J. Siliwaangi*, vol. 7, no. 2, pp. 35–40, 2021.

[3] Safriza and H. Rohayani, "Implementation of Path Testing in White Box Testing for the National Public Transportation Driver Behavior and Competency Assessment System," *LJISSEN-AI. Lovelace J. Inf. Syst. Secur. Educ. Netw. Artif. Intellingence*, pp. 1–16, 2025.

[4] Y. J. Solissa, F. Putra, A. N. Putri, and S. R. C. Nursari, "Pengujian White Box Berbasis Path pada Form Daftar Jobstreet.co.id," *KONSTELASI Konvergensi Teknol. dan Sist. Inf.*, vol. 3, no. 2, pp. 353–362, 2023, doi: 10.24002/konstelasi.v3i2.8287.

[5] A. C. Praniffa, A. Syahri, F. Sandes, U. Fariha, and Q. A. Giansyah, "Pengujian Black Box Dan White Box Sistem Informasi Parkir Berbasis Web Black Box And White Box Testing Of Web-Based Parking Information System," *J. Test. Dan Implementasi Sist. Inf.*, vol. 1, no. 1, pp. 1–16, 2023.

[6] C. T. Pratala, E. M. Asyer, I. Prayudi, and A. Saifudin, "Pengujian White Box pada Aplikasi Cash Flow Berbasis Android Menggunakan Teknik Basis Path," *J. Inform. Univ. Pamulang*, vol. 5, no. 2, pp. 111–119, 2020.

[7] R. I. Ndaumanu, "Pengujian Sistem Informasi Perpustakaan Berbasis Website dengan Basis Path Testing," *Justek J. Sains dan Teknol.*, vol. 6, no. 1, pp. 123–134, 2023.

[8] S. Kalagara, "Cyclomatic Complexity in Software Development," *Fram. J. Eng. Res. Technol.*, vol. 8, no. 16, pp. 46–47, 2020.

[9] V. Siahaan, H. Ginting, and M. Amri, "Cyclomatic Complexity and Maintainability in Modern Software: A Systematic Review," *J. Data Sci. Technol. Comput. Sci.*, vol. 5, no. 1, pp. 8–16, 2025.

[10] H. Rafli *et al.*, "Penerapan Whitebox Testing pada pengujian sistem menggunakan teknik Basis Path," *J. Inf. Syst. Informatics Eng.*, vol. 8, no. 1, pp. 101–111, 2024, [Online]. Available: https://doi.org/10.35145/joisie.v8i1.4229

[11] A. Z. Pitoyo, G. Djuwadi, and P. Yudho, "Nilai Cyclomatic Complexity Konflik Kerja terhadap Pengaruh Pimpinan dan Beban Kerja Karyawan dengan Menggunakan Model Reflektif PLS SEM," *J. Pendidik.*, vol. 3, no. 5, pp. 648–655, 2018.

[12] M. Fahri, D. Ratmananda, M. Rizki Zulfikar, R. S. Putri, and S. R. Natasia, "Analisis Aspek Usability pada Website PDAM XYZ Kota XYZ dengan Metode WEBUSE," *J. Inform. Univ. Pamulang*, vol. 6, no. 2, pp. 358–367, 2021.

[13] A. E. Fajriaty, P. Yuliantoro, M. A. Amanaf, and N. A. Zen, "Prototipe Sistem Monitoring Pemakaian Air PDAM untuk Rumah Tangga Berbasis Aplikasi Android," *Sci. TECH J. Ilmu*

Pengetah. dan Teknol. Vol., vol. 8, no. 2, pp. 124–135, 2022.
[14] P. T. R. Subang, "BAB I," pp. 1–7, 2020.
[15] M. H. Izzaturrahim, M. C. Saputra, and A. Pinandito, "Pengembangan Sistem Informasi Monitoring Kinerja Mesin Gilingan Berbasis Android Studi Kasus PG. Krebet Baru II, Malang," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 5, pp. 2016–2024, 2017.