BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1. Implementasi Sistem

Tahap implementasi merupakan tahap kelanjutan dari kegiatan perancangan sistem. Wujud dari hasil implementasi ini nantinya adalah sebuah sistem yang siap untuk diuji dan digunakan.

4.1.1. Implementasi Perangkat Keras

Spesifikasi perangkat keras (*hardware*) yang digunakan untuk mengimplementasikan aplikasi *game puzzle and the solver* dapat dilihat pada tabel 4.1.

Tabel 4. 1 Perangkat keras yang digunakan

Perangkat keras	Keterangan
Processor	1.83 GHz
Memory	DDR2 2 GB
Harddisk	160 GB HDD
Video Card	*onBoard*
Monitor	Resolusi 1152x864
Mouse	Optic 1 buah
Keyboard	1 buah
Speaker	Simbadda cst5000

4.1.2. Implementasi Perangkat Lunak

Spesifikasi perangkat lunak (*software*) yang digunakan untuk mengimplementasikan aplikasi *game puzzle and the solver* dapat dilihat pada tabel 4.2.

Tabel 4. 2 Perangkat lunak yang digunakan

Perangkat Lunak	Keterangan
Windows 7	Sebagai sistem operasi

4.2. Pengujian Sistem

Pengujian 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.

4.2.1. Rencana Pengujian

Pengujian yang dilakukan yaitu pengujian *white box*, pengujian *black box* dan pengujian beta. Pengujian *white-box* digunakan untuk menguji sistem. Pengujian *white-box* digunakan untuk meyakinkan semua perintah dan kondisi dieksekusi secara minimal. Pengujian *Black Box* Digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak yang dirancang. Pengujian beta digunakan untuk mengetahui tanggapan user terhadap aplikasi *game*, dengan melakukan kuisioner.

Tabel 4. 3 Rencana Pengujian

No	Kelas Uji	Butir Uji	Jenis Pengujian
1	Algoritma Greedy	Uji algoritma kuis tebak kata	*White box*
2	Metode Path finding	Uji Metode path finding menggunakan algoritma A star	*White box*
3	Menu Utama	Memilih Mulai *Game*	*Black Box*
		Memilih Petunjuk	*Black Box*
		Memilih Kredit Pengembang	*Black Box*
		Memilih Keluar	*Black Box*
4	Mulai *Game*	Ketika memilih tombol Mulai *Game*	*Black Box*
		Tombol jeda permainan	*Black Box*
5	Permainan *stage* Minang	Kuis tebak kata stage Minang	*Black Box*
		Mendapatkan inventory Bintang	*Black Box*
6	Permainan *stage* Sunda	Kuis tebak kata stage Sunda	*Black Box*
		Mendapatkan inventory Bintang	*Black Box*
7	Permainan *stage* Papua	Kuis tebak kata stage Papua	*Black Box*
		Mendapatkan inventory Bintang	*Black Box*
8	Petunjuk	Ketika memilih tombol Petunjuk	*Black Box*
9	Kredit Pengembang	Informasi Pengembang	*Black Box*
10	Menu Keluar	Ketika memilih tombol keluar	*Black Box*

4.2.2. Pengujian *White Box*

Pengujian *white-box* digunakan untuk meyakinkan semua perintah dan kondisi pada aplikasi dieksekusi secara minimal. Pengujian *white-box* menggunakan dua *tools* yaitu *flow graph* yang digunakan untuk menggambarkan alur dari algoritma dan *graph matrix* yang digunakan untuk menggenerasi *flow graph*. Adapun pengujian *white-box*nya adalah sebagai berikut:

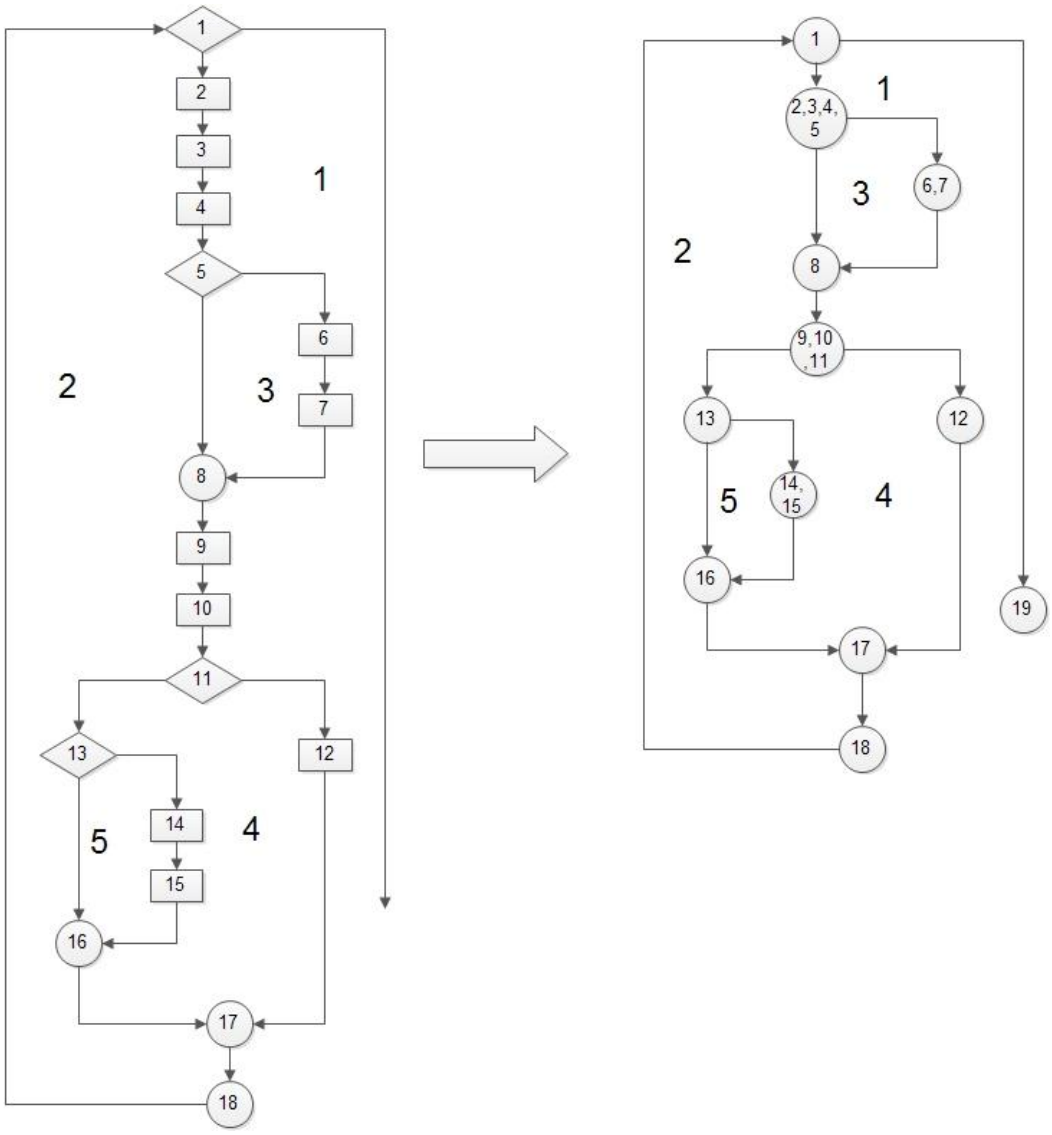
4.2.1.1. Pengujian Algoritma *Greedy* pada kuis Tebak Kata

a. Pengubahan *pseudocode* Algoritma Greedy menjadi *flowchart* kemudian menjadi *flow graph*

   Berikut *pseudocode* algoritma *Greedy* yang digunakan:

```
1.  while salah < 3
2.  x = 0
3.  ada = 0
4.  bandingan = copy(jawaban, x, 1)
5.  if kata = bandingan
6.  replace(copy_jawaban, x, 1) = kata
7.  ada = ada + 1
8.  endif
9.  x = x + 1
10. salah = 0
11. if ada != 0
12. inputkan lagi nilai ke variabel kata
13. else if ada = 0
14. salah = salah + 1
15. inputkan lagi nilai ke variabel kata
16. endif
17. endif
18. endwhile
```

Pengubahan *Flowchart* menjadi *Flow graph* algoritma *Greedy* yang digunakan dapat dilihat pada gambar 4.1.



Gambar 4. 1 Pengubahan *Flowchart* menjadi *Flow Graph* Algoritma *Greedy*

Keterangan gambar 4.1 :

= Menggambarkan kondisi

= Menggambarkan aksi

Dari Gambar 4.1 dapat dihitung *cyclomatic complexity* sebagai berikut :

V(G) = E – N + 2

V(G) = 15 – 12 + 2

V(G) = 3 + 2

V(G) = 5

Di mana :

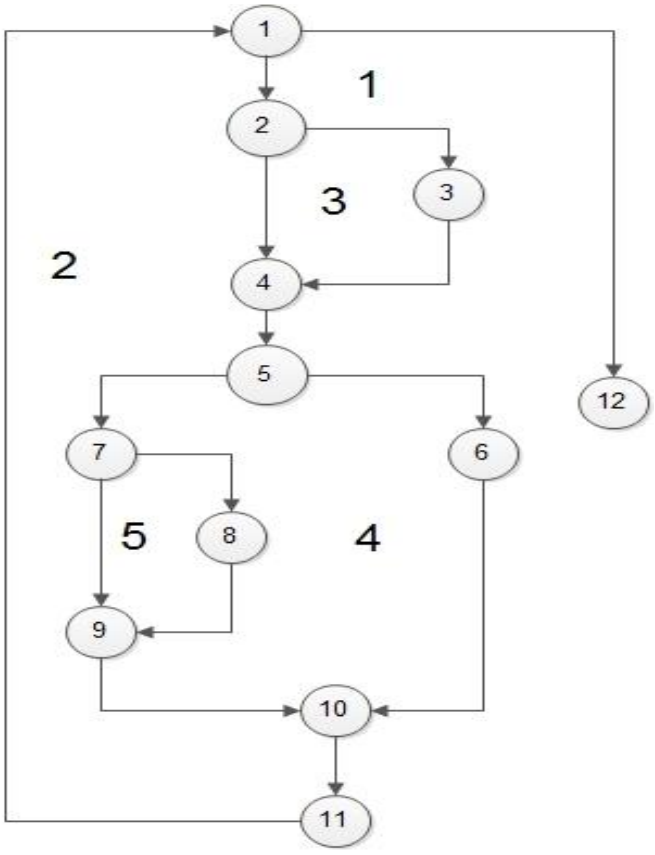
E = jumlah *edge* pada grafik alir

N = jumlah *node* pada grafik alir

Jadi, *cyclomatic complexity* untuk gambar 4.1 adalah 5. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 5 *path* yang terdiri dari :

Path 1	1, 2,3,4,5,6,7, 8,9,10,11,12,17,18,1,19
Path 2	1,2,3,4,5,8,9,10,11,13,16,17,18,1,19
Path 3	1,2,3,4,5,6,7,8,9,10,11,12,17,18,1,19
Path 4	1,2,3,4,5,8,9,10,11,12,13,14,15,16,17,18,1,19
Path 5	1,2,3,4,5,8,9,10,11,13,14,15,16,17,18,1,19

b. Penomoran ulang *flow graph*



Gambar 4. 2 Penomoran ulang *Flow Graph*

c. Graph Matrix

Tabel 4. 4 *Graph Matrix* Algoritma *Greedy*

N	1	2	3	4	5	6	7	8	9	10	11	12	
1		1										1	1
2			1	1									1
3				1									0
4					1								0
5						1	1						1
6										1			0
7								1	1				1
8									1				0
9										1			0
10													0
11	1												0
12													0
												Jumlah +1	5

d. Pengujian *Path* Algoritma *Greedy*

Tabel 4. 5 Pengujian *Path* Algoritma *Greedy*

No	Kasus Uji	Hasil Yang Diharapkan	Hasil Uji Kasus	Keterangan

1	`Salah => 3`
	
	`While salah < 3`
	`:`
	`endwhile`
		Perulangan tidak dilakukan	Salah => 3
			Perulangan tidak dilakukan
				[✓] Alur Terlewati
				[ ] Alur Tidak Terlewati

2	`Memberikan nilai jawaban = TES`
	`Memasukan nilai kandidat = E`
	
	`if kata = bandingan`
	`replace(copy_jawaban, x, 1) = kata`
	`ada = ada + 1`
	`endif`
	`x = x + 1`
	`salah = 0`
	`if ada != 0`
	`inputkan lagi nilai ke`
		Memasukan nilai kandidat= E kedalam jawaban, dan kesalahan = 0	Nilai nilai = E tampil
			-E- . salah = 0
				[✓] Alur Terlewati
				[ ] Alur Tidak Terlewati

	variabel kata else if ada = 0 salah = salah + 1 inputkan lagi nilai ke variabel kata endif endif			
3	Memberikan nilai jawaban = TES Memasukan nilai kandidat = E if kata = bandingan replace(copy_jawaban, x, 1) = kata ada = ada + 1	Memasukan kandidat = E ke dalam himpunan solusi	kandidat =E dimasukan kedalam kandidat solusi dengan nilai = -E-	[✓] Alur Terlewati [ ] Alur Tidak Terlewati
4	Memberikan nilai jawaban = TES Memasukan nilai kandidat = S if ada != 0 inputkan lagi nilai ke variabel kata endif	Memasukan ke dalam jawaban.	Kandidat = S dimasuk dan ditampilkan. Jawaban = -ES	[✓] Alur Terlewati [ ] Alur Tidak Terlewati
5	Memberikan nilai jawaban = TES Memasukan nilai kandidat = Z else if ada = 0 salah = salah + 1 inputkan lagi nilai ke variabel kata endif	Memasukan inputan sebagai kesalahan	Z tidak ada, Salah = +1	[✓] Alur Terlewati [ ] Alur Tidak Terlewati

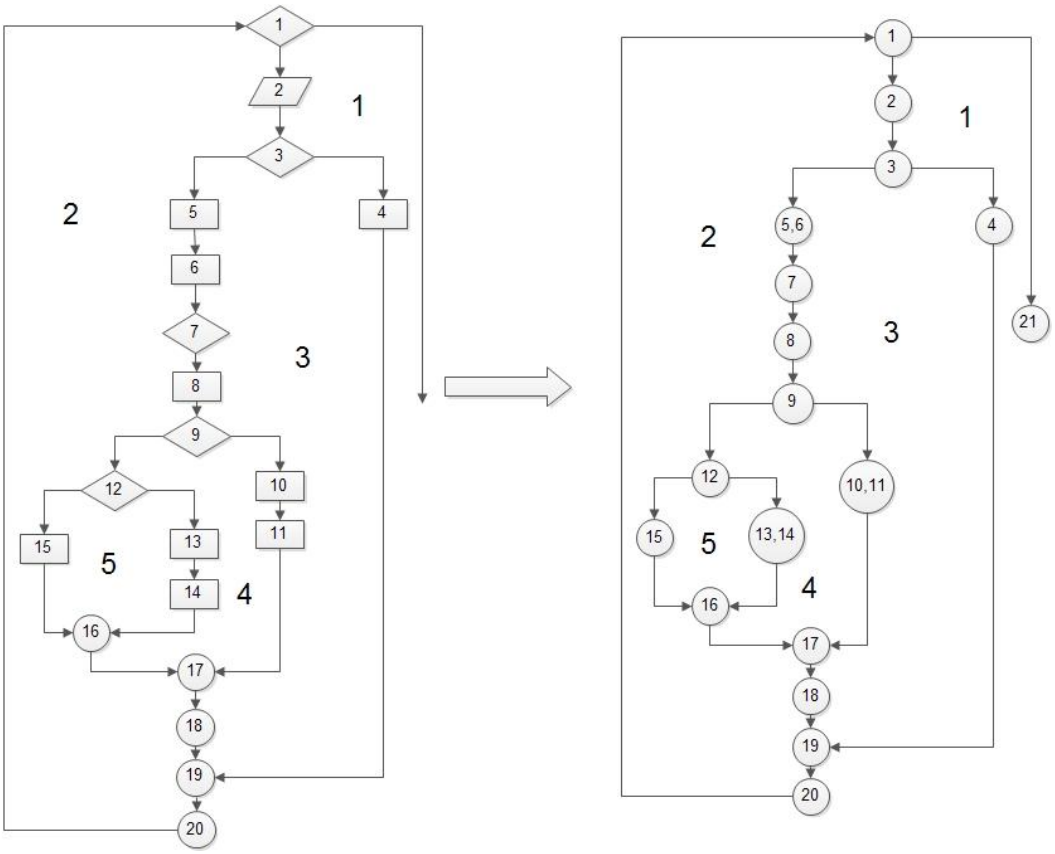
4.2.1.2. Pengujian Metode *Path Finding* pada musuh

a. Pengubahan *pseudocode* menjadi *Flowchart* kemudian menjadi *Flow Graph* Metode *Path Finding*

Berikut adalah *pseudocode* metode *path finding* dengan menggunakan algoritma A *star* dengan heuristik *manhattan distance.*

```
1: while openList tidak kosong
2: Get node n dari openList dengan f(n) terkecil
3: if n = node_tujuan
4: node_tujuan ditemukan
5/6: else masukan n ke dalam close list dan pertimbangkan node
     sebelahnya (n').
7: for setiap node n' dari n
8: Set parent dari  n' ke n
   Set h(n') (memperkirakan jarak heuristik node_tujuan)
   Set g(n') (ditambah cost untuk sampai dari n ke n' )
   Set f(n') = g(n') + h(n')
9: if n' ada di openList dengan f(n') lebih kecil
10: then update n' dengan nilai f(n') terendah yang baru
11: change parent dari n' ke node sekarang
12: else if n' ada di closeList dengan f(n') lebih kecil
13: then update n' dengan nilai f(n') terendah yang baru
14: change parent dari n' ke node sekarang
15: else Add node sekarang ke openList
16: endif
17: endif
18: endfor
19: endif
20: endwhile
```

Flowchart menjadi *Flow Graph* metode *path finding* dapat dilihat pada gambar 4.3.



Gambar 4. 3 *flow graph* metode *path finding*

Keterangan gambar 4.3 :

= Menggambarkan kondisi

= Menggambarkan aksi

Dari Gambar 4.2 dapat dihitung *cyclomatic complexity* sebagai berikut :

V(G) = E – N + 2

V(G) = 21 – 18 + 2

V(G) = 3 + 2

V(G) = 5

Di mana :

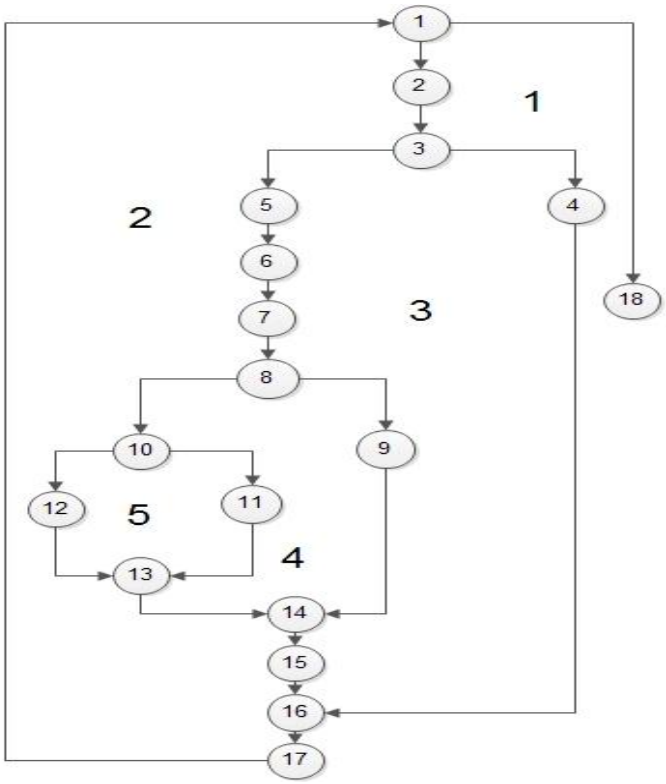
E = jumlah *edge* pada grafik alir

N = jumlah *node* pada grafik alir

Jadi, *cyclomatic complexity* untuk gambar 4.3 adalah 5. Berdasarkan *cyclomatic complexity* tersebut, maka terdapat 5 *path* yang terdiri dari :

Path 1 : 1,2,3,4,19,20,1,21
Path 2 : 1,2,3,5,6,7,8,9,12,15,16,17,18,19,20,1,21
Path 3 : 1,2,3,5,6,7,8,9,10,11,17,18,19,20,1,21
Path 4 : 1,2,3,5,6,7,8,9,10,11,12,13,14,16,17,18,19,20,1,21
Path 5 : 1,2,3,5,6,7,8,9,12,13,14,15,16,17,18,19,20,1,21

b. Penomoran ulang *Flow Graph*



Gambar 4. 4 Penomoran ulang *flow graph*

c. *Graph Matrix*

Tabel 4. 6 *Graph Matrix* Metode *Path Finding*

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
1		1																	0
2			1																0
3				1	1														1
4																1			0
5						1													0
6							1												0

7								1										0
8									1	1								1
9										1				1				1
10											1	1						1
11													1					0
12													1					0
13														1				0
14															1			0
15																1		0
16																	1	0
17	1																	0
18																		0
																	Jumlah	5

d. Pengujian *path*

Tabel 4. 7 Pengujian *path* Metode *Path Finding*

No	Kasus uji	Hasil yang diharapkan	Hasil uji	Keterangan

1	Memberikan nilai Node_tujuan = (3,3) *Unwalkable* = (0,2),(1,2),(2,2) Memasukan node_awal ke dalam open list
	
	While open list tidak kosong do
	:
	endwhile
		Melakukan perulangan ketika open list tidak kosong, dan menemukan tujuan kemudian berhenti	Node_tujuan di temukan pada node (3,3)	[✓] Alur Terlewati
				[ ] Alur Tidak Terlewati

2	Memberikan nilai *Unwalkable* = (0,2),(1,2),(2,2) Memasukan node_awal ke dalam open list Menentukan node_tujuan =(3,3) if n = node_tujuan node_tujuan ditemukan else masukan n ke dalam close list dan pertimbangkan node sebelahnya (n'). endif	Menentukan node n' jika n tidak menemukan node tujuan.	Node n' ditentukan n'= (0,0),(1,0), (1,1)	[✓] Alur Terlewati [ ] Alur Tidak Terlewati

3	Memberikan nilai node_awal = (0,1) node_tujuan =(3,3) *Unwalkable* = (0,2),(1,2),(2,2) Memberikan nilai pada n' = (0,0),(1,0),(1,1) for setiap node n' Set parent dari n' ke n	Menemukan *parent* dari setiap nilai n'	menemukan node dengan nilai (1,1)	[✓] Alur Terlewati [ ] Alur Tidak Terlewati
4	Memberikan nilai node_tujuan =(3,3) *Unwalkable* = (0,2),(1,2),(2,2) n'= (0,2) if n' ada di openList dengan f(n') lebih kecil then update n' dengan nilai f(n') terendah yang baru change parent dari n' ke node sekarang	Menemukan *parent* dari setiap nilai n'.	Menentukan node n' dengan nilai (2,1)	[✓] Alur Terlewati [ ] Alur Tidak Terlewati

5	Memberikan nilai
	Unwalkable = (0,2),(1,2),(2,2)
	Memasukan node_awal ke dalam open list
	Menentukan node_tujuan =(3,3)
	While open list tidak kosong do
	:
	endwhile
		Tidak melakukan perulangan	Tidak melakukan perulangan, open list kosong	[✓] Alur Terlewati
				[ ] Alur Tidak Terlewati

4.2.3. Pengujian *Black Box*

Pengujian *fungsional* yang digunakan untuk menguji sistem yang baru adalah metode pengujian alpha. Metode yang digunakan dalam pengujian ini adalah pengujian *black box* yang berfokus pada persyaratan fungsional dari sistem yang dibangun.

4.2.3.1. Kasus dan Hasil Pengujian

1. Pengujian Menu Utama

Tabel 4. 8 Pengujian Menu Utama

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Menu Utama	Memilih Tombol Mulai *Game*	Menampilkan Area Permainan	[√] Berhasil [ ] Tidak Berhasil
	Memilih Tombol Petunjuk	Menampilkan Petunjuk	
	Memilih Tombol Kredit Pengembang	Menampilkan Kredit Pengembang	
	Memilih Tombol Keluar	Keluar dari permaian	

2. Pengujian Mulai *Game*

Tabel 4. 9 Pengujian Mulai *Game*

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Mulai *Game*	Ketika pemain memilih tombol mulai *game*	Pemain berada di Area permainan	[√] Berhasil [ ] Tidak Berhasil
	Tombol jeda permainan	Jeda permainan	

3. Pengujian Permainan *stage* Minang

Tabel 4. 10 Pengujian Permainan *stage* Minang

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Permainan *stage* Minang	Kuis tebak kata *stage* Minang	Soal dan jawaban *stage* Minang dapat ditampilkan	[√] Berhasil [ ] Tidak Berhasil
	Mendapatkan *inventory*	Bintang yang didapatkan dimasukan	

	Bintang	kedalam *inventory*	

4. Pengujian Permainan *stage* Sunda

Tabel 4. 11 Pengujian Permainan *stage* Sunda

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Permainan *stage* Sunda	Kuis tebak kata *stage* Sunda	Soal dan jawaban *stage* Sunda dapat ditampilkan	[√] Berhasil [ ] Tidak Berhasil
	Mendapatkan *inventory* Bintang	Bintang yang didapatkan dimasukan kedalam *inventory*	

5. Pengujian Permainan *stage* Papua

Tabel 4. 12 Pengujian Permainan *stage* Papua

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Permainan *stage* Papua	Kuis tebak kata *stage* Papua	Soal dan jawaban *stage* Papua dapat ditampilkan	[√] Berhasil [ ] Tidak Berhasil
	Mendapatkan *inventory* Bintang	Bintang yang didapatkan dimasukan kedalam *inventory*	

6. Pengujian Petunjuk

Tabel 4. 13 Pengujian Petunjuk

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Petunjuk	Ketika Pemain memilih tombol petunjuk	Menampilkan papan petunjuk	[√] Berhasil [ ] Tidak Berhasil
	Tombol Kembali	Kembali Ke Menu Utama	

7. Pengujian Kredit Pengembang

Tabel 4. 14 Pengujian Kredit Pengembang

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Kredit Pengembang	Informasi Pengembang	Menampilkan informasi Pengembang	[√] Berhasil [ ] Tidak Berhasil

8. Pengujian Menu Keluar

Tabel 4. 15 Pengujian Menu Keluar

Kelas Uji	Skenario Uji	Hasil yang diharapkan	Kesimpulan
Menu Keluar	Ketika memilih tombol keluar	Keluar dari aplikasi	[√] Berhasil [ ] Tidak Berhasil

4.2.4. Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif dimana diuji secara langsung ke lapangan, dengan menggunakan kuesioner mengenai tanggapan *user* terhadap *game* yang telah dibangun. Kuesioner disebarkan menggunakan teknik *Simple Random Sampling* dimana anggota *sample* dari populasi dilakukan secara acak tanpa memperhatikan strata yang ada dalam populasi. Kuesioner disebarkan kepada 30 orang siswa-siswi sekolah dasar dengan rentang usia antara 6 tahun sampai 11 tahun. Kuesioner terdiri dari 5 pertanyaan dengan sistem penskoran menggunakan skala pengukuran teknik *Likert*, berikut adalah skor untuk jawaban kuesioner :

Tabel 4. 16 Skala Kuesioner

Skala Jawaban	Keterangan	Skor
S	Setuju	3
BS	Biasa saja	2
TS	Tidak Setuju	1

Tabel diatas adalah jawaban dan skor yang diberikan dari setiap pertanyaan kuisioner yang akan dibagikan kepada *user*.

Data yang diperoleh dari pemberian kuesioner kepada responden dapat dianalisis dengan menghitung rata-rata jawaban berdasarkan scoring setiap jawaban dari responden, analisis yang dapat dilakukan yaitu sebagai berikut.

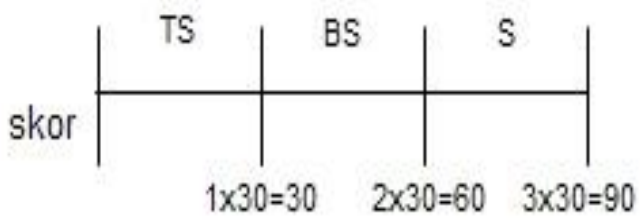
Jumlah skor ideal untuk seluruh item = 3 × jumlah responden

Rata-rata skor dalam persentase :

(jumlah skor : jumlah skor ideal) × 100%

jumlah skor ideal = 3×30 = 90

Secara kontinum dapat digambarkan seperti gambar 4.3.



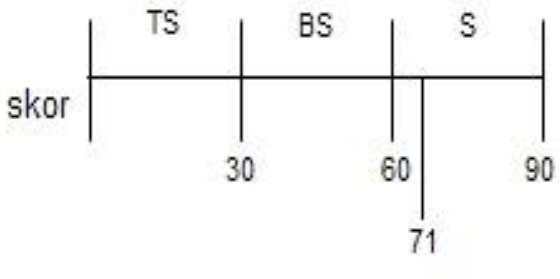
Gambar 4. 5 Skor kontinum

Adapun pertanyaan dan hasil observasi yang telah dibagikan kepada responden adalah sebagai berikut :

1. Apakah game ini dapat menambah pengetahuan adik-adik tentang kebudayaan Sumatera Barat, Jawa Barat dan Papua?

Kode Jawaban	**S**	**BS**	**TS**
Frequensi Jawaban	14	13	3
Skor	14 x 3 = 42	13 x 2 = 26	3 x 1 = 3
Jumlah Skor	71		
Persentase	71/90 * 100% = 78,9%		

Garis Kontinum



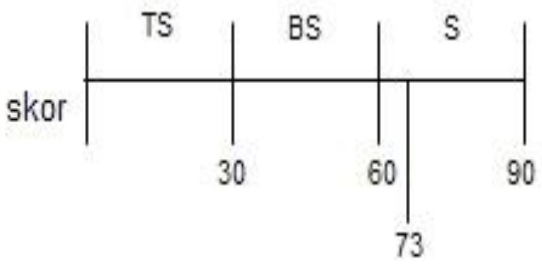
Gambar 4. 6 Garis Kontinum Kuesioner no.1

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap bisa tidaknya aplikasi ini menjadi media alternative pengenalan dan pembelajaran budaya Indonesia adalah 71 dari yang diharapkan adalah 90, atau dapat dikategorikan setuju.

2. Menurut adik-adik, apakah permainan kuis tebak kata yang ada didalam game ini menarik untuk dimainkan?

Kode Jawaban	S	BS	TS
Frequensi Jawaban	15	13	2
Skor	15 x 3 = 45	13 x 2 = 26	2 x 1 = 2
Jumlah Skor	73		
Persentase	73/90 * 100% = 81,1%		

Garis kontinum



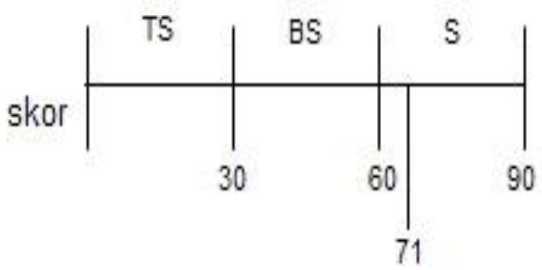
Gambar 4. 7 Garis kontinum Kuesioner no.2

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap bisa tidaknya aplikasi ini menjadikan pengguna tertarik dengan permainan tebak kata adalah 73 dari yang diharapkan adalah 90, atau dapat dikategorikan setuju.

3. Setelah memainkan game ini, apakah adik-adik lebih mengenal kebudayaan Sumatera Barat, Jawa Barat dan Papua?

Kode Jawaban	S	BS	TS
Frequensi Jawaban	14	13	3
Skor	14 x 3 = 42	13 x 2 = 26	3 x 1 = 3
Jumlah Skor	71		
Persentase	71/90 * 100% = 78,9%		

Garis kontinum



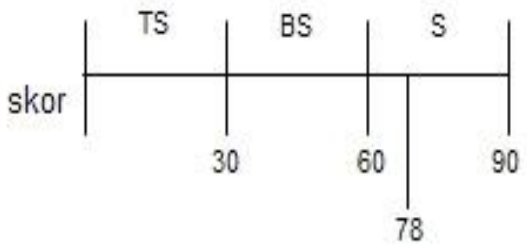
Gambar 4. 8 Garis kontinum kuesioner no.3

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap bisa tidaknya aplikasi ini meningkatkan pengetahuan tentang budaya Indonesia adalah 71 dari yang diharapkan adalah 90, atau dapat dikategorikan setuju.

4. Apakah adik-adik tertarik untuk memainkan lagi game puzzle and the solver ini?

Kode Jawaban	**S**	**BS**	**TS**
Frequensi Jawaban	19	10	1
Skor	19 x 3 = 57	10 x 2 = 20	1 x 1 = 1
Jumlah Skor	78		
Persentase	78/90 * 100% = 86,7%		

Garis kontinum



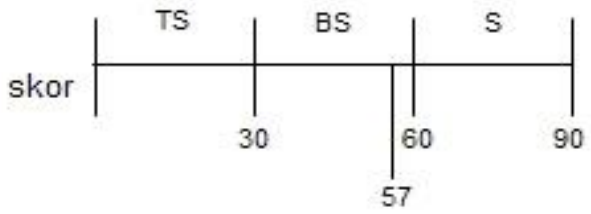
Gambar 4. 9 Garis kontinum kuesioner no.4

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap menarik atau tidaknya aplikasi ini adalah 78 dari yang diharapkan adalah 90, atau dapat dikategorikan setuju.

5. Menurut adik-adik apakah tampilan game ini menarik?

Kode Jawaban	**S**	**BS**	**TS**
Frequensi Jawaban	12	13	5
Skor	12 x 3 = 26	13 x 2 = 26	5 x 1 = 5
Jumlah Skor	57		
Persentase	57/90 * 100% = 63,3%		

Garis Kontinum



Gambar 4. 10 Garis kontinum kuesioner no.5

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap tidak bisanya aplikasi ini dijadikan alternatif pengenalan dan pembelajaran budaya Indonesia adalah 57 dari yang diharapkan adalah 90, atau dapat dikategorikan biasa saja.

4.2.5. Kesimpulan Pengujian

Berdasarkan hasil pengujian terhadap aplikasi game puzzle and the solver, dapat ditarik kesimpulan aplikasi game dapat berfungsi sesuai dengan yang diharapkan serta aplikasi game yang dibangun telah cukup memenuhi tujuan awal pembangunan.