BAB III

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai analisis dan perancangan pada sistem yang dibangun yakni penerapan algoritma *layer by layer* dalam menyelesaikan permainan *rubiks cube* untuk menyusun warna yang sama pada masing – masing sisi rubik. Dalam pembahasan akan dijelaskan yang meliputi komponen – komponen penyelesaian rubik secara *layer by layer*.

3.1 Deskripsi Masalah

Deskripsi masalah adalah suatu gambaran masalah yang diangkat dalam penulisan skripsi tentang penerapan algoritma *layer by layer*. *Layer by layer* merupakan algoritma penyelesaian permainan rubik yang dilakukan secara langkah per langkah (*step by step*). Algoritma *layer by layer* digunakan untuk pemain rubik *beginner* atau pemain rubik pemula yang ingin menyelesaikan permainan rubik. Algoritma *layer by layer* akan diterapkan dalam pencarian langkah pada permainan rubik untuk menyelesaikan rubik menyusun warna yang sama pada setiap sisi yang ada. Pencarian langkah terbaik pada permainan rubik akan diasumsikan dengan titik awal (keadaan pertama) dan titik tujuan (keadaan akhir) yang diimplementasikan pada pencarian langkah. Ruang yang akan dibangun akan berbentuk *cube* (susunan kotak persegi) yaitu bentuk dari permainan rubik supaya mempermudah dalam penerapan cara kerja algoritma

layer by layer. Hasil pencarian algoritma tersebut akan menghasilkan langkah untuk menyelesaikan permainan rubik yaitu dari suatu titik awal ke titik tujuan.

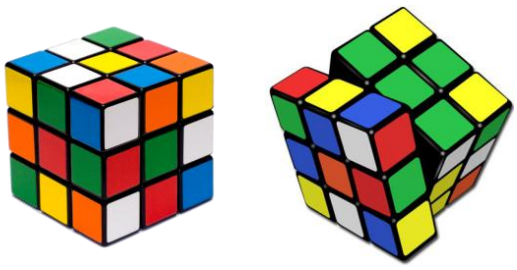
3.2 Komponen Permainan

Untuk menyelesaikan permainan rubik cube 3x3x3 dibutuhkan beberapa komponen, komponen yang dibutuhkan adalah sebagai berikut :

1. Rubik Cube 3x3x3
2. Algoritma atau notasi langkah–langkah utama metode *layer by layer* untuk menyelesaikan rubik.

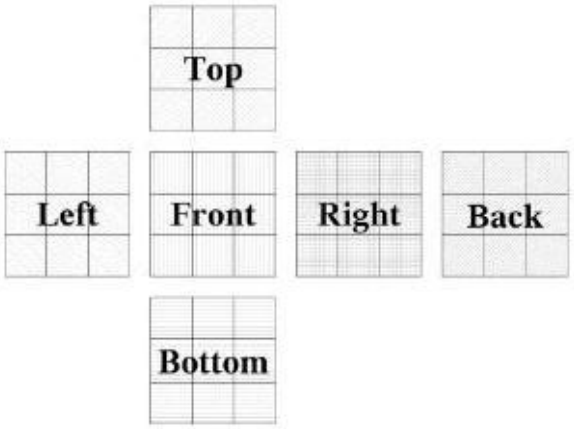
3.2.1 Rubik Cube 3x3x3

Pengguna *Game* ini akan dimainkan oleh satu orang. Pengguna dapat menyelesaikan permainan rubik yang disediakan oleh sistem dalam keadaan teracak. Komputer menyediakan aplikasi rubik cube 3x3x3 sesuai dengan konfigurasi yang telah disediakan oleh aplikasi. Dalam kasus ini banyaknya baris dan kolom pada rubik 3D adalah 3x3x3. Seperti pada gambar 3.1 dibawah ini :



Gambar 3.1. Rubik Cube

Aplikasi rubik juga menyediakan *Rubiks Cube* dengan tampilan dalam bentuk dua dimensi, yaitu:



Gambar 3.2. Tampilan dua dimensi Rubiks Cube

3.2.2 Langkah – Langkah Utama *Layer By Layer*

Pada permainan rubik daftar langkah dibuat terurut yang dimulai dari kotak yang paling tengah karena kotak yang paling tengah adalah kotak yang paling strategis dalam penyusunan deret. Daftar langkah ini merupakan array yang berisi informasi status kotak dan urutan langkah alternatif bagi komputer. *Layer by Layer* dapat menyelesaikan *Rubik's Cube* mulai *layer* pertama, kedua, lalu ketiga. Secara garis besar, penyelesaian Rubik's Cube dengan algoritma *layer by layer* dapat dijelaskan sebagai berikut:

1. Selesaikan *layer* pertama/atas.
    a. Bentuk *cross* disisi atas, sesuaikan dengan warna keempat sisi di samping.
    b. Isi keempat sudut atas dengan kubus yang sesuai.
2. Selesaikan *layer* kedua/tengah.

    Isi keempat kubus pada *layer* kedua dengan kubus yang sesuai.

3. Selesaikan *layer* ketiga/bawah.

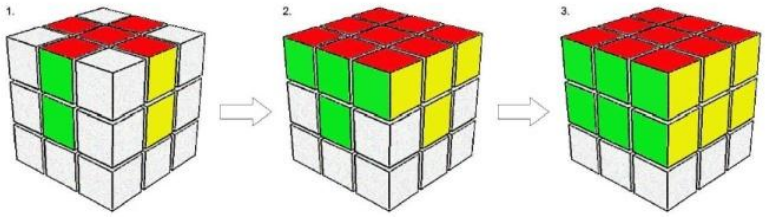
   a. Bentuk *cross* disisi bawah, tanpa merusak kedua *layer* di atas.

   b. Tempatkan keempat sudut bawah di tempat sebenarnya, disesuaikan dengan warna dari ketiga sisi yang bersisian dengannya.

   c. Bentuk supaya sisi bawah memiliki warna yang *uniform*.

   d. Pertukarkan kubus - kubus yang belum sesuai pada tempatnya.

Komputer harus mencari langkah yang sudah berderet dan memberikan nilai kepada pemilik langkah tersebut. Deret didalam rubik dicari dari kotak paling awal yang mengandung indeks kotak sepanjang deret rubik. Pencarian deret rubik dilakukan secara vertical, horizontal, diagonal kiri atas ke kanan bawah, diagonal kanan atas ke kiri bawah atau sebaliknya sesuai dengan *layer – layer* pada rubik seperti digambar 3.3.



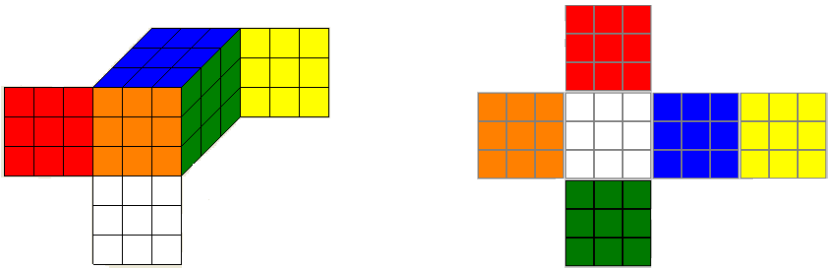
Gambar 3.3. Penyelesaian Rubik

3.3 Analisis Masalah

Algoritma *layer by layer* akan diterapkan pada analisis masalah pada pencarian untuk menyusun warna yang teracak pada permainan *rubiks cube*, untuk mencari penyelesaian dan banyaknya gerakan oleh pemain yang dapat menyelesaikan rubik. Algoritma *layer by layer* digunakan oleh *cuber* pemula

untuk menyelesaikan rubik. Algoritma ini dikenal sangat mudah untuk dihafalkan oleh *cuber* pemula.

Dalam kondisi penyelesaian dengan menggunakan algoritma *layer by layer* pada setiap pergerakan yang dilakukan pada *layer* rubik akan disimpan pada suatu *Array*. *Array* ini akan digunakan untuk melakukan pengecekan apakah kita sudah dapat menyelesaikan rubik *cube* yang teracak atau belum. Pemain dapat menyelesaikan permainan rubik yang teracak kemudian sistem akan memeriksa rubiks *cube* tersebut sudah tersusun setiap warna pada setiap sisinya. Rubik yang belum teracak memiliki keenam sisi warna yang masih tersusun seperti pada gambar 3.4 dibawah ini.



Gambar 3.4. Enam Sisi Rubik Yang Belum Teracak

3.4 Analisis Pemecahan Masalah Dengan Algoritma *Layer By Layer*

Rubik's Cube adalah permainan kubus 3x3x3 yang memiliki warna yang berbeda pada tiap sisinya. Kubus ini terdiri atas dua puluh enam kubus kecil yang disatukan. Satu kubus kecil di tengah dianggap tidak ada karena berperan sebagai sumbu kubus saat memutar sisinya. Pada prinsipnya metode pemecahan *cube* secara *layer by layer* akan selalu melalui tahapan berikut:

1. *Form cross on first layer* – pembentukan silang pada *layer* pertama. Kuncinya adalah pastikan bagian tepi dari setiap *cross* memiliki warna yang sama dengan bagian tengah dari setiap sisi.
2. *Finishing first layer* – penyelesaian *layer* pertama. Tidak ada langkah khusus disini – namun justru ini yang membuat tahapan penyelesaian bisa menjadi panjang jika tidak melakukan perhitungan dari awal.
3. Penyelesaian *second layer* – penyelesaian *layer* kedua. Langkah yang perlu diperhatikan hanya langkah untuk menyelesaikan bagian tepi dari *layer* kedua. Rata-rata setiap tahapan untuk setiap tepi memakan 9 langkah.
4. *Form cross on last layer* – pembentukan silang pada *layer* terakhir. Ada dua langkah yang perlu diperhatikan dalam pembentukan *cross* di*layer* terakhir, yaitu jika sebagian cross sudah terbentuk pada dua sisi dengan kondisi (1) dua sisi berdekatan sudah terbentuk, atau (2) dua sisi berseberangan sudah terbentuk. Didalam tahapan ini tidak perlu langsung menyusun *cross* dengan warna yang sesuai, cukup bentuk *cross* – *regardless it's color*.
5. Atur kesesuaian warna dari *cross* yang telah terbentuk. Satu langkah diperlukan untuk menyelesaikannya.
6. Atur masing - masing sudut pada *layer* terakhir. Pengaturan ini memerlukan paling tidak dua langkah untuk mengatur posisi pada sudut yang benar dan mengatur warna pada setiap sudut.

Kombinasi warna pada *Rubik's Cube* sangatlah banyak. Pertama (*kemungkinan ujung*), kombinasi setiap kotak kecil yang berada di ujung, terdapat kemungkinan 8 tempat yang berbeda dan pada setiap tempat itu satu kotak kecil memiliki 3 warna yang berbeda. Kedua (*kemungkinan rusuk*), kotak kecil yang berada di rusuk, terdapat 12 tempat yang berbeda dan pada setiap tempat itu terdapat 2 warna yang berbeda. Sedangkan(*kemungkinan tengah*) kotak kecil di tengah tiap – tiap sisi tidak akan berubah tempat, hanya ada satu kemungkinan saja. Jadi, banyaknya kombinasi warna yang ada adalah :

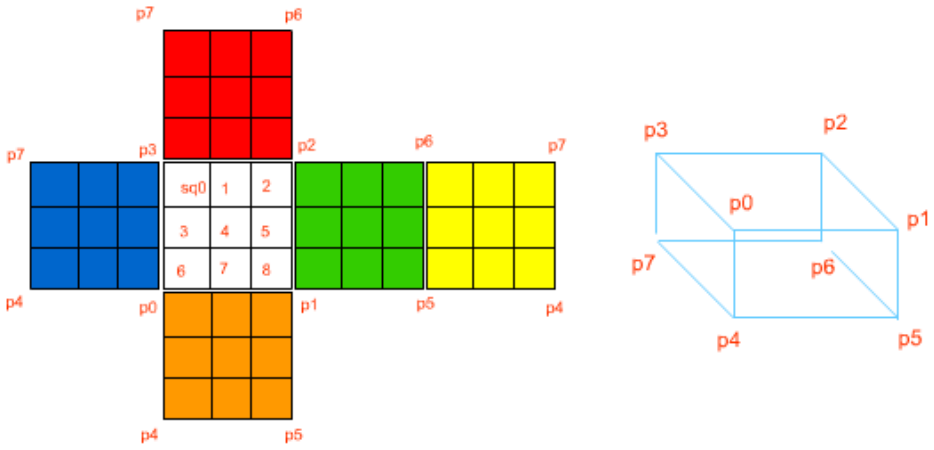
Kombinasi = *kemungkinan ujung* x *kemungkinan rusuk x kemungkinan tengah*

= (8! x 38) x (12! x 212) x 1

= 519.024.039.293.878.272.000

≈ 5,19 x 1020

Penjelasan diatas selanjutnya dikembangkan untuk menyelesaikan permainan kubus rubik yang terdiri dari 6 sisi (*layer*). Setiap sisi terdiri dari 9 kotak kecil. Ketika kubus Rubik diputar oleh pengguna, spesifik "rotasi" akan bergerak dan pengaturan warna sisi masing-masing akan berubah.

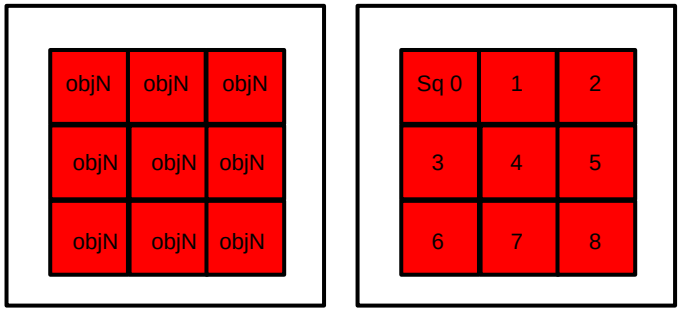


Gambar 3.5. Enam Sisi Rubik Yang Belum Teracak

Dapat didefinisikan warna putih sebagai warna pusat untuk menyelesaikan rubik. Ketika blok paling atas memutar *clockwisely*, melibatkan 1 sisi dan 1 rotasi. Sisi yang memutar adalah warna putih, rotasi warna yang berputar putih persegi dengan notasi (0-1-2-5-8-7-6-3) dengan notasi (4) ditengah. Rotasi cincin sisi, jika dimulai dengan sisi warna merah, maka akan menjadi :

1. Merah persegi dengan notasi 6-7-8
2. Hijau Persegi dengan notasi 0-3-6
3. Orange persegi dengan rotasi 2-1-0
4. Biru persegi dengan notasi 8-5-2

Urutan data ini disimpan dengan *inisial id* atau *keyword* dari setiap sisi. Id sisi putih 1, maka data ini disimpan dalam sisi1, dan rotasi1. Jadi, kesemuanya penyelesaian rubik memiliki 6 set data. Ketika sisi warna no.1 diaktifkan. Persegi-persegi kecil yang ada didalam setiap sisi diinisialkan sebagai [objN] :



[sq(objN, 0), sq(objN, 1), sq(objN, 2), sq(objN, 5), sq(objN, 8), sq(objN, 7), sq(objN, 6), sq(objN, 3), dan sebagai titik pusat sq(objN,4)]

Gambar 3.6. *Inisial id* atau *keyword* dari setiap sisi rubik

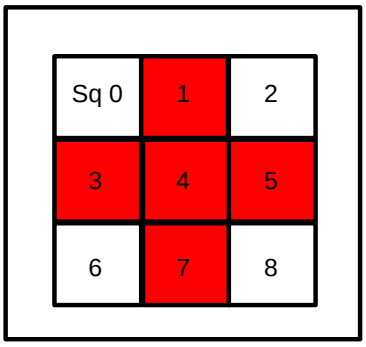
Persegi disini menjadi *array* dan melakukan pergeseran - mendorong dan menyimpannya kembali dengan nilai yang sudah ditentukan. Itu akan menjadi hasil dari rotasi. Bahkan, semua bergerak dengan kombinasi atau varian dari

langkah ini. Ini mungkin searah jarum jam atau berlawanan. Jika rotasi cincin tengah, efeknya adalah sama dengan rotasi berlawanan dari kedua sisi atas dan sisi bawah. Pergerakan persegi ini dikondisikan dengan koordinat – koordinat agar rubik dapat diselesaikan. Koordinat tersebut seperti berikut ini :

1. Data 0 = [sq(4, 0), sq(4, 3), sq(4, 6), sq(1, 0), sq(1, 3), sq(1, 6), sq(5, 0), sq(5, 3), sq(5, 6), sq(3, 8), sq(3, 5), sq(3, 2)];
2. Data 1 = [sq(4, 6), sq(4, 7), sq(4, 8), sq(2, 0), sq(2, 3), sq(2, 6), sq(5, 2), sq(5, 1), sq(5, 0), sq(0, 8), sq(0, 5), sq(0, 2)];
3. Data 2 = [sq(4, 8), sq(4, 5), sq(4, 2), sq(3, 0), sq(3, 3), sq(3, 6), sq(5, 8), sq(5, 5), sq(5, 2), sq(1, 8), sq(1, 5), sq(1, 2)];
4. Data 3 = [sq(4, 2), sq(4, 1), sq(4, 0), sq(0, 0), sq(0, 3), sq(0, 6), sq(5, 6), sq(5, 7), sq(5, 8), sq(2, 8), sq(2, 5), sq(2, 2)];
5. Data 4 = [sq(3, 2), sq(3, 1), sq(3, 0), sq(2, 2), sq(2, 1), sq(2, 0), sq(1, 2), sq(1, 1), sq(1, 0), sq(0, 2), sq(0, 1), sq(0, 0)];
6. Data 5 = [sq(1, 6), sq(1, 7), sq(1, 8), sq(2, 6), sq(2, 7), sq(2, 8), sq(3, 6), sq(3, 7), sq(3, 8), sq(0, 6), sq(0, 7), sq(0, 8)];
7. Data tengah 5 = [sq(0, 3), sq(0, 4), sq(0, 5), sq(1, 3), sq(1, 4), sq(1, 5), sq(2, 3), sq(2, 4), sq(2, 5), sq(3, 3), sq(3, 4), sq(3, 5)];
8. Data tengah 0 = [sq(1, 1), sq(1, 4), sq(1, 7), sq(5, 1), sq(5, 4), sq(5, 7), sq(3, 7), sq(3, 4), sq(3, 1), sq(4, 1), sq(4, 4), sq(4, 7)];
9. Data tengah 3 = [sq(0, 1), sq(0, 4), sq(0, 7), sq(5, 3), sq(5, 4), sq(5, 5), sq(2, 7), sq(2, 4), sq(2, 1), sq(4, 5), sq(4, 4), sq(4, 3)];

Pasangan koordinat ini merupakan inisial yang digunakan agar sistem membaca nilai-nilai sisi menjadi data dan kemudian melakukan pergeseran – push – dan mendorong dari *array* ini dan kemudian menyimpannya kembali ke sisi yang sudah menjadi *inisial* sesuai dengan koordinat yang sudah ada.

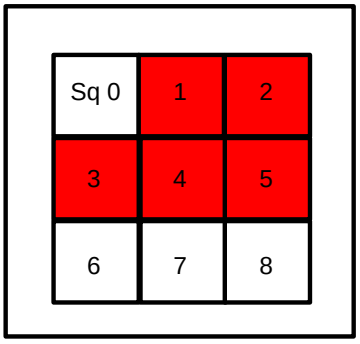
1. Misalnya untuk membuat sebuah *cross* pada salah satu sisi rubik, maka harus memenuhi id yaitu dengan koordinat [sq(5, 3), sq(5, 1), sq(5, 5), sq(5, 7)] dengan sq(5, 4) yang menjadi pusat *cross*. Posisi angka 5 digunakan sebagai acuan utama dalam koordinasi dalam menyelesaikan rubik.



[sq(5, 3), sq(5, 1), sq(5, 5), sq(5, 7)] dengan sq(5, 4) yang menjadi pusat *cross*

Gambar 3.7. Bentuk *cross* pada salah satu sisi rubik

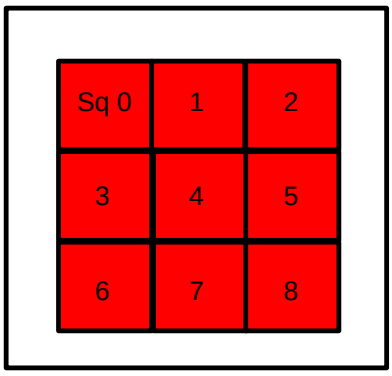
2. Menyelesaikan sudut pada *layer* atau sisi pertama (sisi atas). Maka harus memenuhi id yaitu dengan koordinat sq(5, 4), sq(2, 4), sq(3, 4), dan koordinat sq(2,1).



sq(5, 4), sq(2, 4), sq(3, 4), dan sq(2,1)

Gambar 3.8. Koordinat untuk menyelesaikan sudut

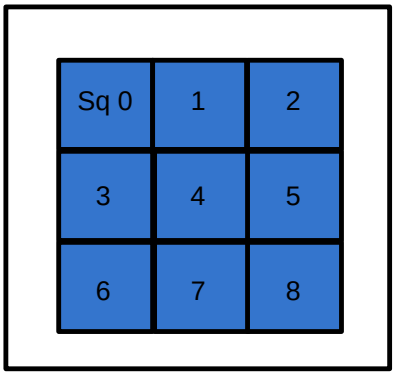
3. Selesaikan sisi pertama dengan memenuhi koordinat :



[sq(5, 3), sq(5, 1), sq(5, 5), sq(5, 7), sq[5,0], sq[5,2], sq[5,6], sq[5,8]], dengan sq(5, 4) yang menjadi pusat

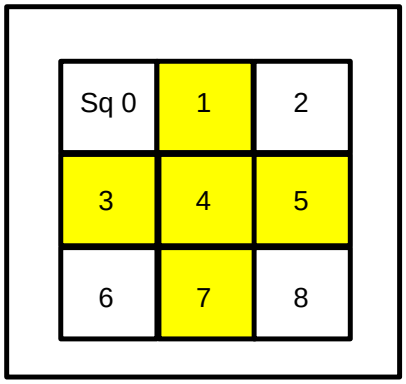
Gambar 3.9. Sisi yang telah selesai disusun warnanya

4. Pada penyelesaian *layer* kedua langkah pertama memenuhi data koordinat sebagai berikut [sq(5, 3), sq(5, 1), sq(5, 5), sq(5, 7)] dengan sq(5, 4) yang menjadi pusat *cross.* Posisi pada koordinat ini sama seperti langkah pertama penyelesaian *layer* satu. Kemudian untuk menyempurnakan warna agar tersusun yaitu dengan melakukan beberapa rotasi.



Gambar 3.10. Penyelesaian *layer* kedua

5. Pada penyelesaian *layer* ketiga langkah pertama yaitu membuat *cross* terlebih dahulu dengan melakukan rotasi agar *layer* bawah keatas, dengan koordinat sebagai berikut :

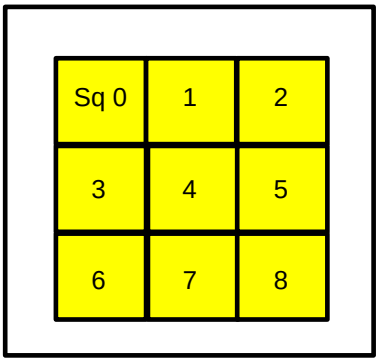


[sq(5, 3), sq(5, 1), sq(5, 5), sq(5, 7)] dengan sq(5, 4) yang menjadi pusat *cross*

Gambar 3.11. Koordinat untuk menyelesaikan *cross* pada *layer* ketiga

6. Benarkan posisi sudut pada *layer* atas agar susunan warna terselesaikan. Penuhi koordinat berikut untuk membentuk sudut pada persegi [sq(0, 8), sq(1, 6), sq(5, 0), sq(2, 6), sq(1, 8), sq(5, 2), sq(2, 8), sq(3, 6), sq(5, 8), sq(0, 6),

sq(5, 6), sq(3, 8)]. Sudut yang telah benar akan memenuhi koordinat seperti berikut : [sq(5, 0), sq(5, 2), sq(5, 8), sq(5, 6), sebagai pusat sq[5,4]].



Gambar 3.12. Penyelesaian *layer* ketiga

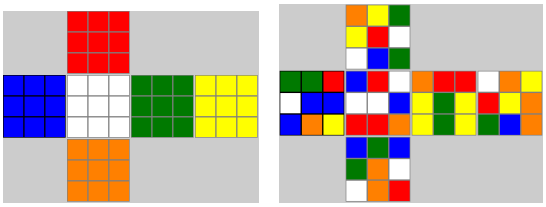
7. Apabila setiap posisi warna pada kubus rubik telah tersusun maka rubik telah dapat diselesaikan.

3.4.1. Performansi Waktu Pada Penyelesaian Rubik

Pada permainan rubik menggunakan format penulisan waktu yang digunakan adalah standar internasional, yaitu dalam menit dan detik yang misalnya dituliskan seperti ini : 1:20,48, yang dibaca “1 menit 20 koma 48 detik”. Tentunya pada aplikasi yang akan dibangun fungsi timer sangat mendukung untuk menghitung performansi penyelesaian rubik antara penyelesaian secara manual dan penyelesaian secara aplikasi. Perbandingan waktu penyelesaian ini diharapkan akan mampu memberikan penjelasan bahwa siapakah yang tercepat dapat menyusun rubik yang teracak agar kembali kekelompok warnanya pada setiap sisi, waktunya yang dihitung dari titik awal acak hingga titik akhir rubik terselesaikan.

3.4.2. Penyelesaian Rubik

Setiap *layer* akan diselesaikan dengan menggunakan algoritma *layer by layer*. Penyelesaian yang diselesaikan dimulai dari menyelesaikan *layer* bawah kemudian *layer* kedua(*layer* tengah) dan yang terakhir yaitu *layer* ketiga (*layer* atas). Dibawah ini merupakan gambar 3.13 yang menunjukkan rubik yang belum teracak kemudian disamping gambar tersebut juga menggambarkan rubik yang sudah teracak. Notasi penyelesaian rubik yang teracak ini adalah sebagai berikut :

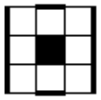
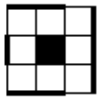
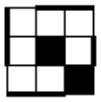
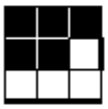
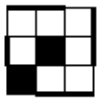
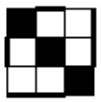
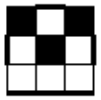
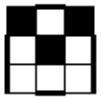
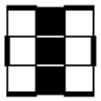
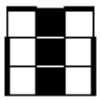
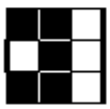
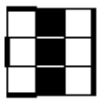
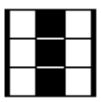


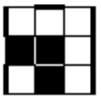
Gambar 3.13. Rubik Dalam Keadaan Teracak

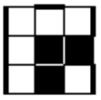
Langkah pada penyelesaian :

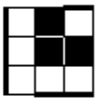
1. *Solve bottom layer* (menyelesaikan lapisan bawah)
2. *Solve middle layer* (menyelesaikan lapisan tengah).
3. *Solve upper/ last layer* (menyelesaikan lapisan atas)
4. *Orienting last layer* (mengorientasikan lapisan atas)
5. *Positioning/ permutating last layer* (memposisikan lapisan atas)
    a. Pada step ini kita mengorientasikan lapisan atas (*Last Layer*) hingga warnanya seragam.
    b. Pada *step* ini kita memposisikan/ mempermutasikan lapisan atas (*Last Layer*) sehingga pada kedudukan/ posisi yang tepat.

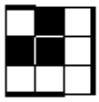
Tabel 3.1. Kode Notasi Penyelesaian Rubiks Cube

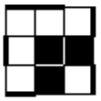
 (R U2) (R' l' U R U') x' U2 (l' U R U') x'	 (l U' R' U) y' (R U R' U')
 U'x'(R U' R' U) x U2 x' (R U' R' U2) x U2 x' U' x	 L d (R U' R' F')
 M (U' r U2') r' (U' R U' M' R')	 R' d' (L' U L F)
 r' (R2' U) (R' U r U2) (r' U M')	 (R U2') (R' l' U R U') (l U2 R')
 R' F R F' U' L R' D' F D L' R U'	 x (U l U R' U' F')
 (L F R U² R' U² R U² R') (F' L')	 (L' U' L U) (r U' L' U)
 L U2 F' L' U' L U F2 U2 F' L'	 (l' U l U R' d') (L' d R)
 M (F' U' F U F U F U' F') M'	 (r U' r' U' L d) (R d' L')
 F R B R' F2 R U B' U' R' F	 (R'U'R U F R B' R' F' B)
 R U B U' B' U B U' B' R'	 (R U) (l U') (R' U) (F' l')
 R U x' U F' U L' U' L U' l'	 (r' U') x U' x' U2 x (U L U' r' U' r)
 (R' U2) (R2' U) (R' U R U2') x' (U' R' U)	 (l U) x U x' U2' x (U' R' U l U l')
 (R' U2 R U R' U' R U R' U R)	
 R U2 R2 U'R2 U'R2 U2 R	
 (R2' D)(R' U2)(R D')(R' U2 R')	

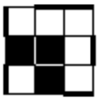

l (U L' U L U' L' U L U2) l'


r' (U' R U' R' U R U' R' U2) r


x' (L U') x (U d) (L U2 r U2 L' U)


x' (R'U) x (U' d') (R' U2 l' U2 R U')


(B L U L' U' L U L' U' B')
or
x' (U r U L' U' L U L' U') x u'


x' (U' l' U' R U R' U' R U) x u

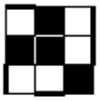

l U L' U L U2' l'

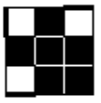

r' U' R U' R' U2 r


(r' U') (R U' l U') (R' U l') (U2' r)

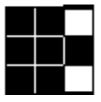

L' R2 B R' B R B2 R' B R' L


F' U' F (r U' r' U r U r')


F U F' (l' U l U' l' U' l)


(L' U R U' L U R')

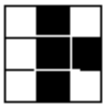

(R U' L' U R' U' L)


l U L' U' R' U L U'

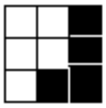

(R' U R U' R2) y' (R'U'RU)y x (R U R' U' l R U')


[U] (R2 U' R2) u R2 y' (L2 U L2) u' L2


(L U' R) D2 (R' U L') (R U' L) D2 (L' U R')

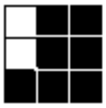

(R2 U) x' (U' D) R2 (U D') x (U R2)

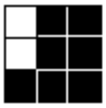

(U R l) U2 x (U' R' U' R U R U) x' U2 x (R U' R)


x' (U2 r U r') U2' x (R D' R D R2)

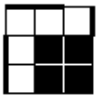
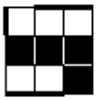
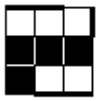
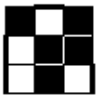
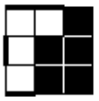
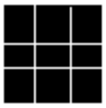
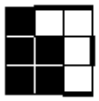
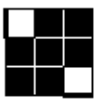

(U' R' U L') (U2 R U' L) (R' U L') (U2 R U' L)


x' (R U' R) D2 (R' U R) D2 R2

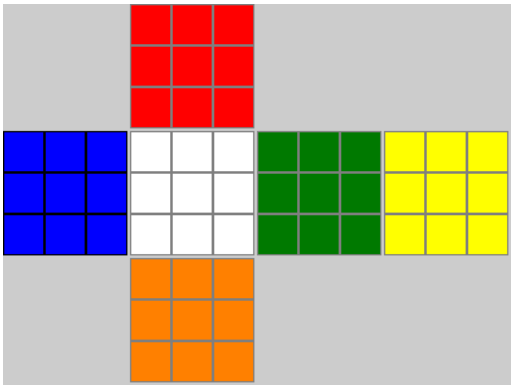

(L U r') z (R' l' u') (R U' R' U R') u R2


y x' (R' l' u') (R U' R U R') u R2 y (R U' R')

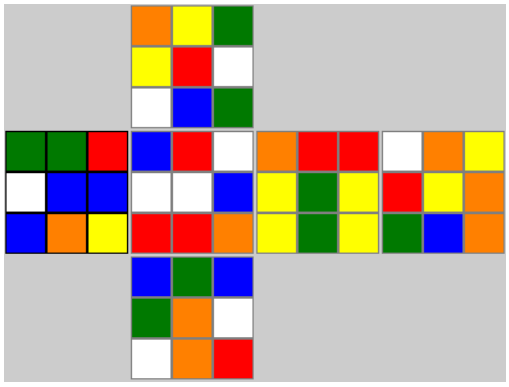

U' x' (U L U' R2 U L' U') x (U2 R2 U' R2 U' R2)

 (r' U2) (R U R' U r)	 x' (R U' R' U) (D R') (D U') (R' U R) (D2 x U)
 (l U2) (L' U' L U' l')	 (l R) (U2 D2) (L' r' U') (l R) (U2 D2) (L' r' U)
 (l' U l) (U l' U' l) (F U' F')	 (L' U R U' L U L' U R' U' L) (U2 R U2 R')
 (r U' r') (U' r U r') (F' U F)	 (R2 U') x' (U' D) R2 (U D') x (U' R2)
 (r U r') (R U R' U' r U' r')	 (U' L' r') U2 x (U L U L' U' L' U') x' U2 x (L' U L'
 (l' U' l) (L' U' L U l' U l)	 x' (U2 l' U' l) U2 x (L' D L' D' L2)
 (R U' R' U2 R U) x' (U F' U' F' l')	 (U L U' R) (U2 L' U R') (L U' R) (U2 L' U R')
 (L' U L U2' L' U') x' (U' F U F r)	 x' (L' U L') D2 (L U' L') D2 L2
 (R2' U l' D') (l U' R2' U l U l')	 y2 R' U' l z' (L r) u (L' U L U' L) u' L2
 (L2 U' r D) (r' U L2 U' r' U' r)	 y' x' (L r u) (L' U L' U' L) u' L2 y' (L' U L)
 (R U) x' U' x (U' R' U l U l')	 Rubik Terselesaikan
 (L' U') x' U x (U L U' r' U' r)	
 R B L B' R' B L' B'	
 M U M' U2 M U M'	
 S R B' R' S' R B R'	

Penjelasan berikut akan memaparkan salah satu penyelesaian rubik yang teracak. Gambar (1) merupakan tampilan 2 dimensi sisi rubik yang belum diacak. Gambar (2) merupakan tampilan 2 dimensi sisi rubik yang sudah teracak. Gambar (3) merupakan tampilan 3 dimensi rubik yang teracak. Lebih jelasnya dapat dilihat pada gambar 3.14 dibawah ini :

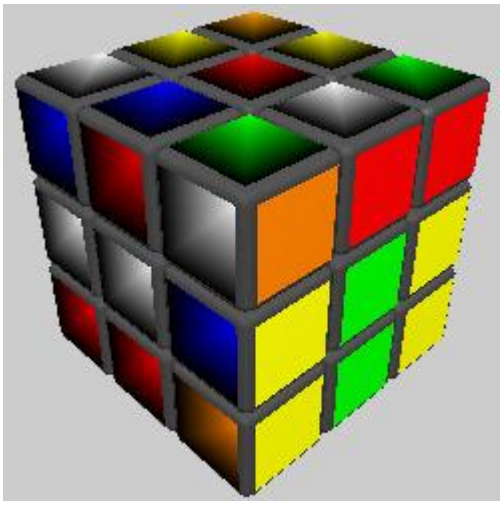


(1)



(2)

(3)

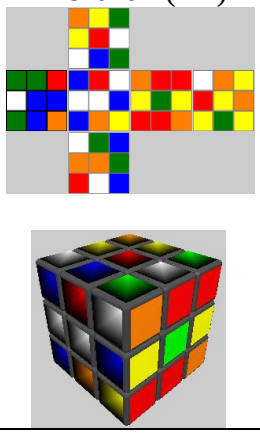
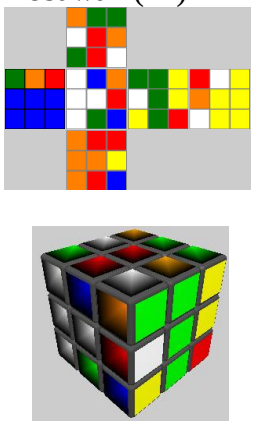
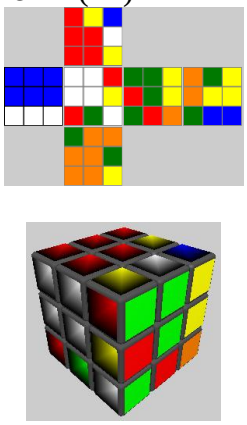
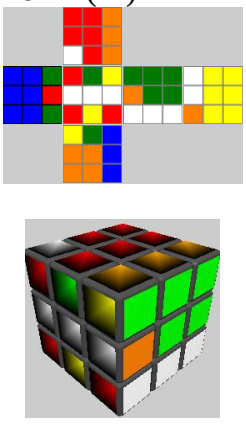
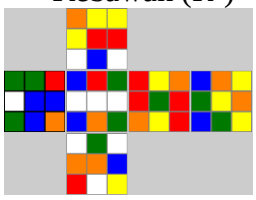
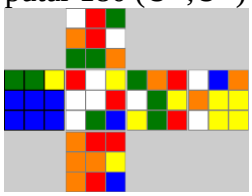
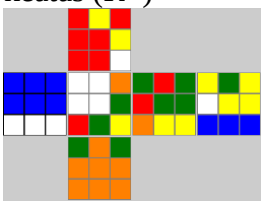
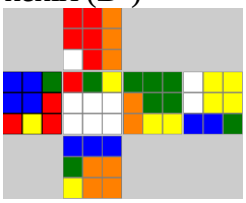
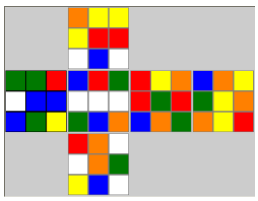
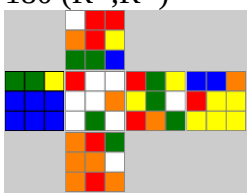
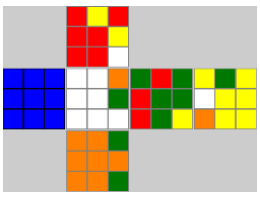
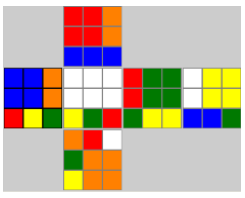


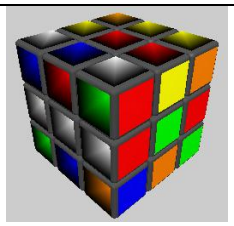
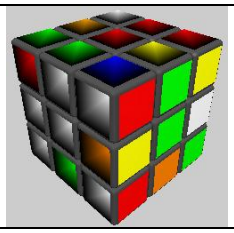
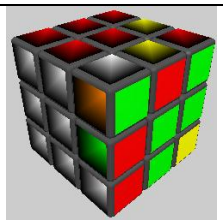
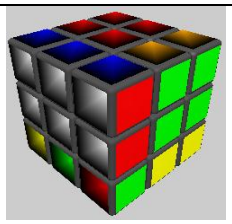
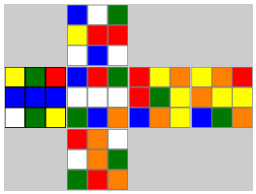
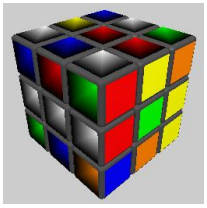
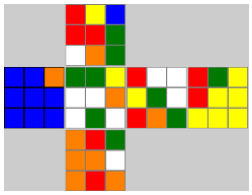
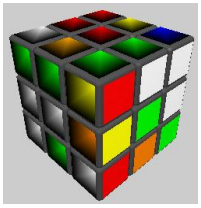
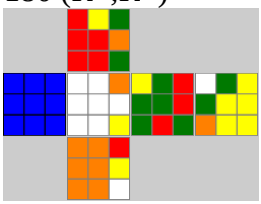
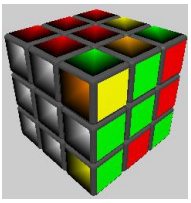
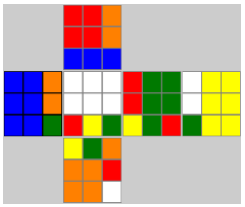
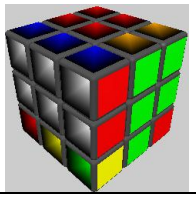
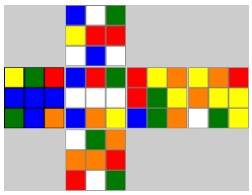
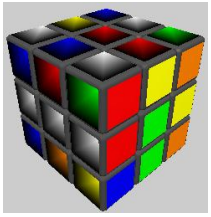
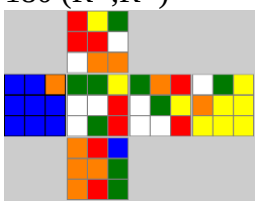
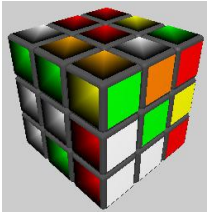
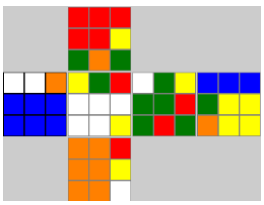
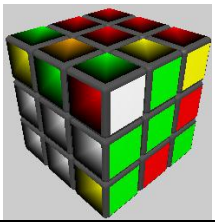
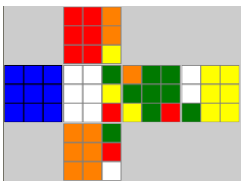
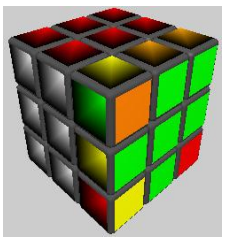
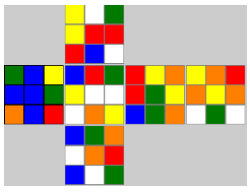
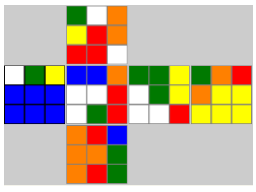
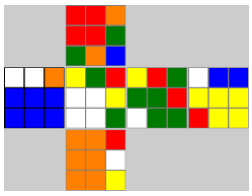
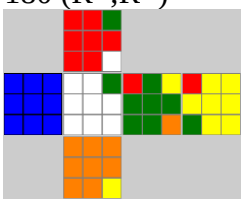
Gambar 3.14. Rubik Dalam Keadaan Teracak

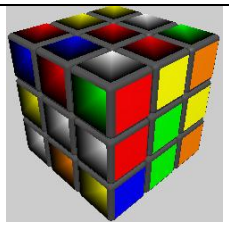
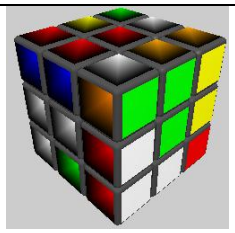
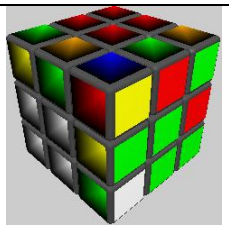
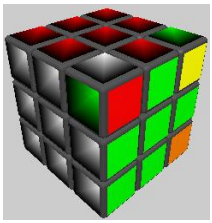
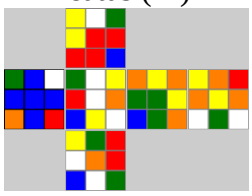
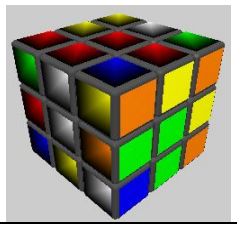
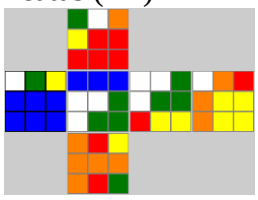
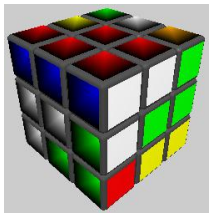
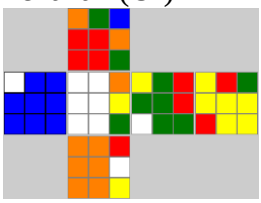
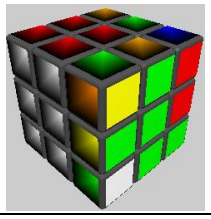
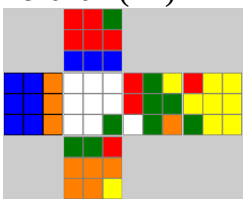
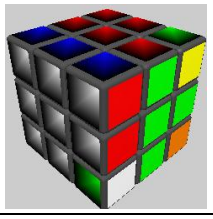
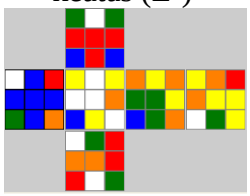
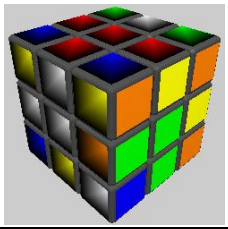
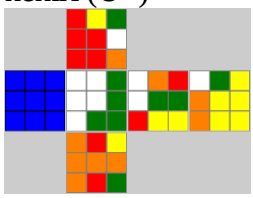
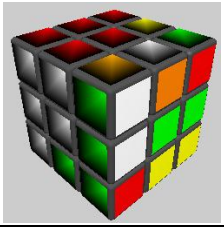
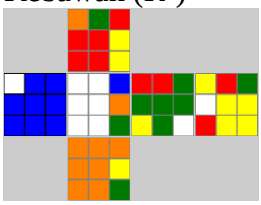
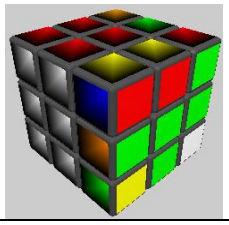
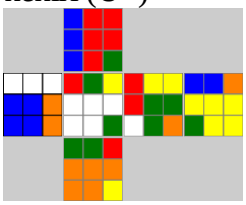
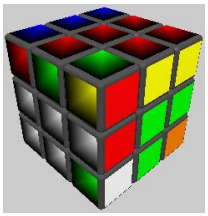
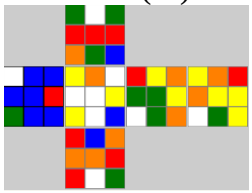
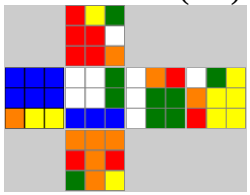
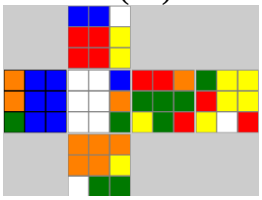
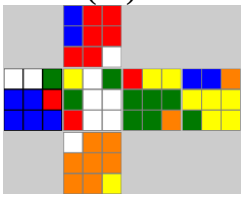
Rubik yang teracak dapat diselesaikan dengan menggunakan algoritma *layer by layer*, pada tabel 3.2 akan menjelaskan tahapan penyelesaian kasus rubik

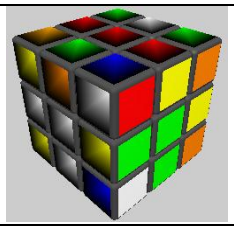
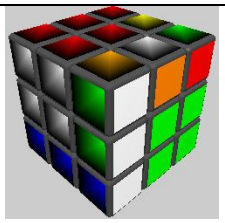
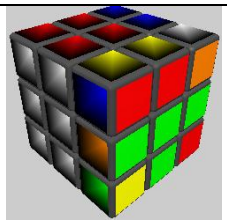
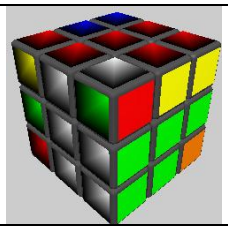
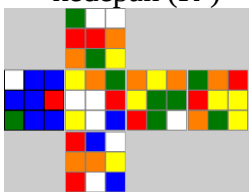
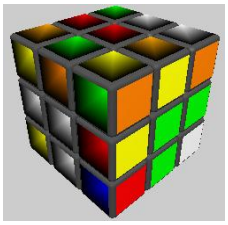
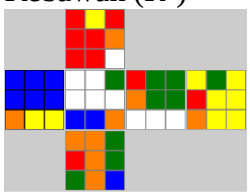
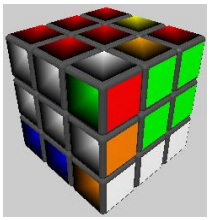
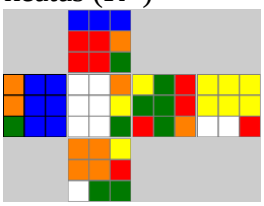
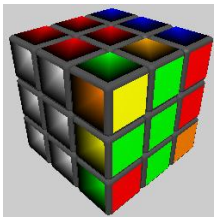
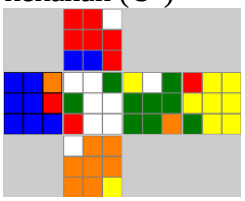
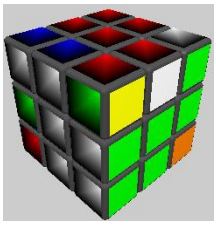
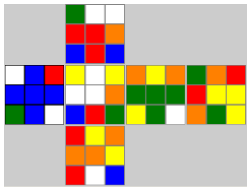
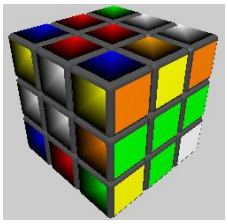
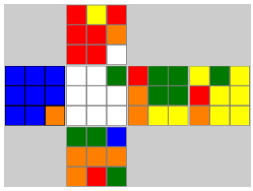
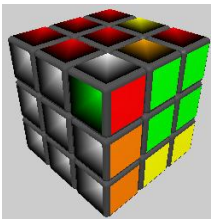
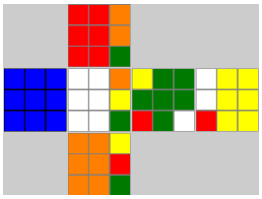
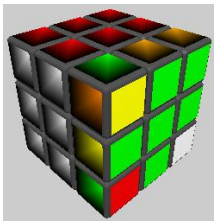
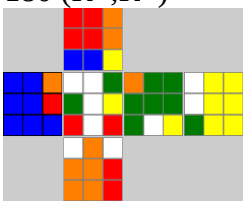
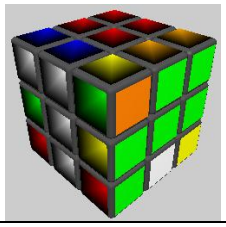
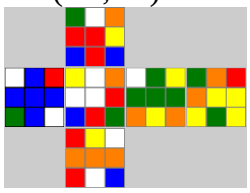
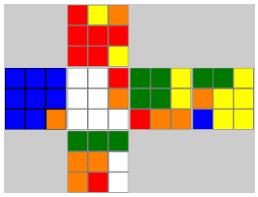
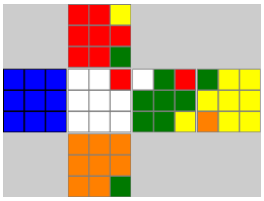
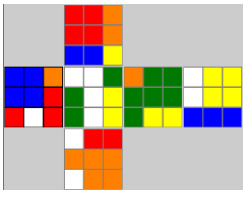
yang teracak pada gambar 3.14 diatas. Tahapan penyelesaian rubik sebagai berikut :

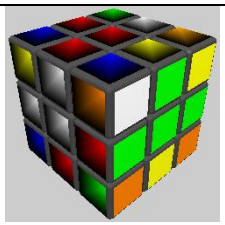
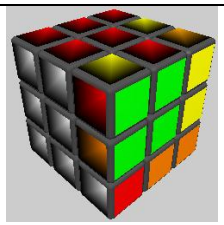
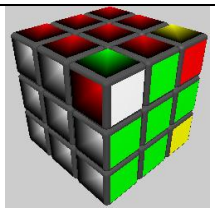
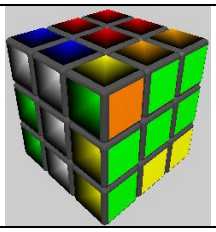
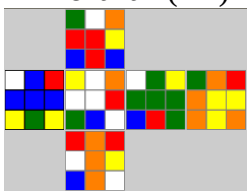
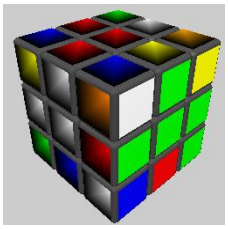
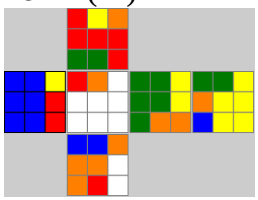
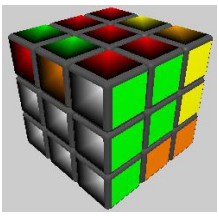
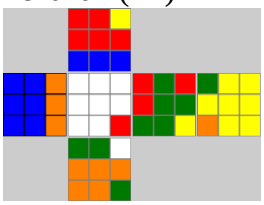
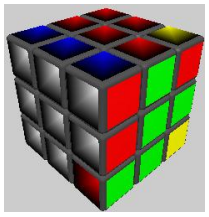
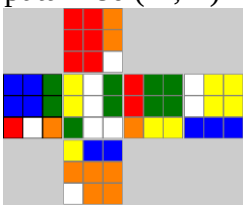
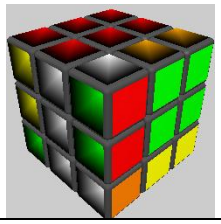
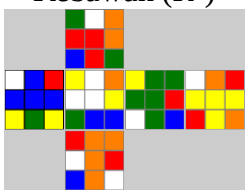
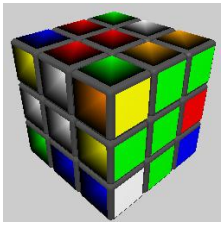
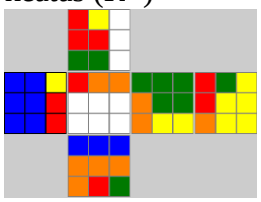
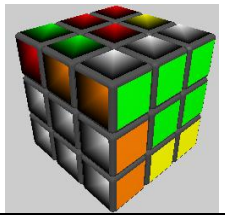
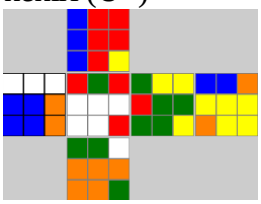
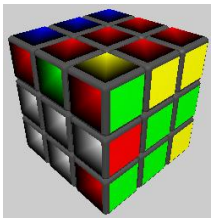
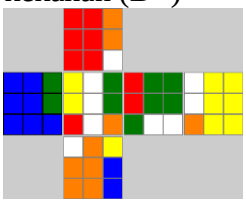
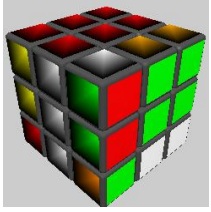
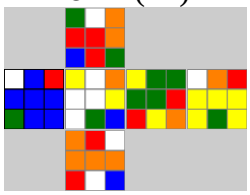
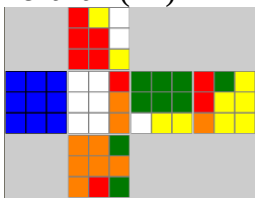
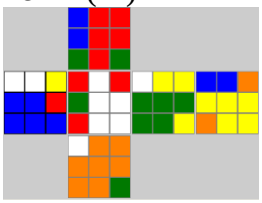
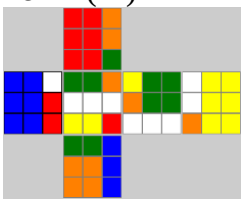
Tabel 3.2. Notasi Penyelesaian Rubik

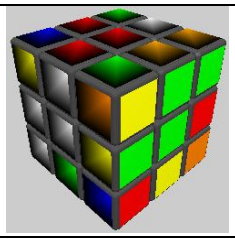
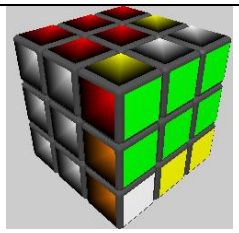
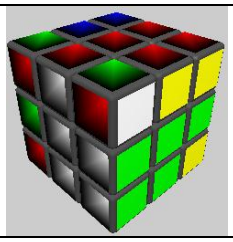
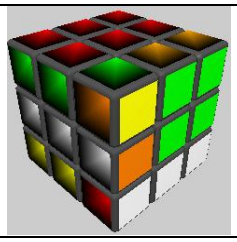
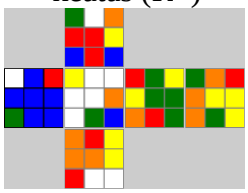
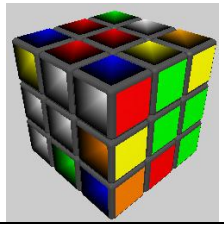
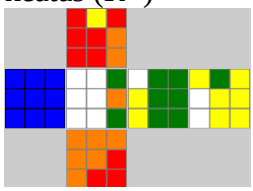
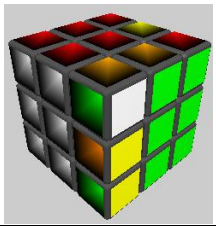
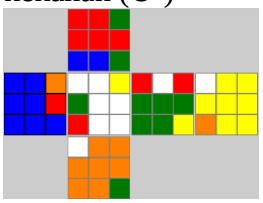
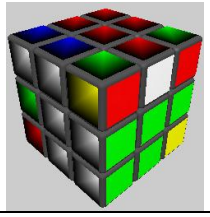
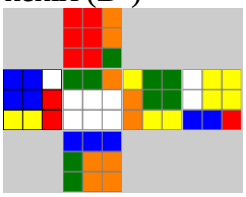
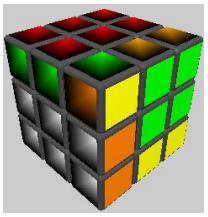
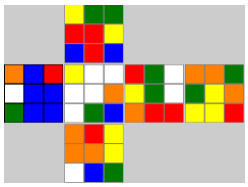
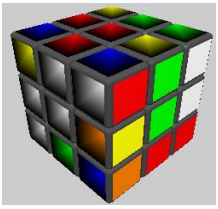
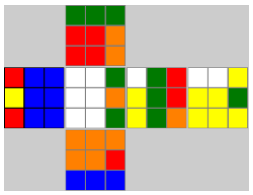
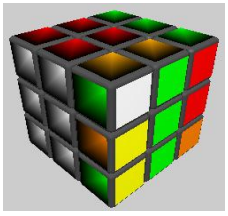
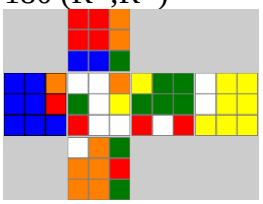
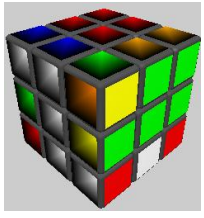
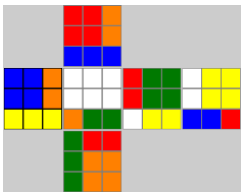
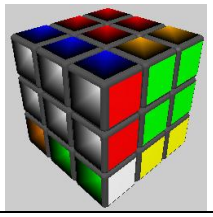
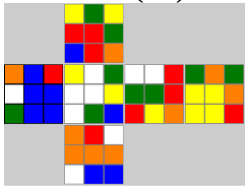
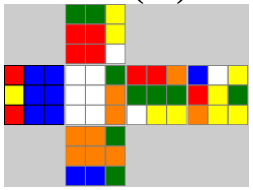
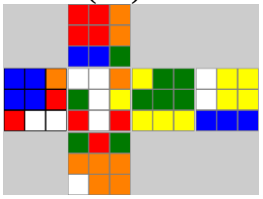
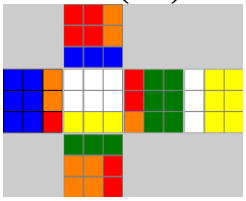
Notasi Gerakan	Notasi Gerakan	Notasi Gerakan	Notasi Gerakan
1. Layer Bawah kekanan (D+) 	21. Layer Kanan Kebawah (R-) 	41. Layer bawah kekiri (D-) 	61. Layer Depan kekiri (F-) 
2. Layer Kanan Kebawah (R-) 	22. Layer atas putar 180 (U+,U+) 	42. Layer kanan keatas (R+) 	62. Layer bawah kekiri (D-) 
3. Layer Bawah kekanan (D+) 	23. Layer kanan putar kebelakang 180 (R+,R+) 	43. Layer Bawah kekanan (D+) 	63. Layer depan putar 180 (F-,F-) 

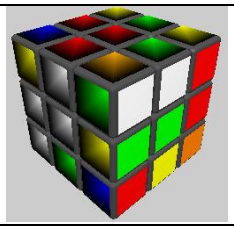
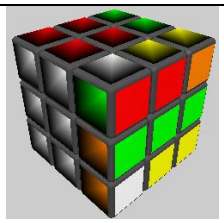
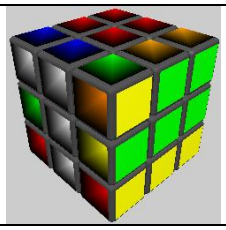
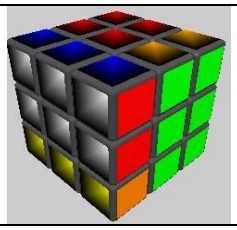
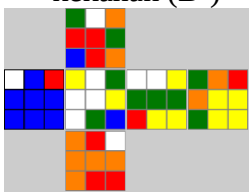
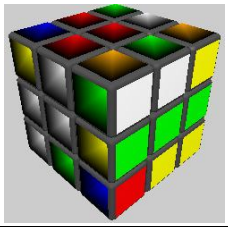
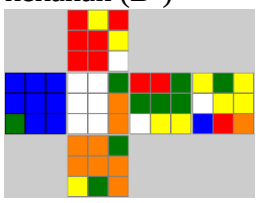
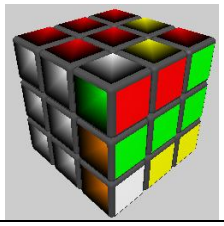
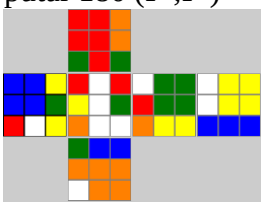
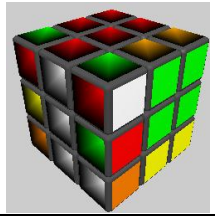
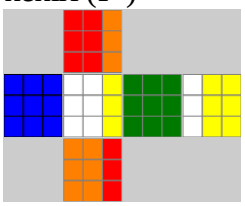
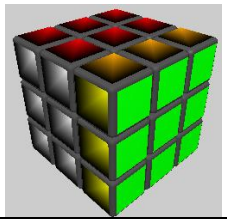
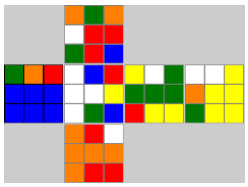
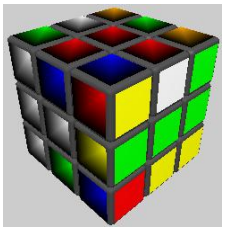
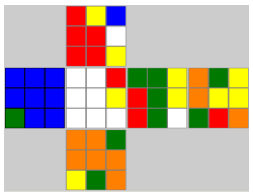
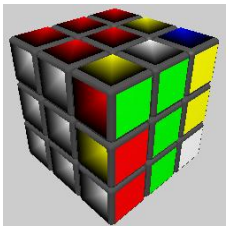
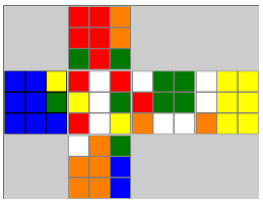
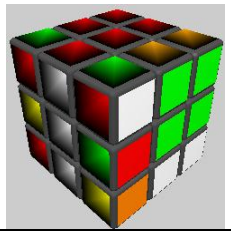
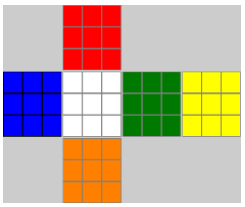
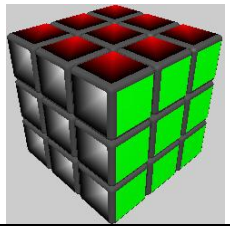
			
4. Layer belakang kekanan (B-)  	24. Layer atas kekanan (U-)  	44. Layer kanan putar kebelakang 180 (R+,R+)  	64. Layer Bawah kekanan (D+)  
5. Layer bawah kekiri (D-)  	25. Layer kanan putar kebelakang 180 (R+,R+)  	45. Layer atas kekiri (U+)  	65. Layer Depan kekiri (F-)  
6. Layer kiri kebawah (L+) 	26. Layer atas kekanan (U-) 	46. Layer Kanan Kebawah (R-) 	66. Layer kanan putar kebelakang 180 (R+,R+) 

			
7. Layer depan keatas (F-)  	27. Layer kanan keatas (R+)  	47. Layer atas kekanan (U-)  	67. Layer depan kekanan (F+)  
8. Layer kiri keatas (L-)  	28. Layer atas kekiri (U+)  	48. Layer Kanan Kebawah (R-)  	68. Layer atas kekiri (U+)  
9. Layer depan keatas (F-) 	29. Layer Bawah kekanan (D+) 	49. Layer belakang kekanan (B-) 	69. Layer Depan kekiri (F-) 

			
10. Layer kiri kedepan (R-)  	30. Layer Kanan Kebawah (R-)  	50. Layer kanan keatas (R+)  	70. Layer atas kekanan (U-)  
11. Layer depan kekanan (F+)  	31. Layer bawah kekiri (D-)  	51. Layer belakang keatas (B+)  	71. Layer kanan putar kebelakang 180 (R+,R+)  
12. Layer kanan putar kebelakang 180 (R+,R+) 	32. Layer Kanan Kebawah (R-) 	52. Layer kanan putar kebelakang 180 (R+,R+) 	72. Layer bawah kekiri (D-) 

			
13. Layer Bawah kekanan (D+)  	33. Layer Depan kekiri (F-)  	53. Layer depan kekanan (F+)  	73. Layer depan putar 180 (F-,F-)  
14. Layer Kanan Kebawah (R-)  	34. Layer kanan keatas (R+)  	54. Layer atas kekiri (U+)  	74. Layer Bawah kekanan (D+)  
15. Layer bawah kekiri (D-) 	35. Layer depan kekanan (F+) 	55. Layer Depan kekiri (F-) 	75. Layer Depan kekiri (F-) 

			
16. Layer kanan keatas (R+)  	36. Layer kanan keatas (R+)  	56. Layer atas kekanan (U-)  	76. Layer bawah kekiri (D-)  
17. Layer belakang keatas (B+)  	37. Layer belakang keatas (B+)  	57. Layer kanan putar kebelakang 180 (R+,R+)  	77. Layer depan putar 180 (F-,F-)  
18. Layer Kanan Kebawah (R-) 	38. Layer Kanan Kebawah (R-) 	58. Layer bawah kekiri (D-) 	78. Layer Bawah kekanan (D+) 

			
19. Layer belakang kekanan (B-)  	39. Layer belakang kekanan (B-)  	59. Layer depan putar 180 (F-,F-)  	79. Layer Depan kekiri (F-)  
20. Layer atas kekanan (U-)  	40. Layer Kanan Kebawah (R-)  	60. Layer Bawah kekanan (D+)  	80. Layer kanan putar kebelakang 180 (R+,R+) / Selesai  

3.4.3 Konversi Kode Notasi Rubik

Terminologi yang biasa dipakai untuk menjelaskan suatu algoritma *Rubik's Cube* sebagai berikut:

1. Rubik's Cube terdiri atas enam buah sisi: depan (*front*), belakang (*back*), kiri (*left*), kanan (*right*), atas (*up*), dan bawah (*down*).
2. Gerakan memutar sisi Rubik's Cube searah jarum jam disimbolkan dengan huruf pertama dari nama sisi tersebut dalam bahasa Inggris. Untuk yang berlawanan arah dengan jarum jam, ditambahkan tanda petik ('). Contohnya, F berarti memutar sisi depan (***f****ront*) searah jarum jam, sedangkan L' berarti memutar sisi kiri (***l****eft*) berlawanan arah dengan jarum jam.

Pada aplikasi yang akan dirancang kode notasi tersebut mengalami perubahan pada algoritma yang digunakan dalam aplikasi. Perubahan ini bertujuan agar sistem lebih mudah dalam membaca perintah untuk menjalankan kode notasi yang dituliskan. Perubahan dapat dilihat pada tabel 3.3 dan tabel 3.4 sebagai berikut :

Tabel 3.3. Konversi Notasi Rubik Searah Jarum Jam

Kode Notasi Putar Searah Jarum Jam	Kode Notasi Pada Aplikasi
F (*front* atau sisi depan)	**F+**
B (*back* atau sisi belakang)	**B+**
U (*up* atau sisi atas)	**U+**
D (*down* atau sisi bawah)	**D+**
R (*right* atau sisi kanan)	**R+**
L (*left* atau sisi kiri)	**L+**

Tabel 3.4. Konversi Notasi Rubik Berlawanan Arah Jarum Jam

Kode Notasi Putar Berlawanan Arah Jarum Jam	Kode Notasi Pada Aplikasi
F' (*front* atau sisi depan)	**F-**
B' (*back* atau sisi belakang)	**B-**
U' (*up* atau sisi atas)	**U-**
D' (*down* atau sisi bawah)	**D-**
R' (*right* atau sisi kanan)	**R-**
L' (*left* atau sisi kiri)	**L-**

3.5 Analisis Kebutuhan Non Fungsional

Analisis non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

Pada analisis kebutuhan sistem non fungsional ini dijelaskan analisis mengenai analisis masukan, analisis keluaran, perangkat keras (*hardware)*, perangkat lunak (*software)*, dan pengguna (*user*) sebagai bahan analisis kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diterapkan.

a. Analisis masukan

Analisis masukan dilakukan untuk mengetahui masukan data apa saja yang diperlukan oleh sistem.

b. Analisis keluaran

Analisis keluaran dilakukan untuk mengetahui keluaran dari sistem yang dibutuhkan oleh *user* yaitu berupa jalur dan pelacakan pencarian dari titik awal ke titik tujuan.

c. Analisis kebutuhan perangkat keras *(hardware)*

Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras yang sesuai dengan kebutuhan aplikasi. Tabel 3.5 adalah spesifikasi minimun perangkat keras yang dibutuhkan agar dapat menjalankan aplikasi permainan *rubik cube* secara optimal.

Tabel 3.5. Spesifikasi minimum perangkat keras

Komputer	**Spesifikasi perangkat keras**
Prosesor	Prosesor 1.6 Ghz
Resolusi layar	1366 x 768 pixel
Memori	Memori 128 Mb
Harddisk	Harddisk 20 GB
Keyboard dan mouse	-

d. Analisis kebutuhan perangkat lunak (*software*)

Penerapan algoritma layer by layer dalam permainan rubim cube memiliki kebutuhan perangkat lunak (*software*) pada tabel 3.6.

Tabel 3.6. Spesifikasi perangkat lunak

Komputer	Spesifikasi perangkat lunak
Sistem Operasi	Window Xp, Window 7
Bahasa Pemograman	*Macromedia Flash* 8.0

Dari hasil pengamatan perangkat lunak, terdapat beberapa hal penting yang harus diperhatikan agar sistem yang akan dirancang dapat berjalan dengan baik dan sesuai dengan yang diharapkan. Pengguna diharapkan dapat memenuhi kebutuhan perangkat lunak yang telah direkomendasikan agar aplikasi dapat berjalan dengan baik.

e. Analisis pengguna

Pengguna yaitu *public* yang dapat mengerti dan memahami komputer sehingga dapat menggunakan aplikasi yang akan dibangun dan mengerti gerakan dasar cara menyelesaikan rubik.

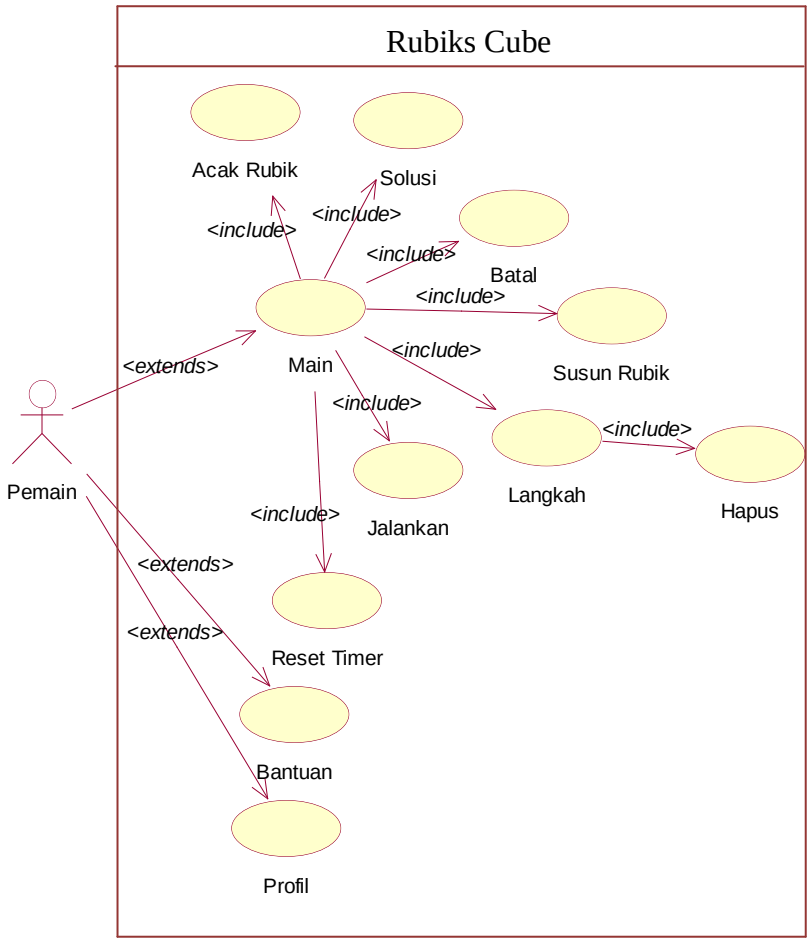
3.6 Analisis Kebutuhan Fungsional

Pemodelan yang digunakan pada aplikasi yang akan dibuat adalah analisis perancangan dan pengembangan perangkat lunak berbasis objek, dimana alat yang digunakan untuk pengembangan perangkat lunak, yaitu menggunakan pemodelan UML(*Unified Modeling Language*).

3.6.1 *Use Case* Diagram

Use case diagram ini digunakan untuk menggambarkan hubungan sejumlah external aktor dengan *use case* yang terdapat dalam sistem. *Use case*

diagram ini hanya menggambarkan keadaan lingkungan sistem yang dapat dilihat dari luar oleh aktor.



Gambar 3.15. Use Case Diagram Aplikasi Penyelesaian rubik

A. Definisi Use Case

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case*.

Table 3.7. Definisi *use case*

No	*Use case*	Deskripsi
1.	Main	Proses untuk menampilkan tampilan permainan rubik

2.	Acak Rubik	Proses untuk mengacak posisi dan warna rubik
3.	Solusi	Proses untuk menyelesaikan rubik yang teracak
4.	Batal	Proses untuk menghentikan gerakan notasi rubik
5.	Susun Rubik	Proses untuk mengembalikan susunan warna rubik kembali pada kelompok warnanya.
6.	Langkah	Proses yang berguna untuk menampilkan pesan langkah notasi gerakan rubik
7.	Hapus	Proses untuk menghapus pesan pada menu langkah
8.	Profil	Proses untuk menampilkan profil
9.	Bantuan	Proses untuk menampilkan fitur yang dapat digunakan pada aplikasi
10.	Jalankan	Proses untuk menjalankan kode notasi rubik yang ditulis
11.	*Reset Timer*	Proses untuk mengulang waktu

B. Skenario Use Case

Skenario Use Case berfungsi untuk menjelaskan lebih detail mengenai modul-modul yang terdapat dalam Aplikasi klasifikasi ini. Adapun penjelasan mengenai keterangan tabel Skenario Use Case adalah sebagai berikut :

1. Ringkasan merupakan deskripsi mengenai kegunaan atau fungsi modul.

2. Rasional merupakan deskripsi dari Use Case.

3. Pengguna menggunakan deskripsi mengenai pengguna Use Case.

4. Prekondisi merupakan deskripsi keadaan sebelum modul dijalankan.

5. Aliran dasar merupakan alur kerja pengguna.

6. Aliran alternatif merupakan aliran dasar lain.

7. Post kondisi merupakan deskripsi keadaan setelah modul dijalankan.

Tabel 3.8 Skenario *Use Case* Menu Main

Identifikasi	
Nomor	1
Nama	Main
Tujuan	Menampilkan tampilan untuk permainan rubik
Deskripsi	Proses untuk menampilkan tampilan permainan
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman utama	
2. Pemain memilih tombol main	3. Menampilkan tampilan permainan rubik
Kondisi akhir	Rubik dalam keadaan teracak

Tabel 3.9 Skenario *Use Case* Menu Acak Rubik

Identifikasi	
Nomor	2
Nama	Acak Rubik
Tujuan	Mengacak posisi dan warna rubik
Deskripsi	Proses untuk mengacak posisi dan warna rubik
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman main	
2. Pemain memilih tombol acak rubik	3. Cek data acak rubik
	4. Data acak valid
	5. Sistem akan mengacak rubik

	6. Mengacak rubik kemudian menampilkan rubik yang sudah teracak kemudian sistem berhenti
Kondisi akhir	Rubik dalam keadaan teracak

Tabel 3.10 Skenario *Use Case* Menu Solusi

Identifikasi	
Nomor	3
Nama	Solusi
Tujuan	Menyelesaikan rubik yang teracak
Deskripsi	Proses untuk menyusun kembali warna rubik yang teracak
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman main	
2. Pemain memilih tombol solusi	3. Cek data penyelesaian rubik
	4. Data penyelesaian valid
	5. Rubik terselesaikan sistem berhenti
	6. Menampilkan rubik dalam keadaan tersusun sesuai dengan kelompok warnanya.
Kondisi akhir	Rubik terselesaikan, warna tersusun kembali disetiap sisi

Tabel 3.11 Skenario *Use Case* Sub Menu Batal

Identifikasi	
Nomor	4
Nama	Batal
Tujuan	Menghentikan gerakan rubik
Deskripsi	Proses untuk menghentikan rubik yang bergerak
Aktor	Pemain Rubik
Skenario Utama	

Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman main	
2. Pemain memilih tombol batal	3. Cek data rubik
	4. Menghentikan notasi gerakan rubik
Kondisi akhir	Rubik berhenti bergerak

Tabel 3.12 Skenario *Use Case* Menu *Reset* Rubik

Identifikasi	
Nomor	5
Nama	Susun Rubik
Tujuan	Mengembalikan warna rubik tersusun pada setiap sisinya
Deskripsi	Proses untuk membuat rubik kembali dalam keadaan tidak teracak
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman main	
2. Pemain memilih tombol *reset rubik*	3. Mengecek data *reset rubik* untuk mengembalikan rubik kembali tersusun
	4. Mengembalikan rubik kembali tersusun sesuai dengan kelompok warnanya
Kondisi akhir	Rubik dalam keadaan tidak teracak

Tabel 3.13 Skenario *Use Case* Menu Langkah

Identifikasi	
Nomor	6
Nama	Langkah
Tujuan	Menampilkan pesan yang berisi kode notasi gerakan

	rubik
Deskripsi	Proses untuk menampilkan pesan yang menjelaskan kode notasi yang dilakukan oleh gerakan rubik
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman main	
2. Pemain memilih tombol langkah	3. Menampilkan kotak pesan yang berisi kode notasi gerakan rubik
Kondisi akhir	Menampilkan kotak pesan berisi kode notasi rubik

Tabel 3.14 Skenario *Use Case* Menu Hapus

Identifikasi	
Nomor	7
Nama	Hapus
Tujuan	Menghapus kode notasi pada menu langkah
Deskripsi	Proses untuk menghapus kode notasi gerakan rubik yang ada pada kotak pesan menu langkah
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman utama	
2. Pemain memilih tombol hapus	3. Menghapus kode notasi pada menu langkah
Kondisi akhir	Kotak pesan pada menu langkah kembali kosong

Tabel 3.15 Skenario *Use Case* Menu Profil

Identifikasi	
Nomor	8
Nama	Profil
Tujuan	Menampilkan profil
Deskripsi	Proses untuk menampilkan tampilan tentang profil

Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman utama	
2. Pemain memilih tombol profil	3. Menampilkan tampilan yang memberikan keterangan tentang profil
Kondisi akhir	Menampilkan tampilan profil

Tabel 3.16 Skenario *Use Case* Menu Bantuan

Identifikasi	
Nomor	9
Nama	Bantuan
Tujuan	Menampilkan tampilan bantuan
Deskripsi	Proses untuk menampilkan kotak pesan yang berisikan bantuan pada aplikasi yang dibangun
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman utama	
2. Pemain memilih tombol bantuan	3. Menampilkan tampilan yang memberikan keterangan tentang fitur bantuan yang dapat digunakan pada aplikasi
Kondisi akhir	Menampilkan tampilan bantuan

Tabel 3.17 Skenario *Use Case* Menu Jalankan

Identifikasi	
Nomor	10
Nama	Jalankan
Tujuan	Menjalankan kode notasi rubik
Deskripsi	Proses untuk menjalankan kode notasi rubik sesuai yang

	dituliskan pada *textbox*
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman main	
2. Pemain memilih tombol jalankan	3. Cek kode notasi rubik untuk dijalankan
	4. Data kode valid
	5. Jalankan kode notasi yang diinputkan
Kondisi akhir	Menampilkan tampilan rubik yang bergerak sesuai dengan kode notasi yang diinginkan oleh pemain

Tabel 3.18 Skenario *Use Case* Menu *Reset Timer*

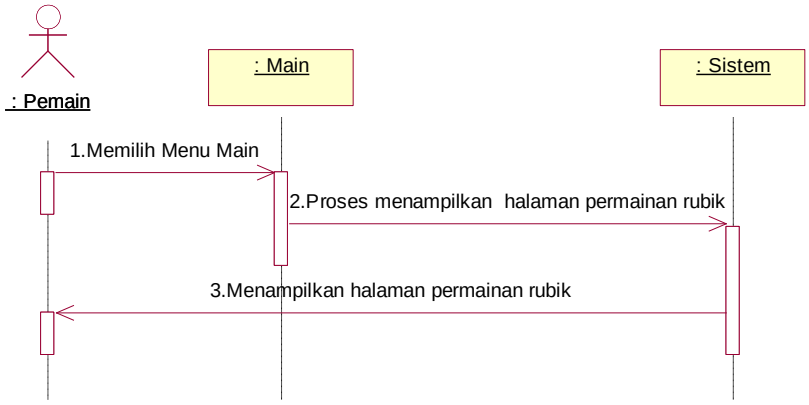
Identifikasi	
Nomor	11
Nama	*Reset Timer*
Tujuan	Untuk mengulang waktu dari pertama
Deskripsi	Proses untuk me-*reset* waktu kembali kepada semula
Aktor	Pemain Rubik
Skenario Utama	
Kondisi Awal	
Aksi actor	Reaksi sistem
1. Pemain berada pada halaman main	
2. Pemain memilih tombol *reset* waktu	3. Mengulang waktu jadi kembali seperti semula
Kondisi akhir	Menampilkan waktu kembali seperti semula.

C. Sequence Diagram

Interaksi yang terjadi pada sistem akan digambarkan dengan *Sequence Diagram*. *Sequence Diagram* digunakan untuk menggambarkan interaksi antara sejumlah *object* dalam urutan waktu. Kegunaannya untuk menunjukkan rangkaian

pesan yang dikirim antara *object* juga interaksi antar *object* yang terjadi pada titik tertentu dalam eksekusi sistem.

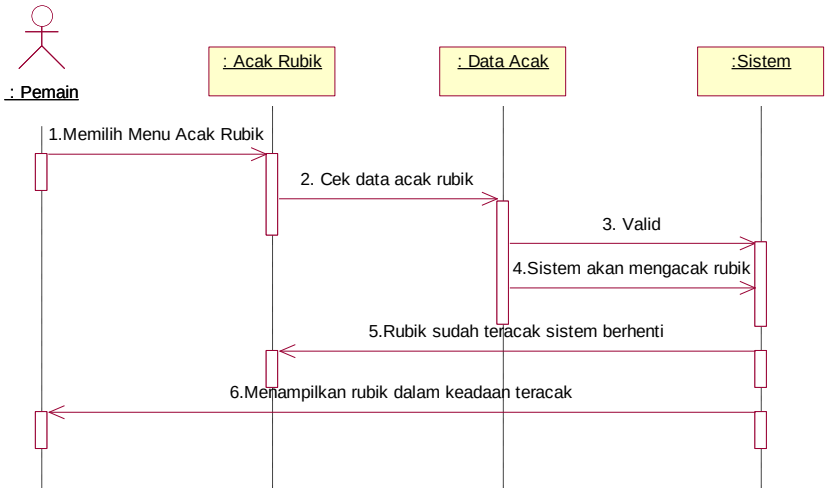
1. *Sequence Diagram* Menu Main



Gambar 3.16. *Sequence Diagram* Main

Sequence Diagram menu main menjelaskan *actor* memilih menu main, kemudian sistem menampilkan tampilan permainan rubik.

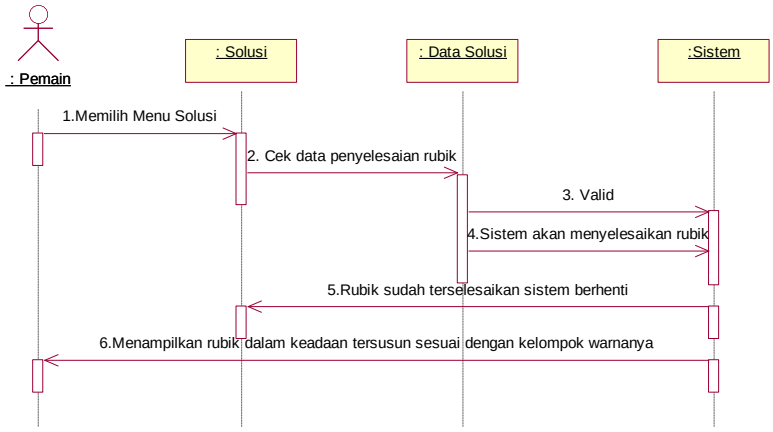
2. *Sequence Diagram* Menu Acak Rubik



Gambar 3.17. *Sequence Diagram* Menu Acak Rubik

Sequence Diagram menu Acak Rubik menjelaskan *actor* memilih menu acak rubik, kemudian sistem akan mengacak rubik dan berhenti apabila acakan telah selesai.

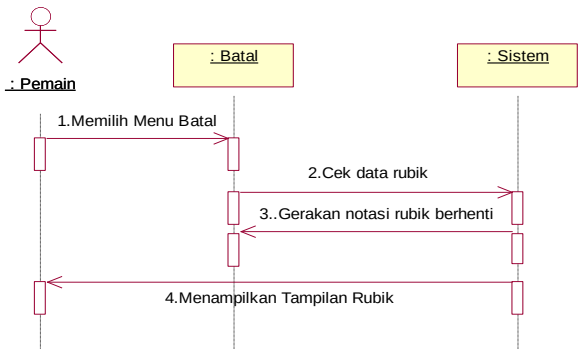
3. *Sequence Diagram* Menu Solusi



Gambar 3.18. *Sequence Diagram* Solusi

Sequence Diagram menu solusi menjelaskan *actor* memilih menu solusi. Ketika *actor* memilih menu solusi sistem akan menyelesaikan *rubik cube* yang teracak dengan algoritma *layer by layer*.

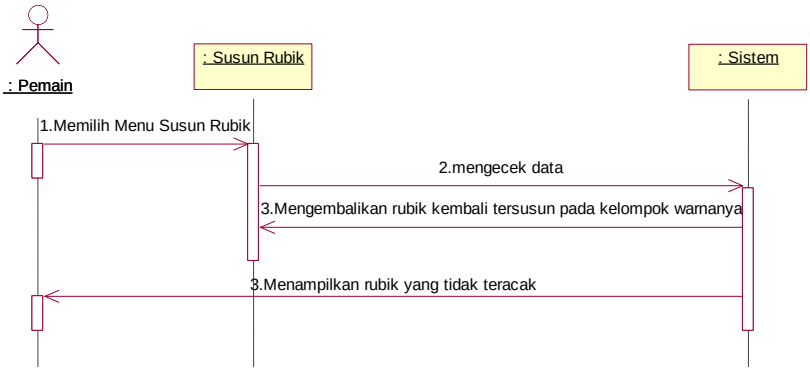
4. *Sequence Diagram* Menu Batal



Gambar 3.19. *Sequence Diagram* Menu Batal

Sequence Diagram menu batal menjelaskan *actor* memilih menu batal. Ketika *actor* memilih menu batal sistem akan menghentikan gerakan notasi rubik yang sedang bergerak.

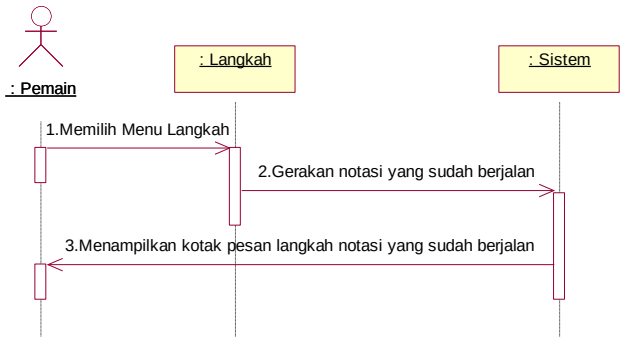
5. *Sequence Diagram* Menu Susun Rubik



Gambar 3.20. *Sequence Diagram* Menu Susun Rubik

Sequence Diagram menu susun rubik menjelaskan *actor* memilih menu susun rubik. Ketika *actor* memilih menu susun rubik akan menampilkan rubik dalam keadaan tidak teracak yaitu warna–warna rubik pada setiap sisi telah tersusun sesuai kelompoknya.

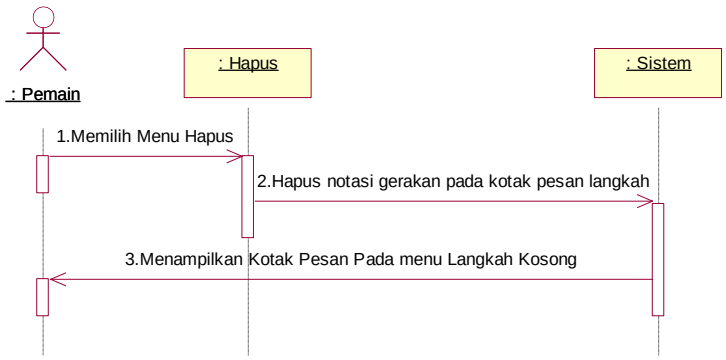
6. *Sequence Diagram* Menu Langkah



Gambar 3.21. *Sequence Diagram* Menu Langkah

Sequence Diagram menu langkah menjelaskan *actor* memilih menu langkah. Ketika *actor* memilih menu langkah sistem akan menampilkan tampilan yang menginformasikan kode notasi gerakan rubik yang pernah berjalan.

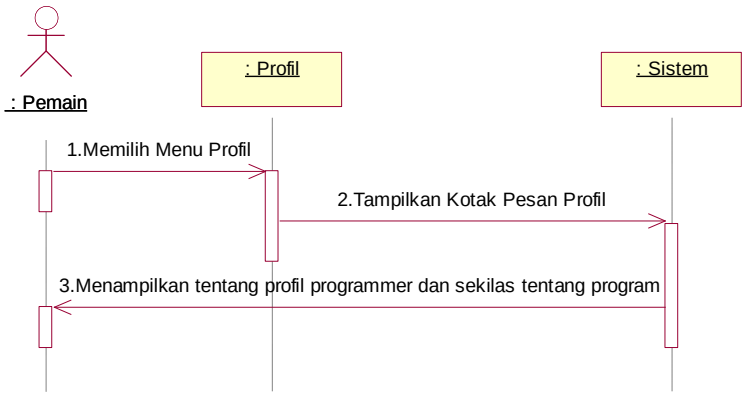
7. *Sequence Diagram* Menu Hapus



Gambar 3.22. *Sequence Diagram* Menu Hapus

Sequence Diagram menu hapus menjelaskan *actor* memilih menu hapus. Ketika *actor* memilih menu hapus sistem akan menghapus kode notasi gerakan yang ada pada tampilan "menu langkah".

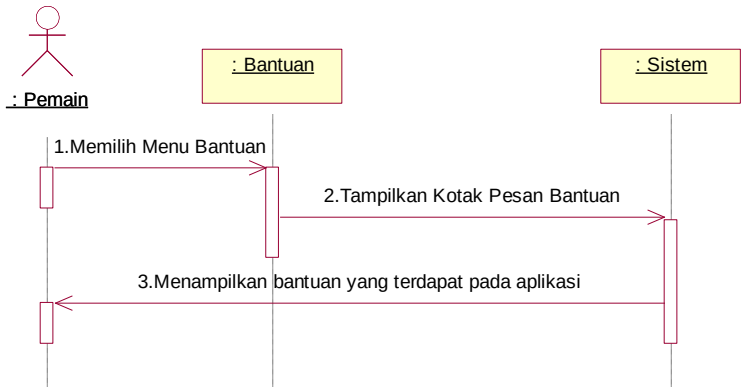
8. *Sequence Diagram* Menu Profil



Gambar 3.23. *Sequence Diagram* Menu Profil

Sequence Diagram menu profil menjelaskan *actor* memilih menu profil. Ketika *actor* memilih menu profil sistem akan menampilkan informasi profil.

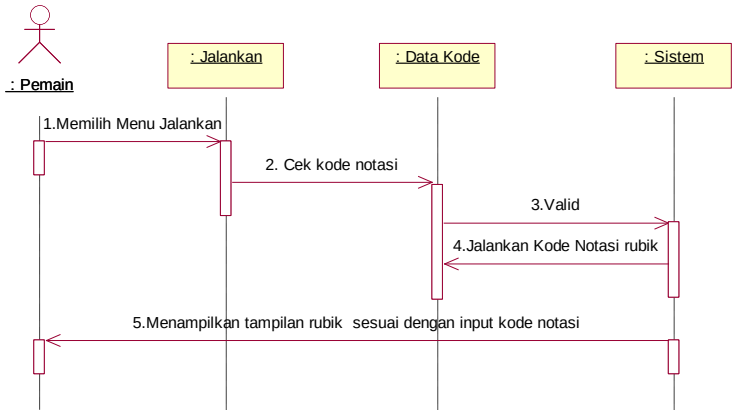
9. *Sequence Diagram* Menu Bantuan



Gambar 3.24. *Sequence Diagram* Menu Bantuan

Sequence Diagram menu bantuan menjelaskan *actor* memilih menu bantuan. Ketika *actor* memilih menu bantuan sistem menampilkan informasi mengenai fitur – fitur dan navigasi yang ada pada aplikasi.

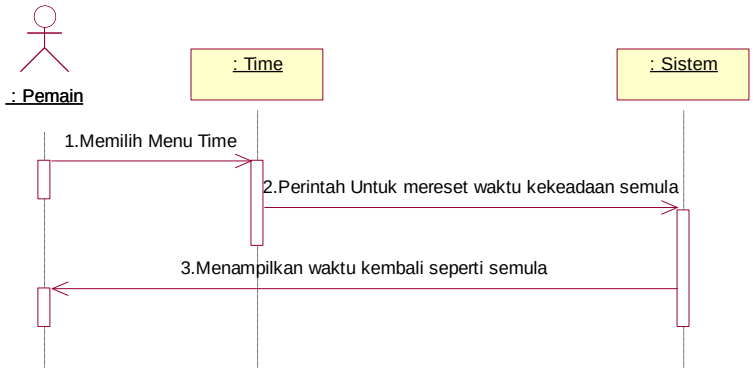
10. *Sequence Diagram* Menu Jalankan



Gambar 3.25. *Sequence Diagram* Menu Jalankan

Sequence Diagram menu jalankan menjelaskan *actor* memilih menu jalankan, kemudian sistem akan menggerakkan rubik dengan mengikuti kode notasi yang sudah diinputkan.

11. *Sequence Diagram* Menu *Reset Timer*



Gambar 3.26. *Sequence Diagram* Menu *Reset Timer*

Sequence Diagram menu *reset timer* menjelaskan *actor* memilih menu *reset timer*. Ketika *actor* memilih menu tersebut sistem akan me-*reset* ulang waktu menjadi seperti semula.

D. *Activity Diagram*

Activity diagram adalah teknik untuk menggambarkan logika *procedural*, proses bisnis, dan alur kerja. Dalam beberapa hal, diagram ini mirip dengan sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dengan notasi diagram alir adalah mendukung *behavior parallel*.

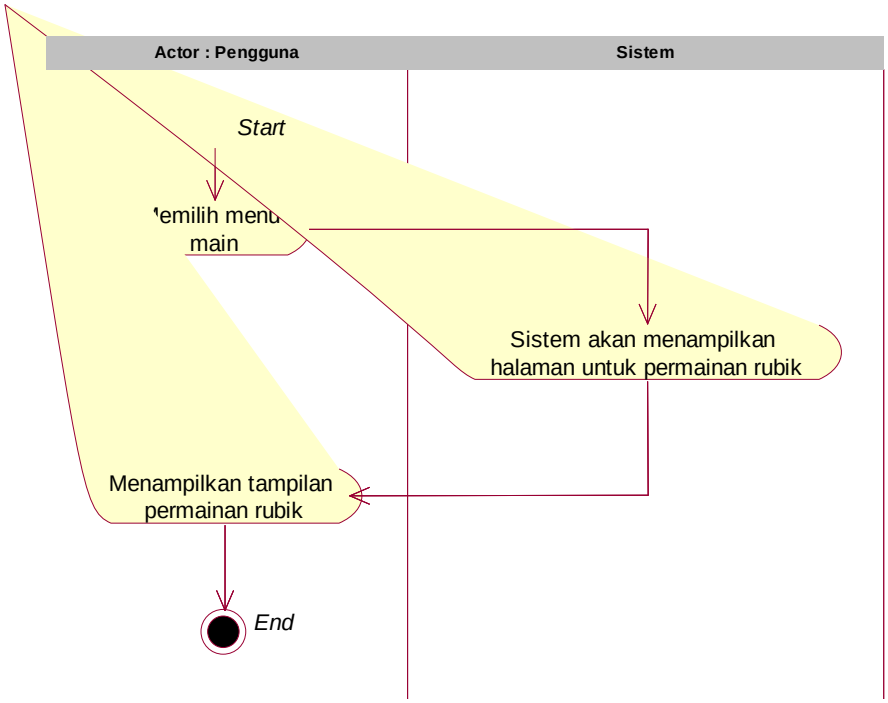
Diagram ini dapat memodelkan sebuah alur kerja dari suatu aktivitas ke aktivitas lainnya atau dari suatu aktivitas ke dalam sesaat. Seringkali bermanfaat

bila kita membuat sebuah *activity diagram*, juga sangat berguna ketika kita ingin menggambarkan perilaku dalam berbagai *use case* beriteraksi.

Activity diagram memungkinkan siapapun yang melakukan proses untuk memilih dalam melakukannya. Dengan kata lain, diagram hanya menyebutkan aturan – aturan rangkaian dasar yang harus kita ikuti.

1. *Activity Diagram* Menu Main

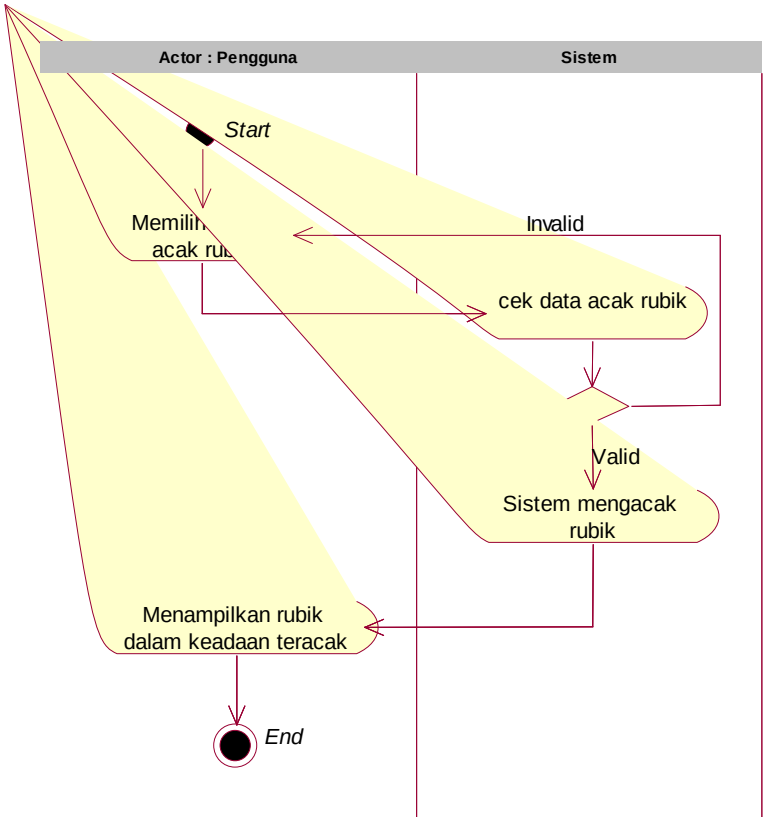
Menggambarkan alur kerja (*workflow*) dari proses memilih menu main. *Activity diagram* Menu main sebagai berikut :



Gambar 3.27. *Activity Diagram* Menu Main

2. *Activity Diagram* Menu Acak Rubik

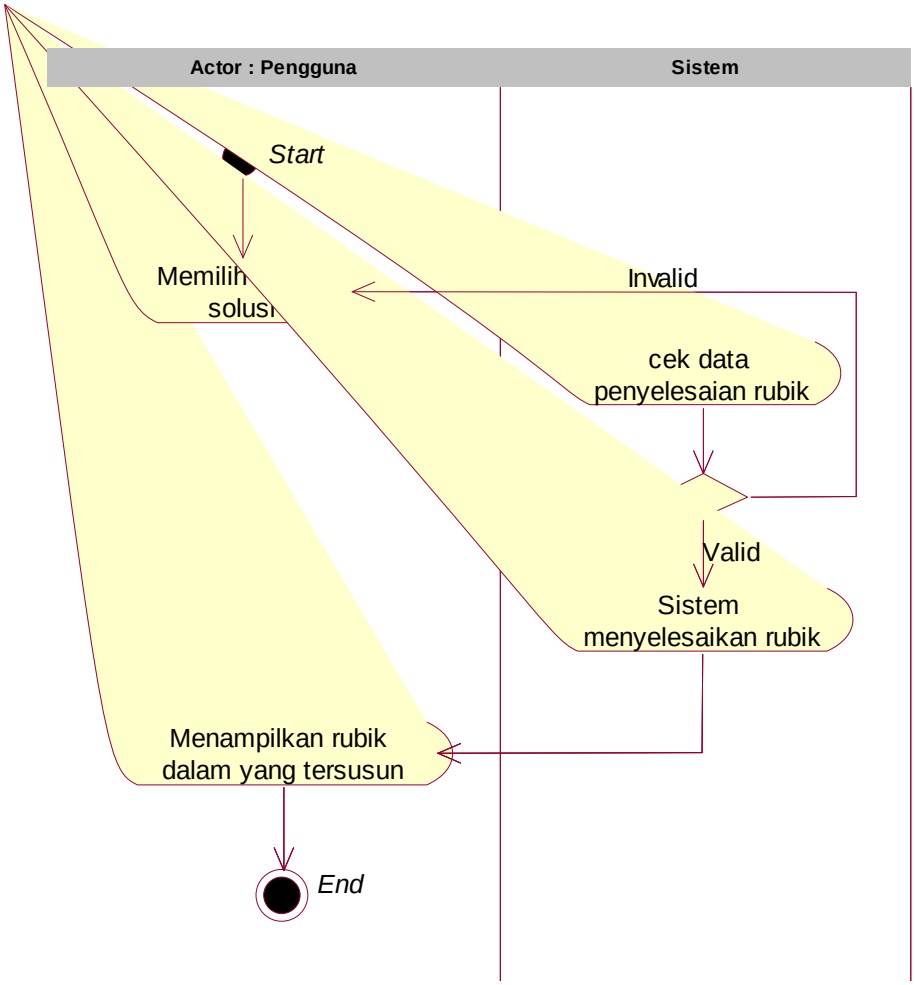
Menggambarkan alur kerja (*workflow*) dari proses memilih menu acak rubik. *Activity diagram* Menu acak rubik sebagai berikut :



Gambar 3.28. *Activity Diagram* Menu Acak Rubik

3. Activity Diagram Menu Solusi

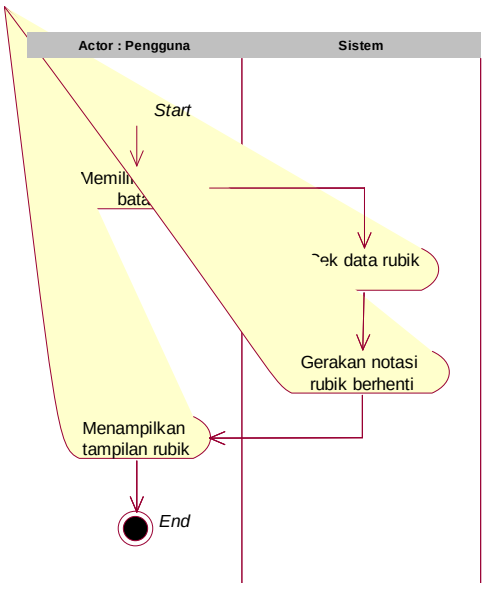
Menggambarkan alur kerja (*workflow*) dari proses menu solusi. *Activity diagram* Menu solusi sebagai berikut :



Gambar 3.29. *Activity Diagram* Menu Solusi

4. *Activity Diagram* Menu Batal

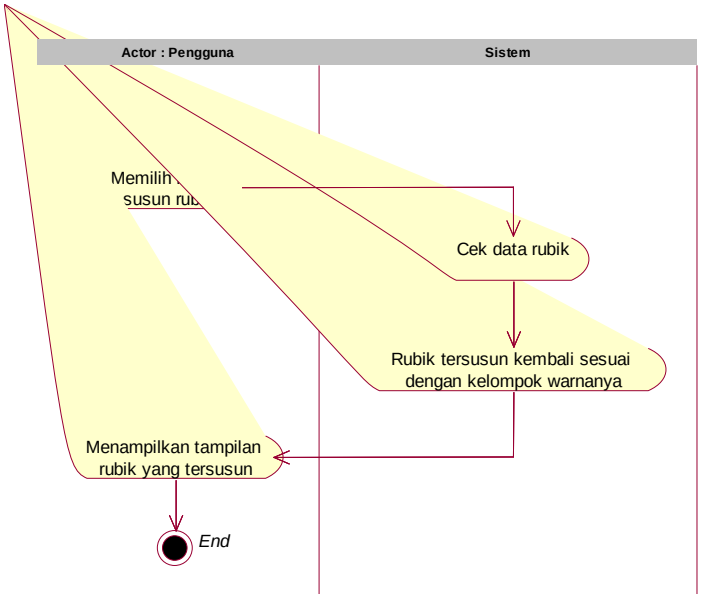
Menggambarkan alur kerja (*workflow*) dari proses menu batal. *Activity diagram* Menu Batal sebagai berikut :



Gambar 3.30. *Activity Diagram* Menu Batal

5. *Activity Diagram* Menu Susun *Rubik*

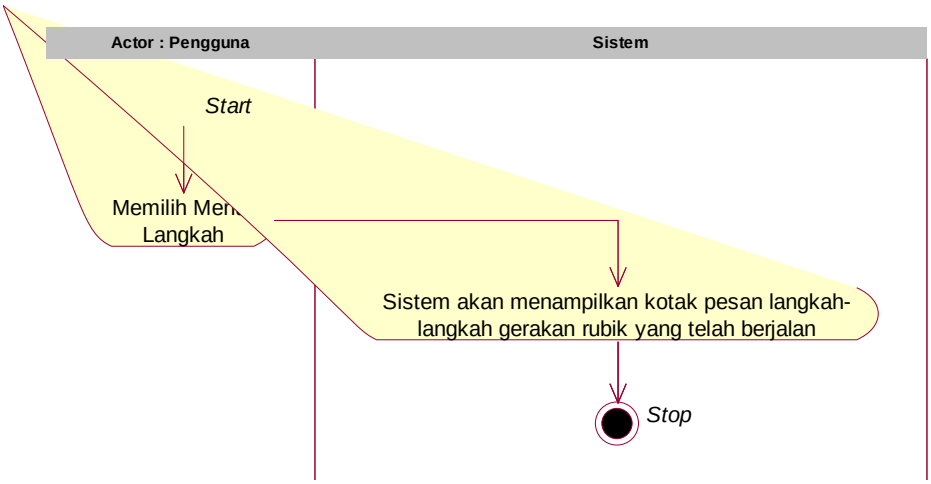
Menggambarkan alur kerja *(workflow)* dari proses menu susun *rubik*. *Activity diagram* Menu susun *rubik* sebagai berikut :



Gambar 3.31. *Activity Diagram* Menu Susun Rubik

6. *Activity Diagram* Menu Langkah

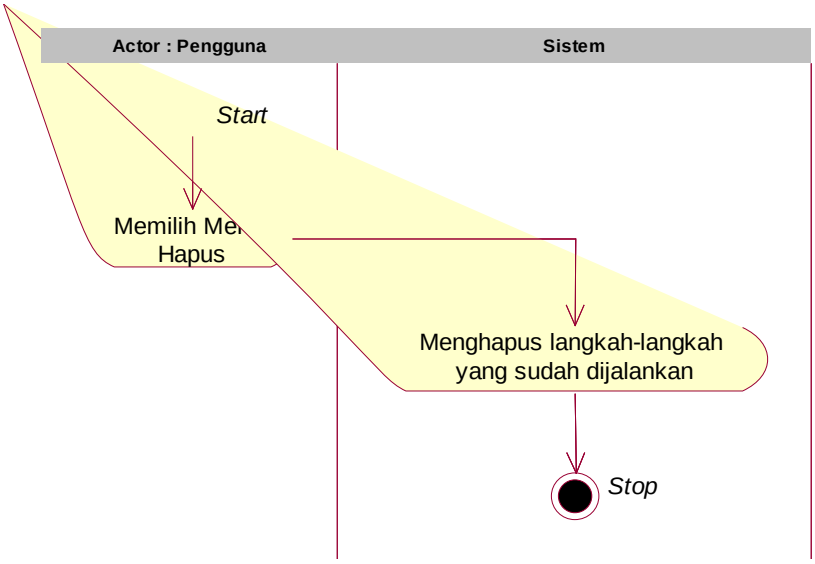
Menggambarkan alur kerja *(workflow)* dari proses menu langkah. *Activity diagram* Menu langkah sebagai berikut :



Gambar 3.32. *Activity Diagram* Menu Langkah

7. *Activity Diagram* Menu Hapus

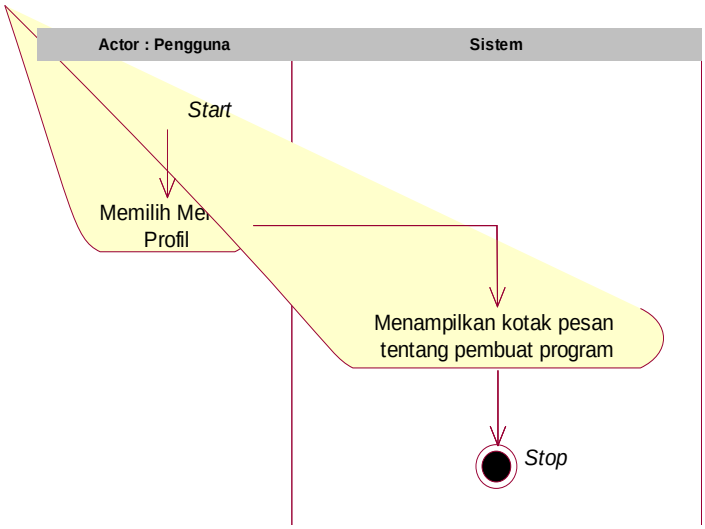
Menggambarkan alur kerja *(workflow)* dari proses menu hapus. *Activity diagram* Menu hapus sebagai berikut :



Gambar 3.33. *Activity Diagram* Menu Hapus

8. *Activity Diagram* Menu Profil

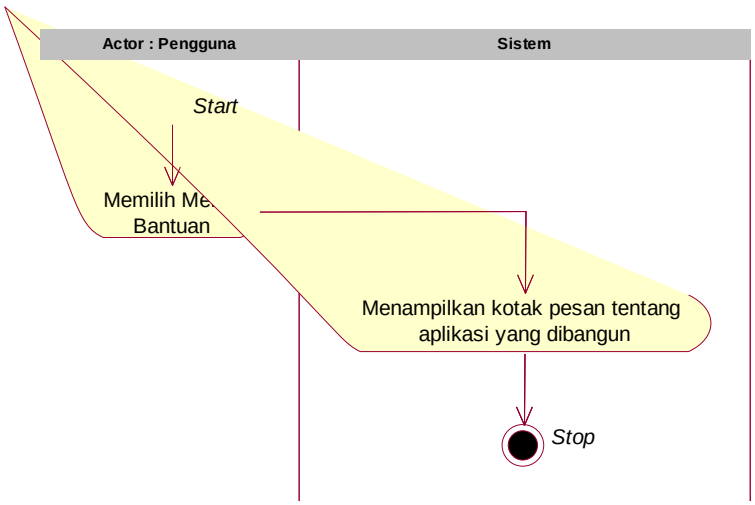
Menggambarkan alur kerja *(workflow)* dari proses menu profil. *Activity diagram* Menu Profil sebagai berikut :



Gambar 3.34. *Activity Diagram* Menu Profil

9. *Activity Diagram* Menu Bantuan

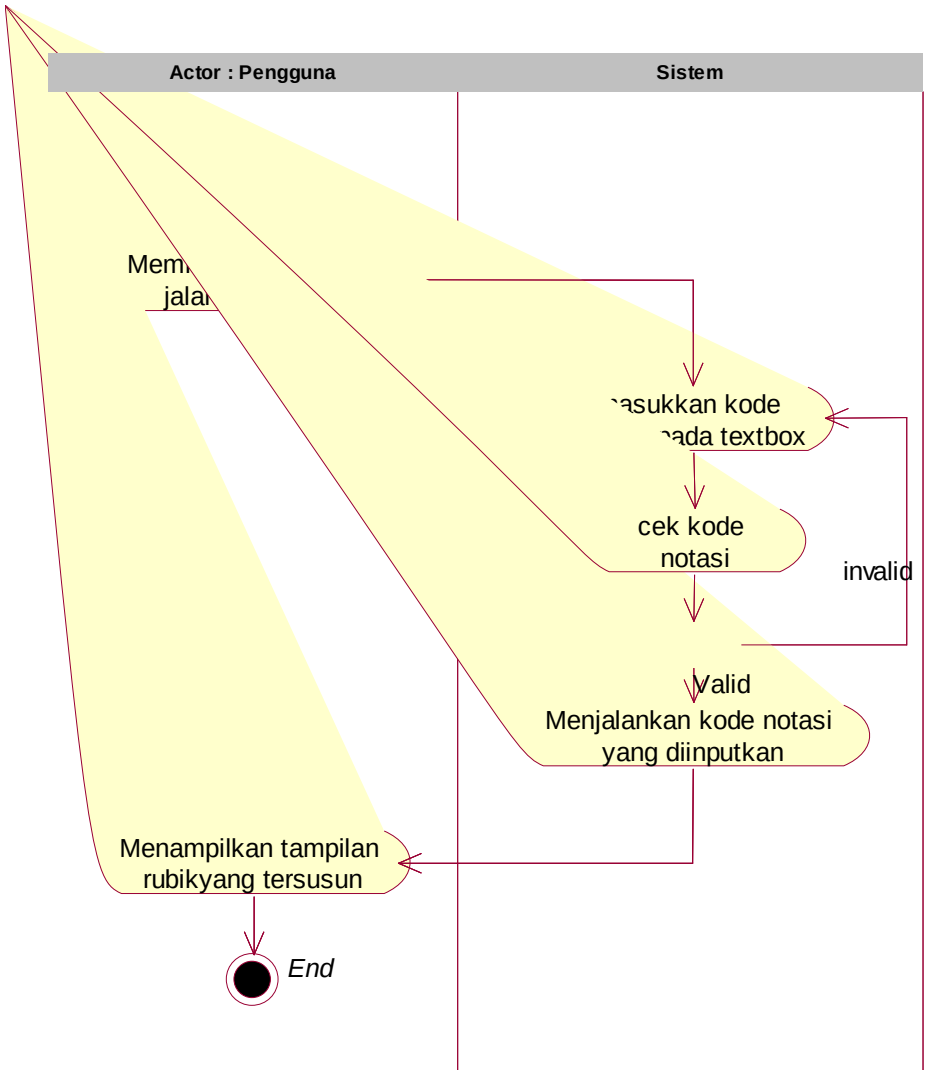
Menggambarkan alur kerja *(workflow)* dari proses bantuan. *Activity diagram* Menu Bantuan sebagai berikut :



Gambar 3.35. *Activity Diagram* Menu Bantuan

10. *Activity Diagram* Menu Jalankan

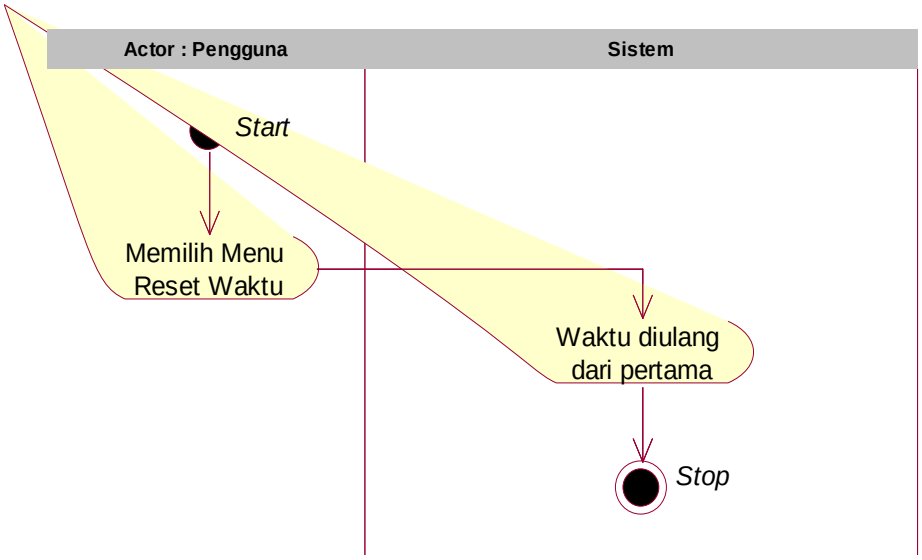
Menggambarkan alur kerja *(workflow)* dari proses menu jalankan. *Activity diagram* Menu Jalankan sebagai berikut :



Gambar 3.36. *Activity Diagram* Menu Jalankan

11. *Activity Diagram Reset Timer*

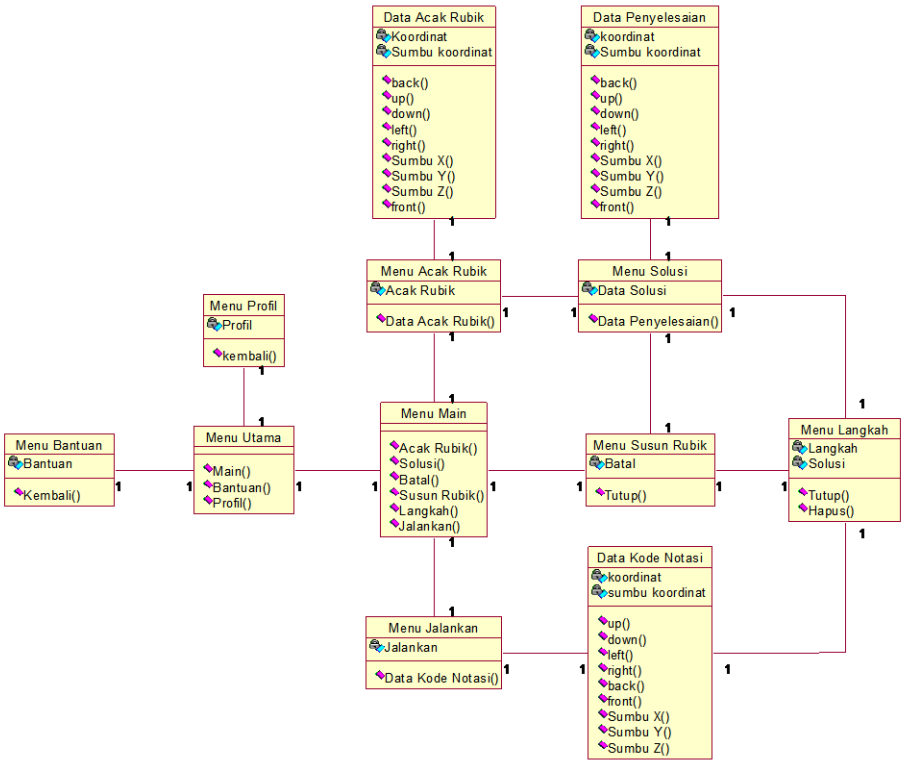
Menggambarkan alur kerja *(workflow)* dari proses menu *Reset Timer*. *Activity diagram* Menu *Reset Timer* sebagai berikut :



Gambar 3.37. *Activity Diagram* Menu *Reset Timer*

E. *Class Diagram*

Class diagram berfungsi untuk menggambarkan relasi atau hubungan antar class. Class diagram untuk aplikasi permainan rubik sebagai berikut :



Gambar 3.38. *Class Diagram* Aplikasi Permainan Rubik

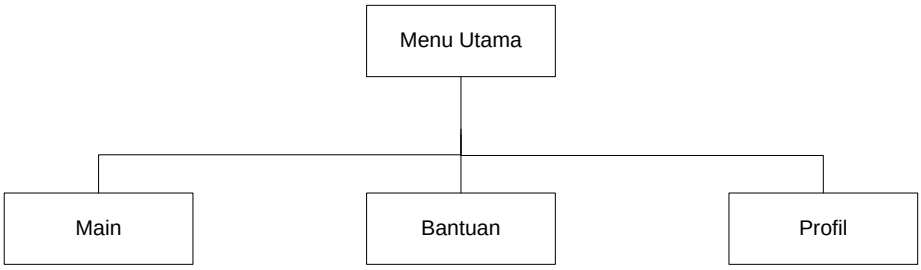
6.7 Perancangan Sistem

Perancangan sistem merupakan metodologi pengembangan suatu perangkat lunak yang dilakukan setelah melalui tahapan analisis. Perancangan bertujuan untuk memberikan gambaran secara terperinci dan merupakan tahap lanjutan dari analisis sistem. Dimana dalam hal ini digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengodean kedalam suatu bahasa pemrograman. Desain umum yang akan diaplikasikan bertujuan untuk memberikan gambaran secara umum kepada penggunaan tentang sistem yang akan dibangun.

3.7.1 Perancangan Struktur Menu

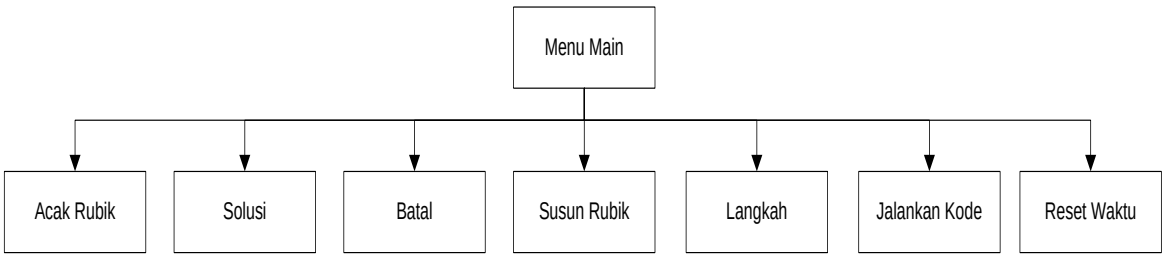
Perancangan struktur menu berisikan menu yang berfungsi memudahkan *user* didalam menggunakan sistem. Pada gambar dibawah ini dijelaskan perancangan struktur menu aplikasi Penyelesaian Permainan *Rubiks Cube*.

1. Perancangan Struktur Menu Utama



Gambar 3.39. Struktur menu utama

2. Perancangan Struktur Menu Main



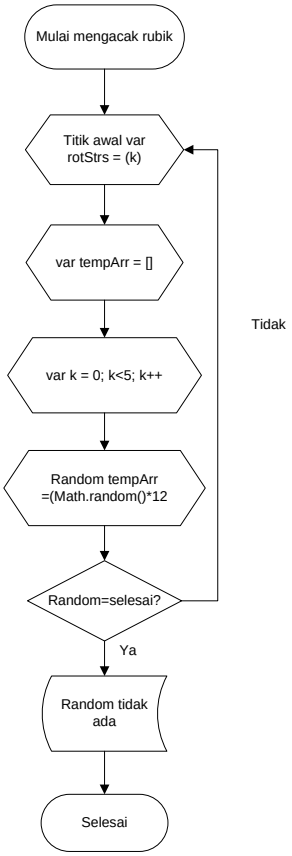
Gambar 3.40. Struktur menu main

3.7.2 Perancangan Prosedural

Bagan alir sistem *(flowchart system)* merupakan bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah dan merupakan cara penyajian dari suatu algoritma, sehingga dapat menjelaskan urutan prosedur-prosedur yang ada dalam sistem.

Pada gambar 3.38. dijelaskan bahwa cara kerja sistem yang akan dibangun yaitu untuk mengacak rubik sesuai dengan random yang ditentukan.

1. **Proses Pengacakan Rubik**

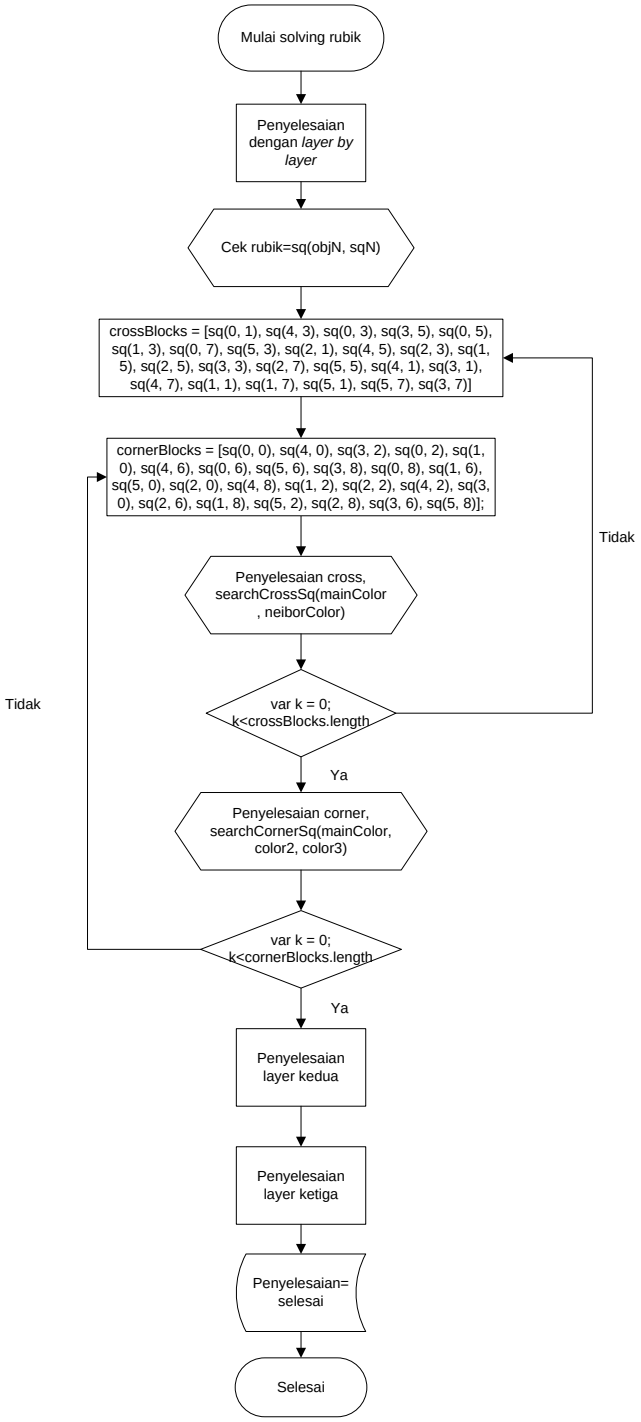


Gambar 3.41. *Flowchart* Proses Pengacakan Rubik

2. **Proses Penyelesaian Rubik**

Setelah *user* memilih menu acak rubik pada halaman main, maka pengacakan rubik pun dilakukan. Diagram alir algoritma *layer by layer* dapat dilihat pada gambar 3.42.

Pada gambar 3.42 menjelaskan cara kerja algoritma *layer by layer*. Algoritma ini bekerja dari titik awal penyelesaian rubik ke titik tujuan yaitu rubik tersusun kembali sesuai dengan kelompok warnanya.



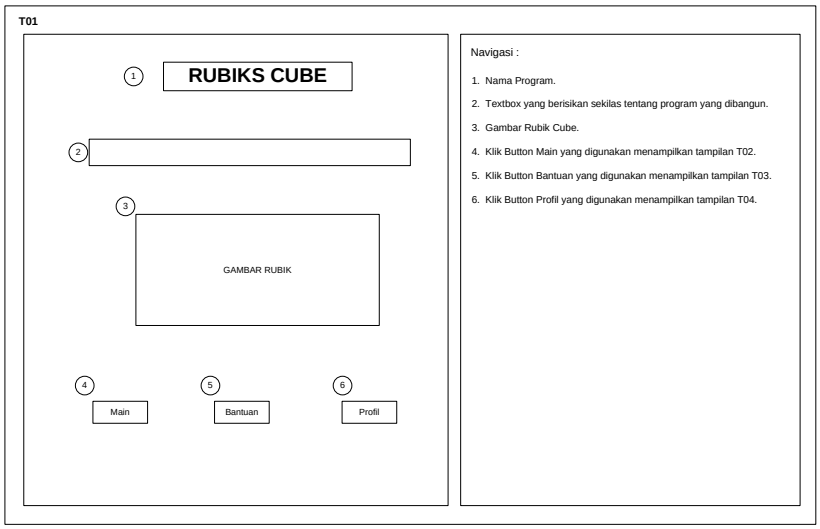
Gambar 3.42. *Flowchart* Proses Penyelesaian Rubik

3.7.3 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Perancangan antarmuka aplikasi *Rubik Cube* terbagi menjadi empat rancangan antarmuka yaitu rancangan antarmuka tampilan utama, rancangan antarmuka langkah penyelesaian, rancangan antarmuka profil, dan rancangan antarmuka bantuan serta penjelasan untuk tombol – tombol yang dapat digunakan pada aplikasi.

a. Rancangan tampilan utama

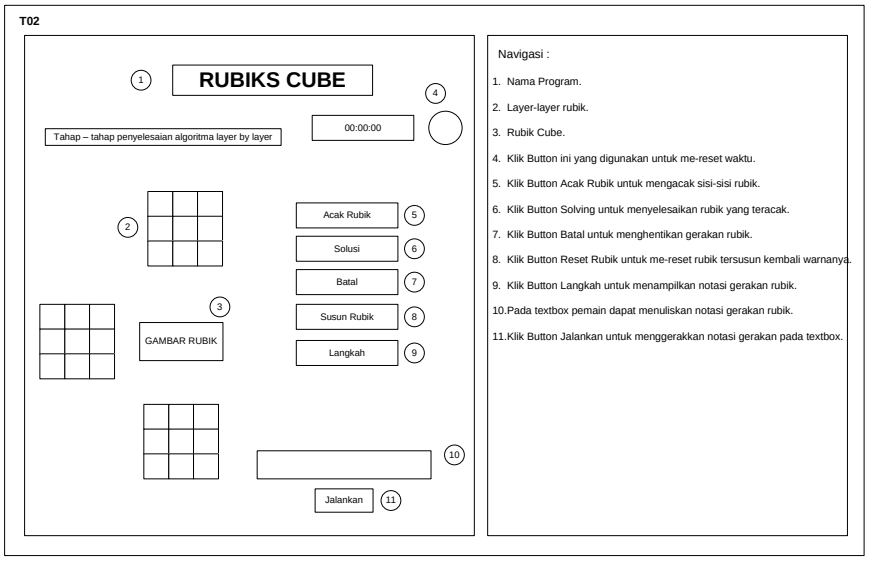
Rancangan antarmuka tampilan utama digunakan untuk menampilkan aplikasi utama. Pada rancangan antarmuka terdapat tombol - tombol yang mempunyai fungsi tertentu yang berguna untuk menjalankan aplikasi. Rancangan antarmuka utama dapat dilihat pada gambar 3.43 dibawah ini.



Gambar 3.43. Rancangan Tampilan Utama

b. Rancangan Tampilan Main

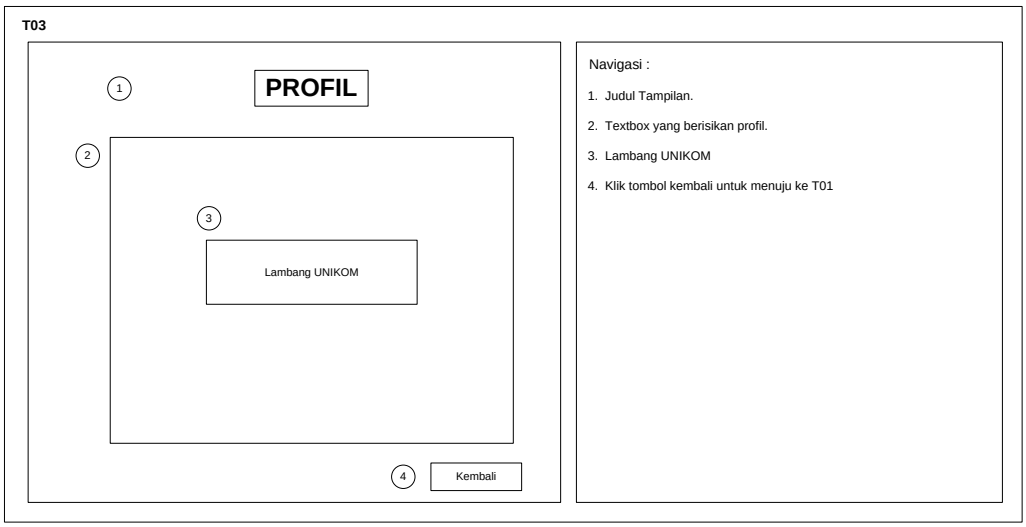
Rancangan tampilan main digunakan untuk menampilkan permainan rubik. Pada tampilan main terdapat beberapa menu yang digunakan oleh pemain untuk mengacak rubik kemudian menyelesaikan rubik. Tampilan tersebut untuk lebih jelas dapat dilihat pada gambar 4.44 dibawah ini.



Gambar 3.44. Rancangan Tampilan Main

c. Rancangan Tampilan Profil

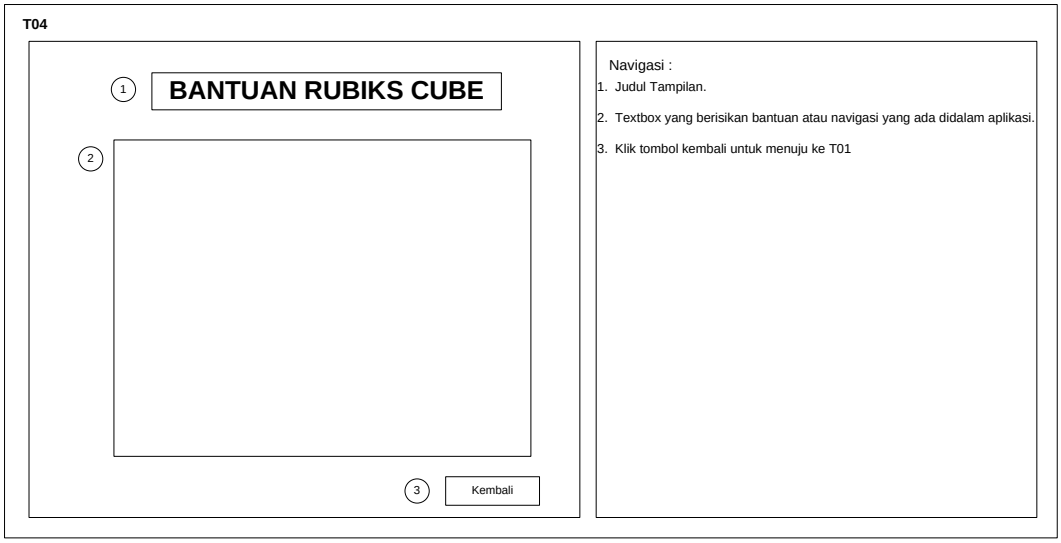
Rancangan tampilan profil digunakan untuk menampilkan informasi tentang program dan pembuatan program.



Gambar 3.45. Rancangan Tampilan Profil

d. Rancangan Tampilan Bantuan

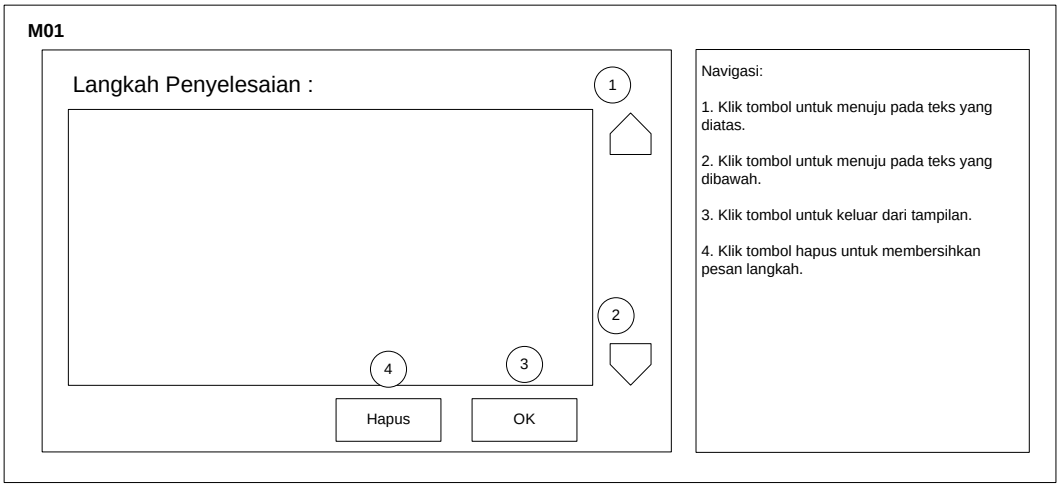
Rancangan tampilan bantuan digunakan untuk menampilkan informasi mengenai navigasi atau fitur – fitur yang ada pada aplikasi dan beberapa penjelasan tentang program, misalnya pada rancangan aplikasi terdapat tombol-tombol yang mempunyai fungsi tertentu yang berguna untuk menjalankan aplikasi. Rancangan antarmuka bantuan dapat dilihat pada gambar 3.46 dibawah ini.



Gambar 3.46. Rancangan Tampilan Bantuan

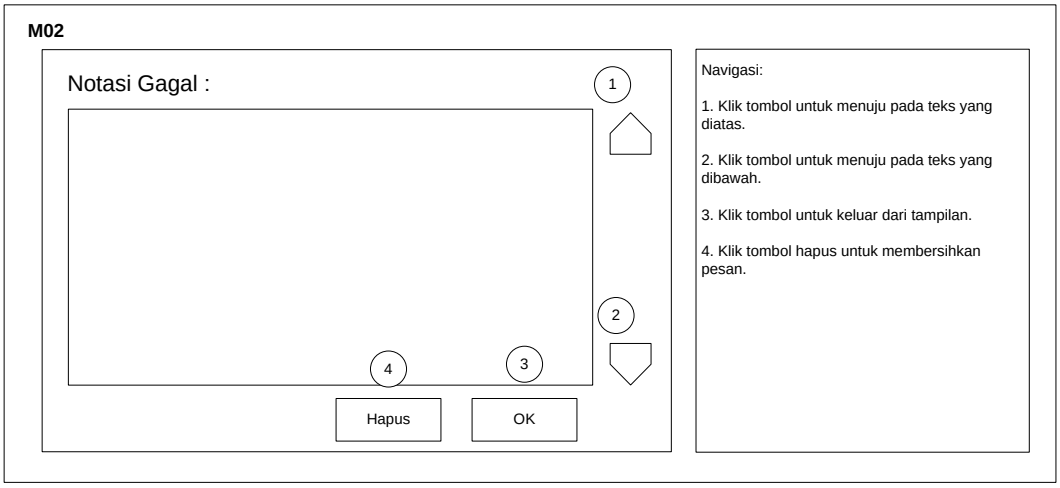
e. Rancangan Tampilan Pesan

Pada rancangan tampilan pesan adalah perancangan yang muncul sebagai informasi kepada pengguna dalam mengoperasikan sistem. Rancangan pesan langkah penyelesaian digunakan pemain untuk menampilkan informasi atau sejarah notasi gerakan yang sudah dijalankan oleh rubik pada aplikasi, tampilan pesan tersebut dapat dilihat pada gambar 3.47 sebagai berikut :



Gambar 3.47. Rancangan Tampilan Pesan Langkah Penyelesaian

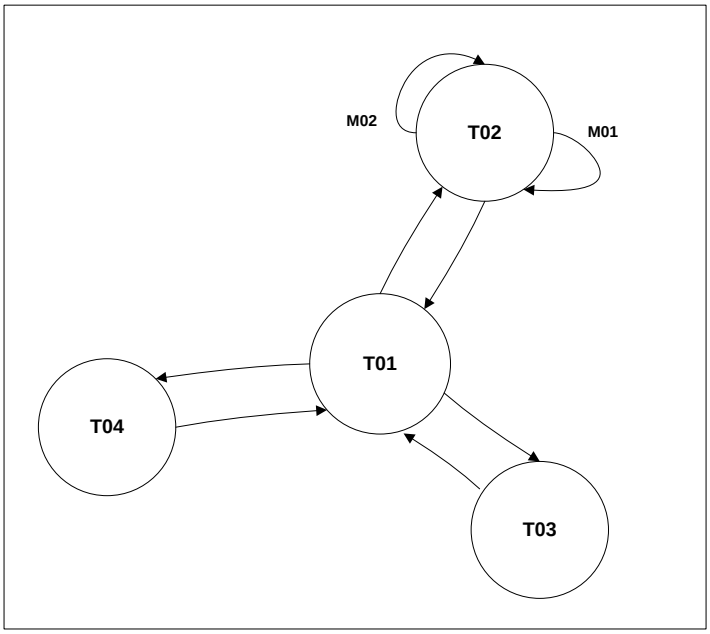
Rancangan pesan pada gambar 3.48 digunakan pemain untuk menampilkan informasi *error* atau gagal dalam menginputkan kode notasi pada *textbox*.



Gambar 3.48. Rancangan Tampilan Pesan Langkah Penyelesaian

f. Jaringan Semantik

Perancangan jaringan semantik menggambarkan alur informasi dari setiap antarmuka aplikasi yang dijelaskan pada gambar 3.49.



Gambar 3.49. Jaringan semantik

Keterangan :

- T01 : Tampilan utama
- T02 : Tampilan Menu Main
- T03 : Tampilan Menu Profil
- T04 : Tampilan Menu Bantuan
- M01 : *Message* yang menampilkan langkah penyelesaian rubik
- M02 : *Message* yang menampilkan gagal dalam menginputkan kode