Jurnal Siliwangi Vol. 9.No.2, 2023 **P-ISSN 2477-3891**
Seri Sains dan Teknologi **E-ISSN 2615-4765**

PENGUJIAN WHITE BOX TERHADAP SISTEM LOGIN BERBASIS WEB MENGGUNAKAN TEKNIK BASIS PATH

Muhamad Septian Jaelani

Program Studi Informatika Fakultas Teknik Universitas Siliwangi Tasikmalaya
197006034@student.unsil.ac.id

Abstrak

Sebuah aplikasi harus melewati fase pengujian agar tidak ada kesalahan atau cacat didalamnya. Proses pengujian dilakukan untuk menemukan kesalahan atau cacat dari sebuah aplikasi. Metode pengujian *white box* merupakan suatu pengujian yang dilakukan pada tingkat alur perangkat lunak. Pada pengujian *white box* terdapat beberapa teknik, salah satunya *basis path* yang dapat digunakan untuk mengukur kompleksitas kode program dan pendefinisian alur yang akan dieksekusi. Melalui teknik ini, dapat diketahui alur eksekusi kode program dan harus melalui minimal satu kali setiap alur. Tujuan penelitian ini, menerapkan metode *white box testing* berbasis *path* untuk mengetahui kompleksitas dan jalur eksekusi kode program dari sebuah sistem login. Tahapan yang dilakukan pada penelitian ini, diantaranya: membuat *flowchart*, membuat *flow graph*, menghitung *Cyclomatic Complexity* (CC), menentukan *independent path*, melakukan *unit test* dengan bantuan *library testing* Karma. Berdasarkan hasil percobaan pada penelitian ini menunjukkan bahwa, prototipe aplikasi sistem login yang dibuat memiliki tiga buah alur yang dapat dilalui sehingga memiliki tingkat kompleksitas kecil.

Kata kunci : basis path, pengujian, login, white box, aplikasi

Abstract

An application must go through a testing phase so that there are no errors or defects in it. The testing process is carried out to find errors or defects in an application. The white box testing method is a test that is carried out at the software flow level. In white box testing there are several techniques, one of which is the path basis that can be used to measure the complexity of the program code and define the flow to be executed. Through this technique, the flow of program code execution can be known and must go through at least once each flow. The purpose of this study is to apply the path-based white box testing method to determine the complexity and path of program code execution of a login system. The stages carried out in this study include: creating a flowchart, creating a flow graph, calculating Cyclomatic Complexity (CC), determining independent paths, conducting unit tests with the help of the Karma testing library. Based on the results of the experiment in this study, it shows that the prototype of the login system application that was created has three flows that can be passed so that it has a low level of complexity.

Keywords*: path basis, testing, login, white box, application*

I. PENDAHULUAN

Pengujian perangkat lunak memiliki fungsi penting dalam proses pengembangan aplikasi. Melalui pengujian akan didapat galat atau error dari suatu perangkat lunak. Sehingg diketahui tinggkat pemenuhan persyaratan dan menentukan perbedaan dari hasil yang diharapkan dengan hasil sebenarnya [1]. Pengujian merupakan suatu proses penting dan memakan waktu yang tidak sedikit, karena proses pengujian yang lengkap dan tidak menemukan kesalahan atau error, akan menghasilkan kualitas perangkat lunak yang sempurna. Untuk menjamin kualitas perangkat lunak tersebut, terdapat dua metode pengujian, yaitu secara fungsional (*Black Box*) dan secara sistematik (*White Box*) [1].

Aplikasi yang akan diuji pada penelitian ini adalah sebuah sistem login. Sistem login ini berbasis web yang dibuat dengan bahasa HTML, CSS, dan juga Javascript. Sistem login bertujuan untuk melakukan log in ke sebuah user yang sudah terdaftar pada web. Untuk melakukan log in ini, pengguna harus memasukkan sebuah username dan juga passwordnya. Pada penelitian ini akan diuji proses pengisian username dan password, contohnya apa yang akan terjadi saat memasukkan username atau password yang salah. Metode yang akan digunakan pada pengujian ini adalah metode white box testing. White box testing adalah pengujian perangkat lunak dari segi desain dan kode program dengan cara memeriksa lojik dari kode program tersebut[2]. Pengujian *white box* ini memiliki beberapa teknik,

tetapi pada penelitian ini akan menggunakan teknik *basis path* yaitu salah satu teknik pengujian struktur kontrol yang menjamin bahwa semua *statement* dalam setiap *independent path* progam dieksekusi minimal satu kali[2].

II. METODOLOGI PENELITIAN

Pengumpulan data pada penilitian ini dilakukan dengan menggunakan pengujian secara otomatis dengan bantuan sebuah library testing yaitu karma. Penelitian ini menggunakan analisis kuantitatif dengan menghitung sebuah nilai *Cyclomatic Complexity* (CC) untuk menghasilkan sebuah kesimpulan.

Penelitian ini dimulai dengan membuat sebuah diagram *flowchart* dari sistem login yang akan diuji. Diagram *flowchart* ini akan menunjukkan jalur jalur yang akan dilalui oleh program saat digunakan. Kemudian tahap berikutnya akan membuat sebuah *flow graph* dari sistem login dengan melihat *flowchart* yang dibuat sebelumnya. Dengan menggunakan *flow graph* ini, kita akan mendapatkan informasi informasi yang akan digunakan pada tahap berikutnya, yaitu menghitung nilai *Cyclomatic Complexity* (CC).

Nilai *Cyclomatic Complexity* (CC) adalah atribut yang secara umum digunakan untuk menghitung tingkat kompleksitas kode program (Zhang, 2007). Nilai ini merupakan pendekatan ukuran kompleksitas yang dilakukan untuk mengukur dan mengontrol jumlah alur melalui program (McCabe, 1976)[10]. Menggunakan nilai ini, kita akan menemukan tingkat kompleksitas dari sistem login yang diuji.

Tahap berikutnya adalah melakukan pengujian unit test secara otomatis dengan bantuan library testing Karma. Pengujian ini akan menguji langsung pada *source code* dari sistem login sehingga dapat terlihat dengan detail kesalahan yang dapat ditemukan.

Pengujian

Pengujian adalah proses untuk menemukan error pada perangkat lunak sebelum dikirim kepada pengguna[3]. Dengan ditemukannya error tersebut, kita dapat menentukan kualitas dari sebuah perangkat lunak. Karena hal inilah pengujian adalah proses yang penting untuk dilakukan sehingga memakan waktu yang sangat banyak.

Pengujian yang paling tepat adalah melakukan pengujian *white box* sebelum pengguna menggunakannya agar mudah digunakan dan berguna bagi pengguna[4].

Pengujian White Box

Pengujian white box adalah pengujian yang dilakukan untuk menguji dan menganalissi kode program bilamana terjadi kesalahan atau tidak[5]. Pengujian ini dapat memberikan detail jelas dari kode program yang diuji sehingga dapat diketahui kesalahan atau bug yang dimilikinya.

Teknik Basis Path

Teknik *Basis path* adalah salah satu teknik dari pengujian *white box*. Teknik ini lebih fokus ke bentuk internal dari objek dan mencoba untuk menemukan semua jalur dari program. Untuk melakukan pengujian ini, penguji harus mengetahui pengetahuan perangkat lunak yang akan diuji, sehingga kebanyakan yang menggunakan teknik ini adalah software developer[6]. Hasil yang didapat dari menggunakan teknik ini adalah jalur jalur yang akan dilalui oleh program dan juga tingkat kompleksitas dari program.

Flow Graph

Flow graph atau *Control Flow Graph* (CFG) adalah *graph* berarah yang merepresentasikan aliran dari sebuah program, setiap CFG terdiri dari node dan edge. Node merepresentasikan perintah, sedangkan edge merepresentasikan kontrol transfer antar node (Watson dan McCabe 1996)[7]. Flow graph ini hampir sama dengan *flow chart* tetapi tidak menggambarkan secara detail proses yang terjadi pada setiap blok notasi.

Cyclomatic Complexity

Cyclomatic Complexity adalah sebuah satuan perangkat lunak yang memberikan ukuran kuantitatif dari logika kompleksitas keempat. Saat digunakan dalam konteks metode pengujian basis path, satuan ini menunjukkan alur independen pada progam yang diuji[8]. Nilai ini akan memberitahu tingkat kompleksitas dari program yang diuji. Nilai *Cyclomatic Complexity* dapat ditemukan menggunakan *flow graph* dengan rumus berikut

$$V(G) = E - N + 2$$

Dimana N adalah banyaknya node pada *flow graph* dan E adalah banyaknya edge pada *flow graph*.

Unit Test

Unit test adalah proses menguji sebagian atau komponen tertentu dalam kode program untuk menentukan apakah komponen tersebut berfungsi dengan benar[9]. Contohnya pada penelitian ini akan dilakukan pengujian unit test pada sistem login

sebuah aplikasi web, dan tidak melakukan testing pada aplikasi web secara keseluruhan.

III. HASIL DAN PEMBAHASAN

Program sistem login yang akan diuji pada penelitian ini dapat diakses pada link berikut https://loginer.netlify.app/ menggunakan username "septian" dan password "Jael". Pada penelitian ini, hanya akan menguji pada sistem login yang digunakan pada web tersebut, tidak keseluruhan aplikasi webnya. Gambar 1 merupakan potongan source code sistem login.

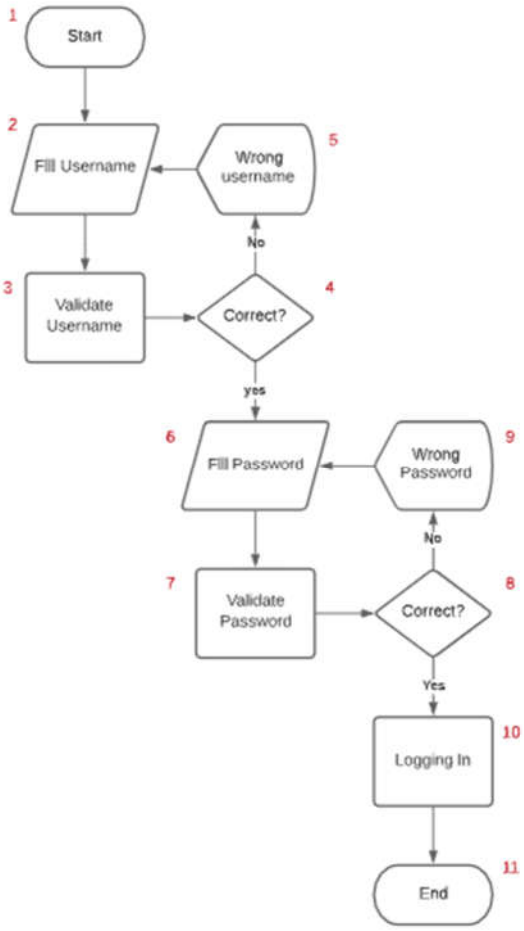
```
loggingIn(event) {
    event.preventDefault()
    const username =
document.getElementById("user");
    const password =
document.getElementById("pwd");

password.nextElementSibling.textContent
= "";

username.nextElementSibling.textContent
= "";

    if (password.value === 'Jael' &&
username.value === 'septian') {
      sessionStorage.setItem('logged-
users', JSON.stringify({
        id: +new Date(),
        username: username.value,
        password: password.value
      }));
      username.value = '';
      password.value = '';

      if
(sessionStorage.getItem('logged-users')
!== null) {
        LoginAuth.renderLogged();
      }
    } else if (username.value !==
'septian' && password.value !== 'Jael')
{
      LoginAuth.renderFalses();
    } else if (username.value !==
'septian') {
      LoginAuth.renderFalseUser();
    } else if (password.value !==
'Jael') {
      LoginAuth.renderFalsePass();
    }
  },

  renderFalses() {
    const username =
document.getElementById("user");
    const password =
document.getElementById("pwd");

username.nextElementSibling.textContent
= "Wrong Username";

password.nextElementSibling.textContent
= "Wrong Password";
    setTimeout(() => {

username.nextElementSibling.textContent
= "";

password.nextElementSibling.textContent
= "";
    }, 2000);
  },

  renderFalseUser() {
    const username =
document.getElementById("user");

username.nextElementSibling.textContent
= "Wrong Username";
    setTimeout(() => {

username.nextElementSibling.textContent
= "";
    }, 2000);
  },

  renderFalsePass() {
    const password =
document.getElementById("pwd");

password.nextElementSibling.textContent
= "Wrong Password";
    setTimeout(() => {

password.nextElementSibling.textContent
= "";
    }, 2000);
  },

  renderLogged() {
    const contentBox =
document.getElementById("content-box");
    const loginBox =
document.getElementById("login-box");
    const loggedBox =
document.getElementById("logged-box");
    const loginBtn =
document.getElementById("login-btn");
    const logoutBtn =
document.getElementById("logout-btn");

    contentBox.classList.add("hide");
    loginBox.classList.add("hide");
    loggedBox.classList.remove("hide");
    loginBtn.classList.add("hide");
    logoutBtn.classList.remove("hide");
  },
```

Gambar 1.Potongan source code sistem login.

A. Pembuatan Flowchart

Gambar 1 menampilkan potongan kode program dari form login yang akan diuji. Dari potongan kode program tesebut selanjutnya dibuat visualisasi ke dibuat *flowchart* seperti ditampilkan pada gambar 2.

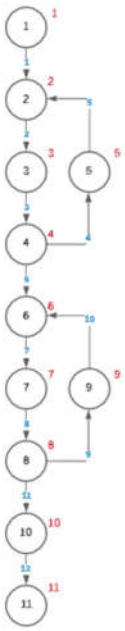
P-ISSN 2477-3891
E-ISSN 2615-4765



Gambar 2. *Flowchart* sistem login.

B. Pembuatan Flow graph

Flowgraph yang dibuat, mengacu kepada *flowchart* pada gambar 2. Hasil *Flowgraph* yang dibuat ditampilkan pada gambar 3.



Gambar 3. *Flow graph* Sistem Login.

C. Menghitung nilai Cyclomatic Complexity (CC)

Informasi yang didapat dari *flow graph*, dijadikan acuan untuk menghitung nilai *Cyclomatic Compexity*. Jumlah node dari sistem login sebanyak 11 dan jumlah edge sebanyak 12, maka nilai *Cyclomatic complexity* dari sistem login ini :

$$V(G) = E - N + 2$$
$$= 12 - 11 + 2$$
$$= 3$$

Didapatkan nilai *Cyclomatic Complexity* sebesar 3, ini berarti tingkat kompleksitas dari sistem login menurut *flow graph* yang dibuat adalah kecil, sehingga resiko terjadinya kesalahan sangat kecil.

D. Menentukan Independent Path

Setelah nilai CC ditemukan, yaitu 3, ini berarti sistem login memiliki *independent path* sebanyak tiga buah, yaitu

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

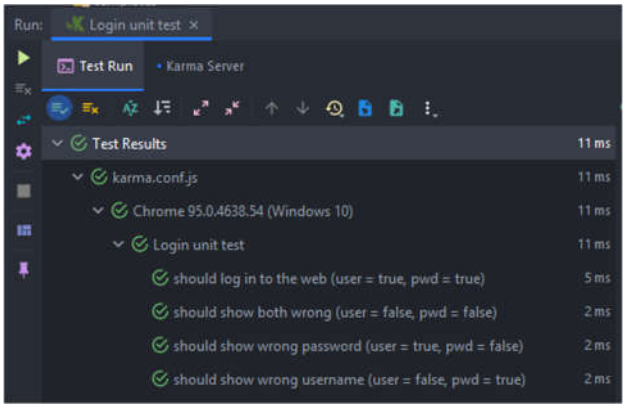
E. Pengujian Unit Test Sistem Login

Setelah ditemukan *independent path* dari sistem login, dilakukan pengujian *unit test* pada *source code* secara langsung. Pengujian ini akan dilakukan secara otomatis menggunakan *library testing* Karma. Gambar 4 menampilkan potongan kode program untuk melakukan unit tes secara otomatis.

```
Import LoginAuth from
"../src/scripts/utils/login-auth";

describe('Login unit test', () => {
  const addDocument = () => {
    document.body.innerHTML = `
      <button id="login-btn">Log
In</button>
      <button id="logout-btn"
class="hide">Log Out</button>
      <div class="content__bg">
        <div id="content-box"
class="content__box">
          <h1 id="welcome-
text">Welcome</h1>
        </div>
        <div id="login-box"
class="login__box hide">
          <h2
class="login__title">Login</h2>
          <form action="" id="login-form"
class="login__form">
            <div class="login__input">
              <label
for="user">Username</label>
              <input type="text"
id="user" required>
```

```
              <span
class="error__text"></span>
            </div>
            <div class="login__input">
              <label
for="pwd">Password</label>
              <input      type="password"
id="pwd" required>
              <span
class="error__text"></span>
            </div>
            <div class="login__submit">
              <input    id="login-submit"
type="submit">
            </div>
          </form>
        </div>
        <div              id="logged-box"
class="content__box hide">
          <h1   id="logged-text">You   are
Logged In</h1>
        </div>
      </div>
    `;
  };

  const renderLogin = (usr, pwd) => {
    const           loginBox           =
document.getElementById("login-box");
    const          contentBox          =
document.getElementById("content-box");
    const           username           =
document.getElementById("user");
    const           password           =
document.getElementById("pwd");
    const           loginForm          =
document.getElementById("login-form");

    loginBox.classList.remove("hide");
    contentBox.classList.add("hide");


expect(loginBox.classList.contains("hide"
)).toBeFalse();

expect(contentBox.classList.contains("hid
e")).toBeTrue();
    loginForm.addEventListener("submit",
LoginAuth.loggingIn);

    username.value = usr;
    password.value = pwd;
  }
      . . . . .

  });
})
```

Gambar 4. Potongan kode program untuk Unit tes



Gambar 5. Pengujian *unit test* secara otomatis.

IV. KESIMPULAN

Terdapat sebuah sistem login berbasis web yang dibuat menggunakan bahasa HTML, CSS dan Javascript. Pada sistem login tersebut, dilakukan sebuah pengujian *white box* menggunakan teknik *basis path*. Pengujian tersebut dilakukan untuk mengetahui jalur jalur yang dapat dilalui oleh program dan juga tingkat kompleksitas dari program tersebut. Untuk melakukan pengujian *white box* dengan teknik *basis path* dibuatkan *flowchart* dan juga *flow graph* dari sistem login tersebut. Kemudian menghitung nilai *Cyclomatic complexity* (CC) menggunakan informasi yang didapat dari *flow graph* yang dibuat. Nilai *Cyclomatic Complexity* yang didapat sebesar 3, ini berarti tingkat kompleksitas dari program adalah kecil, sehingga resiko terjadinya kesalahan sangat kecil. Setelah itu, dilakukan pengujian *unit test* sistem login secara otomatis menggunakan *library testing* Karma. *Unit test* yang dilakukan menghasilkan semua test success.

DAFTAR PUSTAKA

[1] 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, p. 111, 2020, doi: 10.32493/informatika.v5i2.4713.

[2] C. P. C. Munaiseche, G. C. Rorimpandey, T. Informatika, F. Teknik, and U. N. Manado, "Penerapan Metode Basis Path Analysis dalam Pengujian White Box Sistem Pakar," pp. 124–128.

[3] A. Rouf, "Pengujian Perangkat Lunak Dengan Menggunakan Metode White Box dan Back Box," vol. vol 8 no1, pp. 1–7, 2012, [Online]. Available: http://www.ejournal.himsya.ac.id/index.php/HIMSYATECH/article/view/28/27.

[4] Khoirunnisya, "Analisis white box testing pada aplikasi web pemesanan sablon kaos," *J.*

E-Bisnis, vol. XV, no. 01, pp. 1–7, 2021.

[5] E. sita Eriana, "Pengujian Sistem Informasi Aplikasi Perpustakaan Berbasis Web Dengan White Box Testing," *J. Teknol. Inf. ESIT*, vol. XV, no. 10, pp. 28–33, 2020.

[6] D. Madhavi, "A White Box Testing Technique in Software Testing: Basis Path Testing," *J. Res.*, vol. 02, no. 04, pp. 12–17, 2016.

[7] R. A. R. Fitriani and I. Hermadi, "Instrumentasi Kode Program Secara Otomatis untuk Path Testing," *J. Ilmu Komput. dan Agri-Informatika*, vol. 5, no. 1, p. 40, 2018, doi: 10.29244/jika.5.1.40-50.

[8] M. E. Khan, "Different approaches to white box testing technique for finding errors," *Int. J. Softw. Eng. its Appl.*, vol. 5, no. 3, pp. 1–14, 2011, doi: 10.5121/ijsea.2011.2404.

[9] M. S. Lasasi Purnomo Budi; Tama, Ishardita Pambudi, "Rancang Bangun on-Line Public Access Catalogue (Opac) Pada Sistem Informasi Manajemen Ruang Baca Teknik Industri Universitas Brawijaya," *J. Rekayasa dan Manaj. Sist. Ind.*, no. Vol 2, No 2 (2014), pp. p325-336, 2014, [Online]. Available: http://jrmsi.studentjournal.ub.ac.id/index.php/jrmsi/article/view/86.

[10] F. Pranata, F. Pradana, and T. A. Kurniawan, "Pengembangan Sistem Perhitungan Kompleksitas Kode Sumber Berdasarkan Metrik Halstead dan Cyclomatic Complexity," 2016.