Pengujian *White Box* Berbasis Path pada Form Daftar Jobstreet.co.id

Y J Solissa[1], F Putra[2], A N Putri*[3], S R C Nursari[4]

[1,2,3,4]Program Studi Teknik Informatika, Universitas Pancasila

E-mail: yosuasollisa@gmail.com[1], farhanputra123410@gmail.com[2], adelianurlinap@gmail.com[3], sri.rezeki.candra.n@univpancasila.ac.id[4]

Abstrak. Aplikasi kini memegang peranan penting dalam kehidupan sehari-hari. Dengan menggunakan aplikasi, berbagai tugas menjadi lebih sederhana. Oleh karena itu, menguji aplikasi sebelum digunakan oleh pengguna sangat penting untuk memverifikasi keandalan dan keamanannya. Pengujian *White Box* yang memanfaatkan pendekatan teknik *Basis Path* merupakan salah satu metodologi pengujian yang dapat digunakan pada penelitian ini. Teknik Basis Path memungkinkan penguji melihat dan memahami kode program secara langsung, sehingga mengurangi risiko dan meningkatkan keandalan aplikasi. Pengujian *White Box* dengan teknik Basis Path dilakukan pada situs web JobStreet.co.id yang merupakan platform untuk mencari dan merekrut pekerjaan. Situs ini sangat mementingkan interaksi antara pencari kerja dan dunia usaha. Pengujian *White Box* dipilih sebagai metode untuk memastikan keamanan, keandalan, dan kualitas kode di balik fungsi penting di situs web. Pengujian dibagi menjadi beberapa tahap dimulai dari pengumpulan data, pembuatan *Flow Graph*, perhitungan *Cyclomatic Complexity*, penentuan *Independent Path*, hingga pengujian *Test Case*. Hasil yang diperoleh dari pengujian *White Box* dengan teknik Basis Path pada situs web JobStreet.co.id yaitu terdapat 4 *Independent Path* yang kemudian diuji menggunakan scenario *test case*. Pengujian berlangsung dengan baik, dan sistem *form* pendaftaran Jobstreet.co.id tidak menunjukkan adanya kesalahan logika pada fungsi daftar.

Kata kunci: pengujian; *White Box*; Basis Path; web; aplikasi

Abstract. *Applications now play a crucial role in daily life. By using applications, various tasks become simpler. Therefore, testing applications before they are used by users is essential to verify their reliability and security. White Box testing using the Basis Path technique is one testing methodology that can be used in this research. The Basis Path technique allows testers to directly view and understand the program code, reducing risks and enhancing application reliability. White Box testing with the Basis Path technique was conducted on the JobStreet.co.id website. JobStreet.co.id is a platform for job seekers and employers, emphasizing the interaction between job seekers and the business world. White Box testing was chosen as a method to ensure the security, reliability, and quality of the code behind the website's essential functions. The testing process is divided into several stages, starting from data collection, creating a Flow Graph, calculating Cyclomatic Complexity, determining Independent Paths, and conducting Test Case testing. The results obtained from White Box testing using the Basis Path technique on the JobStreet.co.id website reveal the presence of 4 Independent Paths, which were subsequently tested using test cases. The testing proceeded smoothly, and the*

registration form system of Jobstreet.co.id did not experience any logical errors in the registration function.

Keywords: *testing; White Box; Basis Path; web; application*

1. Pendahuluan

Pada era modern saat ini, berbagai macam aplikasi dapat menunjang aktivitas manusia. Banyak aspek kehidupan menjadi lebih sederhana dengan penggunaan aplikasi, di mana aplikasi tersebut dapat menggantikan prosedur manual atau konvensional dengan cara yang lebih efektif. Oleh karena itu, mengevaluasi program sebelum digunakan merupakan langkah penting dalam membatasi kemungkinan kerugian dan menjamin keandalan dari aplikasi [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 operasional 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 penguji 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* [3]. 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* [4].

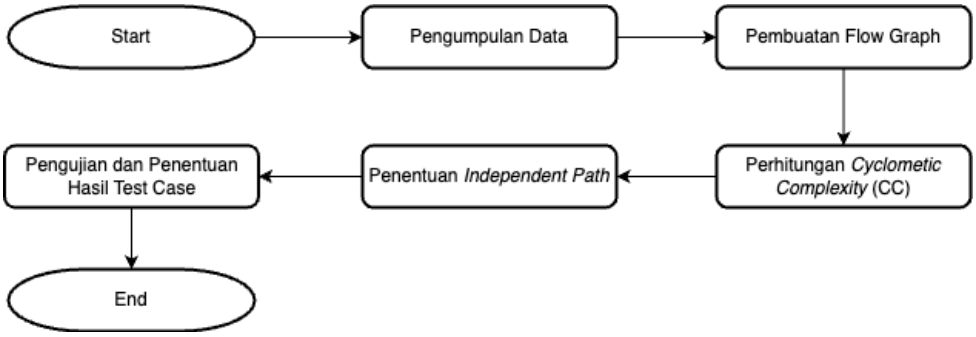
White box dengan *basis path* lebih sesuai untuk menguji perangkat lunak dibandingkan dengan menggunakan metode yang lainnya karena *basis path* akan menghasilkan sejumlah *test case* dengan cakupan pengujian yang lebih menyeluruh dibandingkan dengan teknik lainnya [5]. Teknik *basis path* terdiri dari *flowgraph notation* yang merupakan gambaran dari alur kontrol program, *cyclomatic complexity* yang merupakan perhitungan untuk menentukan jumlah dari jalur pengujian, dan *independent path* merupakan penentuan jalur pengujian yang hanya boleh dilewati sekali [6]. Teknik *basis path* adalah pendekatan pengujian yang berkonsentrasi pada pengujian rute eksekusi program. Untuk memastikan bahwa setiap komponen perangkat lunak diuji secara menyeluruh, penguji dapat mengembangkan skenario pengujian menyeluruh dengan mengetahui rute-rute ini [7]. Pengujian *white box* mungkin lebih efektif memverifikasi keamanan, ketergantungan, dan efisiensi kode komputer dengan menggabungkan pendekatan *basis path* [8]. Untuk dapat menerapkan pengujian *white box* dengan *basis path*, berikut adalah langkah-langkah yang perlu dilakukan 1). menggambarkan *flow* atau aliran dari proses program, 2) menentukan *Cyclomatic Complexity*, 3) menentukan jalur dasar yang sesuai dari jumlah *Cyclomatic Complexity*, dan 4) mendefinisikan kasus-kasus uji untuk setiap jalur dasar yang telah ditentukan.

Cyclomatic Complexity merupakan perangkat lunak *metric* yang menyediakan ukuran kuantitatif terhadap kompleksitas logis suatu program. *Cyclomatic Complexity* membantu penguji untuk mengetahui seberapa rumit atau kompleks logika yang diterapkan pada suatu program [9]. Semakin tinggi hasil perhitungan CC, maka semakin kompleks dan semakin rumit suatu kode untuk dilakukan pengujian [10]. *Cyclomatic Complexity* dapat menentukan minimal *test case* pada jalur independen yang dihasilkan untuk dapat diuji. 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][12].

JobStreet.co.id merupakan situs untuk memfasilitasi hubungan antara dunia usaha dan pencari kerja sebagai platform Indonesia untuk perekrutan dan pencarian kerja. Untuk dapat memastikan bahwa situs web JobStreet.co.id berfungsi dengan baik, memberikan efektivitas, dan memberikan kemudahan kepada pengunjungnya, dilakukanlah pengujian *White Box* yang berfokus pada pendekatan *Basis Path*.

2. Metode

Pengujian tahap awal Jobstreet.co.id dapat dilihat pada Gambar 1 yang dimulai dengan pengumpulan data yang melibatkan analisis dokumen dengan observasi *source code* terhadap sistem. Pembuatan *flowgraph* menjadi langkah penting dalam memvisualisasikan alur kerja sistem dan memudahkan identifikasi jalur-jalur independen [13]. Dilanjutkan dengan perhitungan *Cyclomatic Complexity* (CC), kompleksitas kode dapat diukur untuk memastikan pengujian mencakup seluruh jalur yang berpotensi [14]. Menentukan *independent path* menjadi landasan untuk menentukan *sequence* pengujian [15]. Selanjutnya, penentuan *test case* dilakukan untuk merinci skenario pengujian yang mencakup berbagai aspek fungsional dari sistem tersebut. Tahap pengujian dilakukan sesuai *test case* yang telah ditetapkan, dan hasilnya dievaluasi dengan cermat. Setelah itu, kesimpulan dan saran diajukan berdasarkan hasil temuan dalam pengujian untuk meningkatkan kualitas sistem. Dengan demikian, penelitian ini diakhiri dengan tahap *end* setelah mengimplementasikan saran perbaikan yang relevan.



Gambar 1. Tahap Penelitian.

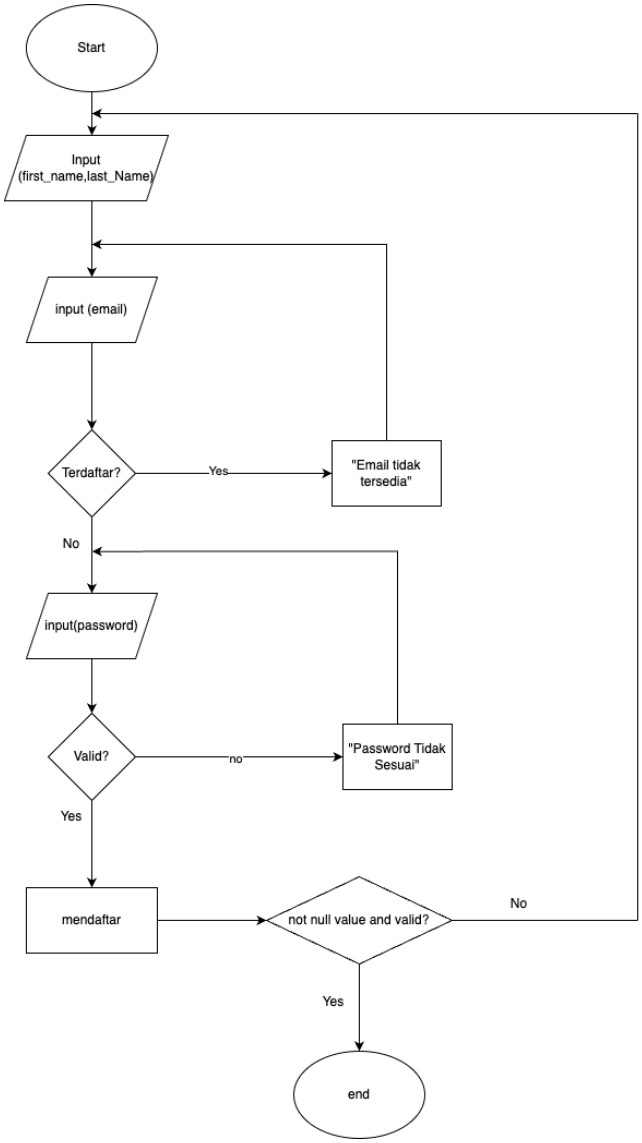
2.1. Pengumpulan Data

Tahapan awal dalam penelitian ini yaitu pengumpulan data. *White box testing* merupakan teknik pengujian yang melibatkan analisa kode program pada sistem. Oleh karena itu, data yang digunakan pada penelitian ini yaitu data sekunder berupa *source code* sistem Jobstreet.co.id yang diperoleh melalui inspeksi web Jobstreet.co.id secara langsung. File yang akan diuji yaitu file simple-signup.php dengan bahasa pemrograman HTML, CSS, dan JS untuk *frontend* dan bahasa pemrograman PHP untuk *backend*.

2.2. Alur Pendaftaran Jobstreet.co.id

Salah satu *form* yang akan diuji pada *website* Jobstreet.co.id adalah *form* pendaftaran. *Flowchart* dimulai dengan permintaan untuk memasukkan nama depan dan belakang pengguna. Kemudian, sistem akan memvalidasi alamat email pengguna. Jika alamat email sudah terdaftar, sistem akan menampilkan pesan eror "Email tidak tersedia". Jika alamat email tidak terdaftar, sistem akan meminta pengguna untuk memasukkan kata sandi. Kemudian, sistem akan memeriksa apakah kata sandi yang dimasukkan oleh pengguna valid. Jika *password* tidak valid maka sistem akan menampilkan pesan eror "Password Tidak Sesuai". Jika kata sandi valid, sistem akan meminta pengguna untuk konfirmasi proses pendaftaran. Jika pengguna mengkonfirmasi pendaftaran, sistem akan membuat akun pengguna dan menampilkan pesan sukses.

Secara keseluruhan, alur pada sistem *form* pendaftaran Jobstreet.co.id menguraikan proses sederhana untuk membuat akun pengguna. Hal tersebut mencakup langkah-langkah untuk memeriksa apakah alamat email sudah terdaftar, memvalidasi kata sandi, dan mengonfirmasi pendaftaran. Hal ini dapat membantu memastikan bahwa akun pengguna dibuat dengan aman dan akurat. *Flowchart form* pendaftaran Jobstreet.co.id dapat dilihat seperti Gambar 2 di bawah ini.



Gambar 2. *Flowchart* Alur Pendaftaran Jobstreet.co.id.

2.3. Pembuatan Flowgraph

Pada tahap ini, peneliti membangun *flowgraph* berdasarkan alur kerja *form* pendaftaran Jobstreet.co.id. *Flowgraph* memberikan visualisasi terkait alur *sequence*, *if-else*, hingga *looping* yang terdapat pada sistem *form* pendaftaran Jobstreet.co.id.

2.4. Perhitungan Cyclometic Complexity

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. $V(G) = P\ (Predicate\ Node) + 1$

2.5. 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.

2.6. Pengujian dan Penentuan Hasil Test Case

Pengujian pada *form* pendaftaran Jobstreet.co.id diuji sesuai dengan *independent path* yang telah ditentukan. Pengujian dilakukan dengan menyusun skenario pengujian dalam bentuk tabel yang mencakup ID, Rute, Hasil yang Diharapkan, Hasil Pengujian, dan Keterangan.

Tabel 1. Template Skenario 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 *form* pendaftaran Jobstreet.co.id. 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 *form* pendaftaran Jobstreet.co.id berdasarkan hasil pengujian yang telah dilakukan.

3. Hasil dan Pembahasan

3.1. Source Code

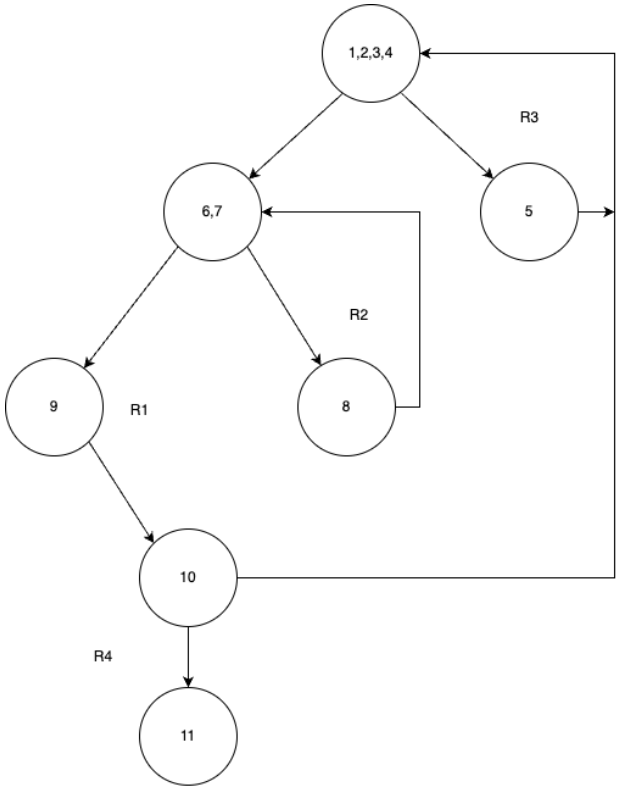
Pada Tabel 2 di bawah ini diketahui bagian-bagian *source code* simple-signup.php pada *form* daftar Jobstreet.co.id.

Tabel 2. *Source Code form* Daftar Jobstreet.co.id.

Source Code
<form id="login" class="form" name="login" method="post">
0: <input id="referer_url" name="referer_url" type="hidden" value="">
1: <input id="mobile_referer" name="mobile_referer" type="hidden" value="">
2: <input id="login_id" class="form-control" name="login_id" type="text" value="" placeholder="Email">
3: <input id="password" class="form-control" name="password" type="password" value="" autocomplete="off" placeholder="Kata sandi">
4: <input id="remember" class="pull-left" name="remember" type="checkbox" value="on" checked="">
5: <button id="btn_login" class="btn btn-primary btn-block btn-block-style" name="btn_login" type="submit" onclick="javascript:return onClickSubmit(this);">
6: <input name="login" type="hidden" value="1">

3.2. Flowgraph

Setelah diketahui bentuk *source code* dari simple-signup.php, dibangun *flowgraph* yang menunjukkan fungsi daftar Jobstreet.co.id. Dari *flowgraph* Gambar 3, terdapat jumlah *edge* (E) = 9, yang merupakan garis untuk menghubungkan *node*. Jumlah *node* (N) = 7 yang berbentuk lingkaran menggambarkan aktivitas, jumlah *predicate* (P) = 2 yang merupakan node bercabang, dan jumlah *region* (R) = 4 yang menandakan suatu area dalam *flowgraph*, yang dapat dilihat dengan simbol R1 hingga R4 pada Gambar 3.



Gambar 3. *Flow Graph.*

Pada Tabel 3 terdapat rincian *action* dari masing-masing *node*, yaitu :

Tabel 3. Deskripsi *Node.*

Node	Action
Node 1 (1,2,3,4)	Start -> Input (first_name,last_name) -> Input (email) -> Terdaftar ?
Node 2 (5)	Terdaftar (“Email Tidak Tersedia”)
Node 3 (6,7)	Tidak Terdaftar -> Input(password) -> Valid ?
Node 4 (8)	Tidak Valid (“Password Tidak Sesuai”)
Node 5 (9)	Valid -> Mendaftar
Node 6 (10)	Value tidak kosong dan Valid ?
Node 7 (11)	Valid -> End

3.3. Cyclomatic Complexity

Merujuk pada Gambar 3 telah diketahui jumlah *edge*, *node*, *predicate*, dan *region*, maka selanjutnya yaitu perhitungan *Complexity Cyclomatic* yang menghasilkan perhitungan seperti berikut:

Tabel 4. Perhitungan *Cyclomatic Complexity.*

Keterangan	Jumlah
Node	7
Edge	9
Predikat	3
V(G) : E – N + 2	9 - 7 = 2 + 2
	= 4
V(G) : P + 1	3 + 1
	= 4

Tabel 4 menyajikan perhitungan *Complexity Cyclomatic* berdasarkan jumlah *node*, *edge*, dan predikat dalam suatu program. Dengan 7 *node*, 9 *edge*, dan 3 predikat, *Complexity Cyclomatic* dihitung menggunakan rumus V(G) = E - N + 2, yang menghasilkan nilai 4. Selanjutnya, nilai ini juga dapat dihitung dengan rumus alternatif V(G) = P + 1, dan hasilnya adalah 4.

Tabel 5. *Penentuan Region.*

Region	Path
Region 1	1,2,3,4 –> 6,7 -> 9 -> 10 -> 1,2,3,4
Region 2	1,2,3,4 -> 6,7 -> 8 -> 6,7
Region 3	1,2,3,4 -> 5 -> 1,2,3,4
Region 4	1,2,3,4 –> 6,7 -> 9 -> 10 -> 11

Tabel 5 menunjukkan penentuan *region* dalam program tersebut, yang merupakan jalur eksekusi program dari satu simpul ke simpul lainnya. *Region* 1 mencakup *path* 1,2,3,4 –> 6,7 -> 9 -> 10 -> 1,2,3,4, jika *node* 9 gagal maka akan diteruskan ke *node* 10 dan *looping* ke *node* 1,2,3,4. *Region* 2 mencakup *path* 1,2,3,4 -> 6,7 -> 8 -> 6,7, jika *node* 6,7 gagal maka diteruskan ke *node* 8 dan *looping* ke *node* 6,7. *Region* 3 mencakup *path* 1,2,3,4 -> 5 -> 1,2,3,4, jika *node* 1,2,3,4 maka diteruskan ke *node* 5 dan *looping* ke *node* 1,2,3,4. Terakhir *Region* 4 mencakup *path* 1,2,3,4 –> 6,7 -> 9 -> 10 -> 1,2,3,4, region ini merupakan skenario jika seluruh *node* berhasil dilalui. Penentuan *region* ini memberikan pemahaman yang lebih mendalam tentang kompleksitas jalur eksekusi program dan dapat digunakan dalam pengujian dan analisis perangkat lunak.

3.4. Independent Path

Berdasarkan perhitungan *Complexity Cyclomatic* didapatkan hasil *independent path* untuk fungsi daftar yaitu sebanyak 4 dengan jalur *independent path*, seperti pada Tabel 6 berikut :

Tabel 6. *Penentuan Independent Path.*

Jalur	Flow
Jalur 1	1,2,3,4 –> 6,7 -> 9 -> 10 -> 1,2,3,4
Jalur 2	1,2,3,4 -> 6,7 -> 8 -> 6,7
Jalur 3	1,2,3,4 -> 5 -> 1,2,3,4
Jalur 4	1,2,3,4 –> 6,7 -> 9 -> 10 -> 11

Jalur-jalur tersebut dapat diuraikan sebagai berikut: Jalur 1, pendaftaran dimulai dengan memasukkan informasi pengguna, validasi keberadaan email, jika email tidak terdaftar, validasi password dilakukan, dan jika password valid, pendaftaran dilanjutkan. Jika ada data kosong dan tidak valid, proses akan kembali ke langkah awal pendaftaran (Node 1). Jalur 2 menggambarkan pendaftaran dengan langkah serupa dengan Jalur 1, tetapi jika password tidak valid, proses kembali ke langkah sebelumnya (Node 3). Jalur 3 menunjukkan pendaftaran dengan email yang sudah terdaftar akan menampilkan "Email Tidak Tersedia" (Node 3) dan kembali menuju (Node 1). Jalur 4 menggambarkan pendaftaran dengan validasi email yang sama seperti Jalur 1, tetapi seluruh validasi berhasil hingga mencapai langkah akhir dan selesai (Node 7).

3.5. Hasil Pengujian Test Case

Setelah diketahui hasil perhitungan *Complexity Cyclomatic* dan penentuan *independent path*, maka selanjutnya dilakukan pengujian *test case*. Hasil yang didapatkan dari pengujian *test case* fungsi daftar Jobstreet.co.id yaitu :

Tabel 7. Hasil Pengujian Form Daftar

ID	Rute	Rincian Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
01	1->2->3->4->6->7->9->10->1->2->3->4	Pengguna tidak menginput nama depan, nama belakang, email, dan password	Sistem akan meminta pengguna untuk mengisi kolom nama depan dan belakang serta email dan password	Sistem akan mengingatkan pengguna untuk mengisi kolom nama depan, belakang, email, serta password	Berhasil
02	1->2->3->4->6->7->8->6->7	Pengguna memasukkan password kategori lemah, berjumlah kurang dari 8 karakter, tidak ada huruf besar, dan tidak ada angka	Sistem akan memperingati pengguna untuk membuat password yang sesuai dengan standar keamanan dan akibatnya pengguna tidak bisa mendaftar	Sistem menolak pendaftaran pengguna dan muncul pesan peringatan kepada pengguna di kolom password	Berhasil
03	1->2->3->4->5->1->2->3->4	Pengguna memasukkan email yang sudah terdaftar	Sistem akan memperingati pengguna bahwa email yang dimasukkan tersebut telah terdaftar dan pengguna tidak bisa mendaftar akun	Sistem menolak pendaftaran pengguna dan muncul peringatan untuk mengganti alamat email yang dimasukkan	Berhasil
04	1->2->3->4->6->7->9->10->11	Pengguna memasukkan nama depan, nama belakang, email, dan password dengan benar serta valid	Sistem akan memproses pembuatan akun dan pengguna akan diproses ke langkah selanjutnya yakni mengisi profil diri	Sistem menerima serta memproses pendaftaran akun dan mengarahkan pengguna untuk mengisi profil diri	Berhasil

Berdasarkan hasil pengujian yang telah dilakukan pada rincian Tabel 7 didapatkan hasil total dari 4 *independent path*, seluruh *test case* mendapatkan keterangan uji "Berhasil". *Test case* mencakup berbagai skenario, termasuk penanganan *input* tidak valid dan memberikan peringatan yang sesuai. Hasil tersebut menunjukan bahwa *form* pendaftaran Jobstreet.co.id berjalan sesuai hasil yang diharapkan. Keberhasilan ini dapat diartikan bahwa sistem mampu mengelola proses pendaftaran dengan baik, memberikan respons yang tepat terhadap *input* yang diberikan, dan memastikan bahwa setiap *independent path* dapat diuji secara efektif.

4. Kesimpulan

Berdasarkan hasil pengujian *White Box* dengan teknik *Basis Path* pada *form* pendaftaran Jobstreet.co.id. Didapatkan informasi dari hasil perhitungan *Cyclometic Complexity* yaitu : 4 *region* dan 4 *independent path*. Setelah melalui uji coba terhadap keempat *independent path* tersebut, seluruh *test case* mendapatkan keterangan "Berhasil".

Dari hasil ini, dapat disimpulkan bahwa pengujian berjalan dengan baik dan sistem *form* pendaftaran Jobstreet.co.id tidak mengalami kesalahan logika pada fungsi daftar. Penggunaan metode *Basis Path* dalam pengujian *White Box* terbukti efektif, memberikan kepercayaan tambahan terhadap keandalan sistem dengan melakukan pengujian setiap jalur eksekusi program secara menyeluruh. Dengan demikian, sistem pendaftaran Jobstreet.co.id dapat dianggap siap untuk digunakan dengan tingkat keandalan yang tinggi dengan memastikan pengguna dapat mengakses layanan pendaftaran dengan lancar dan tanpa mengalami kendala logika.

Referensi

[1] A. P. Kusuma and B. Setiawan, "White Box Testing Pada Sistem Pemesanan Desain Sablon Berbasis Web," *Jurnal Teknika*, vol. 10, no. 2, pp. 1040–1044, 2018. doi:10.30736/teknika.v10i2.241

[2] M. F. Londjo, "Implementasi White Box Testing dengan Teknik Basis Path Pada Pengujian Form Login," *Jurnal Siliwangi Sains Teknologi*, vol. 7, no. 2, pp. 35–40, 2021.

[3] M. I. Shiddiq, "Implementasi White Box Testing Berbasis Path pada Form Login Aplikasi Berbasis Web," *Jurnal Siliwangi Sains Teknologi*, vol. 8, no. 1, pp. 1–6, 2022.

[4] A. C. Praniffa *et al.*, "Pengujian Black Box dan White Box Sistem Informasi Parkir Berbasis Web," *Jurnal Testing dan Implementasi Sistem Informasi*, vol. 1, no. 1, pp. 1–16, 2023.

[5] J. B. Sie, I. A. Musdar, and S. Bahri, "Pengujian White Box Testing Terhadap Website Room Menggunakan Teknik Basis Path," *KHARISMA Tech*, vol. 17, no. 2, pp. 45–57, Sep. 2022. doi:10.55645/kharismatech.v17i2.235

[6] S. Kalagara, "Cyclomatic Complexity in Software Development," *International Journal of Engineering Research & Technology (IJERT)* , vol. 2, no. 16, Oct. 2020.

[7] I. K. Wairooy, "Teknik Dalam White-Box Dan Black-box Testing," SoCS Binus University: School of Computer Science, https://socs.binus.ac.id/2020/07/02/teknik-dalam-white-box-dan-black-box-testing/ (accessed Nov. 28, 2023).

[8] R. Setiawan, "White Box Testing Untuk Menguji Perangkat Lunak," Dicoding Blog, https://www.dicoding.com/blog/white-box-testing/ (accessed Nov. 28, 2023).

[9] Meiliana, "Software testing: Perhitungan Cyclomatic Complexity," School of Computer Science, https://socs.binus.ac.id/2016/12/29/software-testing-perhitungan-cyclomatic-complexity (accessed Nov. 30, 2023).

[10] T. J. McCabe, "Cyclomatic Complexity Definition & Calculation," Jellyfish.co, https://jellyfish.co/library/cyclomatic-complexity/ (accessed Nov. 30, 2023).

[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," *Jurnal Pendidikan: Teori, Penelitian, dan Pengembangan*, vol. 3, no. 5, pp. 648–655, May 2018.

[12] C. Ebert, J. Cain, G. Antoniol, S. Counsell and P. Laplante, "Cyclomatic Complexity," *IEEE Software*, vol. 33, no. 6, pp. 27-29, Nov.-Dec. 2016, doi: 10.1109/MS.2016.147.

[13] A. N. Hesanty, "Cara membuat Flowchart Dalam 4 Langkah Mudah (Lengkap)!," Niagahoster Blog, https://www.niagahoster.co.id/blog/cara-membuat-flowchart/ (accessed Nov. 30, 2023).
[14] Meilina, "Basis Path Testing: Flow Graph," School of Computer Science, https://socs.binus.ac.id/2016/12/30/basis-path-testing-flow-graph (accessed Nov. 30, 2023).
[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," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 2, no. 5, pp. 2016–2024, Mei 2018.