Jurnal Siliwangi Vol. 8.No.2, 2022 **P-ISSN 2477-3891**
Seri Sains dan Teknologi **E-ISSN 2615-4765**

PENGUJIAN WHITE BOX BERBASIS PATH PADA FORM AUTENTIKASI BERBASIS MOBILE

Wisnu Arya Nugraha

Program Studi Informatika Fakultas Teknik Universitas Siliwangi Tasikmalaya
e-mail: 197006033@student.unsil.ac.id

Abstrak

Pembuatan perangkat lunak memerlukan sebuah pengujian untuk dapat memastikan kualitas semaksimal mungkin. Penelitian ini melakukan pengujian terhadap sebuah perangkat lunak yang dirancang menggunakan flutter. Aplikasi berisi suatu fungsi yang melakukan autentikasi berupa login, yang terdiri dari Username dan Password. Pengujian dilakukan menggunakan whitebox testing dengn metode Basic Path Testing. Dilakukan dua cara pengujian yakni secara manual dan otomatis menggunakan library testing yang disediakan flutter. Langkah yang dilakukan dalam pengujian adalah membuat flowchart, membuat flowgraph, menghitung cyclomatic complexity. Hasil pengujian pada penelitian ini didapatkan cyclomatic complexity yaitu jalur independen berjumlah 3 yang artinya resiko error pada aplikasi tersebut terbilang rendah. Hasil uji berdasarkan tabel skenario mendapatkan hasil valid, hal ini menunjukan bahwa aplikasi ini dapat berjalan dengan baik tanpa adanya error. Pengujian otomatis juga dilakukan menggunakan bantuan library yang tersedia, dengan hasil sesuai dengan tujuan.

Kata Kunci : flutter, login, pengujian, whitebox, path.

Abstract

Pembuatan perangkat lunak memerlukan sebuah pengujian untuk dapat memastikan kualitas semaksimal mungkin. Penelitian ini melakukan pengujian terhadap sebuah perangkat lunak yang dirancang menggunakan flutter. Aplikasi berisi suatu fungsi yang melakukan autentikasi berupa login, yang terdiri dari Username dan Password. Pengujian dilakukan menggunakan whitebox testing dengn metode Basic Path Testing. Dilakukan dua cara pengujian yakni secara manual dan otomatis menggunakan library testing yang disediakan flutter. Langkah yang dilakukan dalam pengujian adalah membuat flowchart, membuat flowgraph, menghitung cyclomatic complexity. Hasil pengujian pada penelitian ini didapatkan cyclomatic complexity yaitu jalur independen berjumlah 3 yang artinya resiko error pada aplikasi tersebut terbilang rendah. Hasil uji berdasarkan tabel skenario mendapatkan hasil valid, hal ini menunjukan bahwa aplikasi ini dapat berjalan dengan baik tanpa adanya error. Pengujian otomatis juga dilakukan menggunakan bantuan library yang tersedia, dengan hasil sesuai dengan tujuan.

***Keywords**: flutter, login, testing, whitebox, path.*

I. PENDAHULUAN

Pada saat seorang developer atau programmer membuat sebuah perangkat lunak, tentunya akan ditemukan kesalahan atau error pada proses-proses tertentu. Untuk menghindari banyaknya bug atau error, maka diperlukan pengujian perangkat lunak sebelum perangkat lunak siap digunakan. Tujuan utama dari pengujian perangkat lunak adalah untuk memastikan bahwa perangkat lunak yang dihasilkan sesuai dengan kebutuhan yang telah ditentukan. Pengujian perangkat lunak dibagi menjadi dua yaitu: whitebox testing, dan black box testing. Whitebox testing adalah pengujian yang dikembangkan berdasarkan pada kode program, sedangkan pengujian blackbox adalah pengujian fungsional yang dibuat berdasarkan spesifikasi dari klien[1].

Pada penelitian kali ini, dikembangkan sebuah *prototype* perangkat lunak yang memiliki fitur login dengan fungsi verifikasi yang akan melakukan autentikasi user ke halaman utama. Aplikasi dibuat menggunakan flutter sehingga menghasilkan bentuk luaran sebagai sebuah aplikasi multi-platform yang dapat digunakan pada beberapa perangkat yang berbeda.

II. METODOLOGI

Penelitian ini dilakukan dengan menerapkan whitebox testing berbasis path atau path testing. Tahapan yang dilakukan dengan menggambarkan flow graph dan menentukan cyclomatic complexity. Penelitian 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. Tahap berikutnya membuat sebuah *flow graph* dari sistem login dengan melihat *flowchart* yang dibuat sebelumnya, berikutnya menghitung nilai *Cyclomatic Complexity* (CC).

Terdapat beberapa tahap dalam proses penelitian ini, diantaranya: analisis data, proses pengerjaan, prosedur cara kerja sistem, dan skenario pengujian sistem.

Bahasa Pemrograman Dart

Dart merupakan bahasa pemrograman yang dikembangkan google untuk membuat aplikasi mobile (android), front-end, web, IoT, back-end (CLI), dan Game[2]. Bahasa pemrograman Dart menerapkan konsep pemrograman berorientasi objek (OOP). Struktur dari dart sendiri menyerupai bahasa pemrograman C, java, javascript[3], dan swift, yakni menggunakan C-Style syntax. Dart menggunakan beberapa alat pendukung yakni sebuah virtual machine[4] untuk menjalankan program yang dibuat.

Flutter

Flutter adalah sebuah framework open source yang diciptakan oleh Google yang ditujukan untuk membuat aplikasi mobile, dekstop, dan web dengan satu basis kode [5]. Flutter menggunakan bahasa pemrograman dart sebagai intinya. Pengkodean flutter dirancang untuk memudahkan developer dalam mengembangkan aplikasi, karena fitur yang disebut widget memudahkan dalam mebuat objek-objek yang diperlukan pada aplikasi.

Pengujian Perangkat Lunak

Pengujian adalah sebuah aktivitas dalam pengembangan perangkat lunak. Kegiatan ini berisi observasi terhadap eksekusi yang dilakukan oleh perangkat lunak, serta melakukan pemeriksanaan validasi atas hasil yang diberikan [6]. Dengan dilakukannya software testing, pengembang dapat mengidentifikasi error atau kecacatan pada perangkat lunak yang dikembangkan.

Strategi software testing dapat dilakukan secara manual atau otomatis [7]. Pengujian manual, merupkan suatu pendekatan yang lebih tradisional, penguji menyiapkan suite tes yang menurut mereka akan lebih baik menjalankan program. Alat pengujian perangkat lunak otomasi membantu dalam menghasilkan kasus uji dari spesifikasi program atau dari data teks aslinya. Pengujian manual adalah teknik pengujian dimana penguji menyiapkan test case secara manual dan mengeksekusinya untuk mengidentifikasi cacat pada perangkat lunak.

Whitebox Testing

Strategi Pengujian whitebox adalah pengujian yang dikembangkan berdasarkan pada kode program. Penguji dalam white box testing harus memiliki pengetahuan tentang kode dan penulisan kasus uji dengan parameter yang sesuai. Hal ini terutama menyangkut dengan aliran kontrol dan aliran data suatu program[1].

Pada pengujian whitebox terdapat beberapa teknik dalam pengujiannya diantaranya: Data Flow Testing, Control Flow Testing, Basis Path / Path Testing, dan Loop Testing[1].

Basis Path Testing

Basis Path merupakah salah satu metode dalam pengujian perangkat lunak menggunakan teknik whitebox testing. Metode ini menyiapkan serangkaian langkah independen untuk melihat korespondensi studi kasus dengan menganalisa cyclomatic complexity dari kontrol struktur. Pengujian didasarkan pada kontrol flow graph yang merepresentasikan beberapa informasi kode utama[8].

Flow Graph

Grafik alir (Flow Graph) adalah sebuah notasi sederhana yang merepresentasikan aliran kontrol dari sebuah struktur program. Dalam sebuah grafik alir, anak panah disebut sebagai sisi (edge, E) yang merepresentasikan aliran kontrol, kemudian lingkaran disebut sebagai simpul (node, N) yang merepresentasikan satu atau lebih aksi/pernyataan logis, daerah yang dibatasi oleh sisi dan simpul disebut area (region, R), simpul yang mengandung keputusan disebut sebagai (predicate node, P) yaitu simpul yang mengeluarkan lebih dari satu sisi[9].

Cyclomatic Complexity

Cyclomatic complexity, V(G) adalah sebuah besaran yang menyatakan ukuran tingkat kompleksitas sebuah program. Angka ini menentukan jumlah jalur dasar yang harus diuji minimal sekali dari sebuah program[10]. Berdasarkan studi yang telah dibuat oleh beberapa industri menyatakan semakin besar nilai V(G) maka semakin besar probabilitas terjadinya kesalahan program.

Analisa Data

Data yang digunakan dalam penelitian adalah source code file sebagai pembentuk aplikasi, diantaranya:

- main.dart
- page_home.dart
- main_test.dart

Pengujian dilakukan dengan menguji fungsi yang terdapat pada main.dart sebagai file utama.

Proses Pengerjaan

Proses yang diuji diambil dari file main.dart dengan tujuan menguji fungsi yang ada. Proses pengerjaan yang dilakukan disesuaikan dengan teknik basic path testing pada whitebox testing, diantaranya:

a. membuat flowchart sistem
b. membuat flowgrph dari flowchart sistem
c. menentukan nilai flowgraph
d. menghitung cyclomatic complexity
e. membuat jalur independen
f. melakukan test case skenario otomatis dan manual

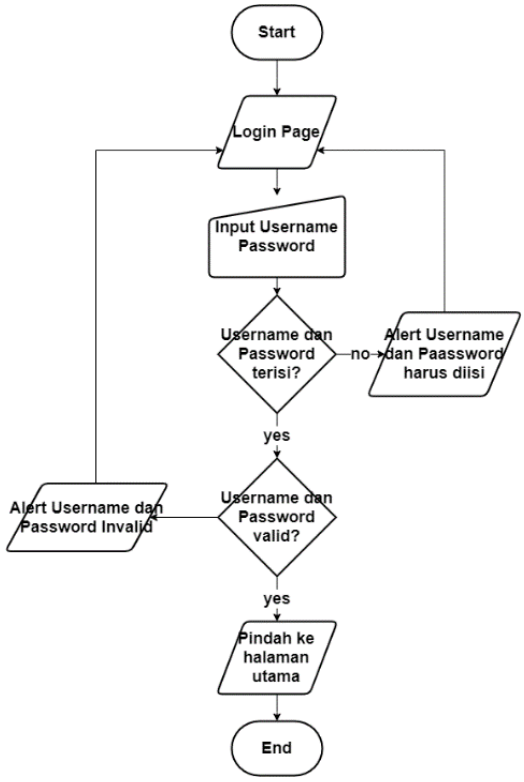
III. HASIL DAN PEMBAHASAN

Source Code Program Login pada main.dart

Tabel 1. Source Code Fungsi Login

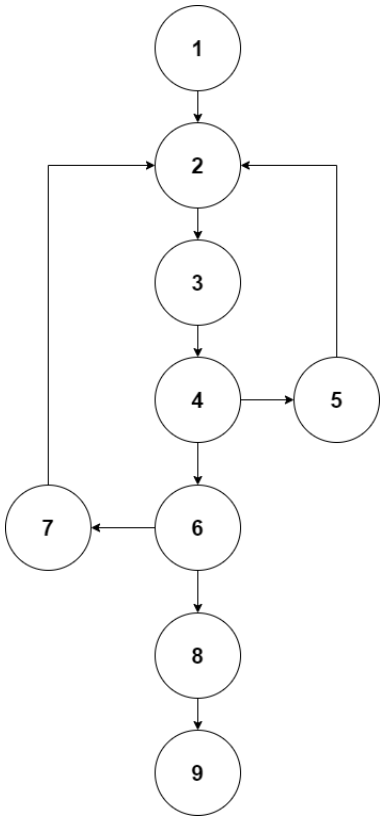
```
Column(crossAxisAlignment:
CrossAxisAlignment.center,
children: <Widget>[
TextFormField( //cek data field nya
kosong
  validator: (value) {if
(value!.isEmpty) {
return 'Please Input Username';}
if (value != 'wisnu') {
return 'Username tidak ditemukan';}
if (value == 'wisnu') {unameTrue =
true;}
return null;},controller:
myUsernameController, decoration:
InputDecoration( hintText: 'Input
Username',),
),
TextFormField(//cek data field kosong
validator: (value) {
if (value!.isEmpty) {
return 'Please Input Password';}
if (value != 'siliwangi') {
return 'Password salah';}
if (value == 'siliwangi') {
passTrue = true;}
return null;},
// maxLength: 16,
// maxLengthEnforced: true,
controller:
myPasswordController,obscureText:
true,
decoration: InputDecoration(hintText:
'Input Password',),),
SizedBox(
height: 25.0,
),
MaterialButton(
minWidth: 85.0,
height: 50.0,
color: Colors.green,
textColor: Colors.white,
onPressed: () {
nUsername =
myUsernameController.text;
nPassword =
myPasswordController.text;
if
(_formKey.currentState!.validate()) {
if (unameTrue && passTrue) {
//aksi pindah
Navigator.push(context,MaterialPageRo
ute(
builder: (context) => PageHome(
nama: nUsername,
password:   nPassword, // variable
yang di pass ke page home
)));
                    }
                  }
                },
                child: const
Text('Submit'),
              )
            ],
          ),
```

Flowchart Login Function



Gambar 1. Flow Chart Fungsi Login

Flow Graph Fungsi Login dari Flowchart



Gambar 2. Flow Graph Fungsi Login

Gambar 2 adalah flowgraph dari flowchart fungsi login pada gambar 1, dari gambar tersebut diketahui node dari fungsi login berjumlah 9, edge berjumlah 10, region berjumlah 3, dan predicate node berjumlah 2. Setelah diketahui nilai dari flowgraph, maka untuk perhitungan flowgraph adalah :

1. V(G) = R = 3
2. V(G) = E – N + 2
   = 10 – 9 +2
   = 3
3. V(G) = P + 1
   = 2 + 1
   = 3

Dari perhitungan didapat jalur independen sebagai berikut :

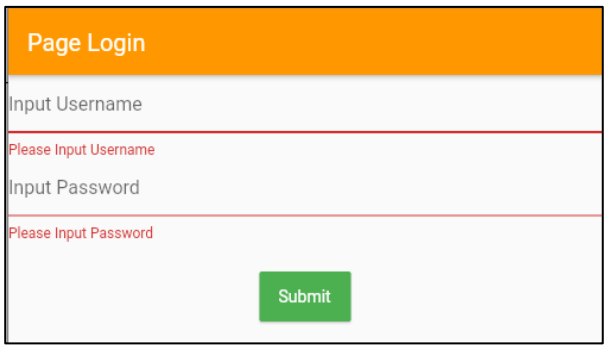
1. 1-2-3-4-5
2. 1-2-3-4-5-6-7
3. 1-2-3-4-5-6-7-8-9

Test Case

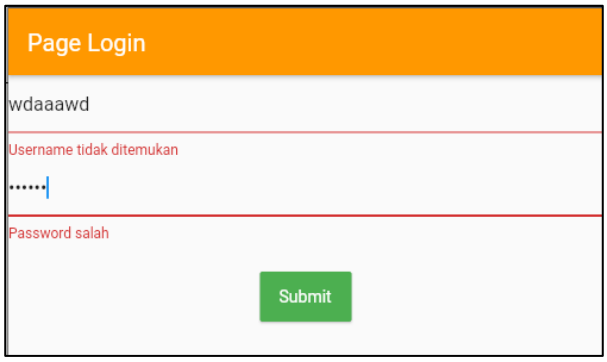
Setelah diketahui jalur independen, selanjutnya dibuat tabel test case seperti ditampilkan pada tabel 2.

Tabel 2. Test Case Model

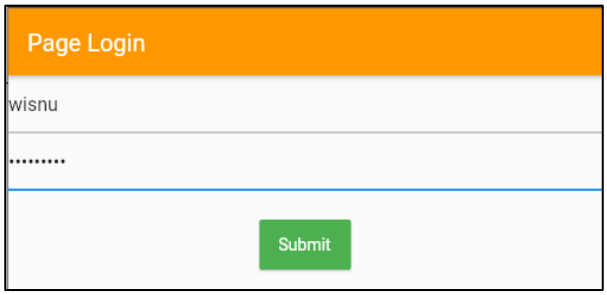
No	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
1	Input username dan pass word kosong	Muncul alert username password harus diisi	Muncul alert username password harus diisi	Valid
2	Input user name pass word yang salah	Muncul alert username tidak ditemukan atau password salah	Muncul alert username tidak ditemukan atau password salah	Valid
3	Input username pass word sesuai	Beralih ke halaman utama	Beralih ke halaman utama	Valid



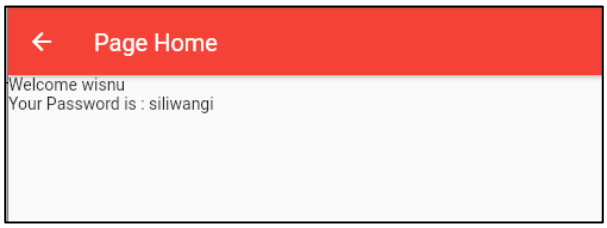
Gambar 3. Testing Case 1



Gambar 4. Testing Case 2



Gambar 5. Testing Case 3



Gambar 6. Halaman Utama

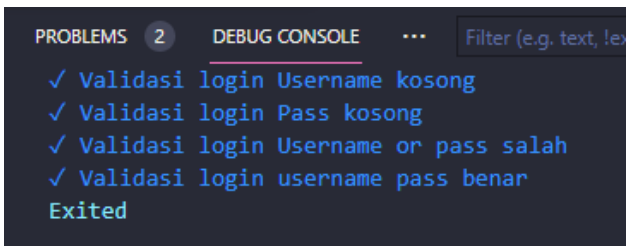
Dari tabel test case berdasarkan jalur independen yang didapat melalui hasil perhitungan Cyclomatic Complexity dengan melakukan tiga pengujian didapatkan hasil keduanya adalah valid yang artinya tidak ditemukan error pada fungsi login.

Automatic Testing

Untuk testing otomatis, digunakan unit testing yang tersedia pada library flutter, yakni unit test dan widget test. Disini digunakan unit test untuk melakukan pengujian pada fungsi login. Code yang dibuat untuk menjalankan testing ditampilkan pada tabel 3

Tabel 3. Source Code Unit Testing

```
import
'package:flutter/material.dart';
import
'package:flutter_test/flutter_test.da
rt';

import
'package:white_box_test/main.dart';

void main() {
  group('Validasi login', () {
    test('Username kosong', () {
      PageLogin model = PageLogin();
      model.nUsername = '';

      expect(model.unameTrue, false);
    });
    test('Pass kosong', () {
      PageLogin model = PageLogin();
      model.nPassword = '';
      expect(model.passTrue, false);
    });

    test('Username or pass salah', ()
{
      PageLogin model = PageLogin();
      model.nUsername = 'adwasd';
      model.nPassword = 'dwaawd';
      expect(model.passTrue, false);
      expect(model.unameTrue, false);
    });

    test('username pass benar', () {
      PageLogin model = PageLogin();
     model.nUsername = 'wisnu';
      model.nPassword = 'siliwangi';
      expect(model.unameTrue, true);
      expect(model.passTrue, true);
    });
  });
}
```



Gambar 7. Output Automatic Testing

Didapat hasil sesuai yang di inginkan yakni jika kredensial username dan password salah, maka akan menghasilkan nilai false. Kemudian apabila benar akan menghasilkan nilai true.

IV. KESIMPULAN DAN SARAN

Telah dilakukan whitebox testing dengan menerapkan teknik berbasis path. Hasil pengujian tersebut didapatkan cyclomatic complexity yaitu jalur independen berjumlah 3 yang artinya resiko error pada aplikasi tersebut terbilang rendah. Hasil uji berdasarkan tabel uji skenario mendapatkan hasil valid, hal ini menunjukan bahwa aplikasi ini dapat berjalan dengan baik tanpa adanya error. Pengujian otomatis juga dilakukan menggunakan bantuan library yang tersedia, dengan hasil sesuai dengan tujuan.

DAFTAR PUSTAKA

[1] R. Subagia, R. Alit, and F. A. Akbar, "Pengujian white box pada sistem informasi monitoring skripsi program studi informatika," *J. Inform. dan Sist. Inf.*, vol. 01, no. 2, pp. 539–547, 2020.
[2] F. Strocco and P. Dissertation, "Type Soundness in the Dart Programming Language," 2016.
[3] Unicode Consortium., "The Unicode standard," p. 1040, 2000.
[4] The Dart Team, "The standalone VM," 2016.
[5] L. Dagne, "Flutter for Cross-Platform App and SDK Development," *Metrop. Univ. Appl. Sci.*, no. May, 2019, [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/172866/Lukas Dagne Thesis.pdf?sequence=2&isAllowed=y.
[6] A. Bertolino and I. A. Faedo, "Software Testing Research : Achievements , Challenges , Dreams Software Testing Research : Achievements , Challenges , Dreams," no. September 2007, 2007.
[7] I. K. WAIROOY, "Software Testing." https://socs.binus.ac.id/2020/06/30/software-testing/ (accessed Oct. 28, 2021).
[8] Z. Zhang and L. Mei, "An improved method of acquiring basis path for software testing," *ICCSE 2010 - 5th Int. Conf. Comput. Sci. Educ. Final Progr. B. Abstr.*, no. 2, pp. 1891–1894, 2010, doi: 10.1109/ICCSE.2010.5593820.
[9] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The query-flow graph: Model and applications," *Int. Conf. Inf. Knowl. Manag. Proc.*, pp. 609–617, 2008, doi: 10.1145/1458082.1458163.
[10] C. Ebert and J. Cain, "Cyclomatic Complexity," *IEEE Softw.*, vol. 33, no. 6, pp. 27–29, 2016, doi: 10.1109/MS.2016.147.