BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Pada bagian ini akan dijelaskan mengenai analisis sistem terhadap penerapan algoritma *Fast Fourier Transform* untuk pengolahan sinyal suara dan *Euclidean Distance* untuk algoritma pencocokkan dengan kamus data pada *Speech Recognition* terhadap *game* catur.

Untuk dapat melakukan analisis sistem maka dilakukan beberapa langkah yaitu sebagai berikut:

1. Analisis Masalah
2. Analisis Metode
3. Analisis Kebutuhan Non-Fungsional
4. Analisis Kebutuhan Fungsional

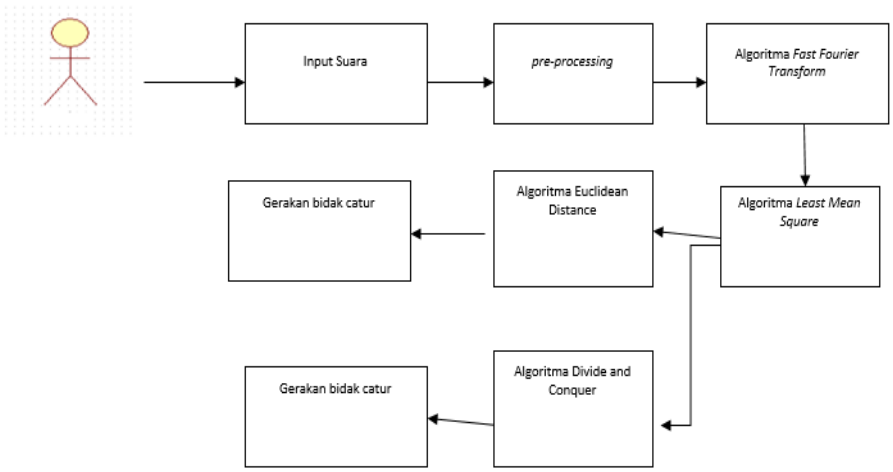
3.1.1 Analisis Masalah

Permainan catur merupakan suatu game strategi yang dapat dimainkan oleh 2 orang. Permainan menggunakan papan berwarna hitam dan putih dengan bidak sebagai alat permainannya. Terdapat 6 jenis bidak yang berbeda dengan langkah yang berbeda (Pion, Benteng, Kuda, Mentri, Ratu dan Raja) dan 64 kotak yang terdiri dari 8 kotak huruf (A, B, C, D, E, F, G, H) dan 8 kotak angka (1, 2, 3, 4, 5, 6, 7, 8) yang semua kotaknya dapat dilalui oleh semua pion. Proses utama dalam *Speech Recognition* pada *game* catur ini adalah proses pencocokkan dengan kamus data. Proses tersebut bertujuan untuk memperoleh suara yang sesuai atau cocok agar menghasilkan hasil akhir berupa gerakan bidak catur dan merupakan suatu proses yang sangat penting karena merupakan proses dimana pengenalan pola suara terjadi. Kendala yang dihadapi pada proses pengenalan pola suara adalah menggunakan algoritma yang tepat untuk mendapatkan hasil yang akurat. Pada penelitian ini, digunakan 2 algoritma yang sama – sama dapat menghitung jarak dari suatu pola dan sukses diterapkan pada penelitian sebelumnya. Pola tersebut

dapat diterapkan pada array 2 dimensi menyesuaikan dengan kamus data game catur ini. Tujuan penggunaan 2 metode tersebut adalah untuk melakukan perbandingan algoritma manakah yang lebih akurat dan lebih cocok diterapkan pada game catur ini.

3.2 Deskripsi Umum Sistem

Deskripsi umum dalam pengolahan suara ini digambarkan sebagai berikut:



Gambar 3.1 Proses Pengenalan Suara Secara Umum

Deskripsi secara umum dalam pengenalan suara ini dimulai dari tahap masukan suara yang dilakukan oleh pengguna. Setelah suara dimasukan atau direkam, sinyal suara akan disimpan ke dalam template penyimpanan sementara pada sistem dan akan masuk ke tahap pengolahan sinyal suara, yaitu proses pengubahan sinyal analog menjadi sinyal digital *(Analog to Digital Convertion)*. Setelah diubah menjadi sinyal digital, sinyal suara tersebut diolah menggunakan algoritma *Fast Fourier Transform (FFT)* untuk diubah menjadi sinyal suara dalam domain frekuensi. Tahap selanjutnya adalah melakukan *filtering* / pengurangan *noise* menggunakan algoritma *Least Mean Square (LMS)*. Tahap selanjutnya adalah proses pencocokkan dengan kamus data, proses tersebut menggunakan algoritma *Divide and Conquer* dan *Euclidean Distance* yang dapat mencari jarak terdekat antara 2 bilangan. Hasil akhir dari proses ini adalah gerakan bidak catur yang sesuai

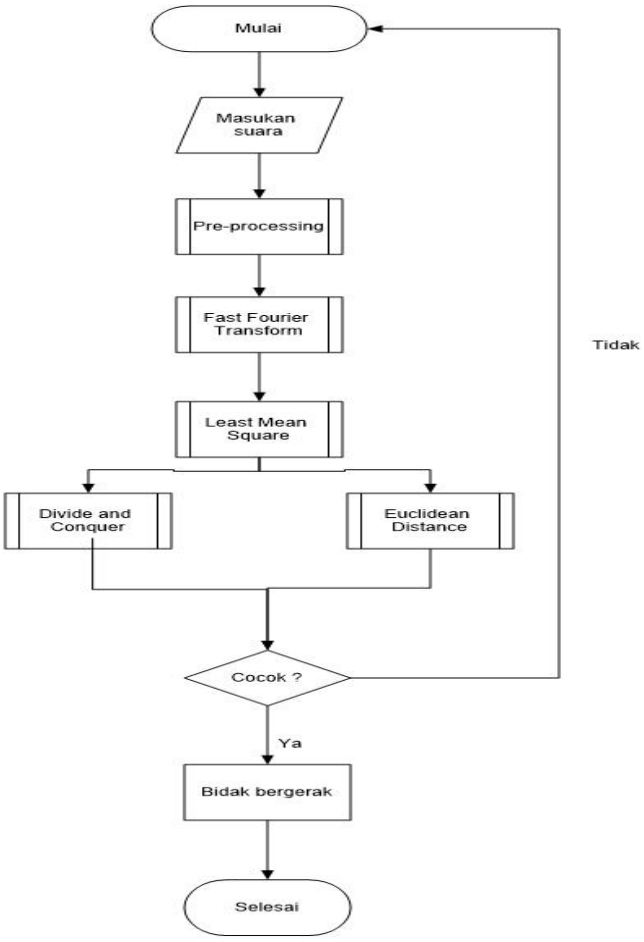
dengan hasil pencocokkan, antara suara yang masuk dengan suara yang ada pada kamus data.

3.3 Analisis metode

Metode pengenalan suara agar komputer dapat mengenali sinyal masukan yang digunakan pada *game* catur ini dirancang dengan menggunakan algoritma *Fast Fourier Transform*, menggunakan filtering *noise Least Mean Square*, dan algoritma pencocokkan *Divide and Conquer* dan *Euclidean Distance*.

3.3.1 Analisis pengolahan sinyal suara

Dalam analisis pengolahan sinyal suara ini dibagi menjadi beberapa tahap, tahap pertama yaitu *pre-processing* atau mengubah sinyal analog menjadi sinyal digital dengan menggunakan *analog to digital converter*, tahap selanjutnya adalah proses *Fast Fourier Transform*, lalu hasil dari proses FFT akan di filter untuk dikurangi *noise* nya. Kemudian akan dicocokan menggunakan algoritma *Divide and Conquer* dan *Euclidean Distance*. Alur diagram proses pengolahan suara ini digambarkan sebagai berikut:



Gambar 3.2 Alur pengolahan suara

3.4 Analisis basis data

Basis data adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh suatu informasi yang terkandung di dalamnya. Basis data pada aplikasi ini disebut dengan kamus data. Pada game catur ini, kamus data menyimpan informasi berupa gerakan bidak catur. Berbentuk *array* 2 dimensi dengan 3 bentukan yang berbeda, mempunyai 10 pembicara yang mengisi data – data seperti pada gambar 3.3:

Pion1	Pion2	Pion3	Pion4	Pion5	Pion6	Pion7	Pion8	Benteng1	Benteng2	Kuda1	Kuda2	Mentri1	Mentri2	Ratu	Raja

A	B	C	D	E	F	G	H

1	2	3	4	5	6	7	8

Gambar 3.3 Data pada kamus data

Gambar tabel diatas, mewakili gerakan untuk jenis pion, kotak angka dan huruf. Setiap kotak gerakan bidak dan kotak, berisikan nilai sinyal suara yang telah diproses menggunakan proses *Speech Recognition*. Nilai sinyal suara ini yang akan dicocokan dengan hasil sinyal suara masukan dan menghasilkan gerakan bidak catur pada aplikasi.

3.5 Analisis Input suara

Input suara merupakan suatu gerakan atau aksi yang dilakukan oleh pemain catur dalam memainkan permainan tersebut. Ada 3 format suara yang wajib dilakukan oleh pemain untuk menggerakan bidak catur, yaitu:

1. Suara format bidak (Pion1, Benteng2, Raja dan lain – lain).
2. Suara format kotak huruf (A, B, C dan lain – lain).
3. Suara format kotak angka (1, 2, 3 dan lain – lain).

Format suara tersebut akan dilakukan sekaligus dengan jeda waktu 1 detik per format dan menunggu komputer dapat mengenali suara. Durasi rekaman pemain memasukan suara adalah 3 detik. Pemain memasukan suara menggunakan *microphone* yang digunakannya. *Microphone* berfungsi untuk menyalurkan sinyal suara yang keluar dari suara pemain agar dapat direkam oleh komputer. Setelah sinyal suara tersimpan pada suatu komputer, sinyal suara tersebut masih bersifat analog, agar dapat dikenali oleh komputer, sinyal tersebut harus diubah ke dalam bentuk digital.

3.6 *Pre Processing*

Sinyal suara yang akan diproses dalam pengolahan sinyal suara ini bersifat analog sehingga agar dapat di kenali oleh komputer, sinyal analog tersebut harus

dikonversi menjadi sinyal digital yang berupa urutan angka yang dinamakan *analog to digital converter*.



Gambar 3.4 Skema *analog to digital converter*

Keterangan konsep kerja ADC pada aplikasi ini, sebagai berikut:

1. *Sampling* adalah konversi sinyal kontinu dalam domain waktu menjadi sinyal diskrit, melalui proses *sampling* sinyal pada selang waktu tertentu. Sehingga jika $x0(t)$ adalah sinyal input, maka outputnya adalah $x0(t)$, dengan T adalah interval *sampling*.
2. *Quantization* adalah proses untuk membulatkan nilai data kedalam bilangan bilangan tertentu yang telah ditentukan terlebih dahulu.
3. *Coding* pada proses ini, tiap nilai diskrit yang telah didapat, di representasikan dengan angka binary n-bit.

Proses ini dilakukan oleh aplikasi *Audacity* untuk mendapatkan sampel data sinyal suara.

3.6.1 Akuisisi data

Tahap ini adalah proses pengambilan data berupa sinyal suara yang diperoleh dengan cara merekam suara melalui mikrofon yang dihubungkan dengan komputer. Format perekaman suara yang digunakan adalah *.wav* dengan frekuensi sampling 44100Hz, 16 *bits/sample* dan *channel mono* yang sudah ditentukan terlebih dahulu yang bernilai 1 dan durasi rekaman yang diambil adalah 3 detik. Menggunakan rumus perhitungan (2.1), seperti berikut:

$$x = 44100 * 3 * \left(\frac{16}{8}\right) * 1 = 264600\ byte$$

Data sampling yang telah dihitung akan digunakan untuk menghitung *sample rate*. Perhitungan *sample rate* dengan Ts (durasi rekaman) adalah 3 detik. Menggunakan rumus (2.2), seperti berikut:

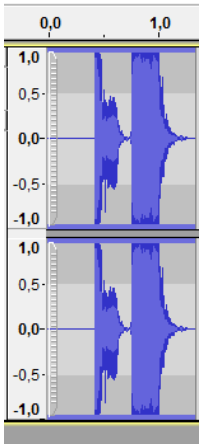
$$\frac{x}{Ts} = \frac{264600}{3} = 88200\ Hz$$

Untuk mendapatkan *sample point*, *sample rate* yang digunakan 88200 dengan waktu pengambilan data setiap 1,7 milidetik. Perhitungan *sample* point menggunakan rumus (2.3), seperti berikut:

$$88200 * 0.00017 = 15\ sample\ points$$

Pada penelitian ini, diambil data sebanyak 15 pada setiap masukan suaranya. Data ini diambil dengan menggunakan bantuan aplikasi *Audacity*. Cara mengambil datanya adalah sebagai berikut:

1. Buka file *.wav* yang telah direkam sebelumnya dan berisikan data suara "Pion4". Seperti gambar dibawah ini 3.5:



Gambar 3.5 Potongan sinyal suara pion4

2. Pilih menu *Analyze*, kemudian pilih *Sample Data Export*.
3. Pilih berapa jumlah *sample point* yang ingin didapatkan, dan skala pengukuran yang dipilih adalah *linear*.

Setelah sampel data di *export*, hasilnya adalah *file .txt* yang dapat di lihat datanya pada tabel 3.1:

Tabel 3.1 Sampel data suara

Data Suara Pion4	Sinyal Masukan (D) dalam Hertz
1	0,07354
2	0,09422
3	0,08081
4	0,07847
5	0,07738
6	0,08892
7	0,06929
8	0,06893
9	0,08556
10	0,09768
11	0,08802
12	0,08593
13	0,07265
14	0,08274
15	0,08186

3.6.2 *Windowing*

Proses selanjutnya adalah proses windowing atau *window hamming* untuk meminimalkan diskontinyuitas pada bagian awal dan akhir sinyal dengan menggunakan rumus (2.4), seperti berikut:

$$W_0 = 0{,}54 - 0{,}46\cos(\frac{2*3.14*0}{15-1}) = 0{,}08$$

.....

$$W_{14} = 0{,}54 - 0{,}46\cos(\frac{2*3.14*14}{15-1}) = 0{,}08002$$

Nilai pada Windowing dapat dilihat pada tabel 3.2:

Tabel 3.2 Windowing

Wi	Windowing (W)
0	0,08000
1	0,12551
2	0,25303
3	0,43733
4	0,64195
5	0,82639
6	0,95417
7	0,99999
8	0,95480
9	0,82754
10	0,64338
11	0,43876
12	0,25417
13	0,12614
14	0,08002

Berikut adalah contoh algoritma *windowing hamming*:

```
Procedure windowing
i.s : Sinyal hasil normalisasi telah diketahui
f.s : Sinyal hasil windowing diketahui
Kamus
X : double
N,i :int
Algoritma
For i= 0 to N-1 do
Begin
        W[i] = 0,54 – 0,46 cos ( 2*Pi*i)/(N-1);
        X[i] = D[i] * W[i];
End
Endfor
```

Gambar 3.6 Potongan algoritma *windowing hamming*

Setelah mendapatkan data *windowing*, maka data tersebut akan di kalikan dengan data sampel suara awal, untuk mendapatkan sinyal yang mengurangi diskontinyuitas, maka dimasukan dengan rumus (2.5), seperti berikut:

$$X_0 = 0{,}07354 * 0{,}08 = 0{,}00588$$

.....

$$X_{14} = 0{,}08186 * 0{,}08002 = 0{,}00655$$

Dengan cara yang sama dilakukan pada titik yang lain, dan di peroleh nilai seperti tabel 3.3 dibawah ini:

Tabel 3.3 Hasil Windowing

Xi	Di * Wi
0	0,00588
1	0,01182
2	0,02044
3	0,03431
4	0,04967
5	0,07348

6	0,06611
7	0,06892
8	0,08169
9	0,08083
10	0,05663
11	0,03771
12	0,01846
13	0,01043
14	0,00655

3.6.3 Algoritma *Fast Fourier Transform*

Fast Fourier Transform (FFT) adalah suatu algoritma yang mengimplemetasikan *Discreet Fourier Transform* (DFT) dengan teknik perhitungan yang cepat serta memanfaatkan sifat periodikal dari transformasi fourier. *Fast Fourier Transform* merupakan operasi matematika yang bertujuan untuk dekomposisi dari suatu sinyal domain waktu ke sinyal domain frekuensi. DFT dilakukan dengan memasukan sebuah transformasi, dengan panjang vektor N.

Berikut ini adalah perhitungan algoritma FFT dengan menggunakan rumus (2.6), seperti berikut:

Tabel 3.4 Tabel perhitungan algoritma FFT

N	$\boldsymbol{F(x)}$	**Perhitungan DFT**	**Hasil DFT**
1	$F(0)$	$\frac{1}{15}$ [0.00588 (cos $\left(2 * 3.14 * 0 * \frac{0}{15}\right) - j \sin(2 * 3.14 * 0 * \frac{0}{15})$) + 0.01182 (cos $\left(2 * 3.14 * 0 * \frac{1}{15}\right) - j \sin(2 * 3.14 * 0 * \frac{1}{15})$) ..... + 0.01043 $\left(\cos\left(2 * 3.14 * 0 * \frac{13}{15}\right) - j \sin(2 * 3.14 * 0 * \frac{13}{15})\right)$ + 0.00655 (cos $\left(2 * 3.14 * 0 * \frac{14}{15}\right) - j \sin(2 * 3.14 * 0 * \frac{14}{15})$)] $= \frac{1}{15}$ [0.00588 (cos(0) − j sin(0)) + 0.01182 (cos(0) − j sin(0))	0.04153

		…….. $+\big(0.01043)\ (\cos(0) - j\ \sin(0))$ $+\ (0.00655)(\cos(0) - j\ \sin(0))\big)\ ]$ $= \frac{1}{15}\ (0.00588 + 0.01182 \ldots . + (-0.01043)$ $+\ (-0.00655))$ $= \frac{1}{15}\ (0.62298)$ $= 0.04153$	
2	$F(1)$	$\frac{1}{15}\ [0.00588\ (\cos\left(2 * 3.14 * 1 * \frac{0}{15}\right) - j\ \sin(2 * 3.14 * 1 * \frac{0}{15}))$ $+\ 0.01182\ (\cos\left(2 * 3.14 * 1 * \frac{1}{15}\right) - j\ \sin(2 * 3.14 * 1 * \frac{1}{15}))$ …….. $+\ (0.01043)\ \left(\cos\left(2 * 3.14 * 1 * \frac{13}{15}\right) - j\ \sin(2 * 3.14 * 1 * \frac{13}{15})\right)$ $+\ (0.00655)\ (\cos\left(2 * 3.14 * 1 * \frac{14}{15}\right) - j\ \sin(2 * 3.14 * 1 * \frac{14}{15}))\ ]$ $= \frac{1}{15}\ [0.00588\ (\cos(0) - j\ \sin(0))$ $+\ 0.01182\ (\cos(0.419) - j\ \sin(0.419))$ …….. $+\big(0.01043\ (\cos(5.443) - j\ \sin(5.443))$ $+\ (0.00655)(\cos(5.861) - j\ \sin(5.861))\big)\ ]$ $= \frac{1}{15}\ [0.00588\ (1) +\ 0.01182\ (0.91363 - 0.40654\ j))$ …….. $+\Big(0.01043\ (0.66707 - (-0.74498\ j)\ )$ $+\ (0.00655)(0.91233$ $-\ (-0.40945\ j))\Big)\ ]$ $= \frac{1}{15}\ (0.00588 + (0.01081$ $-\ 0.00481j) \ldots . + (0.00696$ $-\ (-0.00777j)$ $+\ (0.00597 - (-0.00268j))$ $= \frac{1}{15}\ (-0.28093 - 0.04033j)$ $= -0.01872 - 0.00268j$	-0.01872 $- 0.00268j$

Berikut ini adalah contoh potongan proses algoritma FFT.

```
Procedure FFT
{menghitung sinyal hasil FFT}
i.s : banyaknya data dan sinyal hasil windowing diketahui
f.s : nilai hasil FFT
Kamus
I,n :integer
A,b :double
Algoritma
For i = n div 2 do
Begin
  Z[n] = 0
  For n = 0 to N do
  Begin
       a←X[i] * cos (2*Pi*n*i)/ N
       b← X[i] * sin (2*Pi*n*i)/ N
  End
  Endfor
Z[n] ←sqrt (a^2 +b^2)/N
End
For i = n div 2 +1 to n do
Begin
  X←i-((0+n)div 2)
  Y←((0+n)div 2)-x
  Z[n]←Z[y]
End

Endfor
```

Gambar 3.7 Gambar potongan algoritma FFT

Setelah mendapatkan hasil dari DFT, maka nilai tersebut, akan dimasukan ke rumus (2.7), Tujuan dari penggunaan rumus ini adalah untuk menghilangkan nilai J pada hasil nilai DFT dan untuk memperlihatkan hasil sebenarnya dari nilai FFT.

Maka perhitungannya adalah:

$$|F(0)| = [0.04153^2 + 0^2]^{1/2} = 0.04153$$

$$|F(1)| = [0.01872^2 + 0.00268^2]^{1/2} = 0.01892$$

Maka $F(0) = 0.04153$, Cara yang sama akan dilakukan pada data sinyal lainnya sampai $F(7)$.

Perhitungan FFT mengimplementasikan pencerminan transformasi ganda hasil DFT dengan hanya menghitung nilai setengahnya data sinyal sehingga

perhitungan akan lebih cepat, lalu nilai setengahnya lagi dihitung dengan cara *conjugate* nilai yang telah dihitung dengan DFT. Untuk membagi data sinyal adalah dengan rumus (2.8), seperti berikut:

$$b = \frac{(15+1)}{2} = 8$$

Jadi perhitungan DFT hanya sampai data ke-8 / F(7), dan untuk hasil selanjutnya menggunakan fungsi *conjugate* dari nilai hasil DFT.

Tabel 3.5 Hasil perhitungan FFT

$\boldsymbol{F(x)}$	**Hasil FFT**
$F(0)$	0.04153
$F(1)$	0.01892
$F(2)$	0.01121
$F(3)$	0.00902
$F(4)$	0.00871
$F(5)$	0.00693
$F(6)$	0.00601
$F(7)$	0.00494
$F(8)$	0.00601
$F(9)$	0.00693
$F(10)$	0.00871
$F(11)$	0.00902
$F(12)$	0.01121
$F(13)$	0.01892
$F(14)$	0.04153

Pada tabel 3.5, dapat dilihat bahwa nilai $F(14)$ merupakan nilai *conjugate* (kebalikan) dari $F(0)$, nilai $F(13)$ dengan nilai $F(1)$ dan seterusnya.

3.6.4 Algortima *Least Mean Square*

Tahapan ini merupakan tahapan lanjutan penghilang sinyal – sinyal yang tidak dibutuhkan atau disebut juga dengan *noise*. Menggunakan rumus perhitungan (2.9), (2.10), dan (2.11) dengan langkah – langkah sebagai berikut:

$$d'(0) = 0 * 1 = 0$$

$$e(0) = 0.04153 - 0 = 0.04153$$

$$W_1 = 0 + 0.02 * 0.04153 * 1 = 0.000831$$

...

$$d'(1) = 0.000831 * 2 = 0{,}00166$$

$$e(1) = 0.01892 - 0.00166 = 0.01725$$

$$W_2 = 0.000831 + 0.02 * 0.01725 * 2 = 0.00152$$

Perhitungan akan dilakukan terus sejumlah data pada nilai FFT yaitu sampai mendapatkan W_{15}.

Tabel 3.6 Hasil *adaptif filter Least Mean Square*

W_n	Hasil LMS
W_1	0.00083
W_2	0.00152
W_3	0.00191
W_4	0.00202
W_5	0.00188
W_6	0.00135
W_7	0.00086
W_8	0.00054
W_9	0.00074
W_{10}	0.00064
W_{11}	0.00100
W_{12}	0.00028
W_{13}	0.00224
W_{14}	0.00126
W_{15}	0.01689

3.6.5 Algoritma *Divide and Conquer*

Pada tahapan ini, nilai hasil dari adaptif filter LMS akan melalui proses *binary search* implementasi algoritma *divide and conquer*, ada beberapa tahapan yaitu

membagi sampel yang telah diambil sebelumnya dan pencocokkan kata, dan saat pencocokkan dilakukan algoritma *binary search divide and conquer*.

Berikut adalah metode *divide and conquer* yang akan dilakukan:

1. *Divide*: membagi masalah (hasil proses LMS) menjadi beberapa bagian proses yang lebih kecil.
2. *Conquer*: memecahkan (pencarian biner) masing – masing sub-proses atau bagian menggunakan fungsi *binary search* .
3. *Combine*: menggabungkan solusi masing – masing proses sehingga memperoleh hasil dari solusi tersebut.

Misal didapatkan beberapa data hasil dari adaptif filter LMS:

0.00081	0.00152	0.00191	0.00202	0.00188	0.00135	0.00086	0.00054	0.00074	0.00064	0.00100	0.00028	0.00224	0.00126	0.01689

Sebelum dilakukan pencarian, data diurutkan dahulu agar pencarian lebih cepat. Diurutkan dari terkecil hingga terbesar hasil pengurutan data adalah sebagai berikut:

0.00028	0.00054	0.00064	0.00074	0.00081	0.00086	0.00100	0.00126	0.00135	0.00152	0.00188	0.00191	0.00202	0.00224	0.01089

Mula - mula diambil posisi awal p = 1 dan posisi akhir q = N, lalu cari posisi data tengah dengan rumus (2.8): $dt = \frac{(1+15)}{2} = 8$

Maka nilai tengah berada pada posisi ke 8. Masukan satu data suara baru yang merupakan nilai dari template database dan merupakan hasil dari nilai adaptif filter LMS dan dicari kecocokannya yaitu pion1, nilai yang akan dicocokan adalah 0.00144.

0.00028	0.00054	0.00064	0.00074	0.00081	0.00086	0.00100	0.00126	0.00135	0.00152	0.00188	0.00191	0.00202	0.00224	0.01089

P **dt** **q**

Karena $0.00126 < 0.00144$ maka *q* -> *dt* - 1 = 7, maka dilakukan perulangan untuk mencari nilai tengah lanjutan, yaitu: $dt = \frac{(1+7)}{2} = 4$

0.00028	0.00054	0.00064	0.00074	0.00081	0.00086	0.00100	0.00126	0.00135	0.00152	0.00188	0.00191	0.00202	0.00224	0.01089

P **dt** **q**

Karena $0.00074 < 0.00144$ maka *q* -> *dt* - 1 = 3, maka dilakukan perulangan untuk mencari nilai tengah lanjutan, yaitu: $dt = \frac{(1+3)}{2} = 2$

0.00028	0.00054	0.00064	0.00074	0.00081	0.00086	0.00100	0.00126	0.00135	0.00152	0.00188	0.00191	0.00202	0.00224	0.01089

P **dt** **q**

Karena $0.00054 < 0.00144$ maka *q* -> *dt* - 1 = 1, maka dilakukan perulangan untuk mencari nilai tengah lanjutan, yaitu: $dt = \frac{(1+1)}{2} = 1$

0.00028	0.00054	0.00064	0.00074	0.00081	0.00086	0.00100	0.00126	0.00135	0.00152	0.00188	0.00191	0.00202	0.00224	0.01089

P dt q

Karena P, dt, dan q berada pada satu nilai, maka nilai tersebut merupakan hasil dari pencocokkan tersebut. Hasil pencocokkan tersebut akan melalui proses selanjutnya yaitu proses peramalan error terkecil menggunakan *Mean Square Error* (*MSE*).

Perhitungan *mean square error* untuk mendapatkan nilai *error* terkecil adalah dengan rumus (2.12), seperti dibawah ini:

$$MSE = 0{,}00028 - 0{,}00144 = -0{,}00116$$

Proses dilakukan kembali sesuai dengan banyaknya data pada kamus data dengan hasil sebagai berikut:

Tabel 3.7 Hasil *Divide and Conquer*

Pion acuan	**Hasil *Divide and Conquer***
Pion1	$-0{,}00116$
Pion2	$-0{,}00241$
Pion3	$-0{,}00379$
Pion4	$-0{,}00076$
Pion5	$-0{,}00320$
Pion6	$-0{,}00282$
Pion7	$-0{,}00098$
Pion8	$-0{,}00121$
Benteng1	$0{,}00216$
Benteng2	$0{,}00261$
Kuda1	$0{,}00126$
Kuda2	$0{,}00021$
Mentri1	$0{,}00302$
Mentri2	$0{,}00106$
Ratu	$0{,}00241$
Raja	$0{,}00379$

Pada tabel 3.7, hasil error terkecil adalah $-0{,}00076$ yaitu pion4. Hasil tersebut akan menjadi pencocokkan dengan database menggunakan *Divide and Conquer*.

3.6.6 Proses *Euclidean Distance*

Pada tahap ini, nilai pada perhitungan adaptif filter LMS, akan digunakan sebagai nilai masukan menggunakan perhitungan jarak *euclidean distance*. Nilai pada *template database* akan menjadi nilai acuan pada perhitungan ini, menggunakan rumus (2.13).

I (pion4) = (0.00081, 0.00152,...., 0.00126, 0.01689).

J (pion1) = (0,00144, 0.00131,...., 0.00098, 0.01592).

$$d_{ij} = \sqrt{\begin{matrix}(0.00081 - 0.00144)^2 + (0.00152 - 0.00131)^2 + \cdots \\ + (0.00126 - 0.00098)^2 + (0.01689 - 0.01592)^2\end{matrix}}$$

$$d_{ij} = \sqrt{\begin{matrix}(-0.00063)^2 + (0.00021)^2 + \cdots \\ + (0.00028)^2 + (0.00097)^2\end{matrix}}$$

$$d_{ij} = 0.00231$$

Pada perhitungan diatas, data template acuan dimulai dari pion1 dan melakukan perulangan sampai banyak nya data pada database dengan hasil sebagai berikut:

Tabel 3.8 Perhitungan *Euclidean Distance*

Pion acuan	**Hasil *Euclidean Distance***
Pion1	0.00231
Pion2	0.00119
Pion3	0.00121
Pion4	0.00071
Pion5	0.00171
Pion6	0.00193
Pion7	0.00201
Pion8	0.00194
Benteng1	0.00101
Benteng2	0.00153
Kuda1	0.00121
Kuda2	0.00202
Mentri1	0.00121
Mentri2	0.00192
Ratu	0.00183
Raja	0.00112

Pada tabel 3.8, pion4 memiliki nilai selisih terkecil dibandingkan dengan nilai pion – pion yang lainnya dan merupakan hasil pencocokkan dari *euclidean distance* yang akan digunakan.

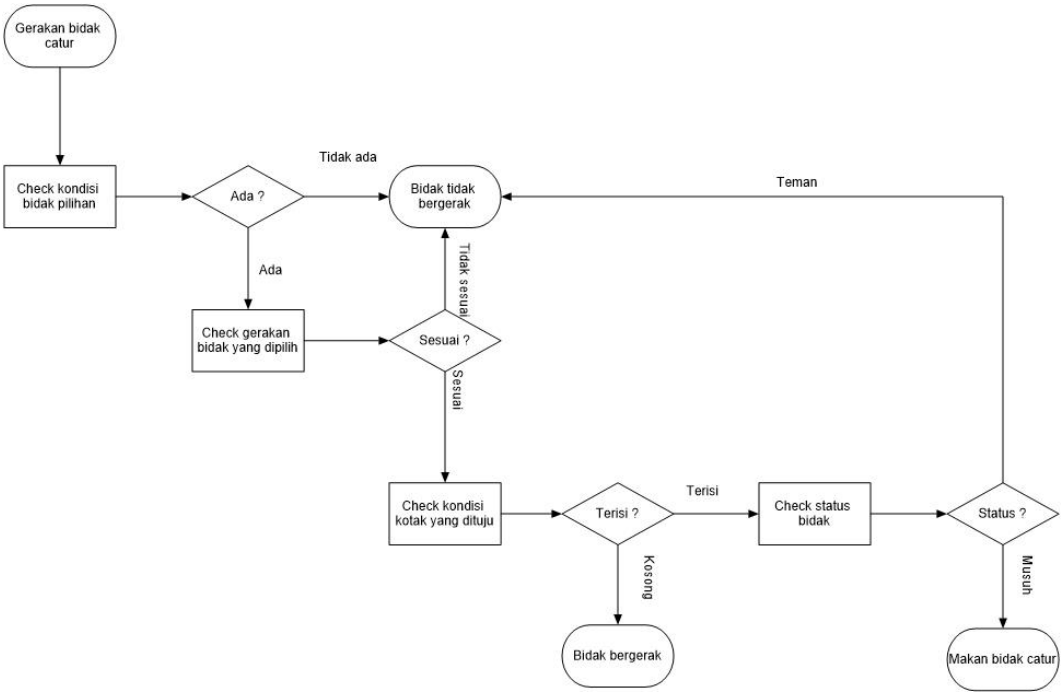
3.6.7 Pencocokkan dengan kamus data

Nilai yang didapat pada proses algoritma pencocokkan, akan menjadi hasil akhir dari proses pencocokkan dengan kamus data. Pada proses *Divide and Conquer* hasil pencocokkan *error* terkecil terdapat pada bidak pion4, dan pada proses *Euclidean Distance* hasil pencocokkan dengan nilai *error* terkecil pun terdapat pada bidak pion4.

Proses selanjutnya adalah mencari nilai pencocokkan pada format huruf dan angka, sesuai dengan suara yang dimasukan oleh pemain. Setelah ke 3 format tersebut mengeluarkan hasil pencocokkan, maka bidak catur akan bergerak sesuai dengan hasil pencocokkan tersebut.

3.7 Analisis gerakan bidak catur

Setelah proses pencocokkan berakhir dan menghasilkan pencocokkan yang sesuai dengan basis data, maka bidak catur akan bergerak dengan alur atau kondisi seperti gambar 3.8:



Gambar 3.8 Alur gerakan bidak catur

3.8 Analisis gerakan NPC

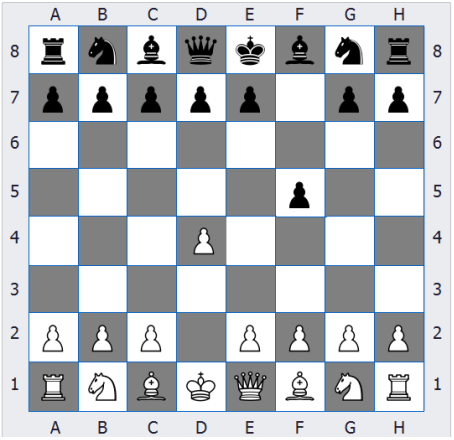
NPC merupakan kepanjangan dari *Non Player Character* yang artinya kemampuan suatu komputer melakukan tindakan balasan dari suatu aksi sistem. Kemampuan ini merupakan suatu *Aritificial Intelligent* atau disebut juga kecerdasan buatan. Pada game catur ini, NPC diterapkan pada bidak musuh, yaitu bidak hitam. Gerakan bidak catur NPC selalu menyesuaikan dengan gerakan bidak putih.

3.8.1 Gerakan NPC terhadap gerakan player putih

Pada game catur ini, gerakan yang telah dilakukan oleh pemain bidak putih adalah gerakan PION4 - D – 4 yang artinya pion dengan nomor urut "4" bergerak ke arah kotak huruf "D" dan angka "4". Setelah pion berpindah dari posisi awal ke kotak tujuan, musuh yang menggunakan AI, akan melakukan tindakan balasan. Tindakan balasan yang dapat dilakukan oleh AI, sebagai berikut:

1. PION1 – A – 5
2. PION2 – B – 5
3. PION6 – F – 5
4. PION7 – G – 5
5. PION8 – H – 5

Menurut langkah AI diatas, AI akan memikirkan langkah yang terbaik yang dapat dilakukan sekarang dan untuk mengantisipasi langkah musuh kedepan, maka gerakan pion yang akan dilakukan adalah PION6 – F – 5 yang artinya pion nomor urut "6" bergerak ke arah kotak huruf "F" dan angka "5".



Gambar 3.9 Gerakan NPC bidak hitam

Gerakan tersebut merupakan gerakan balasan yang dilakukan oleh musuh bidak hitam untuk membalas gerakan yang dilakukan oleh pemain bidak putih.

3.9 Analisis Kebutuhan Non Fungsional

Analisis dan kebutuhan *non-fungsional* meliputi analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, serta analisis pengguna.

3.9.1 Analisis Kebutuhan Perangkat Keras

Analisis perangkat disini menjelaskan tentang kebutuhan perangkat keras yang dapat digunakan untuk menjalankan program *game catur* ini adalah dengan spesifikasi minimal sebagai berikut:

Tabel 3.9 Spesifikasi perangkat keras

Perangkat Keras	
Nama Perangkat	**Spesifikasi**
Processor	Intel CORE i3
RAM	2GB
VGA	1GB
Harddisk	100GB
Headset Microphone	Semua Jenis

3.9.2 Analisis Kebutuhan Perangkat Lunak

Analisis perangkat disini menjelaskan tentang kebutuhan perangkat lunak yang diperlukan untuk membangun aplikasi *Game* Catur. Berikut adalah perangkat lunak yang digunakan untuk membangun aplikasi yang terdiri:

Tabel 3.10 Spesifikasi perangkat lunak

Perangkat Lunak	
Nama Perangkat	**Spesifikasi**
Sistem Operasi	*Microsoft Windows 8.1*
Aplikasi pembuat game	*Visual Studio*
Tools	*DevExpress Universal*

3.9.3 Karakteristik pengguna

Selain dibutuhkannya perangkat lunak dan perangkat keras, *user* atau penguna adalah bagian yang sangat penting untuk dapat menerapkan aplikasi ini. Analisis pengguna diperlukan untuk dapat mengetahui sasaran calon pengguna yang dapat menggunakan aplikasi. Pada penelitian ini pembangunan aplikasi ditujukan untuk pengujian implementasi algoritma sehingga pengguna disini disarankan memiliki karakteristik sebagai berikut:

Tabel 3.11 Spesifikasi pengguna

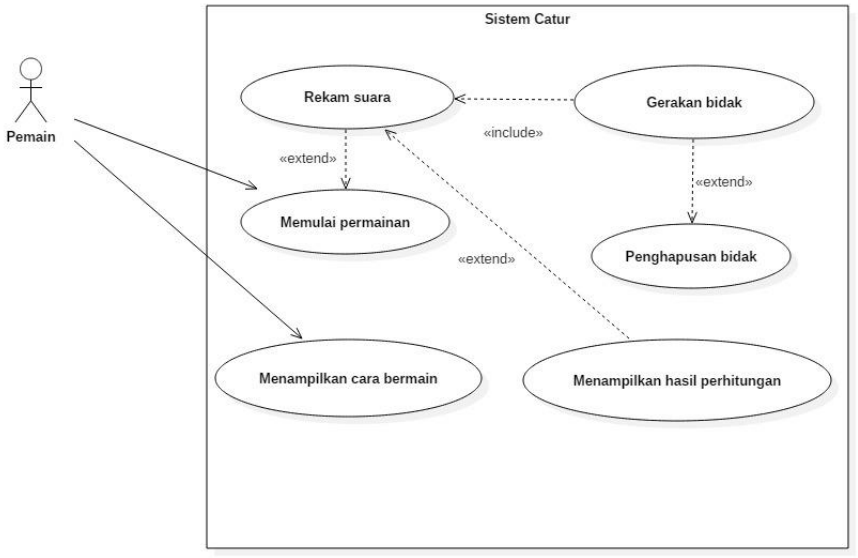
User	
Pengguna	Umum
Hak Akses	Memainkan *game* catur
Keterampilan	Mengerti dan dapat mengoperasikan komputer
Tingkat Pendidikan	Minimal Setara SMP

3.10 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik sesuai kebutuhan. Pada bagian ini akan dibahas tentang *use case diagram*, *activity diagram*, *class diagram*, dan *sequence diagram*.

3.10.1 *Use Case* Diagram

Use Case Diagram adalah diagram yang menggambarkan sebuah kegiatan yang dilakukan oleh *actor* terhadap sistem yang akan dirancang sehingga terjadi interaksi. Pada bagaian ini akan dijelaskan *Use Case* yang akan dirancang beserta definisinya.



Gambar 3.10 Diagram *Use case game* catur

3.10.1.1 Definisi Aktor

Definisi aktor berfungsi untuk menjelaskan aktor yang terdapat pada *use case* diagram. Definisi aktor dapat dilihat pada tabel berikut:

Tabel 3.12 Definisi Aktor

Aktor	Deskripsi
Pemain	Pemain merupakan orang yang memainkan aplikasi *game* dan berinteraksi langsung dengan antar muka aplikasi tersebut.

3.10.1.2 Skenario *Use Case*

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian didalam *use case* tersebut, dimana pengguna memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada pengguna setelah pengguna memberikan perintah pada setiap bagian-bagian *use case*.

1. Skenario *Use case* memulai permainan

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain dengan *use case* memulai permainan yang dijelaskan oleh tabel berikut:

Tabel 3.13 Skenario *Use Case memulai permainan*

Identifikasi		
Nama *Use case*	Memulai permainan	
Deskripsi	Fungsional ini digunakan oleh pemain untuk memulai permainan	
Aktor	Pemain	
Kondisi Awal	Pemain berada di halaman utama sistem	
Kondisi Sukses	Tampilan awal papan catur dapat ditampilkan	
Kondisi Gagal	Tampilan papan catur tidak dapat ditampilkan kepada pencatur	
Skenario	Tahap	Aksi
	1	Pemain memilih memulai permainan
	2	Sistem menampilkan tampilan papan catur
	3	Timer waktu mulai berjalan
Kondisi lain	Tahap	Aksi percabangan
	-	-

2. Skenario *Use case* rekam suara

Skenario *use case* ini menjelaskan spesifikasi *Use case* rekam suara yang dijelaskan oleh tabel berikut:

Tabel 3.14 Skenario *Use case* rekam suara

Identifikasi	
Nama *Use case*	Rekam suara
Deskripsi	Fungsionalitas ini digunakan oleh pemain untuk memasukan suara pada sistem agar sistem dapat merekam suara tersebut

Aktor	Pemain	
Kondisi Awal	Tampilan papan catur telah berhasil ditampilkan kepada pemain	
Kondisi Sukses	Suara berhasil diterima oleh sistem	
Kondisi Gagal	Suara gagal diterima oleh sistem	
Skenario	Tahap	Aksi
	1	Pemain memilih merekam suara
	2	Pemain memasukan suara
	3	Sistem berhasil menerima suara masukan pemain dan merekamnya
	4	Sistem mengolah sinyal suara awal masukan menggunakan proses *pre-processing*
	5	Sistem mengolah sinyal suara menggunakan algoritma *Fast Fourier Transform*
	6	Sistem mengolah suara menggunakan proses pengurangan noise dengan algoritma *Least mean square*
	7	Sistem melakukan pencocokkan menggunakan algoritma *Divide and Conquer* terhadap basis data

	8	Sistem melakukan pencocokkan menggunakan algoritma *Euclidean Distance* terhadap basis data
	9	Bidak bergerak sesuai dengan hasil pencocokkan
Kondisi lain	Tahap	Aksi percabangan
	3.1	Sistem tidak dapat menerima suara masukan pemain (gagal)

3. Skenario *Use Case* menampilkan gerakan bidak

Skenario *use case* ini menjelaskan spesifikasi *Use case* menampilkan gerakan bidak yang dijelaskan oleh tabel berikut:

Tabel 3.15 Skenario *Use case* menampilkan gerakan bidak

Identifikasi		
Nama *Use case*	Gerakan bidak	
Deskripsi	Bidak yang bergerak sesuai dengan masukan suara pemain	
Aktor	-	
Kondisi Awal	Sistem berhasil menerima masukan suara pemain dan pencocokkan dengan basis data berhasil dilakukan	
Kondisi Sukses	Bidak bergerak dan berubah posisi	
Kondisi Gagal	Bidak tidak bergerak	
Skenario	Tahap	Aksi
	1	Input suara dan pencocokkan berhasil dilakukan

	2	Sistem memeriksa status bidak yang dipilih
	3	Jika bidak yang dipilih ada, sistem memeriksa apakah gerakan bidak sesuai dengan jenisnya
	4	Jika gerakan jenis bidak sesuai dengan jenisnya, sistem memeriksa apakah kotak yang dituju kosong atau tidak
	5	Jika kotak yang dituju kosong, bidak berhasil berpindah posisi
Kondisi lain	Tahap	Aksi percabangan
	2.1	Sistem tidak dapat menggerakan bidak karena bidak yang dipilih tidak ada dan memunculkan pesan bidak catur tidak dapat bergerak
	3.1	Sistem tidak dapat menggerakan bidak karena gerakan bidak tidak sesuai dengan jenisnya dan memunculkan pesan bidak catur tidak dapat bergerak
	4.1	Sistem tidak dapat menggerakan bidak

		karena kotak yang dituju sudah ditempati bidak teman dan memunculkan pesan bidak catur tidak dapat bergerak

4. Skenario *Use Case* menampilkan penghapusan bidak

Skenario *use case* ini menjelaskan spesifikasi *Use case* menampilkan penghapusan bidak yang dijelaskan oleh tabel berikut:

Tabel 3.16 Skenario *Use case* menampilkan penghapusan bidak

Identifikasi		
Nama *Use case*	Penghapusan bidak	
Dekripsi	Bidak yang berpindah posisi akan memakan bidak musuh	
Aktor	-	
Kondisi Awal	Sistem telah memeriksa kondisi bidak catur yang terdapat pada kotak yang dituju merupakan bidak musuh	
Kondisi Sukses	Bidak berpindah posisi dengan cara memakan bidak musuh	
Kondisi Gagal	Bidak tidak dapat berpindah posisi dan memakan bidak musuh	
Skenario	Tahap	Aksi
	1	Input suara dan pencocokkan berhasil dilakukan
	2	Sistem memeriksa status bidak yang dipilih
	3	Jika bidak yang dipilih ada, sistem memeriksa

		apakah gerakan bidak sesuai dengan jenisnya
	4	Jika gerakan jenis bidak sesuai dengan jenisnya, sistem memeriksa apakah kotak yang dituju kosong atau tidak
	5	Jika kotak yang dituju ada bidak lain, check status bidak tersebut, apakah teman atau musuh
	6	Jika kotak yang dituju merupakan musuh, maka bidak akan berpindah dengan memakan bidak tersebut.
Kondisi lain	Tahap	Aksi percabangan
	2.1	Sistem tidak dapat menggerakan bidak karena bidak yang dipilih tidak ada dan memunculkan pesan bidak catur tidak dapat bergerak
	3.1	Sistem tidak dapat menggerakan bidak karena gerakan bidak tidak sesuai dengan jenisnya dan memunculkan pesan bidak catur tidak dapat bergerak

	4.1	Sistem tidak dapat menggerakan bidak karena kotak yang dituju sudah ditempati bidak teman dan memunculkan pesan bidak catur tidak dapat bergerak

5. Skenario *Use Case* menampilkan hasil perhitungan

Skenario *use case* ini menjelaskan spesifikasi *Use case* menampilkan hasil perhitungan yang dijelaskan oleh tabel berikut:

Tabel 3.17 Skenario *Use case* menampilkan hasil perhitungan

Identifikasi		
Nama *Use case*	Menampilkan hasil perhitungan	
Deskripsi	Menampilkan perhitungan tahap *pre-processing* ,algoritma *Fast Fourier Transform*, *K-medoid clustering* dan *Fuzzy C-mean clustering* ke dalam suatu tabel	
Aktor	-	
Kondisi Awal	Pemain berada pada tampilan utama sistem dan berhasil melakukan input suara	
Kondisi Sukses	Menampilkan perhitungan pada *template* yang telah disediakan	
Kondisi Gagal	Perhitungan tidak dapat ditampilkan	
Skenario	Tahap	Aksi
	1	Pemain memilih lihat perhitungan
	2	Sistem menghitung sinyal suara yang dimasukan oleh user menggunakan

		proses perhitungan yang telah dibuat sebelumnya
	3	Sistem menampilkan hasil perhitungan
Kondisi lain	Tahap	Aksi percabangan
	-	-

6. Skenario *Use Case* Menampilkan cara bermain

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain dengan *use case* cara bermain yang dijelaskan oleh tabel berikut:

Tabel 3.18 Skenario *Use case* menampilkan cara bermain

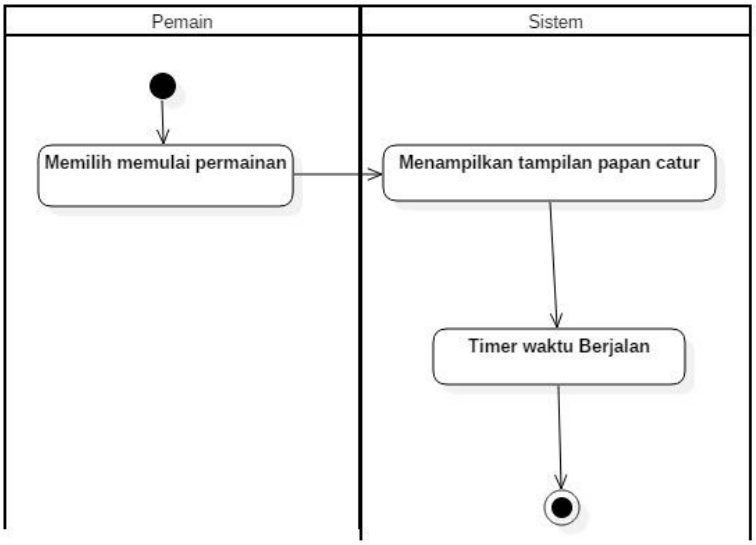
Identifikasi		
Nama *Use case*	Menampilkan cara bermain	
Deskripsi	Memberikan tampilan informasi, bagaimana cara bermain permaianan catur	
Aktor	Pemain berada di halaman utama sistem	
Kondisi Awal	Pemain berada di halaman utama sistem	
Kondisi Sukses	Berhasil menampilkan tampilan cara bermain kepada pemain	
Kondisi Gagal	Tidak dapat menampilan tampilan cara bermain	
Skenario	Tahap	Aksi
	1	Pemain memilih cara bermain
	2	Sistem berhasil menampilkan tampilan cara bermain
Kondisi lain	Tahap	Aksi percabangan
	-	-

3.10.2 *Activity Diagram*

Diagram aktivitas atau *Activity Diagram* menggambarkan *work flow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis. *Activity diagram* digunakan untuk memperjelas atau mempresentasikan sekenario dari *use case*.

1. *Activity diagram* Memulai permainan

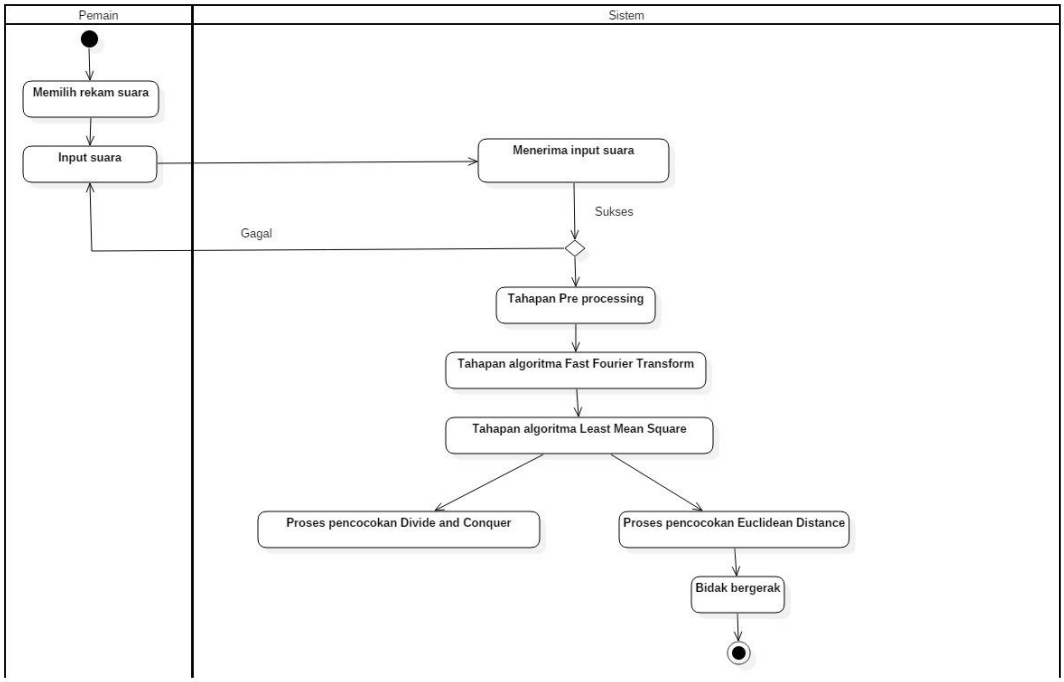
Berikut ini adalah diagram yang menunjukan alur aksi dari memulai permainan yang dapat dilihat pada gambar dibawah ini:



Gambar 3.11 *Activity diagram* memulai permainan

2. *Activity diagram* rekam suara

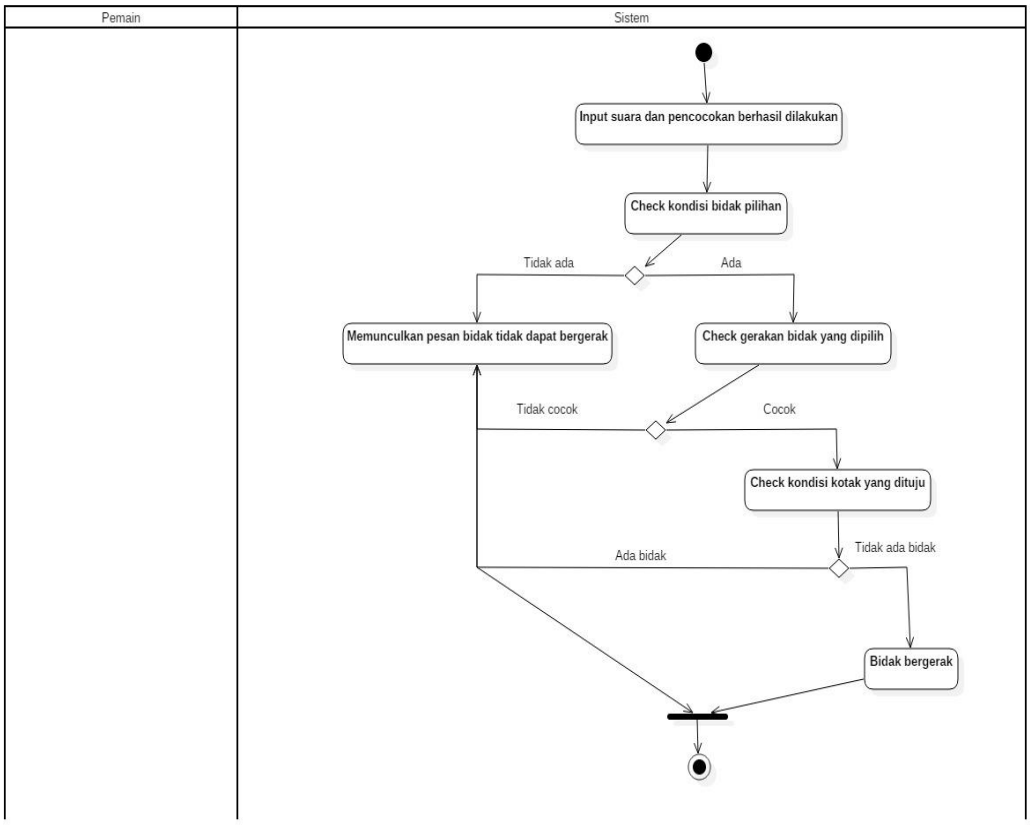
Berikut ini adalah diagram yang menunjukan alur aksi dari input suara yang dapat dilihat pada gambar dibawah ini:



Gambar 3.12 *Activity diagram* rekam suara

3. *Activity diagram* gerakan bidak

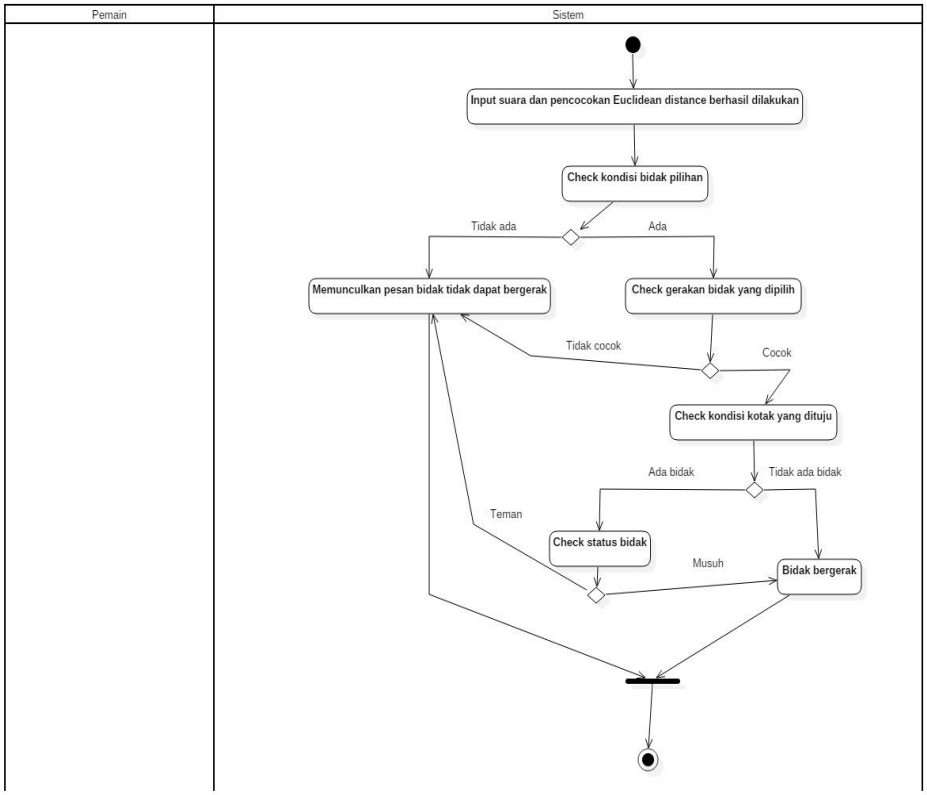
Berikut ini adalah diagram yang menunjukan alur aksi dari gerakan bidak atau gerakan bidak yang dapat dilihat pada gambar dibawah ini:



Gambar 3.13 *Activity diagram* gerakan bidak

4. *Activity diagram* penghapusan bidak

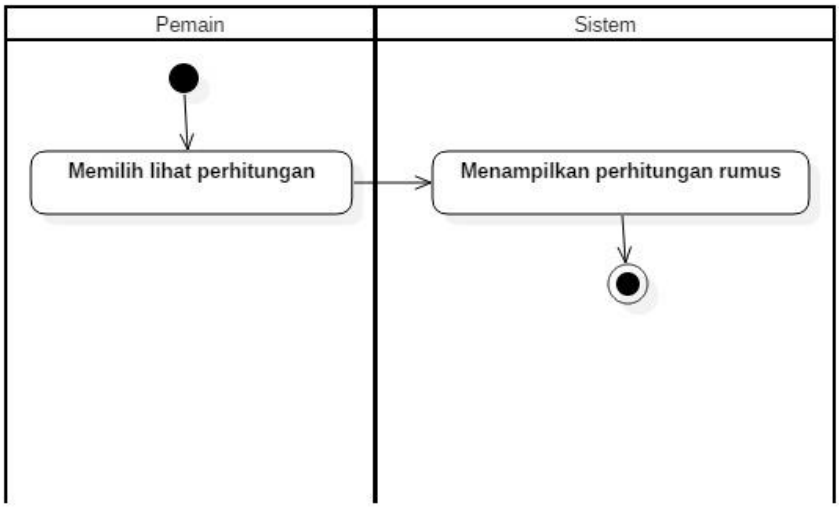
Berikut ini adalah diagram yang menunjukan alur aksi dari hapus atau makan bidak yang dapat dilihat pada gambar dibawah ini:



Gambar 3.14 *Activity diagram* penghapusan bidak

5. *Activity diagram* tampilan tabel perhitungan

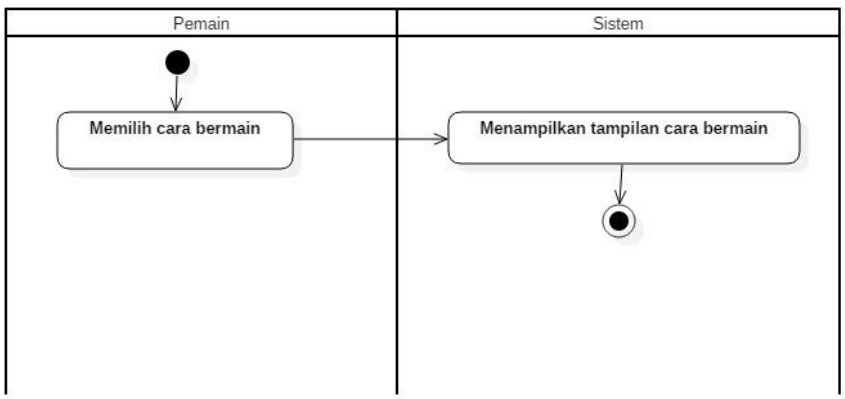
Berikut ini adalah diagram yang menunjukan alur aksi dari tampilan tabel perhitungan atau tabel perhitungan yang dapat dilihat pada gambar dibawah ini:



Gambar 3.15 *Activity diagram* tampilan tabel perhitungan

6. *Activity diagram* tampilan cara bermain

Berikut ini adalah diagram yang menunjukan alur aksi dari tampilan cara bermain atau cara bermain yang dapat dilihat pada gambar dibawah ini:



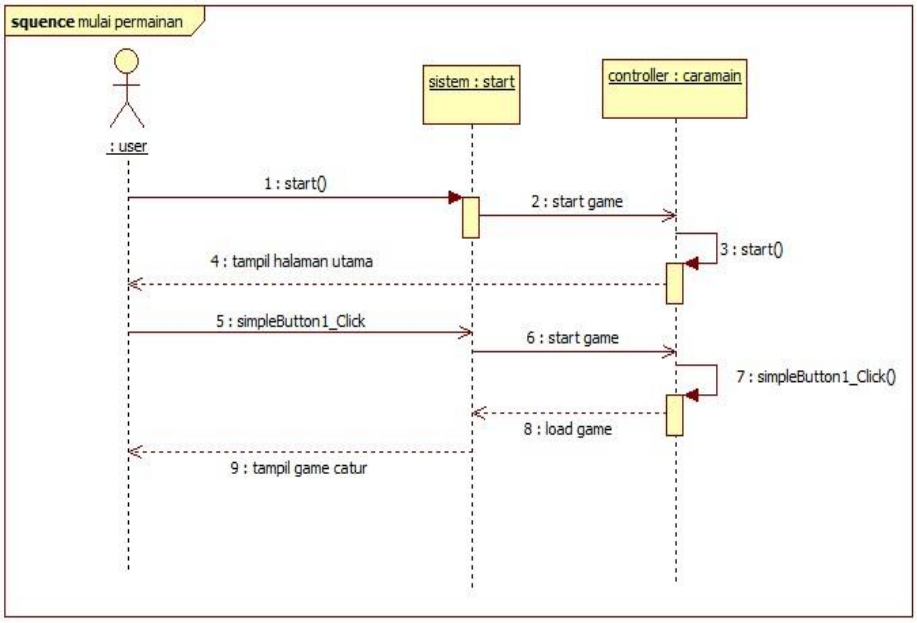
Gambar 3.16 *Activity diagram* tampilan cara bermain

3.10.3 *Sequence Diagram*

Diagram sekuen atau *Sequence Diagram* menggambarkan kelakuan objek atau *class* pada *use case* dengan mendeskripsikan waktu hidup dan *message* yang dikirimkan dan diterima antara *object* atau *class*.

1. *Sequence diagram* memulai permainan

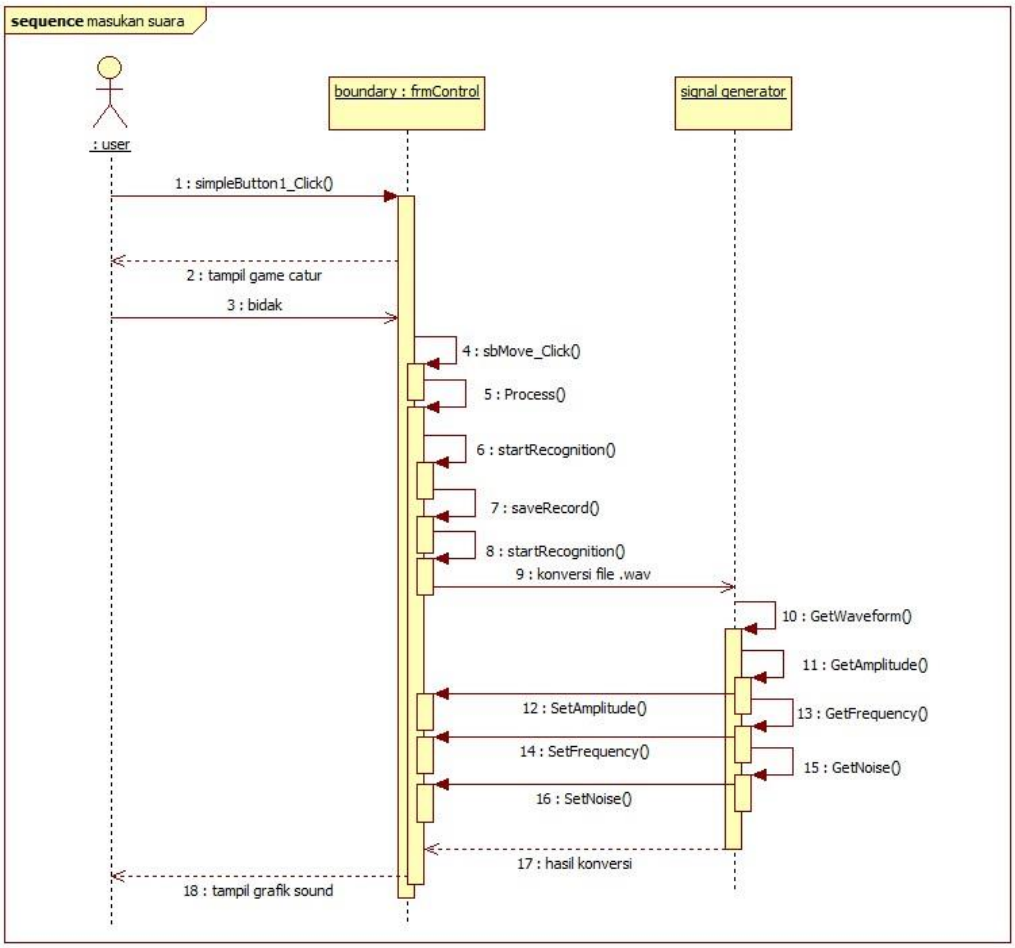
   Diagram *Sequence* yang menunjukan memulai permainan.



Gambar 3.17 Diagram *sequence* memulai permainan

2. *Sequence diagram* rekam suara

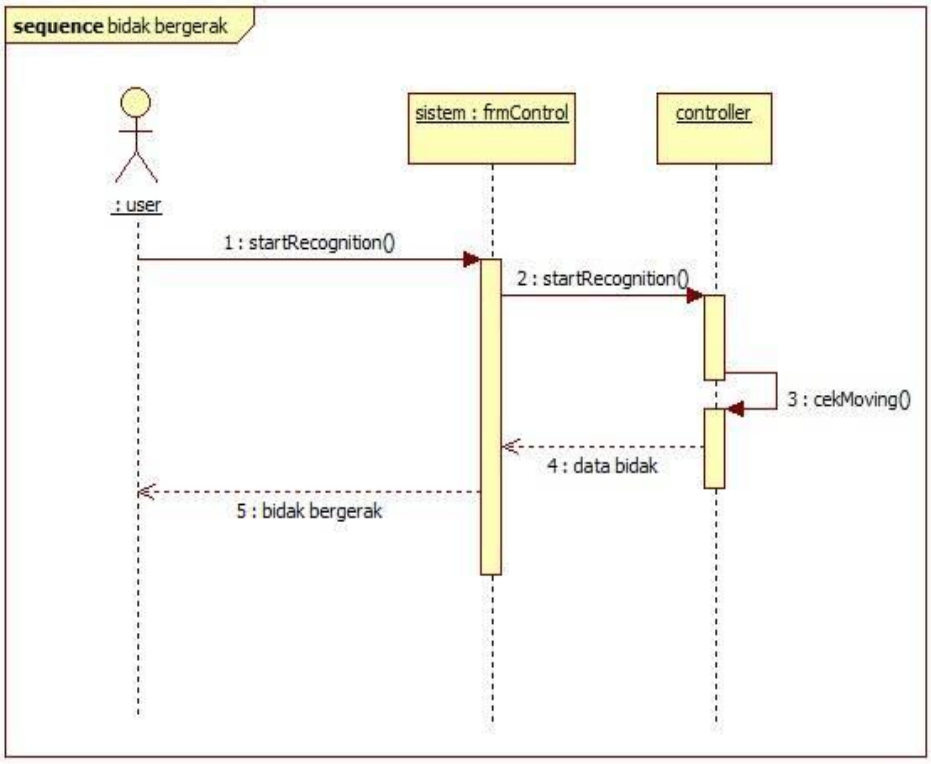
Diagram Sequence yang menunjukan rekam suara.



Gambar 3.18 Diagram *sequence* rekam suara

3. *Sequence diagram* gerakan bidak

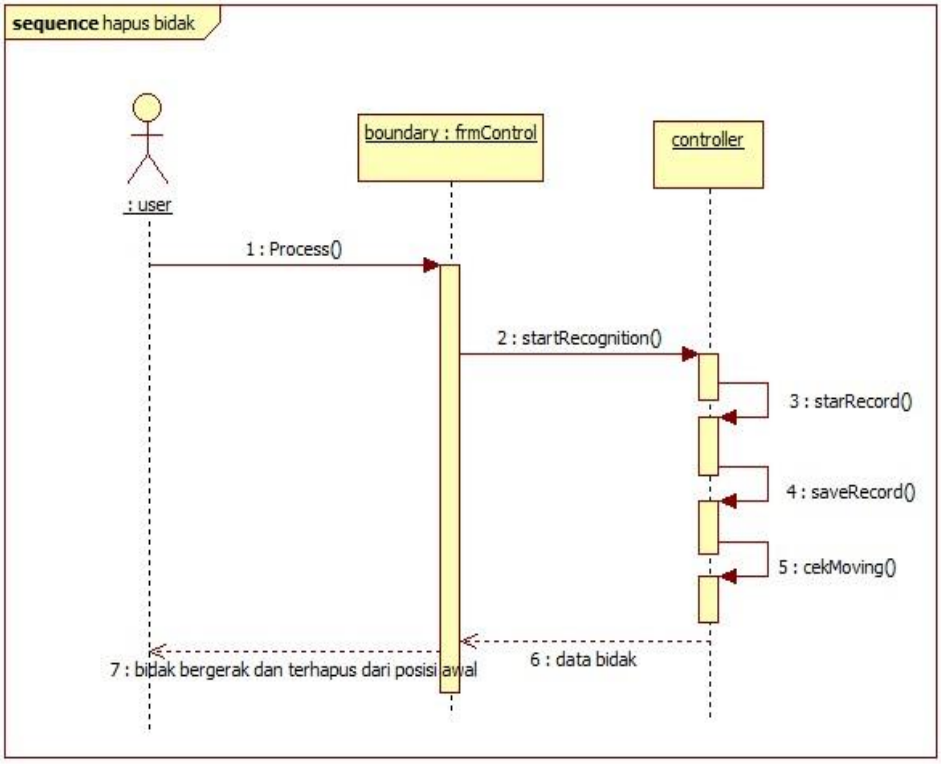
Diagram *Sequence* yang menunjukan gerakan bidak.



Gambar 3.19 Diagram *sequence* gerakan bidak

4. *Sequence diagram* penghapusan bidak

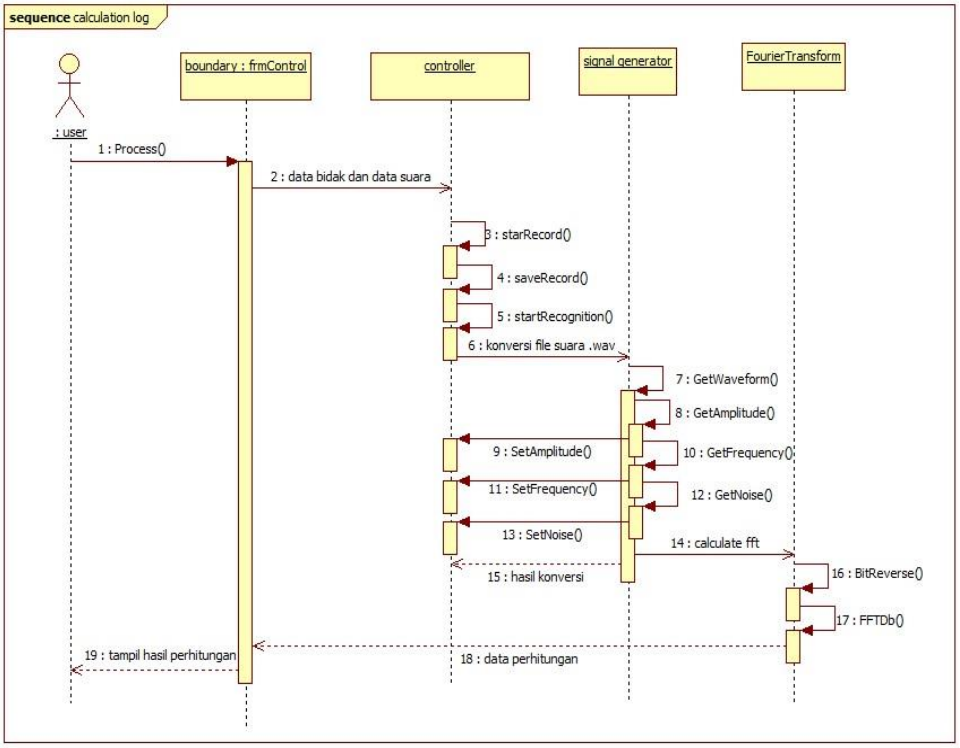
Diagram Sequence yang menunjukan memakan bidak musuh.



Gambar 3.20 Diagram *sequence* penghapusan bidak

5. *Sequence diagram* menampilkan hasil perhitungan

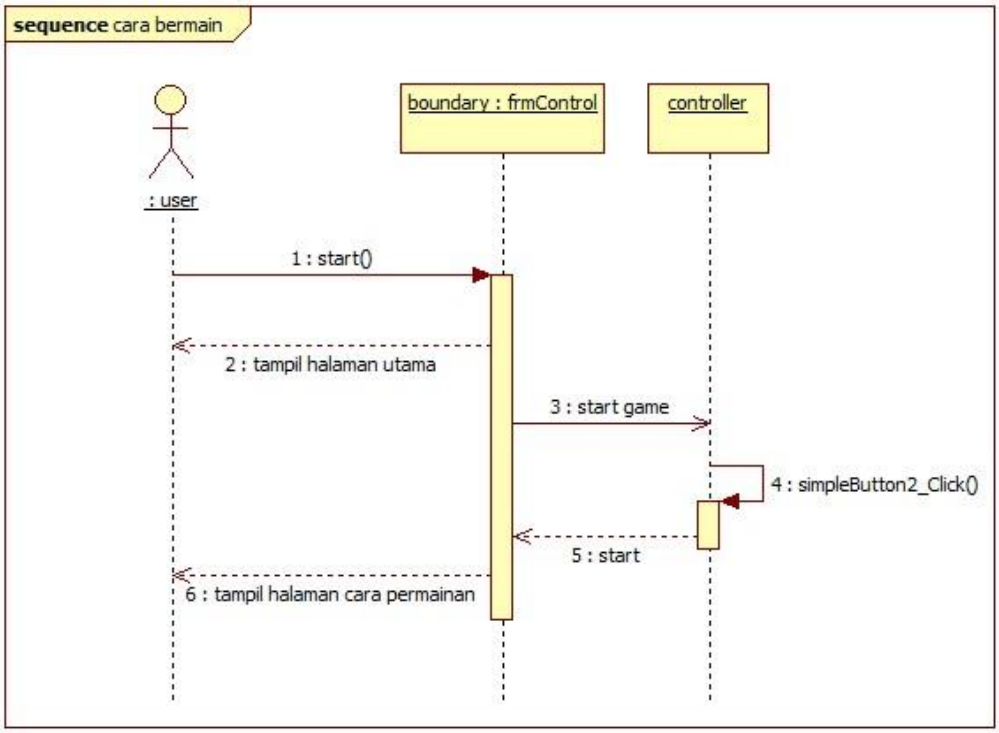
Diagram *Sequence* yang menunjukan hasil perhitungan metode yang digunakan dan pencocokkan pola suara.



Gambar 3.21 Diagram *sequence* menampilkan hasil perhitungan

6. *Sequence diagram* menampilkan cara bermain

Diagram *Sequence* yang menunjukan informasi cara bermain kepada pemain.



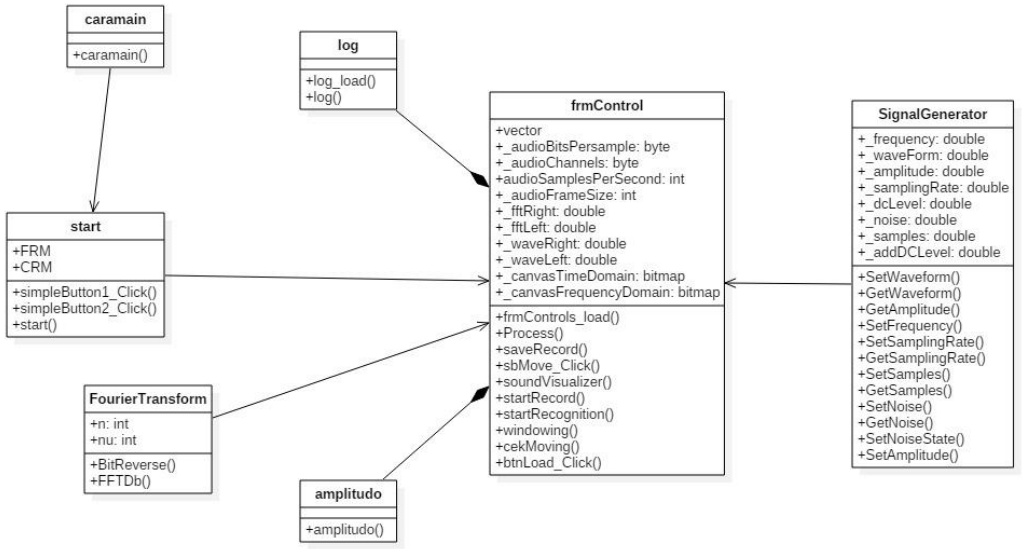
Gambar 3.22 Diagram *sequence* menampilkan cara bermain

3.10.4 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan di buat untuk membangun sistem.

1. *Class diagram*

Diagram kelas yang menggambarkan struktur sistem *game* catur.



Gambar 3.23 *Diagram class*

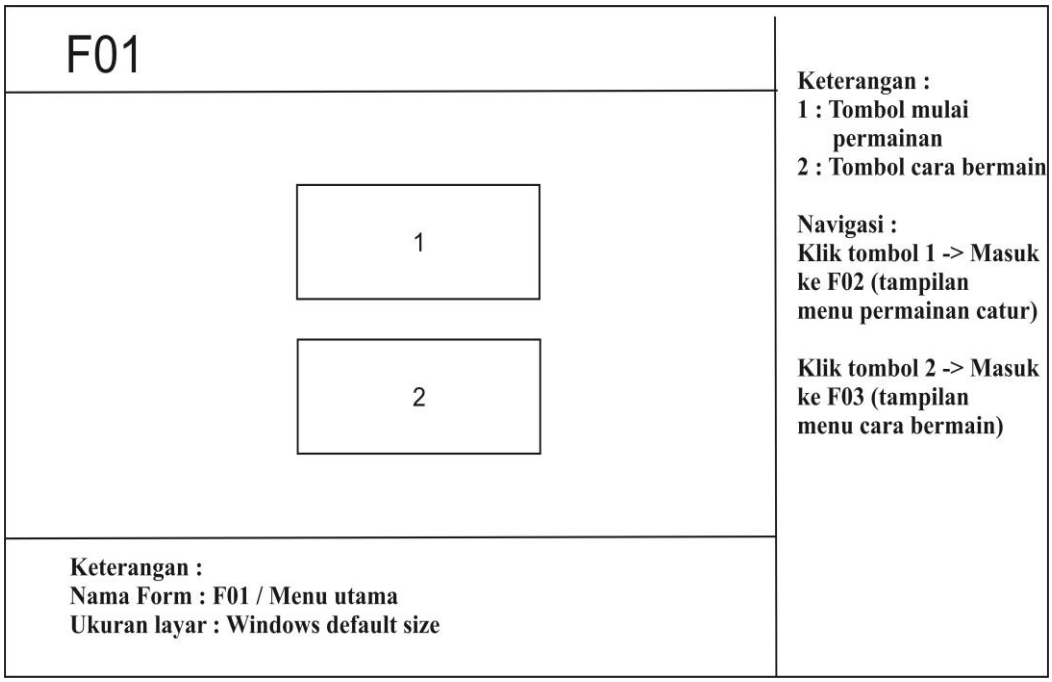
3.11 Perancangan Sistem

Perancangan sistem merupakan tahapan perencanaan aplikasi, desain aplikasi, pembuatan aplikasi dengan menggabungkan beberapa unsur menjadi satu bagian utuh. Pada tahap ini akan dipaparkan mengenai perancangan antar muka dan jaringan semantik.

3.11.1 Perancangan antar muka

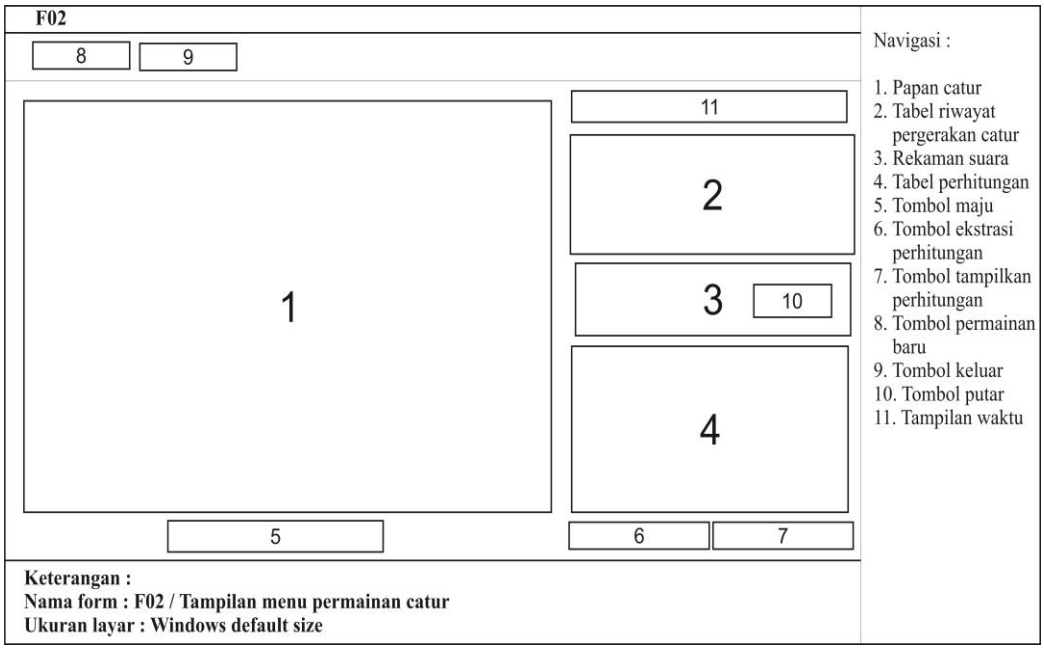
Perancangan antar muka bertujuan untuk memberikan gambaran visual tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam menerapkan aplikasi serta akan mempermudah pembuatan aplikasi. Berikut merupakan perancangan antar muka pada permainan catur.

1. Perancangan tampilan awal



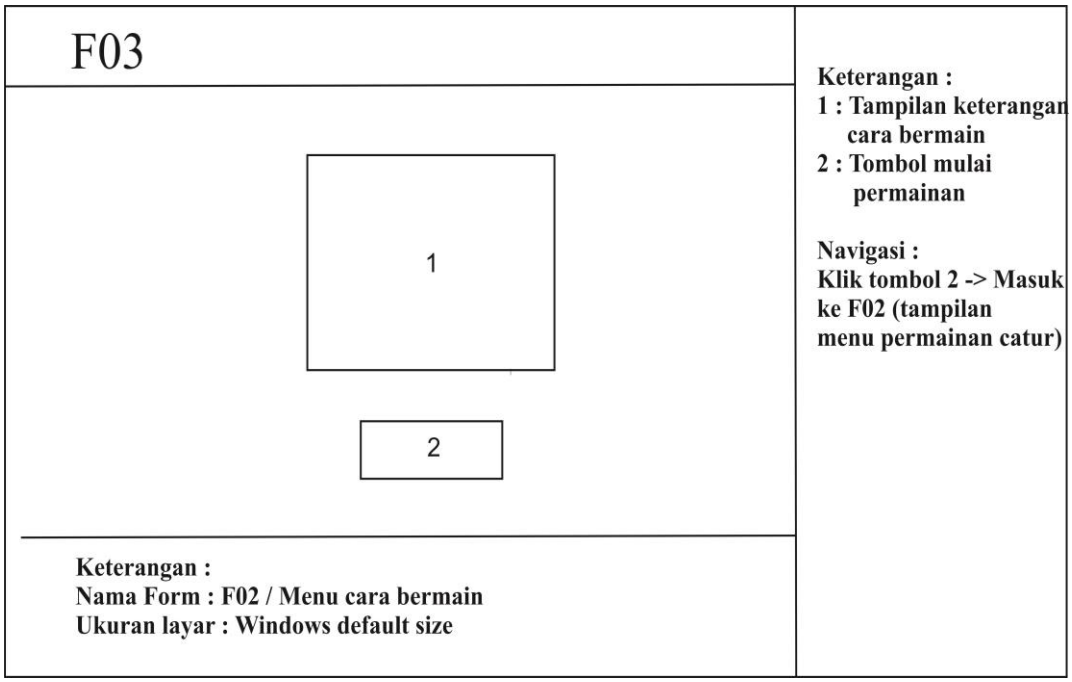
Gambar 3.24 perancangan tampilan awal

2. Perancangan menu permainan catur



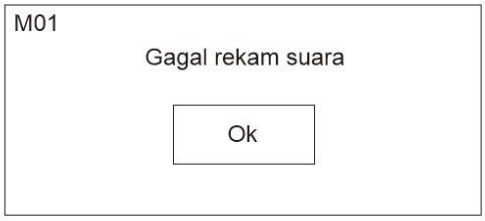
Gambar 3.25 perancangan menu permainan catur

3. Perancangan menu cara bermain



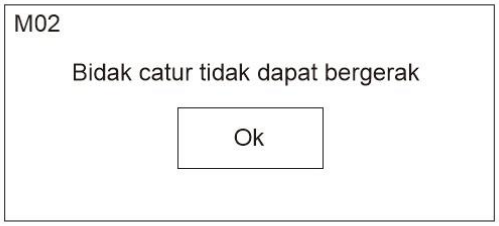
Gambar 3.26 Perancangan menu cara bermain

4. Perancangan antar muka pesan pemberitahuan gagal rekam suara.



Gambar 3.27 Perancangan menu gagal rekam suara

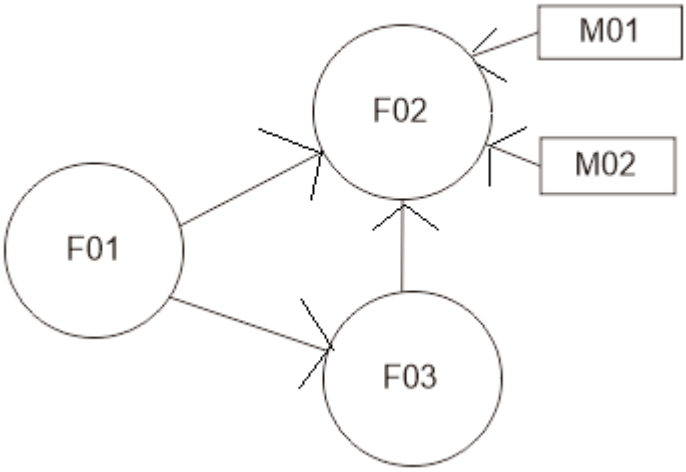
5. Perancangan antar muka pesan pemberitahuan bidak catur tidak dapat bergerak.



Gambar 3.28 Perancangan menu bidak tidak bergerak

3.11.2 Jaringan semantik

Jaringan semantik menggambarkan hubungan navigasi menu dari satu halaman ke halaman lainnya. Berikut merupakan jaringan semantik pada permainan catur yang ditunjukan oleh gambar 3.29.



Gambar 3.29 Jaringan semantik