BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan aplikasi.Tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan, yang terdiri dari penjelasan mengenai lingkungan implementasi, baik itu lingkungan perangkat keras, maupun lingkungan perangkat lunak, serta implementasi program.

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang digunakan untuk menjalankan aplikasi permainan *Detective Code out* tidak harus yang berspesifikasi tinggi. Spesifikasi minimum yang dapat digunakan untuk menjalankan aplikasi permainan *Detective code out* terdapat pada tabel 4.1.

Tabel 4. 1 Implementasi Perangkat Keras

Nama Perangkat	Spesifikasi
Prosesor	1.8 GH
RAM	128 MB
Harddisk	40 GB
Display	Resolusi 1366 x 768
Keyboard	Standar
Mouse	Standar

4.1.2 Implementasi Perangkat Lunak

Implementasi perangkat lunak digunakan untuk menjalankan aplikasi *Detective code out* **,** terdapat pada tabel 4.2.

Tabel 4. 2 Implementasi Perangkat Lunak

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows XP
Tools	Greenfoot+JDK 6

4.1.3 Implementasi Aplikasi

Detective Code Out merupakan aplikasi permainan dengan *genre platformers*. Pemain mendapatkan misi setiap *level*-nya yaitu untuk mengumpulkan bukti berupa huruf-huruf yang tersebar di area permainan. Selama dalam perjalanan mengumpulkan bukti pemain dihadang oleh preman-preman yang menyerang pemain. Pada level pertama pemain dihadapkan dengan segerombolan musuh yang bernama jack, dilevel kedua berhadapan dengan segerombolan musuh bernama Billy dan di level ketiga pemain berhadapan dengan bos yaitu Moggy.

Permainan dikatakan berhasil apabila telah berhasil menglahkan semua musuh yang ada di setiap levelnya, sedangkan permainan dikatakakan gagal apabila pemain terbunuh oleh musuh.

Aplikasi permainan ini berbasis Java desktop yang dibangun dengan menggunakan bahasa pemograman java dan *tools* Greenfoot sehingga permainan ini hanya bisa dijalankan apabila pengguna telah menginstal JVM (*java Virtual Machine)* pada perangkat komputernya.

4.1.4 Implementasi Antarmuka

Implementasi antarmuka pada aplikasi permainan *Detective code Out* adalah sebagai berikut:

1. Antarmuka Menu Utama

Antarmuka menu utama merupakan tampilan menu utama pada saat pertama kali pemain membuka aplikasi permainan. Tampilan dari antarmuka menu utama dapat dilihat pada gambar 4.1.



Gambar 4. 1 Antarmuka Menu Utama

2. Antarmuka Mulai

Antarmuka mulai permainan merupakan tampilan ketika pemain memilih menu mulai permainan sehingga langsung masuk ke *intro*1, *intro* 2, *intro* 3, intro 4, intro 5, level 1, level 2, dan level 3.

A. Antarmuka Mulai *Intro* 1

Tampilan dari antarmuka mulai permainan *intro* 1 dapat dilihat pada gambar 4.2.



Gambar 4. 2 Antarmuka Mulai Permainan Intro 1

B. Antarmuka Mulai Permainan *Intro* 2

Tampilan dari antarmuka mulai permainan *intro* 2 dapat dilihat pada gambar 4.3.



Gambar 4. 3 Antarmuka Mulai *Intro* 2

C. Antarmuka Mulai *Intro 3*

Tampilan dari antarmuka mulai permainan *intro* 3 dapat dilihat pada gambar 4.4



Gambar 4. 4 Antarmuka Mulai *Intro 3*

D. Antarmuka Mulai *Intro* 4

Tampilan dari antarmuka mulai permainan *intro* 3 dapat dilihat pada gambar 4.5.



Gambar 4. 5 Antarmuka Mulai *Intro* 4

E. Antarmuka Mulai Permainan *Intro* 5

Tampilan dari antarmuka mulai permainan *intro* 3 dapat dilihat pada gambar 4.6.



Gambar 4. 6 Antarmuka Mulai *Intro* 5

F. Antarmuka Mulai Level 1

Tampilan dari antarmuka mulai permainan level 1 dapat dilihat pada gambar 4.7



Gambar 4. 7 Antarmuka Mulai Level 1

G. Antarmuka Mulai Level 2

Tampilan dari antarmuka mulai permainan level 2 dapat dilihat pada gambar 4.8.



Gambar 4. 8 Antarmuka Mulai Level 2

H. Antarmuka Mulai Level 3

Tampilan dari antarmuka mulai permainan level 3 dapat dilihat pada gambar 4.9.



Gambar 4. 9 Antarmuka Mulai Level 3

3. Antarmuka Menu Kontrol

Antarmuka menu kontrol merupakan tampilan menu yang menampilkan halaman tombol pengatur dalam permainan.

Tampilan dari antarmuka menu utama dapat dilihat pada gambar 4.10.



Gambar 4. 10 Antarmuka Menu Kontrol

4. Antarmuka Menu *About*

Antarmuka menu about merupakan tampilan menu yang menampilkan halaman tentang pembuat aplikasi.

Tampilan dari antarmuka menu utama dapat dilihat pada gambar 4.11.



Gambar 4. 11 Antarmuka Menu *About*

4.2 Pengujian Sistem

Dalam melakukan pengujian terhadap aplikasi permainan *Detective Code Out* dilakukan dengan dua macam teknik, yaitu teknik pengujian *black-box* dan teknik pengujian *white-box*. Adapun pengujian *black box* berfokus pada menemukan kesalahan program, sedangkan pengujian *white box* berfokus kepada struktur internal (*source* code) program untuk mengetahui apakah masih terjadi *error*. Pengujian dilakukan dengan dua tahap, yaitu tahap pengujian *alpha* dan tahap pengujian *bet*a.

4.2.1 Pengujian *Alpha*

Pada tahap ini, pengujian dilakukan untuk memastikan aplikasi permainan *Detective Code Out* telah berjalan dengan benar dan sesuai dengan kebutuhan yang diinginkan. Pengujian *alpha* menitikberatkan kepada persyaratan fungsional perangkat lunak.

4.2.1.1 Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi permainan *Detective Code Out.* Skenario pengujian dari aplikasi permainan *Detective Code Out* dapat dilihat pada tabel 4.3.

Tabel 4. 3 Skenario Pengujian Aplikasi

No	Komponen yang di uji	Skenario Pengujian	Jenis Pengujian
1	Menu Utama	Piih Menu Mulai	*Black Box*
		Pilih Menu Kontrol	*Black Box*
		Pilih Menu About	*Black Box*
		Pilih Menu *Exit*	*Black Box*
2	Menu Mulai Permainan	Pilih Menu Mulai	*Black Box*
		Pilih Tombol Lanjut *Intro 1*	*Black Box*
		Pilih Tombol Lanjut *Intro* 2	*Black Box*
		Pilih Tombol Lanjut *Intro* 3	*Black Box*

		Pilih Tombol Lanjut *Intro* 4	*Black Box*
		Pilih Tombol Lanjut *Intro* 5	*Black Box*
		Permainan *Level* 1	*Black Box*
3	Menu Petunjuk	Pilih Menu Kontrol	*Black Box*
		Pilih Tombol Lanjut	*Black Box*
		Pilih Tombol Kembali	*Black Box*
4	Menu About	Pilih Menu Lanjut	*Black Box*
		Pilih Tombol Kembali	*Black Box*
5	Menu Keluar	Pilih Menu Keluar	*Black Box*
6	Gerak Karakter	Menekan Tombola Arah Kanan	*Black Box*
		Menekan Tombola Arah Kiri	*Black Box*
		Menekan Tombola Spasi	*Black Box*
		Menekan Tombol Ctrl	*Black Box*
		Menekan Tombol S	*Black Box*
7	Collision Detection	Deteksi tabrakan peluru dengan musuh	*White Box*
		Deteksi tabrakan peluru dengan raja	*White Box*
8	Algoritma A* (A star)	Algoritma A* Pada Musuh *Level* 1	*White Box*
		Algoritma A* Pada Musuh *Level* 2	*White Box*
		Algoritma A* Pada Raja *Level* 3	*White Box*

4.2.1.2 Hasil Pengujian *(Black Box)*

Kasus dan hasil uji perangkat lunak yang sudah dibangun dengan menggunakan metode *black box* berdasarkan pengamatan yang dilakukan dan menghasilkan kesimpulan dari pengujian tersebut, diantaranya sebagai berikut :

A. Pengujian Mulai Permainan

Pengujian mulai permainan dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.4.

Tabel 4. 4 Pengujian Mulai Permainan

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pilih Mulai	Menampilkan *Intro 1*	Dapat menampilkan *Intro 1*	[√] Berhasil [ ] Gagal
Pilih lanjut intro 1	Menampilkan intro 2	Dapat menampilkan intro 2	[√] Berhasil [ ] Gagal
Pilih lanjut intro 2	Menampilkan intro 3	Dapat menampilkan intro 3	[√] Berhasil [ ] Gagal
Pilih lanjut intro 3	Menampilkan intro 4	Dapat menampilkan intro 4	[√] Berhasil [ ] Gagal
Pilih lanjut intro 4	Menampilkan intro 5	Dapat menampilkan intro 5	[√] Berhasil [ ] Gagal
Pilih lanjut intro 5	Menampilkan map level 1	Dapat menampilkan map level 1	[√] Berhasil [ ] Gagal
Menyelesaikan level 1	Menampilkan map level 2	Dapat menampilkan map level 2	[√] Berhasil [ ] Gagal
Menyelesaikan level 2	Menampilkan map level 3	Dapat menampilkan keterangan menang	[√] Berhasil [ ] Gagal

B. Pengujian Tampil Kontrol

Pengujian tampil kontrol dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.5.

Tabel 4. 5 Pengujian Tampil Kontrol

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pilih Kontrol	Menampilkan informasi mengendalikan permainan	Dapat menampilkan informasi mengendalikan permainan	[√] Berhasil [ ] Gagal
Pilih kembali	Menampilkan menu utama	Dapat menampilkan menu utama	[√] Berhasil [ ] Gagal
Pilih skip	Menampilkan intro 1	Dapat menampilkan intro 1	[√] Berhasil [ ] Gagal

C. Pengujian Tampil *About*

Pengujian tampil *about* dengan kasus dan hasil uji, kemudian diamati dan menghasilkan kesimpulan dapat dilihat pada tabel 4.6.

Tabel 4. 1 Pengujian Tampil About

Kasus dan Hasil Uji(Data benar)			
Masukan	Keluaran yang diharapkan	Pengamatan	Kesimpulan
Pilih About	Menampilkan informasi pembuat aplikasi permainan	Dapat menampilkan informasi pembuat aplikasi permainan	[√] Berhasil [ ] Gagal
Klik kembali	Menampilkan menu utama	Dapat menampilkan menu utama	[√] Berhasil [ ] Gagal
Klik skip	Menampilkan intro 1	Dapat menampilkan intro 1	[√] Berhasil [ ] Gagal

4.2.1.3 Hasil Pengujian *(White Box)*

Di bawah ini merupakan kasus algoritma A* (A *star*) dan *collision detection* untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *white box*, berdasarkan skenario pengujian aplikasi yang sudah dijabarkan.

1. Pengujian penelusuran *algoritma A** pada musuh

Pengujian proses algoritma A* merupakan proses pengujian dari pengimplementasian algoritma A* menggunakan pengujian *whitebox*, dan menguji apakah algoritma A* (A Star) berjalan dengan seharusnya pada musuh.

Tabel 4. *6* Pengujian Penelusuran Algoritma A *

1	`while ((maxDepth < maxSearchDistance) && (open.size() != 0)) {`
	`Node current = getFirstInOpen();`
2	`if (current == nodes[tx][ty]) {`
3	`break;`
	`}`
	`removeFromOpen(current);`
4	`addToClosed(current);`
	`for (int x=-1;x<2;x++) {`
	`for (int y=-1;y<2;y++) {`
5	`if ((x == 0) && (y == 0)) {`
6	`continue;`
	`}`
7	`if (!allowDiagMovement) {`
8	`if ((x != 0) && (y != 0)) {`
9	`continue;`
	`}`

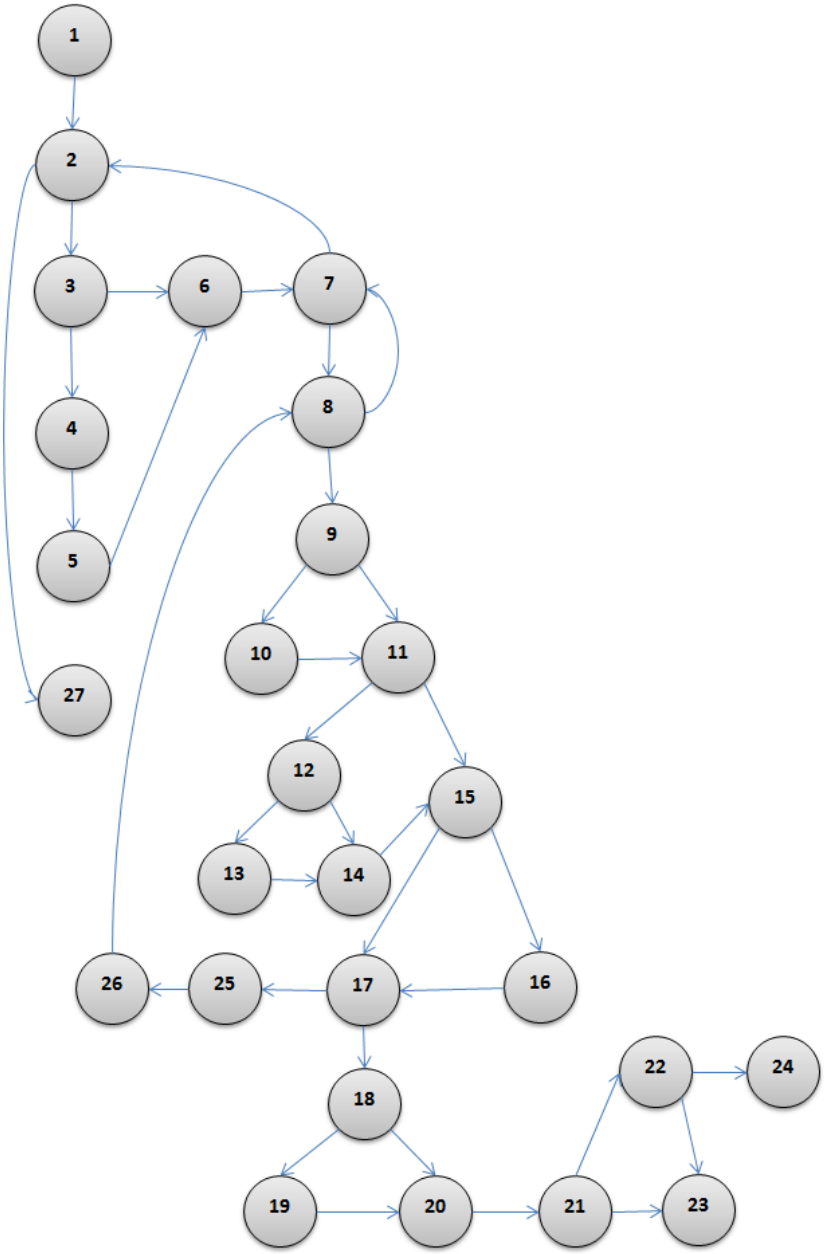
```
10                  }
                    int xp = x + current.x;
                    int yp = y + current.y;
11                  if (isValidLocation(mover,sx,sy,xp,yp)) {
12                     float nextStepCost = current.cost
                       +getMovementCost(mover, current.x,
                       current.y, xp, yp);
                       Node neighbour = nodes[xp][yp];
13                     map.pathFinderVisited(xp, yp);


14                     if (nextStepCost < neighbour.cost) {
15                       if (inOpenList(neighbour)) {
16                            removeFromOpen(neighbour);
                         }
17                       if (inClosedList(neighbour)) {
18                           removeFromClosed(neighbour);
                         }
19                     }
20                     if (!inOpenList(neighbour) &&
                       !(inClosedList(neighbour))) {
                            neighbour.cost = nextStepCost;
                            neighbour.heuristic =
21                          getHeuristicCost(mover, xp,  yp,  tx,
                            ty);
                            maxDepth = Math.max(maxDepth,
22                          neighbour.setParent(current));
                            addToOpen(neighbour);
23

                    }
                 }
              }
```

```
        }
24  }
```

a. *Flow Graph*



Gambar 4. 12 *Flow Graph* A *

Kerangan :

= Menggambarkan kondisi

= Menggambarkan aksi

b. Cyclomatic Complexity (V(G))

V(G) = E – N + 2

V(G) = 34 – 27 + 2

V(G) = 9

Di mana:

E = jumlah edge pada grafik alir

N = jumlah node pada grafik alir

c. Independent path

Path 1 = 1-2-27

Path 2 = 1-2-3-4-5-6-7-2-27

Path 3 = 1-2-3-4-5-6-7-8-7-2-27

Path 4 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-25-26-8-7-2-27

Path 5 = 1-2-3-4-5-6-7-8-9-11-15-17-25-26-8-7-2-27

Path 6 = 1-2-3-4-5-6-7-8-9-11-12-13-14-15-16-17-25-26-8-7-2-27

Path 7 = 1-2-3-4-5-6-7-8-9-11-15-16-17-25-26-8-7-2-27

Path 8 = 1-2-3-4-5-6-7-8-9-11-12-14-15-16-17-25-26-8-7-2-27

Path 9 = 1-2-3-4-5-6-7-8-7-8-9-11-12-14-15-17-25-26-8-7-2-27

d. Graph Matrices

Berikut ini adalah graph matrices dari algoritma A*yang dapat dilihat pada tabel 4.7.

Tabel 4. 7 *Graph Matrices*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	HASIL
1		1																										0
2			1																								1	1
3				1																								0
4					1	1																						1
5						1																						0
6							1																					0
7		1						1																				1
8							1		1																			1
9										1	1																	1
10											1																	0
11												1													1			1
12													1															0
13														1														0
14															1													0
15																1												0
16																	1											0
17																		1							1			1
18										1									1									1
19																				1								0
20																					1							0
21																						1						0
22														1									1					1
23																								1				0
24																									1			0
25																										1		0
26								1																				
27																												9

Kesimpulan:

Berdasarkan hasil pengujian dihasilkan nilai *Cyclomatic Complexity dan Graph matrices* sama yaitu 9, maka dapat disimpulkan bahwa pada proses pencarian dengan algoritma A* (A *star*) berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

4.2.2 Pengujian *Betha*

Pengujian *betha* merupakan pengujian yang dilakukan secara objektif, dimana pengujian langsung melibatkan para pengguna aplikasi permainan secara langsung dengan menggunakan kuesioner mengenai kepuasan para pengguna terhadap aplikasi permainan yang dibangun. Pengujian dilakukan terhadap 30 responden dengan usia minimal 6 tahun dan setiap responden diberikan 5 pernyataan untuk kemudian dilakukan penilaian dengan memilih 5 pilihan yang telah disediakan untuk setiap pernyataannya.

4.2.1.1 Kuesioner

Data hasil kuesioner yang telah diberikan dapat dicari presentasinya dari masing-masing jawaban dengan menggunakan rumus[14] :

$$P = \frac{\sum s}{skorideal} \times 100\% \quad (4.1)$$

dimana,

P = Nilai presentase yang dicari

$\sum s$ = Jumlah skor yang didapatkan dari setiap pilihan jawaban dikalikan dengan nilai yang telah ditetapkan sebelumnya

Skor ideal = Nilai tertinggi pada point pilihan dikalikan dengan jumlah responden

Data yang didapat dari hasil kuesioner terhadap 30 orang responden adalah sebagai berikut :

1. Apakah tampilan aplikasi permainan *Detective code out* secara umum sudah baik?.

Hasil kuesioner untuk pernyataan ke-1 dapat dilihat pada tabel 4.9.

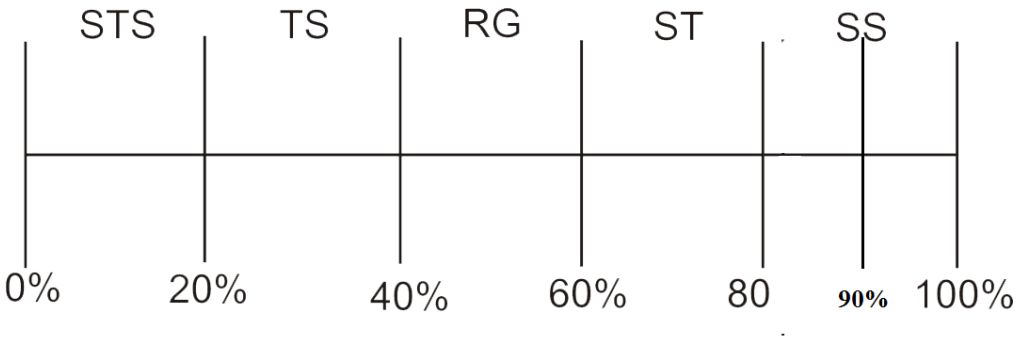
Tabel 4. 8 Hasil Kuesioner Tampilan Aplikasi Permainan

Pernyataan	Jawaban	Skor	Responden	S
1	Sangat Setuju	5	18	90
	Setuju	4	9	36
	Ragu-Ragu	3	3	9
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**135**

P = (135 : 150) x 100% = 90%

Dapat disimpulkan bahwa penilaian terhadap tampilan pada aplikasi permainan *Detective code out* adalah 90% dari yang diharapkan 100%, atau dapat

dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.16.



Gambar 4. 13 Interpretasi Skor Pertanyaan ke-1

2. Apakah Alur permainan pada game *Detective code out* mudah dimengerti?. Hasil kuesioner untuk pernyataan ke-2 dapat dilihat pada tabel 4.10.

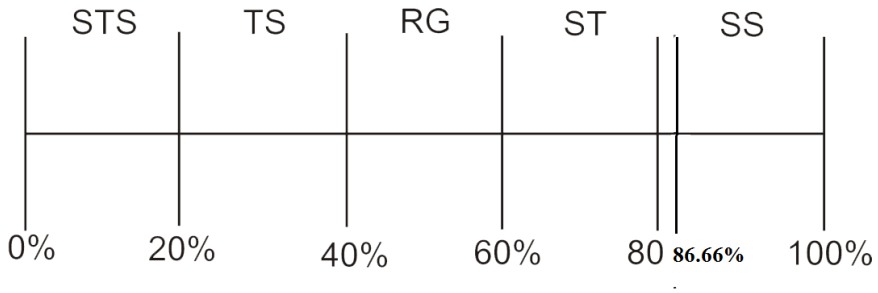
Tabel 4. 9 Alur Permainan Mudah Dimengerti

Pernyataan	Jawaban	Skor	Responden	S
2	Sangat Setuju	5	13	65
	Setuju	4	14	56
	Ragu-Ragu	3	3	9
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**130**

P = (130 : 150) x 100% = 86.66%

Dapat disimpulkan bahwa penilaian terhadap Alur permainan mudah dimengerti pada aplikasi permainan Detective *Code out* adalah 86,67% dari yang

diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.17.



Gambar 4. 14 Interpretasi Skor Pertanyaan ke-2

3. Apakah perilaku musuh yang mengejar karakter utama pada aplikasi permainan *Detective code out* menjadi lebih menantang ketika dimainkan?.

Hasil kuisioner untuk pernyataan ke-3 dapat dilihat pada tabel 4.11.

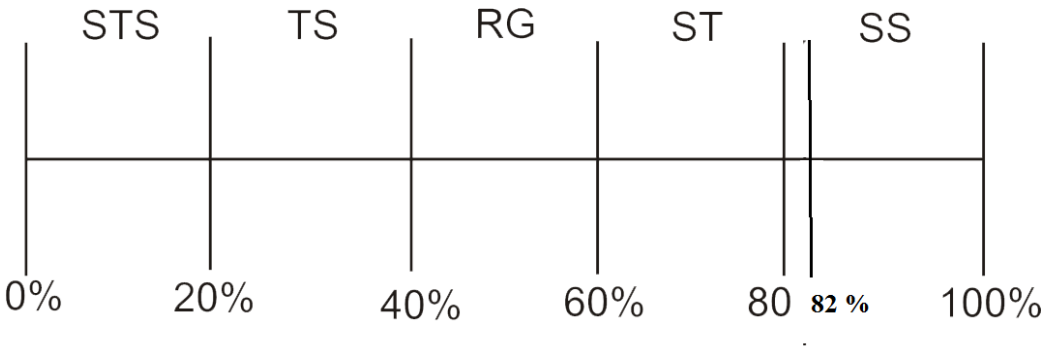
Tabel 4. 10 Perilaku Musuh Terhadap Karakter Utama

Pernyataan	**Jawaban**	**Skor**	**Responden**	**S**
3	Sangat Setuju	5	13	65
	Setuju	4	17	58
	Ragu-Ragu	3	0	0
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**123**

P = (123 : 150) x 100% = 82%

Dapat disimpulkan bahwa penilaian terhadap Perilaku musuh yang mengejar karakter utama menjadi lebih menantang pada aplikasi permainan *Detective code out* adalah 82% dari yang diharapkan 100%, atau dapat

dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.18.



Gambar 4. 15 Interpretasi Pertanyaan ke-3

4. Apakah Aplikasi permainan *Detective Code Out* menarik untuk dimainkan?
   Hasil kuesioner untuk pernyataan ke-4 dapat dilihat pada tabel 4.12.

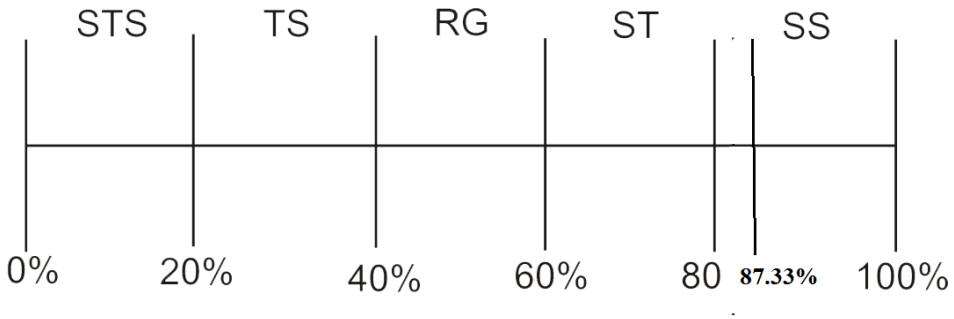
Tabel 4. 11 Permainan Ini Menarik Untuk Dimainkan.

Pernyataan	Jawaban	Skor	Responden	S
4	Sangat Setuju	5	16	80
	Setuju	4	9	36
	Ragu-Ragu	3	5	15
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**131**

P = (131 : 150) x 100% = 87.33%

Dapat disimpulkan bahwa penilaian terhadap Aplikasi permainan ini menarik untuk dimainkan mudah dimengeri adalah 87,33% dari yang diharapkan 100%,

atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.18.



Gambar 4. 16 Interpretasi Pertanyaan ke-4

5. Apakah anda bersedia untuk memainkan kembali aplikasi *Detective Code Out*?

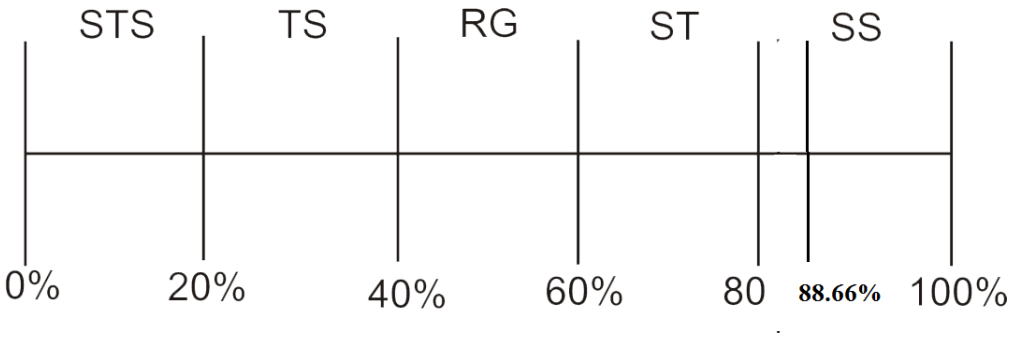
   Hasil kuesioner untuk pernyataan ke-5 dapat dilihat pada tabel 4.13.

Tabel 4. 12 Permainan Ini Menarik Untuk Dimainkan Lagi

Pernyataan	Jawaban	Skor	Responden	S
5	Sangat Setuju	5	14	70
	Setuju	4	15	60
	Ragu-Ragu	3	1	3
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
JUMLAH			**30**	**133**

P = (133 : 150) x 100% = 88.66%

Dapat disimpulkan bahwa penilaian Aplikasi permainan ini menarik untuk dimainkan lagi adalah 88,66% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju, untuk lebih jelasnya dapat dilihat pada gambar 4.19.



Gambar 4. 17 Interpretasi Pertanyaan ke-5