BAB 4

IMPLEMETASI DAN PENGUJIAN

4.1. Implementasi Sistem

Tahap implementasi sistem merupakan tahap pembuatan perangkat lunak, tahap lanjut dari tahap perancangan sistem. Tahap yang dilakukan untuk menerjemahkan perancangan berdasarkan hasil analisis dalam bahasa yang dimengerti oleh komputer serta penerapan perangkat lunak pada keadaan yang sebenarnya.

4.1.1. Implementasi Perangkat Keras

Spesifikasi perangkat keras (hardware) yang digunakan untuk membangun perangkat lunak pencocokan string ini tercantum dalam Tabel 4-1 Spesifikasi Perangkat Keras berikut ini :

Tabel 4-1 Spesifikasi Perangkat Keras

NO	Perangkat Keras	Spesifikasi
1	Processor	Pentium Dual Core T4200 2.00GHz
2	RAM	1 Gb
3	Harddisk	250 Gb
4	VGA	512 Mb
5	Monitor	1024x768

4.1.2. Implementasi Perangkat Lunak

Spesifikasi perangkat keras (hardware) yang digunakan untuk membangun perangkat lunak pencocokan string ini tercantum dalam Tabel 4-2 Spesifikasi Perangkat Lunakberikut ini :

Tabel 4-2 Spesifikasi Perangkat Lunak

NO	Perangkat Lunak	Keterangan
1	Windows 7	Sistem Operasi
2	NetBeans	Tools Pembangun Perangkat Lunak
3	StarUML	Pemodelan Sistem

4.1.3. Implementasi Antarmuka

Implementasi antarmuka dilakukan untuk setiap tampilan program yang dibangun dalam bentuk file code. Berikut ini adalah implementasi antarmuka untuk admin dan pengguna. Berikut ini adalah tabel implementasi antarmuka pada perangkat lunak pencocokan String :

Tabel 4-3 Form Perangkat lunak

NO	Form Antarmuka	Proses	Nama File	Keterangan
1	Form Pencarian	Proses inputan string	UiBrowseBook.java	Tampilan untuk menginputkan kata yang dicari
2	Form Proses Pencarian	Proses pencarian string	UiProcessBook.java	Tampilan hasil pencarian

NO	**Form Antarmuka**	**Proses**	**Nama File**	Keterangan
3	Form Bantuan	Proses tentang bantuan	UiAboutDialog.java	Tampilan bantuan cara penggunaan prangkat lunak
4	Form Tentang	Proses tentang pembuat	UiHelpDialog.java	Tampilan tentang pembuat perangkat lunak

Gambar implementasi antarmuka dapat dilihat pada bagian lampiran A.

4.2. Pengujian Sistem

Pengujian Sistem merupakan hal terpenting yang bertujuan untuk menemukan kesalahan – kesalahan atau kekurangan – kekurangan pada perangkat lunak yang akan diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut. Pada tahap pengujian sistem ini terdiri dari beberapa bagian yaitu :

1. Rencana Pengujian
2. Pengujian *White Box*
3. Pengujian *Black Box*
4. Pengujian Beta
5. Kesimpulan Pengujian

4.2.1. Rencana Pengujian

Rencana pengujian perangkat lunak ini dibagi menjadi 2 bagian yaitu teknik pengujian black box dan teknik pengujian white box. Pengujian black box berfokus pada pengujian semua perintah-perintah dan kondisi yang dieksekusi oleh perangkat lunak dan pengujian ini dilakukan oleh pembangun aplikasi. Pengujian white box dilakukan untuk menguji function pada sistem perangkat lunak dan menguji apakah function tersebut memenuhi kebutuhan sistem. Strategi pengujian akan dilakukan dengan menggunakan strategi pengujian Beta. Strategi pengujian beta bertujuan untuk mengetahui pendapat user sebenarnya terhadap fungsi-fungsi program apakah sudah berjalan sesuai kebutuhan atau tidak. Strategi pengujian ini dilakukan dengan cara memberikan perangkat lunak kepada user yang telah ditentukan tanpa didampingi pembangun perangkat lunak.

Tabel 4-4 Rencana Pengujian

No	Item Uji	Detail Pengujian	Jenis Uji
1	Proses pencocokan algoritam Zhu Takaoka	Uji algoritma	White box
2	Proses pencocokan Karp-Rabin	Uji algoritma	White box
3	Login	Verifikasi data masuk pengguna dan sandi	Black Box
4	Pengolahan data buku	Tambah data buku	Black Box
		Ubah data buku	Black Box
		Hapus data buku	Black Box

5	Pencarian Kata	Pencarian kata	Black Box

4.2.2. Pengujian White Box

Pengujian *white box* bertujuan untuk mengetahui kinerja logika yang dibuat pada sebuah perangkat lunak apakah berjalan dengan baik atau tidak. Pengujian *white box* akan digunakan pada algoritma Zhu Takaoka dan Karp Rabin, untuk mengukur kinerja logika berdasarkan *pseudocode* yang telah dibuat pada tahap analisis.

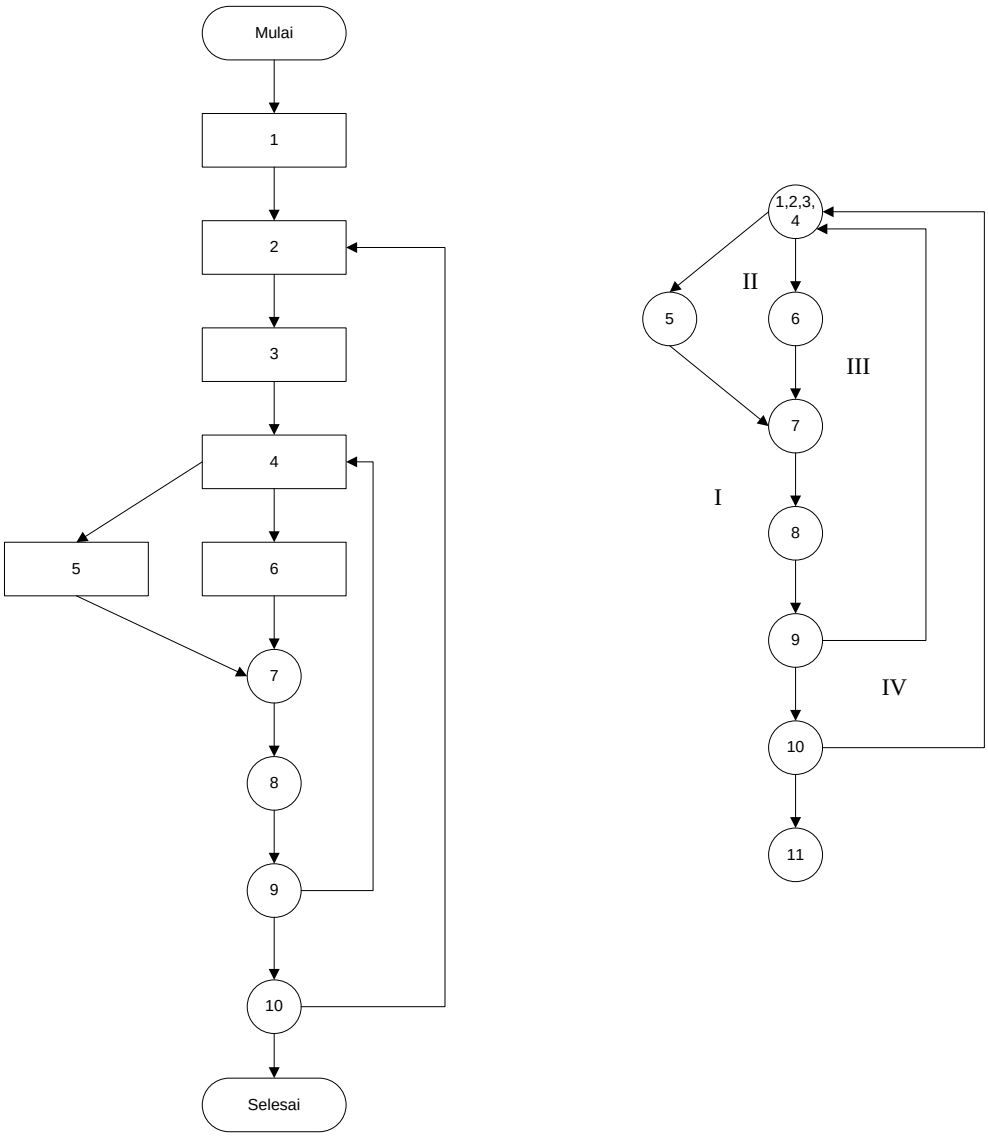
Pengujian *white box* dilakukan dengan beberapa tahap, yaitu :

1. Langkah pertama ubah *pseudocode* menjadi *flowchart.*
2. Ubah *flowchart* menjadi *flowgraph* ke dalam bentuk yang lebih sederhana.
3. Tahap pengujian, dimana tahap pengujian ini dilakukan dengan 5 cara yaitu, menghitung *region*, menghitung *Cyclomatic Complexity*, menghitung *independent path*, menggunakan *graph matriks*, menghitung *predicate node*.

4.2.2.1. Pengujian Algoritma Zhu-Takaoka

Pengujian dilakukan berdasarkan *pseudocode* pada **Error! Reference source not found.**

1. Pengubahan *pseudocode* menjadi *flowchart* kemudian menjadi *flowgarph*.



Gambar 4.1 Pengubahan *Flowchart* menjadi *Flowgraph* algoritma Zhu-Takaoka

2. Tahap Pengujian

   a. Region = 4

   b. $V_{(G)}$ = Edge – Node + 2

      = 10 – 8 + 2

      = 2 + 2

      = 4

c. *Independent Path*

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

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

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

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

d. *Predicate Node*

$V_{(G)}$ = Jumlah node yg memiliki lebih dari 1 jalur + 1

= 3 + 1

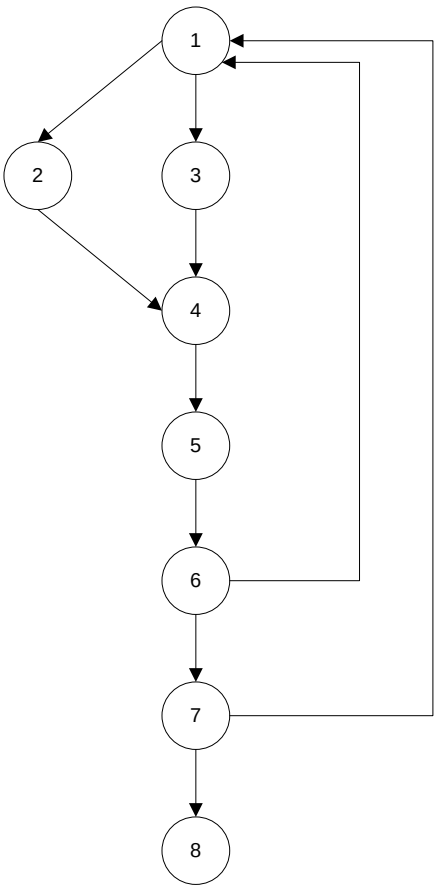
= 4

e. *Graph Matriks*

Graph dapat dilihat pada Gambar 4.2 Graph Matriks

Tabel 4-5 Graph Matriks Algoritma ZhuTakaoka

Node	1	2	3	4	5	6	7	8	
1		1	1						1
2				1					0
3				1					0
4					1				0
5						1			0
6	1						1		1
7	1							1	1
8									0
jumlah									3



Gambar 4.2 Graph Matriks Algoritma Zhu-Takaoka

$V_{(G)}$ = Jumlah *Graph Matriks* + 1

= 3 + 1

= 4

Tabel 4-6 Uji Coba Independent Path

Independent Path ke	**Penggunaan**	**Contoh data masukan**	**Yang Diharapkan**	**Hasil**
1	Ketika Predicate node 1 bernilai false	lampu	Proses looping berhenti	[x] Diterima [ ] Ditolak
2	Ketika Predicate node 1 bernilai True	sutan	Proses looping berhenti	[x] Diterima [ ] Ditolak

3	Ketika Predicate node 2 bernilai True	sutan	Proses looping berhenti	[x] Diterima [ ] Ditolak
4	Ketika Predicate node 3 bernilai True	sutan	Proses looping berhenti	[x] Diterima [ ] Ditolak

Langkah terakhir dalam melakukan pengujian menggunakan metoda basis path testing adalah menyiapkan kasus-kasus uji untuk mengeksekusi semua alur logika yang telah dibuat pada langkah sebelumnya. Kasus uji yang dimaksud disini adalah dengan cara memberikan nilai pada variable yang terlibat. Nilai yang dimasukkan haruslah nilai yang mungkin muncul dan sesuai dengan tipe data yang telah didefinisikan. Test case dibuat dalam bentuk tabel yang tujuannya untuk mempermudah eksekusi setiap basis path, karena nilai-nilai yang dimasukkan dan yang dihasilkan akan lebih mudah diamati.

1. Pengujian Path 1 = 1-2-3-4-5-7-8-9-10-11

   Pada path ini, supaya node-4 dapat menuju ke node 5, maka nilai dari node 4 harus dapat menghasilkan nilai *false*. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari buku tersebut tidak mengandung kata lampu. Agar bernilai *false* digunakan kata lampu sebagai data uji coba.

2. Pengujian Path 2 = 1-2-3-4-6-7-8-9-10-11

   Pada path ini, supaya node-4 dapat menuju ke node 6, maka nilai dari node 4 harus dapat menghasilkan nilai *true*. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari

buku tersebut mengandung kata sutan. Agar bernilai *true* digunakan kata sutan sebagai data uji coba.

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

   Pada path ini, supaya node-9 dapat menuju ke node 4, maka nilai dari node 9 harus dapat menghasilkan nilai *true* agar *looping* dapat berhenti. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari buku tersebut mengandung kata sutan. Agar bernilai *true* digunakan kata sutan sebagai data uji coba.

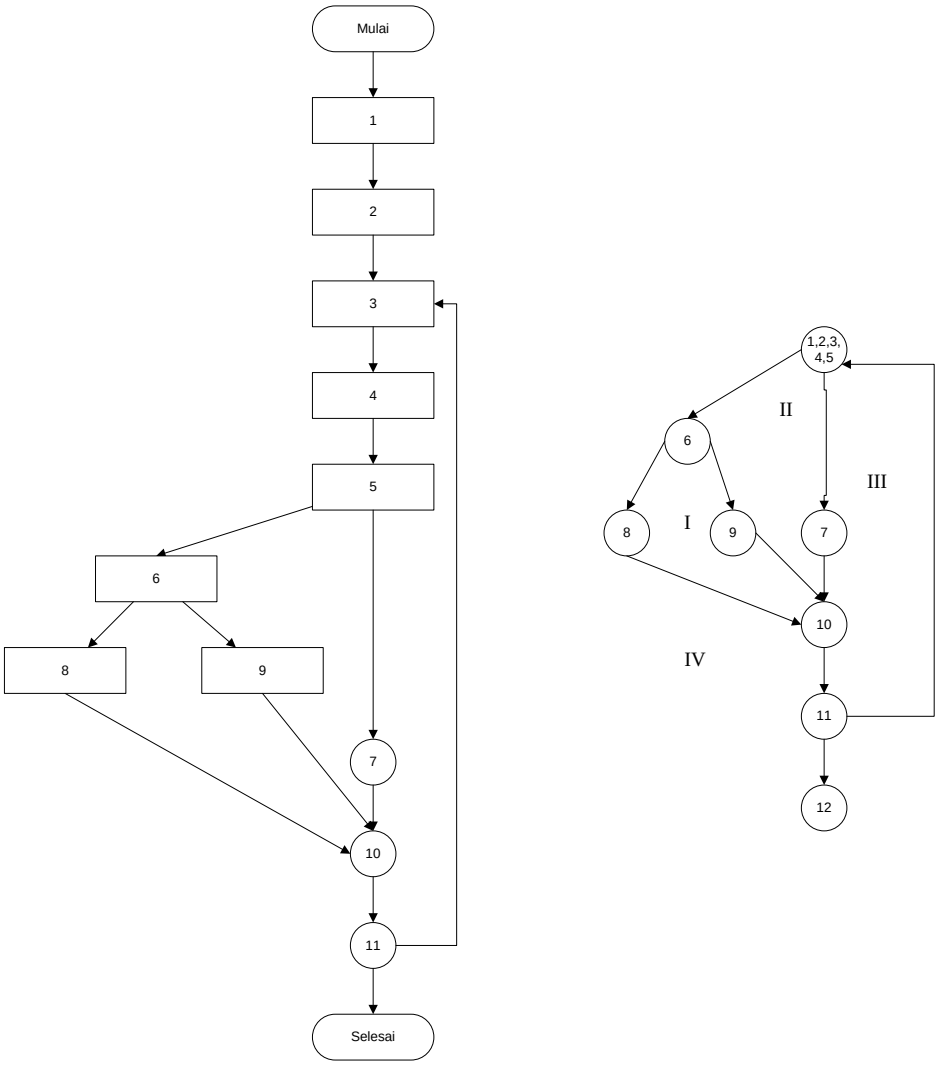
4. Pengujian Path 4 = 1-2-3-4-5-7-8-9-10-2-3-4-5-7-8-9-10-11

   Pada path ini, supaya node-9 dapat menuju ke node 4, maka nilai dari node 9 harus dapat menghasilkan nilai *true* agar *looping* dapat berhenti. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari buku tersebut mengandung kata sutan. Agar bernilai *true* digunakan kata sutan sebagai data uji coba.

4.2.2.2. Pengujian Algoritma Karp-Rabin

Pengujian dilakukan berdasarkan *pseudocode* pada **Error! Reference source not found.**

1. Pengubahan *pseudocode* menjadi *flowchart* kemudian menjadi *flowgarph*.



Gambar 4.3 Pengubahan Flowchart menjadi Flowgraph algoritma Karp-Rabin

2. Tahap Pengujian

   a. Region = 4

   b. $V_{(G)}$ = Edge – Node + 2

      = 10 – 8 + 2

      = 2 + 2

      = 4

   c. *Independent Path*

Path 1 = 1-2-3-4-5-6-8-10-11-12

Path 2 = 1-2-3-4-5-6-9- 10-11-12

Path 3 = 1-2-3-4-5-7-10-11-1-2-3-4-5-7-10-11-12

Path 4 = 1-2-3-4-5-7-10-11-12

d. *Predicate Node*

$V_{(G)}$ = Jumlah node yg memiliki lebih dari 1 jalur + 1

= 3 + 1

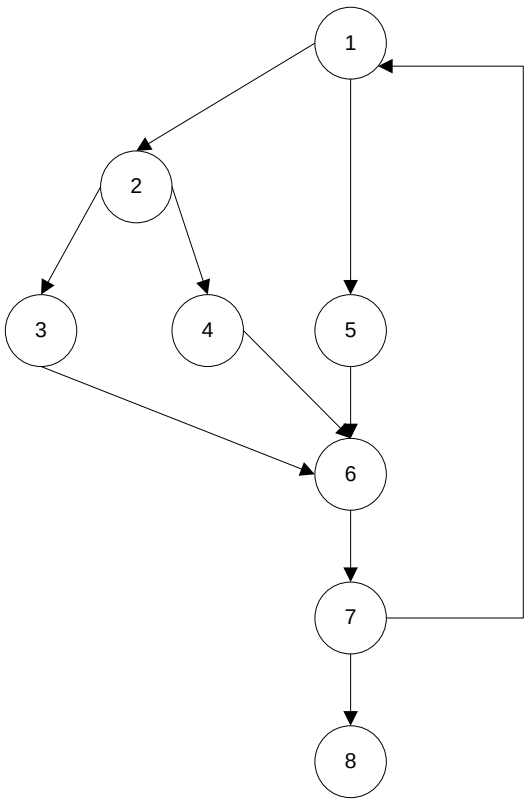
= 4

e. *Graph Matriks*

Graph dapat dilihat pada

Tabel 4-7 Graph Matriks Algoritma Karp-Rabin

Node	1	2	3	4	5	6	7	8	
1		1			1				1
2			1	1					1
3						1			0
4						1			0
5						1			0
6							1		0
7	1							1	1
8									
jumlah									3



Gambar 4.4 Graph Matriks Algoritma Karp-Rabin

$V_{(G)}$ = Jumlah *Graph Matriks* + 1

= 3 + 1

= 4

Tabel 4-8 uji coba path algoritma Zhu-Takaoka

Independent Path ke	**Penggunaan**	**Contoh data masukan**	**Yang Diharapkan**	**Hasil**
1	Ketika predicate node ke 1 bernilai false	lampu	Proses looping berhenti	[x] Diterima [ ] Ditolak
2	Ketika predicate node ke 1 bernilai True	sutan	Proses looping berhenti	[x] Diterima [ ] Ditolak

3	Ketika predicate node ke 3 bernilai True	sutan	Proses looping berhenti	[x] Diterima [ ] Ditolak
4	Ketika predicate node ke 2 bernilai True	sutan	Proses looping berhenti	[x] Diterima [ ] Ditolak

1. Pengujian Path 1 = 1-2-3-4-5-6-8-10-11-12

   Pada path ini, supaya node-6 dapat menuju ke node 8, maka nilai dari node 6 harus dapat menghasilkan nilai *false*. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari buku tersebut tidak mengandung kata lampu. Agar bernilai *false* digunakan kata lampu sebagai data uji coba.

2. Pengujian Path 2 = 1-2-3-4-5-6-9- 10-11-12

   Pada path ini, supaya node-5 dapat menuju ke node 6, maka nilai dari node 5 harus dapat menghasilkan nilai *true* agar *looping* dapat berhenti. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari buku tersebut mengandung kata sutan. Agar bernilai *true* digunakan kata sutan sebagai data uji coba.

3. Pengujian Path 3 = 1-2-3-4-5-7-10-11-1-2-3-4-5-7-10-11-12

   Pada path ini, supaya node-11 dapat menuju ke node 1, maka nilai dari node 11 harus dapat menghasilkan nilai *true* agar *looping* dapat berhenti. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari buku tersebut mengandung kata sutan. Agar bernilai *true* digunakan kata sutan sebagai data uji coba.

4. Pengujian Path 4 = 1-2-3-4-5-7-10-11-12

   Pada path ini, supaya node-5 dapat menuju ke node 7, maka nilai dari node 9 harus dapat menghasilkan nilai *true* agar *looping* dapat berhenti. Maka data uji coba diasumsikan terdapat deskripsi buku dengan jumlah 3000 karakter dimana didalam deskripsi dari buku tersebut mengandung kata sutan. Agar bernilai *true* digunakan kata sutan sebagai data uji coba.

3. Kesimpulan

Berdasarkan pengujian pada setiap metode, dihasilkan nilai Cyclomatic Complexity yang sama yaitu 4. Maka dapat disimpulkan bahwa pengujian white box pada algoritma Karp-Rabin berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.3. Pengujian Black Box

Berdasarkan rencana pengujian yang telah disusun, maka dapat dilakukan pengujian sebagai berikut :

Tabel 4-9 Pengujian Pencarian

Kasus dan Hasil Uji (Data Normal)			
Data Masukan	***Yang Diharapkan***	***Pengamatan***	***Kesimpulan***
Form pencarian diisi dengan : dan	Data hasil pencarian berdasarkan kata kunci “dan” muncul	Muncul hasil pencarian berdasarkan kata kunci “dan”.	[✓] Diterima [ ] Ditolak

Kasus dan Hasil Uji (Data Normal)			
Data Masukan	***Yang Diharapkan***	***Pengamatan***	***Kesimpulan***
Form pencarian diisi dengan : 1945	Data hasil pencarian berdasarkan kata kunci “1945” muncul	Muncul hasil pencarian berdasarkan kata kunci “1945”.	[✓] Diterima [ ] Ditolak
Kasus dan Hasil Uji (Data Kosong)			
Data Masukan	***Yang Diharapkan***	***Pengamatan***	***Kesimpulan***
Text field pencarian kosong	Muncul pesan kesalahan “teks yang dicari kosong”	Dapat menampilkan pesan kesalahan “teks yang dicari kosong.”	[✓] Diterima [ ] Ditolak

4.2.3.1. Pengujian Performansi

Pada pengujian performansi kedua algoritma ini mendapatkan beberapa inputan yang berbeda dari tiap-tiap percobaan yaitu

1. Kata huruf kecil semua,
2. Kata dengan huruf besar semua
3. Angka
4. Tiga suku kata.

Di uji coba teradap deskripsi buku dengan jumlah karakter sebanyak 1858.

Tabel berikut akan menjelaskan hasil ujicoba.

Tabel 4-10 hasil uji coba algoritma Zhu-Takaoka

No	Inputan	Panjang sumber karakter	Posisi ditemukan (karakter ke)	Algotima Zhu-Takaoka	
				Waktu (millisecond)	Memori (kb)
1	prajurit	1858	1479	124	724
2	TKR	1858	1125	118	221
3	1945	1858	925	110	518
4	Mayor Daan Mogot	1858	1147	106	668

Untuk selanjutnya uji performansi algoritma Karp-Rabin

Tabel 4-11 hasil uji coba algoritma Karp-Rabin

No	Inputan	Panjang sumber (karakter)	Posisi ditemukan (karakter ke)	Algotima Karp-Rabin	
				Waktu (millisecond)	Memori (kb)
1	prajurit	1858	1479	95	102
2	TKR	1858	1125	105	142
3	1945	1858	925	100	234
4	Mayor Daan Mogot	1858	1147	104	538

4.2.4. Kesimpulan Pengujian.

Berdasarkan hasil pengujian terhadap perangkat lunak dapat ditarik kesimpulan perangkat lunak dapat berfungsi sesuai dengan yang diharapkan serta perangkat lunak yang dibangun telah cukup memenuhi tujuan awal pembangunan dan algoritma Karp-Rabin cendrung lebih cepat dan efesien dalam pencarian kata.