BAB V
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari perancangan *game design document* dan perancangan *technical design document*. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi algoritma, dan implementasi antarmuka aplikasi.

5.1. Spesifikasi Sistem

Perangkat lunak ini dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

5.1.1. Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan dijelaskan pada Tabel 5.1.

Tabel 5.1. Spesifikasi perangkat keras komputer

Notebook Sony VAI O VGN-SR129E	
Processor	Intel (R) Core (TM) Duo CPU P8400 @ 2.26 GHz
Memory (RAM)	4 GB
Harddisk	250 GB
Motherboard	Mobile Intel PM45 Express Chipset
Graphic Card	ATI RADEON 3400 series
Monitor	*13.3-Inch Led Backlit Screen With XBRITE-Eco Technology*

Sumber: Implementasi

5.1.2. Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan perangkat lunak dijelaskan pada Tabel 5.2.

Tabel 5.2. Spesifikasi perangkat lunak komputer

Notebook Sony VAI O VGN-SR129E	
Operating System	Microsoft Windows 7 (6.1) Ultimate Edition (Build 7600)
DirectX Version	DirectX 11
Programming Language	Java
Software Development Kit	Java Development Kit 6 Update 18
Programming Environment	Java Runtime Environment 6
Android Operating System	Android 2.1 API 7
Editor	Eclipse Helios Service Release 1
ADT	ADT plugins versi 15

Sumber: Implementasi

5.2. Batasan Implementasi

Batasan dalam mengimplementasikan perangkat lunak ini adalah sebagai berikut

1. Perangkat lunak ini berjalan pada *smart phone* dengan sistem operasi Android dengan versi minimal SDK 7 serta dapat terintegrasi dengan jaringan lokal.
2. Pembuatan permainan Sarinande Angklung menggunakan *AndEngine*.
3. Pembuatan permainan Sarinande Angklung menggunakan *android* API 7.
4. Pembuatan permainan Sarinande Angklung menggunakan angklung dengan jenis angklung sarinande dengan tangga nada satu oktaf yang terdiri dari delapan nada.
5. Implementasi *file beat* yang digunakan dalam permainan Sarinande Angklung menggunakan struktur XML.
6. Pengaksesan *file-file* pada permainan Sarinande Angklung terbatas pada direktori *sdcard/Sarinande.*
7. Pengonversian *file beat* menjadi *file audio* dilakukan dengan mengaktifkan *microphone* dan merekam semua suara yang ada di lingkungan sekitar *smartphone*.
8. Hasil pengonversian *file beat* menjadi *file audio* berupa file .AMR.

9. Fitur multiplayer hanya dapat dimainkan pada jaringan local yang terkoneksi pada satu *access point*.
10. Fitur multiplayer pada permainan Sarinande Angklung ini berbasis *client server* menggunakan protokol TCP.
11. Penanganan eksepsi pada *multiplayer* hanya terbatas pada pengiriman dan penerimaan pesan antara *server* dan *client*.

5.3. Implementasi penyimpanan data pada *file beat*

File beat merupakan tempat penyimpanan informasi yang didesain dengan menggunakan format XML dan dikemas dalam sebuah *file* dengan tipe data .ANG. Informasi pada file *beat* terdiri dari *title beat*, *author beat*, judul musik, nada, dan waktu *beat* akan keluar. Implementasi penyimpanan data pada file *beat* ditunjukkan pada Tabel 5.3.

Tabel 5.3. Implementasi XML pada *file beat*

No	XML
1	`<?xml version="1.0" encoding="ISO-8859-1"?>`
2	`<angklung>`
3	`  <detail>`
4	`  <title>gundul gundul pacul</title>`
5	`  <author>hendra</author>`
6	`  <song>lagu_01.mp3</song>`
7	`  </detail>`
8	`  <collection>`
9	`  <node time="6336"  node="do" />`
10	`  <node time="9106"  node="re" />`
11	`  <node time="9805"  node="mi" />`
12	`  <node time="11041" node="fa" />`
13	`  <node time="11805" node="sol" />`
14	`  <node time="12451" node="la" />`
15	`  </collection>`
16	`</angklung>`

Sumber: Implementasi

Struktur XML pada *file beat* terdiri dari beberapa elemen yaitu

1. **Angklung**

   Elemen angklung merupakan *root* elemen pada file *beat*. Elemen angklung dibuka dengan menggunakan *tag* pembuka <angklung> dan ditutup dengan *tag* penutup </angklung>. Elemen inilah yang menunjukkan bahwa *file* merupakan *file beat* permainan Sarinande

Angklung. Di dalam elemen angklung terdapat dua anak elemen yaitu *detail* dan *collection*.

2. **Detail**

   Detail merupakan anak elemen dari elemen angklung. Elemen detail dibuka dengan menggunakan *tag* pembuka <detail> dan ditutup dengan *tag* penutup </detail>. Elemen detail menunjukkan identitas pada file *beat*. Identitas *file beat* terdiri dari *title, author,* dan *song*.

3. ***Title***

   Elemen *title* dibuka dengan menggunakan *tag* pembuka <title> dan ditutup dengan *tag* penutup </title>. Elemen *title* ini menunjukkan nama atau judul pada file *beat*.

4. ***Author***

   Elemen *author* dibuka dengan menggunakan *tag* pembuka <author> dan ditutup dengan *tag* penutup </author>. Elemen *author* ini menunjukkan nama orang yang membuat *file beat*.

5. ***Song***

   Elemen *song* dibuka dengan menggunakan *tag* pembuka <song> dan ditutup dengan *tag* penutup </song>. Elemen *song* ini menunjukkan nama *audio* yang akan diputar saat menjalankan *game*.

6. ***Collection***

   Collection merupakan anak elemen dari elemen angklung. Elemen *collection* dibuka dengan menggunakan *tag* pembuka <collection> dan ditutup dengan *tag* penutup </collection>. Elemen collection merupakan elemen yang menyimpan semua notasi pada *beat* dimana setiap notasi dilambangkan dengan elemen *node*.

7. ***Node***

   Elemen *node* dibuka dengan menggunakan *tag* pembuka <node> dan ditutup dengan *tag* penutup </node>. Elemen node melambangkan sebuah nada pada partitur. Elemen node memiliki dua atribut yaitu time dan tone. Atribut *time* menunjukkan waktu kapan nada harus dibunyikan dan atribut *node* menunjukkan nada apa dibunyikan.

5.4. Implementasi Kode

Perangkat lunak permainan Sarinande Angklung memiliki beberapa proses atau *method* yang terdapat pada beberapa *class*. Penulisan skripsi ini hanya kode dari beberapa proses (operasi) saja sehingga tidak semua *method* akan dicantumkan. Implementasi kode ini akan direpresentasikan dalam bentuk *pseudocode* dari kode – kode tersebut.

5.4.1. Kode penggunaan *touch* pada angklung delapan nada

Permainan Sarinande Angklung menggunakan delapan nada angklung pada salah satu *gameplay*. Penggunaan *touch* pada angklung dibagi menjadi dua yaitu inisialisasi id angklung dan pengolahan *input touch*. Implementasi kode penggunaan *touch* pada angklung delapan nada ditunjukkan pada Tabel 5.4 dan Tabel 5.5.

Tabel 5.4. Kode inisialisasi id angklung

Pseudo Code

```
ALGORITHM NAME: Inisialisasi id angklung
DECLARATION:
      TYPE sound IS Sound
      tone IS List
      allsprite IS List
      texture IS Texture
      angklung IS TextureRegion
      angklungSprite IS Sprite
      positionX IS float
      positionY IS float
      itemClickIndex IS int
      a IS int
      game IS Thread //thread for touch input
INPUT: touch IS TouchEvent
DESCRIPTION:
  1    tone <- CREATE OBJECT List
  2    allSprite <- CREATE OBJECT List
  3    texture <- CREATE OBJECT Texture(128,128);
  4    angklung <- CREATE OBJECT  TextureRegion ("texture", "path_gambar.png");
  5    FOR a=0 to 7 DO
  6          Sound <- CREATE OBJECT Sound("nada_" + a + ".mp3")
  7         CALL tone.add(sound)
  8    ENDFOR
  9    positionX <- 50
 10    positionY <- 180
 11    FOR a=0 to 7 DO
 12       angklungSprite <- CREATE OBJECT Sprite (positionX,  positionY, angklung)
 13       CALL angklungSprite.setAreaTouchId(a)
 14       positionX <- positionX + CALL angklungSprite.getWidth() + 50
 15       CALL allsprite.add(angklungSprite)
 16    ENDFOR
```

Sumber: Implementasi

Penjelasan dari kode inisialisasi id angklung pada Tabel 5.4., yaitu

1. Baris 1 menjelaskan penginstansiasian objek pada variabel *tone* dengan tipe data List yang akan menampung objek *Sound*.
2. Baris 2 menjelaskan penginstansiasian objek pada variabel *allsprite* dengan tipe data List yang akan menampung objek *Sprite*.
3. Baris 3 menjelaskan penginstansiasian objek *Texture* pada variabel *texture*, dimana variabel ini dibuat untuk menyediakan memory sebesar 128 pixel x 128 pixel.
4. Baris 4 menjelaskan penginstansiasian objek *TextureRegion* pada variabel *angklung*, dimana variabel ini dibuat untuk mengalokasikan gambar angklung pada memory.
5. Baris 5 – 8 menjelaskan pengintansian objek Sound sebanyak delapan objek. Objek sound ini digunakan untuk mengakses file sound yang berada di asset. Objek yang telah dibuat akan ditampung pada variabel *tone*.
6. Baris 9 menjelaskan pemberian nilai pada variabel positionX dengan nilai 50.
7. Baris 10 penjelasan pemberian nilai pada variabel positionY dengan nilai 180.
8. Baris 11 – 16 menjelaskan pembuatan objek Sprite sebanyak delapan buah. Kemudian objek diberi id untuk pemberian identitas pada sprite. Pada baris 14 menjelaskan pemberian id angklung untuk area touch. Objek yang dibuat kemudian disimpan pada variabel *allsprite*.

Tabel 5.5. Kode pengolahan *input touch* pada angklung

Pseudo Code

```
ALGORITHM NAME: Pengolahan input touch pada angklung
DECLARATION:
      TYPE
      a IS int
INPUT: touch IS TouchEvent
DESCRIPTION:
  1    IF CALL touch.getPointerID() > -1 THEN
  2                FOR a=0 to 7 DO
  3                     IF CALL touch.getPointerID() == b THEN
  4                          CALL tone.get(a).play()
  6                     ELSE
  7                          CALL tone.get(a).stop()
  8                     ENDIF
  9                ENDFOR
 10    ENDIF
```

Sumber: Implementasi

Penjelasan dari kode pengolahan input touch angklung pada Tabel 5.5., yaitu

1. Baris 1 – 10 menjelaskan proses pemasukan input *touch screen.* Ketika pemain menyentuh layar, maka inputan tersebut akan dicek apakah id pada variabel *touch* sama dengan id angklung. Jika sama akan membunyikan suara angklung.

5.4.2. Kode penggunaan *accelerometer* pada angklung satu nada

Permainan Sarinande Angklung menggunakan angklung satu nada pada salah satu *gameplay*. Untuk memasukkan input pada *game* menggunakan sensor *accelerometer*. Implementasi kode penggunaan *accelerometer* pada nada angklung satu nada ditunjukkan pada Tabel 5.6.

Tabel 5. 6. Kode penggunaan *accelerometer* pada angklung

Pseudo Code

```
ALGORITHM NAME: Penggunaan accelerometer pada angklung
DECLARATION:
      TYPE sound IS Sound
      tone IS List
      a IS int
      x IS Float
      y IS Float
      deltaX IS Float
      deltaY IS Float
      lastX IS Float
      lastY IS Float
      sensor IS Thread
INPUT: event IS SensorEvent, chooseId IS int
DESCRIPTION:
 1    tone <- CREATE OBJECT List
 2    vibrator <- CALL getSystemService (Context.VIBRATOR_SERVICE)
 3    FOR a=0 to 7 DO
 4         Sound <- CREATE OBJECT Sound("nada_" + a + ".mp3")
 6         CALL tone.add(sound)
 7    ENDFOR
 8    WHILE CALL sensor.isAlive() DO
 9         x <- CALL event.getX()
10         y <- CALL event.getY()
11         IF event.sensor.getType()== Sensor.TYPE_ACCELEROMETER THEN
12              IF isInitial == FALSE THEN
13                   lastX <- x
14                   lastY <- y
15                   isInitial <- TRUE
16              THEN
17                   deltaX <- lastX - x
18                   deltaY <- lastY - y
19                   lastX <- x
20                   lastY <- y
21                   IF deltaX > deltaY THEN
22                        CALL tone.get(chooseId).play()
23                        IF CALL option.getVibrator == TRUE THEN
```

```
24                      CALL vibrator.vibrate(60)
25                    ENDIF
26                  ENDIF
27                ENDIF
28              ENDIF
29  ENDWHILE
```

Sumber: Implementasi

Penjelasan dari kode penggunaan *accelerometer* pada angklung pada Tabel 5.6., yaitu

1. Baris 1 menjelaskan penginstansiasian objek pada variabel *tone* dengan tipe data List yang akan menampung objek *Sound*.
2. Baris 2 menjelaskan proses penginstansiasian objek vibrator
3. Baris 3 – 7 menjelaskan pengintansian objek Sound sebanyak delapan objek. Objek sound ini digunakan untuk mengakses file sound yang berada di asset. Objek yang telah dibuat akan ditampung pada variabel *tone*.
4. Baris 8 menjelaskan perulangan akan terus dilakukan jika sensor masih aktif.
5. Baris 9 menjelaskan pengambilan posisi x pada *smartphone*.
6. Baris 10 menjelaskan pengambilan posisi y pada *smartphone*.
7. Baris 12 menjelaskan proses seleksi kondisi apakah sensor yang aktif adalah sensor *accelerometer*. Jika bernilai benar maka akan melanjutkan ke proses selanjutnya.
8. Baris 13 – 16 menjelaskan proses seleksi kondisi jika variabel isInitial bernilai false maka akan menginialisasi posisi x terakhir pada lastX, menginialisasi posisi y terakhir pada lastY, dan memberikan nilai true pada isInitial.
9. Baris 21 – 29 menjelaskan proses penghitungan selisih posisi x yang dimasukkan pada variabel deltaX dan selisih posisi y yang dimasukkan pada variabel deltaY. Jika nilai pada deltaX lebih dari deltaY maka akan membunyikan suara angklung. Jika pengaturan vibrasi adalah aktif maka *smartphone* akan bergetar.

5.4.3. Kode pendeteksian *file beat* pada SD card

Fitur multiplayer dan touch game pada Sarinande Angklung, menggunakan file beat. Dengan demikian pemain harus memilih *file beat*. Sebelum *file beat* dapat dipilih, langkah yang dilakukan sebelumnya adalah mendeteksi semua *file beat* yang terdapat pada SD card. Implementasi pendeteksian file beat ditunjukkan pada Tabel 5.7.

Tabel 5.7. Kode pendeteksian *file beat*

Pseudo Code

```
ALGORITHM NAME: Pendeteksian file beat
DECLARATION:
      TYPE fileExtensions IS String <- ".ang"
      FolderList IS List
      FileList IS List
      dirpath IS STRING <- "sdcard/Sarinande";
      dir IS File
      objek IS File
      file IS File
      a IS int
      b IS int
DESCRIPTION:
 1    FolderList <- CREATE OBJECT List
 2    FileList <- CREATE OBJECT List
 3    dir <- CREATE OBJECT File(dirpath)
 4    IF CALL dir.listFiles().length > 0 THEN
 5          FOR objek : CALL dir.listFiles() DO
 6             IF CALL objek.isDirectory() == TRUE THEN
 7                CALL FolderList.add(objek)
 8             ELSE
                IF CALL objek.isFIle() == TRUE AND CALL
                objek.getName().endsWith(fileExtensions) THEN
 9                CALL FileList.add(objek)
10             ENDIF
11          ENDFOR
12          SET a <- 0
13          WHILE a < CALL FolderList.size() DO
14             objek <- CREATE OBJECT File (CALL FolderList.get(a).
               getAbsolutePath())
15             IF CALL objek.isDirectory() == TRUE THEN
16                CALL FolderList.add(objek)
17             ENDIF
18             a <- a + 1
19          ENDWHILE
20          FOR objek : FolderList DO
21             IF CALL objek.listFiles().length > 0 DO
22                FOR file: CALL objek.listFiles() DO
23                   IF file.isFiles() == TRUE AND CALL
24                   file.getName().endsWith(fileExtensions)
25                     CALL FileList.add(objek)
26                   ENDIF
27                ENDFOR
28             ENDIF
29          ENDFOR
30   ENDIF
31   RETURN FileList
```

Sumber: Implementasi

Penjelasan dari kode pendeteksian *file beat* pada Tabel 5.7., yaitu

1. Baris 1 menjelaskan penginstansiasian objek pada variabel *FolderList* dengan tipe data List yang akan menampung objek *File* yang merupakan direktori
2. Baris 2 menjelaskan penginstansiasian objek pada variabel *FileList* dengan tipe data List yang akan menampung objek *File* yang merupakan *file*.
3. Baris 3 menjelaskan penginstansiasian objek pada variabel *dir* dengan tipe data *File* yang akan menampung objek *File* dengan direktori sdcard/Sarinande
4. Baris 4 -13 menjelaskan proses pengecekan pertama untuk mengumpulkan semua direktori dan file yang berekstensi .ANG pada sdcard/Sarinande. Dimana semua file yang merupakan direktori disimpan pada variabel FolderList dan semua file yang berekstensi .ANG akan disimpan pada variabel FileList
5. Baris 12 – 19 menjelaskan proses pengecekan kedua untuk mengumpulkan semua direktori yang berada di dalam folder yang telah dikumpulkan sebelumnya. Kemudian semua folder tersebut akan disimpan pada variabel FolderList.
6. Baris 20 - 30 menjelaskan proses pengecekan ketiga untuk mengumpulkan semua file yang berekstensi .ANG yang berada di dalam folder yang telah dikumpulkan semuanya dan akan disimpan pada variabel FileList.
7. Bari 31 menjelaskan proses pengembalian semua nilai pada variabel FileList

5.4.4. Kode pembacaan *file beat*

Setelah file beat dipilih langkah selanjutnya adalah mendapatkan nilai-nilai yang berada pada elemen-elemen file beat. Nilai-nilai yang didapatkan kemudian akan disimpan pada variabel List. Implementasi kode pembacaan file beat ditunjukkan pada Tabel 5.8, 5.9, 5.10, 5.11. dan 5.12.

Tabel 5.8. Implementasi kode pembacaan *file beat: Init*

Pseudo Code
ALGORITHM NAME: Pembacaan file beat: Init DECLARATION:

```
      TYPE spf IS SAXParserFactory
      time IS List
      node IS List
      sax IS SAXParser
      gf IS GameFile
      tempVal IS String
DESCRIPTION:
 1    time <- CREATE OBJECT List
 2    node <- CREATE OBJECT List
 3    spf <- CALL SAXParserFactory
 4    BEGIN
 5         sax <- CALL spf.newSAXParser()
 6         gf <- CALL GameFile.getInstance()
 7         path <- CALL gf.getGameFile().getPath()
 8         CALL sp.parse(CREATE OBJECT InputSource(CREATE OBJECT
           FileInputStream("path")))
 9    EXCEPTION
 10        WHEN ParserConfiguration e
 11          WRITE e
 12        WHEN SAXException e
 13          WRITE e
 14        WHEN IOExcepetion e
 15          WRITE e
 16   END
```

Sumber: Implementasi

Tabel 5.9. Implementasi kode pembacaan *file beat: start element*

Pseudo Code

```
ALGORITHM NAME: Pembacaan file beat: startElement
DECLARATION:
      TYPE value1 IS String
      value2 IS String
INPUT: uri IS String, localName IS String, qName IS String, atribut IS Attributes
DESCRIPTION:
 1    IF CALL qName.equals("node") THEN
 2       value1 <- CALL atribut.getValue("time")
 3       value2 <- CALL atribut.getValue("node")
 4       time.add(value1)
 5       node.add(value2)
 6    ENDIF
```

Sumber: Implementasi

Tabel 5.10. Implementasi kode pembacaan *file beat: characters*

Pseudo Code

```
ALGORITHM NAME: Pembacaan file beat: characters
DECLARATION:
INPUT: ch IS char[], start IS int, length IS int
DESCRIPTION:
 1    tempVal <- CREATE OBJECT String(ch, start, length)
 2
```

Sumber: Implementasi

Tabel 5.11. Implementasi kode pembacaan *file beat: end element*

Pseudo Code

```
ALGORITHM NAME: Pembacaan file beat: endElement
```

```
DECLARATION:
INPUT: uri IS String, localName is String, qName IS String
DESCRIPTION:
 1    CASE qName OF
 2       title  : title <- tempVal BREAK
 3       author : author <- tempVal BREAK
 4       song   : song <- tempVal BREAK
 5    ENDCASE
```

Sumber: Implementasi

Tabel 5.12. Implemnetasi kode pembacaan *file beat*: Penyimpanan hasil pembacaan

Pseudo Code
ALGORITHM NAME: Pembacaan file beat: Penyimpanan hasil pembacaan DECLARATION: INPUT: uri IS String, localName is String, qName IS String DESCRIPTION: 1 CALL gf.setTitle(title) 2 CALL gf.setAuthor(author) 3 CALL gf.setMusicGame(song) 4 CALL gf.setTime(time) 5 CALL gf.setNode(node)

Sumber: Implementasi

Proses pembacaan *file beat* merupakan pengambilan nilai-nilai pada elemen XML *file beat*. Pembacaan file beat menggunakan SAXParser. Secara garis besar, terdapat tiga bagian pada XML, yaitu elemen *start*(*tag* pembuka), elemen *end*(*tag* penutup), dan *characters*(nilai yang terdapat diantara tag). Dengan demikian proses yang dilakukan dalam pembacaan *file beat* adalah init, pembacaan pada startElement, pembacaan pada characters, pembacaan endElement, dan penyimpanan hasil pembacaan. Kode-kode pada gambar 5.8, 5.9, 5.10, 5.11, dan 5.12. merupakan method-method yang terdapat pada kelas SAXParserXML.java. Penjelasan pada implementasi kode pembacaan file beat adalah

A. Init

1. Baris 1 - 5 merupakan proses instansiasi objek yang akan digunakan pada proses selanjutnya.
2. Baris 6 -7 merupakan proses pengambilan *path* direktori file yang akan dibaca. Kemudian path direktori tersebut akan disimpan pada variabel path.
3. Baris 8 -16 merupakan proses eksekui pembacaan XML pada file dengan nilai yang terdapat pada variabel path. Proses pembacaan

dilakukan dengan mengambil setiap baris pada *file beat* kemudian akan dicek bagian mana yang merupakan startElement, bagian mana yang merupakan characters, dan bagian mana yang merupakan endElement

B. Pembacaan startElement

Baris 1 -6 menjelakan proses seleksi kondisi jika start element merupakan "node" maka akan diambil atributnya yaitu *time* dan *node*. Kemudian akan disimpan pada variabel List.

C. Pembacaan characters

Baris 1 – 2 menjelaskan proses instansiasi objek String dari setiap pembacaan karakter diantara *start element* dan *end element*.

D. Pembacaan endElement

Baris 1 -5 menjelaskan proses pengambilan nilai sebelum endElement. Jika end element sama dengan *title* maka akan diambil nilainya dan dimasukkan ke dalam variabel *title*. Jika end element sama dengan *author* maka akan diambil nilainya dan dimasukkan ke dalam variabel *author*. Jika end element sama dengan *song* maka akan diambil nilainya dan dimasukkan ke dalam variabel *song.*

E. Penyimpanan hasil pembacaan

Baris 1 -5 menjelaskan proses penyimpanan data dimana akan disimpan pada kelas GameFile. GameFile merupakan sebuah kelas yang hanya dapat diinstansiasi sebanyak satu kali.

5.4.5. Kode pembuatan kotak *beat*

Setelah proses pembacaan *file beat* selesai dilakukan langkah selanjutnya adalah memroses nilai-nilai yang didapat agar dapat ditampilkan pada layar berpa kotak *beat*. Implementasi kode pembuatan kotak beat ditunjukkan pada Tabel 5.13.

Tabel 5.13. Implementasi kode pembuatan kotak *beat*

Pseudo Code

```
ALGORITHM NAME: Pembuatan kotak beat
DECLARATION:
      TYPE tap IS LIST
      tr IS TextureRegion
      scene IS Scene
      nodeList IS List
      timeList IS List
      game IS GameFile
      notasi IS String
      time IS Float
      beat IS AbstractTap
      a IS int
      switcher IS int
DESCRIPTION:
 1    tap <- CREATE OBJECT List
 2    scene <- CREATE OBJECT Scene
 3    tr <- CREATE OBJECT TextureRegion ("path_gambar.png")
 4    nodeList <- CREATE OBJECT List
 5    timeList <- CREATE OBJECT List
 6    game <- CALL GameFile.getInstance()
 7    nodeList <- CALL game.getNode()
 8    timeList <- CALL game.getTime()
 9    FOR a=0 to CALL nodeList.size() DO
10          notasi <- CALL nodeList.get(a)
11          time <- CALL Float.parseFloat(timeList.get(a))/1000
12          CASE notasi OF
13                do    : beat <- CREATE OBJECT DoTap(tr, timing, scene)
14                        BREAK
15                re    : beat <- CREATE OBJECT ReTap(tr, timing, scene)
16                        BREAK
17                mi    : beat <- CREATE OBJECT MiTap(tr, timing, scene)
18                        BREAK
19                fa    : beat <- CREATE OBJECT FaTap(tr, timing, scene)
20                        BREAK
21                sol   : beat <- CREATE OBJECT SolTap(tr, timing, scene)
22                        BREAK
23                la    : beat <- CREATE OBJECT LaTap(tr, timing, scene)
24                        BREAK
25                si    : beat <- CREATE OBJECT SiTap(tr, timing, scene)
26                        BREAK
27                do2   : beat <- CREATE OBJECT Do2Tap(tr, timing, scene)
28                        BREAK
29              default: beat <- NULL BREAK
30          ENDCASE
31          CALL tap.add(beat)
32    ENDFOR
```

Sumber: Implementasi

Tabel 5.14. Implementasi kode penentuan posisi kotak *beat*

Pseudo Code

```
ALGORITHM NAME: Penentuan posisi kotak beat
DECLARATION:
      TYPE speed IS Float <- 100
      distanceToAreaHit IS Float <- 100
      x IS  <- 0
      y IS float
      beat IS Sprite
INPUT: tr IS TextureRegions, timing IS float, scene IS Scene
DESCRIPTION:
1    y <- distanceToAreaHit -(timing * speed)
2    beat <- CREATE OBJECT Sprite(CALL getXValue(), y, tr)
3    CALL beat.setSpeedY(speed);
4    CALL scene.addEntity(beats);
```

Sumber: Implementasi

Penjelasan dari kode pembuatan kotak *beat* pada Tabel 5.13., yaitu

1. Baris 1 - 6 menjelaskan proses penginstansiasian objek yang akan digunakan dalam proses selanjutnya.
2. Baris 7 – 8 menjelaskan proses pengambilan hasil pembacaan elemen pada file beat. Dimana data node akan disimpan pada variabel nodeList dan data time akan disimpan pada variabel timeList.
3. Baris 9 menjelaskan proses perulangan yang digunakan untuk membuat objek AbstractBeat sejumlah data yang ada di nodeList.
4. Baris 11 menjelaskan proses pengoversian data pada timeList yang bertipe data string menjadi tipe data float. Hasil pengonversian ini akan digunakan untuk menentukan posisi Y pada kotak *beat*.
5. Baris 10 – 30 menjelaskan proses pembuatan objek AbstractBeat dimana nilai string pada NodeList akan menentukan kelas turunan dari AbstractBeat mana yang akan dibuat. Pada saat membuat objek AbstractBeat, konstruktor pada AbstractBeat akan melakukan proses penentuan posisi kotak beat pada layar yang ditunjukkan pada Tabel 5.14. Penjelasan implementasi penentuan posisi kotak beat, yaitu
    a. Baris 1 menjelaskan proses penghitungan posisi Y dimana didapatkan dari pengurangan antara jarak dari layar menuju ke area hit dengan hasil perkalian antara waktu dan kecepatan turunnya *beat*
    b. Baris 2 menjelaskan proses instansiasi objek Sprite
    c. Baris 3 menjelaskan proses penentuan kecepatan kotak beat pada sumbu Y sebesar 100
    d. Baris 4 menjelaskan proses penambahan objek *sprite* ke dalam *scene*.
6. Baris 31 menjelaskan proses penyimpanan objek AbstractBeat yang dibuat pada variabel tap

5.4.6. Kode pengoneksian *client* ke *server*

Permainan multiplayer pada Sarinande Angklung menggunakan konsep client server. Dimana satu *smartphone* berperan sebagai *server* dan tujuh *smartphone* lainnya berperan sebagai *client*. Protokol yang digunakan dalam server client ini adalah TCP. Implementasi pengoneksian *client* ke *server* pada sisi *client* ditunjukkan pada Tabel 5.15. dan sisi *server* ditunjukkan pada Tabel 5.16.

Tabel 5.15. Implementasi pengoneksian *client* ke *server* pada sisi *server*

Pseudo Code

```
ALGORITHM NAME: TCP Server
DECLARATION:
      TYPE server IS SocketServer
      connection IS List
      PORT IS int <- 6666
      client IS Socket
      isWaiting IS Boolean <- true
DESCRIPTION:
1     connection <- CREATE OBJECT List
2     BEGIN
3          server <- CREATE OBJECT SocketServer(PORT)
4     EXCEPTION
5        WHEN IOException e
6           WRITE e
7     END
8     WRITE CALL AndroidIP.getIPAdress()
9
10    WHILE isWaiting == TRUE DO
11       BEGIN
12          IF CALL connection.size() == 7 THEN
13              isWaiting <- false
14          ELSE
15              client <- CALL server.accept()
16              CALL connection.add(client)
17          ENDIF
18       EXCEPTION
19          WHEN IOException e
20              WRITE e
21       END
22    ENDWHILE
```

Sumber: Implementasi

Tabel 5.16. Implementasi pengoneksian *client* ke *server* pada sisi *client*

Pseudo Code

```
ALGORITHM NAME: TCP Client
DECLARATION:
      TYPE PORT IS int <- 6666
      client IS Socket
      ipServer IS String
DESCRIPTION:
1     WRITE "Enter server IP"
2     READ ipServer
3     BEGIN
4          client <- CREATE OBJECT Socket(ipServer,PORT)
5     EXCEPTION
6        WHEN IOException e
7           WRITE e
8     END
```

Sumber: Implementasi

Penjelasan kode pengoneksian *client* ke *server* pada Tabel 5.15 dan 5.16 yaitu

Dari sisi server

1. Baris 1 menjelaskan penginstansiasian objek pada variabel *conncetion* dengan tipe data List yang akan menampung semua objek *socket client* yang terkoneksi pada *server*
2. Baris 2 – 7 menjelaskan proses penginstansiasian objek ServerSocket pada variabel *server* dengan PORT 6666. Ketika objek diinstansiasi, server akan langsung mulai berjalan.
3. Baris 10 – 22 menjelaskan proses penungguan *client.* Jika client yang terkoneksi sama dengan tujuh maka proses akan diberhentikan dan jika tidak proses akan terus dilanjutkan. Ketika *client* terkoneksi akan disimpan pada variabel *connection.*

Dari sisi client

1. Baris 1 – 2 menjelaskan proses pengambilan input berupa ip server yang kemudian akan disimpan pada variabel ipserver.
2. Baris 3 – 8 menjelaskan proses pembuatan objek Socket ke ipserver pada PORT 6666. Proses ini sekaligus pengoneksian dari *client ke server.*

5.4.7. Kode penambahan *score* pada permainan *touch game*

Proses penambahan score dapat dilakukan ketika pemain telah memilih beat yang akan dimainkan. Setelah beat dipilih kemudian akan diproses dan akan menuju layar permainan. Score akan bertambah ketika angklung ditekan dengan benar sesuai dengan *beat*-nya. Implementasi kode penambahan *score* pada permainan *touch game* ditunjukkan pada Tabel 5.17.

Tabel 5.17. Implementasi kode penambahan *score*

Pseudo Code

```
ALGORITHM NAME: Penambahan score
DECLARATION:
      TYPE score IS int
      inGame IS Thread
      areaHit IS rectangle
      scene IS Scene
      factory IS TapCollector
      startTime IS long
      finishTime IS long
      elapseTime IS long
      beat IS AbstractTap
      beat2 IS AbstractTap
      listBeat IS List
      file IS GameFile
      time IS String
      maxScore IS Float
      finalScore IS Float
      rank IS String
INPUT: touch IS TouchEvent
DESCRIPTION:
 1    areaHit <- CREATE OBJECT Rectangle(0,0,720,20)
 2    file <- CALL GameFile.getInstance()
 3    time <- CALL file.getTime().get(CALL file.size() -1)
 4    finishTime <- CALL Long.parseToLong(time) + 1000
 5    listBeat <- CREATE OBJECT List
 6    factory <- CALL TapCollector.getInstance()
 7    CALL factory.tapLoads()
 8    listBeat <- CALL factory.getAllTaps()
 9    startTime <- CALL System.CurrentTimeMilis()
10    score <- 0
11    elapseTime <- CALL System.CurrentTimeMilis() - startTime
12    WHILE elapseTime <= finishTime TRUE DO
13       FOR beat : listBeat DO
14           IF CALL beat.collidesWith(areaHit) THEN
15               beat.setInAreaHit(true)
16               beat2 <- beat
17           ELSE
18               beat.setInAreaHit(true)
19           ENDIF
20       ENDFOR
21       IF   CALL   touch.getPointerID()   >   -1   AND   beat2   !=   NULL      AND
22       beat2.getInAreaHit()   ==   TRUE   AND   CALL   touch.getPointerID()   ==
23       beat.getIdNode() THEN
24           score <- score + 1
25           CALL good()
26           CALL super.executeAction(touch)
27       ELSE
28           CALL miss()
29           CALL super.executeAction(touch)
30       ENDIF
31       elapseTime <- CALL System.CurrentTimeMilis() - startTime
32    ENDWHILE
33    maxScore <- CALL factory.getAllTaps()
34    finalScore <- score/maxScore
35    IF finalScore == 1 THEN
36       rank <- "A"
37    ELSE IF finalScore >= 0.75 AND finalScore < 1 THEN
38       rank <- "B"
39    ELSE IF finalScore >= 0.5 AND finalScore < 75 THEN
40       rank <- "C"
41    ELSE IF finalScore >= 0.25 AND finalScore < 0.5 THEN
42       rank <- "D"
43    ELSE
44       rank <- "E"
45    ENDIF
```

Sumber: Implementasi

Penjelasan implementasi kode penambahan score pada Tabel 5.17. yaitu

1. Baris 1 menjelaskan proses penginstansian objek *Rectangle* pada variabel areaHit. Variabel inilah yang akan digunakan sebagai area dimana pemain harus menyentuh angklung ketika kotak beat berada di daerah area hit.
2. Baris 2 – 4 menjelaskan proses pengambilan waktu terakhir pada elemen node yang akan digunakan untuk menentukan kapan permaian harus berakhir. Waktu tersebut akan disimpan pada variabel finishTime.
3. Bari 5 - 8 menjelaskan proses untuk memanggil method tapLoads() yang digunakan untuk membuat kotak beat. Hasil dari pembuatan kotak beat akan disimpan pada listBeat.
4. Baris 9 menjelaskan proses pengambilan waktu yang diperlukan untuk menentukan awal permainan dan disimpan pada variabel startTime.
5. Baris 10 menjelaskan proses inisialiasasi pada variabel score dengan nilaii 0.
6. Baris 11 - 12 menjelaskan proses perulangan while dimana perulangan akan melakukan pengecekan apakah permainan sudah berakhir atau belum. Pengecekan dilakukan membandingkan nilai pada variabel finishTime dengan hasil pengurangan antara waktu sekarang dengan nilai pada startTime. Jika nilai finishTime sama dengan nilai hasil pengurangan maka permainan akan berhenti. Jika tidak maka permainan akan terus dilanjutkan.
7. Baris 13 - 20 menjelaskan proses pengecekan setiap kotak beat. Jika kotak beat menyentuh daerah area maka kotak beat diberi flag setInAreaHit yang bernilai *true*. Jika tidak maka nilai flag bernilai false Kotak beat yang berada di area hit maka akan disimpan pada variabel *beat*.
8. Baris 21 - 31 menjelaskan proses pengecekan *input touch* yang diterima. Jika id angklung yang disentuh sama dengan id pada kotak beat maka *score* akan bertambah dan menampilkan tulisan *good.* Jika tidak maka akan mengeluarkan nilai *miss.*
9. Baris 33 – 45 merupakan proses penenentuan *grade* dari hasil *score* yang didapat.

5.4.8. Kode pembuatan *file beat*

Selain dapat memainkan file beat, permainan ini juga menyediakan fitur untuk membuat *file beat*. Pembuatan file beat dilakukan dua tahap yaitu dengan merekam semua aktivitas pemain pada saat menyentuh angklung lalu kemudian menyimpannya dalam sebuah file dengan format XML. *File beat* yang telah dibuat dapat dimainkan pada *touch game* atau *multiplayer*. Implementasi penangkapan aktivitas angklung ditunjukkan pada Tabel 5.18.

Tabel 5.18. Implementasi kode penangkapan aktivitas angklung

Pseudo Code

```
ALGORITHM NAME: Penangkapan aktivitas angklung
DECLARATION:
      TYPE Nodelist IS List
        start_time IS Long
        elapse_time IS Long
        a IS Int
        isFinish IS boolean <- false
        node IS String
        mcf IS MakeComposerFile
        cf IS ComposerFile
INPUT: touch IS TouchEvent
DESCRIPTION:
 1    nodelist <- CREATE OBJECT List
 2    cf <- CALL ComposerFile.getInstance()
 3     file_a <- CALL cf.getFileMusik()
 4     music <- CREATE OBJECT Music(file_a)
 5     CALL music.play()
 6    startime <- CALL System.CurrentTimeMilis()
 7    WHILE isFinish == FALSE THEN
 8       IF CALL touch.getPointerID() > -1 THEN
 9          CASE CALL touch.getPointerID() OF
10               0 : node <- "do" BREAK
11               1 : node <- "re" BREAK
12               2 : node <- "mi" BREAK
13               3 : node <- "fa" BREAK
14               4 : node <- "sol" BREAK
15               5 : node <- "la"   BREAK
16               6 : node <- "si" BREAK
17               7 : node <- "do2" BREAK
18          ENDCASE
19       ENDIF
20       elapse_time <- CALL System.CurrentTimeMilis() - start_time
21       save_node <- "<node time=\""+elapse_time+"\"type=\""+type+"\"
22       node=\""+node+"\" />"
23       CALL nodelist.add(save_node)
24       node <- ""
25       READ isFinish
26    ENDWHILE
27    CALL cf.setNode(nodelist)
28    mcf <- CALL MakeComposerFile.getInstance()
29    CALL mcf.MakeFile()
```

Sumber: Implementasi

Penjelasan implementasi kode penangkapan aktivitas angklung pada Tabel 5.18., yaitu

1. Baris 1 - 2 merupakan proses penginstansiasian objek yang akan digunakan pada proses selanjutnya.
2. Baris 3 – 5 merupakan proses pembuatan file musik yang digunakan sebagai background music pada composer.
3. Baris 7 – 26 menjelaskan proses perekaman aktivitas yang dilakukan oleh pemain. Ketika pemain menyentuh angklung maka setiap angklung yang ditekan akan disimpan waktu dan nada yang ditekan. Penyimpanan waktu dan nada dikemas dalam format XML, <node time=”x” node=”do” / >. Semua data tersebut disimpan pada variabel List.
4. Baris 27 menjelaskan proses penyimpanan nilai nodelist pada objek ComposerFile.
5. Baris 28 – 29 menjelaskan proses pemanggilan method makeFile untuk memulai pembuatan *file beat.*

Setelah semua aktivitas telah ditangkap, langkah berikutnya adalah menyimpan aktivitas – aktivitas tersebut pada file dengan format XML. Implementasi pembuatan *file beat* ditunjukkan pada Tabel 5.19.

Tabel 5.19. Implementasi kode pembuatan *file beat*

Pseudo Code

```
ALGORITHM NAME: Pembuatan file beat
DECLARATION:
        TYPE save_node IS String
        title IS String
        author IS String
        file_path IS String
        beat_path IS String
        beat_folder_path IS String
        detail IS String
        xml IS String
        base_path IS String <- "sdcard/Sarinande"
        beat_file IS File
        beat_folder IS File
        file_a IS File
        file_b IS File
        out IS FileOutputStream
        fout IS OutputStreamWriter
        input IS InputStream
        output IS OutputStream
        music IS Music
        input_byte IS byte
        cf IS ComposerFile
DESCRIPTION:
 1    cf <- CALL ComposerFile.getInstance()
 2    title <- CALL cf.getTitle()
 3    author <- CALL cf.getAuthor()
```

```
4   beat_folder_path <- base_path +"\"+title
5   beat_path <- beat_folder_path +"\" + title + ".ang";
6   beat_folder <- CREATE OBJECT File(beat_folder_path)
7   beat_file <- CREATE OBJECT File(beat_path)
8   beat_folder.mkdirs()
9   beat_file.createNewFile()
10  judul <- "none"
11  IF cf.getNone == false THEN
12     judul <- cf.getFileMusik().getName()
13     beat_folder_path <- beat_folder_path + CALL
14     cf.getFileMusik().getPath()file_a <- CREATE OBJECT
15     File(cf.getFileMusik().getPath())
16     file_b <- CREATE OBJECT File(beat_folder_path)
17     input <- CREATE OBJECT InputStream(file_a)
18     output <- CREATE OBJECT OutputStream(file_b)
19     buffer <- CREATE OBJECT byte[1024]
20     WHILE (length <- CALL input.read()) > 0 DO
21        CALL output.write(buffer, 0,length)
22     ENDWHILE
23     CALL input. close()
24     CALL output.close()
25  ENDIF
26  detail="<angklung><detail><title>"+ title +
27  "</title><author>"+ author +"</author><song>" +
28  judul +"</song></detail>"
29  FOR a=0 to cf.getNode().size DO
        node <- node + nodelist.get(a)
30  ENDFOR
31  xml <- detail +"<collection>"+node+"</collection></angklung>"
32  out <- CREATE OBJECT FileOutputStream(beat_file)
33  fout <- CREATE OBJECT OutputStreamWriter(out)
34  CALL fout.write(xml)
35  CALL fout.close()
36  CALL out.close()
```

Sumber: Implementasi

Penjelasan implementasi kode pembuatan *file beat* pada Tabel 5.19. yaitu

1. Baris 1 merupakan proses penginstansiasian objek ComposerFile yang akan digunakan pada proses selanjutnya.
2. Baris 2 - 3 menjelaskan proses pengambilan data *title* dan *author* dari objek ComposerFile
3. Baris 4 – 7 menjelaskan proses instansiasi Objek file pada variabel file_beat dan folder_beat. Dimana objek file ini akan digunakan untuk pembuatan file dan folder beat di sdcard/Sarinande
4. Baris 8 – 9 menjelaskan proses pembuatan folder dan file untuk beat.
5. Baris 11 – 25 menjelaskan proses penduplikasian file audio yang ada pada sdcard/composer ke dalam *folder beat*. Proses ini akan dilakukan jika composer menggunakan *background* musik.
6. Baris 26 – 31 menjelaskan proses pengemasan semua data author, title, song, dan data yang ada pada nodelist untuk dikemas dalam bentuk XML.

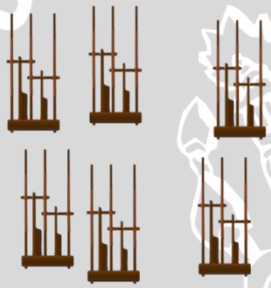
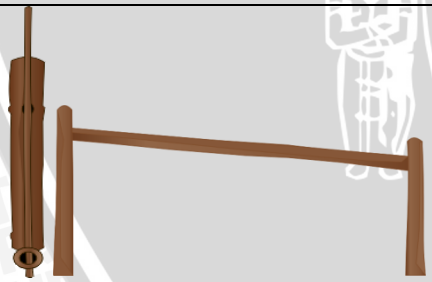
Pengemasan data dalam bentuk XML dibuat mengikuti struktur implentasi XML pada *file beat*.

7. Baris 32 – 36 menjelaskan proses penulisan variabel xml pada variabel file_beat.

5.5. Implementasi Karakter

Karakter atau elemen-elemen yang dibutuhkan pada permainan Sarinande Angklung telah dijelaskan di *game design document* pada BAB Perancangan. Implementasi dari elemen permainan ditunjukkan pada Tabel 5.20.

Tabel 5.20. Implementasi karakter pada Sarinande Angklung

Game Object	**Nama**	**Keterangan**
	Angklung satu nada	Angklung satu nada diimplementasikan dalam bentuk gambar *sprite*. Hal itu dilakukan untuk membuat efek getar ketika *smartphone* digerakkan.
	Angklung delapan nada	Angklung delapan nada dibagi menjadi dua bagian yaitu angklung dan tempat angkung.
	*touch game beat*	-
	*Multiplayer beat*	-
	*Area hit*	-

GOOD	*Good*	-
MISS	*Miss*	-
	*Hand cursor*	

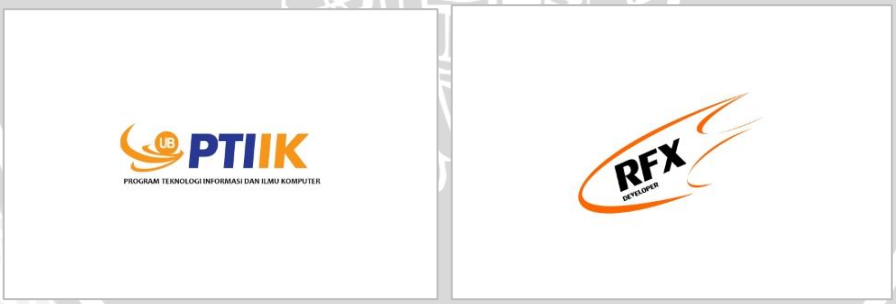
Sumber: Implementasi

5.6. Implementasi Antarmuka dan HUD

Implementasi antarmuka perangkat lunak ini terdiri dari 20 halaman. Berikut merupakan penjelasan dari implementasi antarmuka tersebut.

5.6.1. Halaman *splash screen*

Halaman *splash screen* adalah tampilan yang pertama keluar ketika aplikasi dijalankan. Pada splash screen akan keluar logo PTIIK dan logo *developer*. Tampilan halaman splash screen ditunjukkan pada Gambar 5.1.



Gambar 5. 1. Tampilan halaman *splash screen*

Sumber: Implementasi

5.6.2. Halaman *main menu*

Halaman main menu merupakan halaman yang keluar setelah halaman splash screen muncul. Halaman main menu merupakan penghubung antara fitur-fitur yang ada di game Sarinande Angklung. Terdapat enam tombol ketika halaman main menu pertama kali muncul yaitu

- ***Play***, tombol *play* adalah tombol yang digunakan untuk memulai permainan. Ketika tombol ini diklik maka akan muncul dua tombol lagi yaitu tombol *shake* dan dan tombol *touch*. Tombol *shake* digunakan untuk memainkan angklung dengan cara digoyangkan dan tombol *touch* digunakan untuk memainkan angklung dengan cara menyentuh *smartphone*. Ketika tombol *shake* diklik maka akan menuju ke halaman *menu shake* dan ketika tombol *touch* diklik akan menuju ke halaman *list beat*.
- ***Composer,*** tombol *composer* adalah tombol yang digunakan untuk menjalankan fitur *composer*.
- ***About***, tombol *about* adalah tombol yang digunakan untuk melihat *menu about*.
- ***Highscore***, tombol *highscore* adalah tombol yang digunakan untuk melihat *menu highscore*.
- ***Setting***, tombol *setting* adalah tombol yang digunakan untuk melihat *menu setting*. Tombol *setting* berbentuk roda gigi dengan warna abu-abu.
- ***How to play,*** tombol *how to play* adalah tombol yang digunakan untuk melihat *menu how to play*. Tombol *how to play* ditunjukkan pada sebuah lingkaran yang terdapat tanda seru.

Tampilan halaman *main menu* ditunjukkan pada Gambar 5.2.

Gambar 5.2. Tampilan halaman *main menu*
Sumber: Implementasi

5.6.3. Halaman *menu shake*

Halaman *menu shake* adalah halaman untuk memilih bermain angklung secara *multiplayer* atau *single player*. Ketika tombol *multiplayer* diklik maka akan menuju ke halaman *multiplayer* menu dan jika tombol *single player* diklik maka akan menuju ke halaman *single player*. Tampilan halaman menu shake ditunjukkan pada Gambar 5.3.

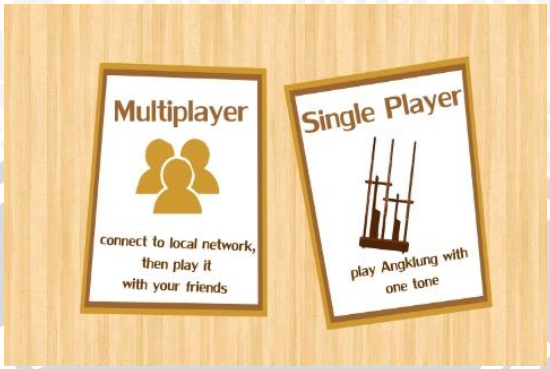
5.6.4. Halaman *list beat*

Halaman *list beat* merupakan tampilan untuk memilih *file beat*. Semua file yang berekstensi .ANG akan ditampilkan pada halaman *list beat*. Halaman *list beat* digunakan pada permainan *multiplayer* dan *touch game*. Ketika salah satu *beat* dipilih maka akan menuju ke halaman selanjutnya. Ketika bermain pada *mode touch* maka akan menuju ke halaman *menu touch* dan jika bermain *multiplayer* maka akan menuju ke halaman *server waiting*. Tampilan halaman *list beat* ditunjukkan pada Gambar 5.4.

5.6.5. Halaman *menu touch*

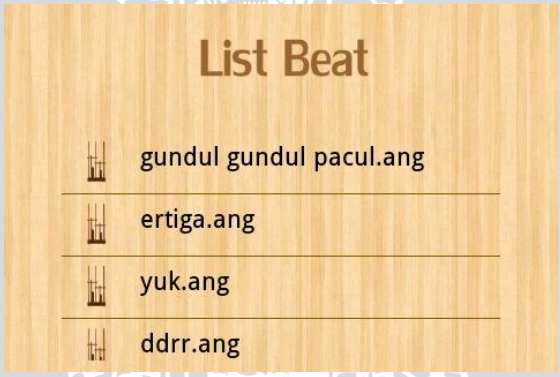
Halaman *menu touch* merupakan tampilan untuk memilih fitur pada menu *touch*. Jika tombol *record* diklik maka akan menuju ke halaman *record* dan jika

tombol *game* diklik maka akan menuju ke halaman menu *touch game*. Tampilan halaman *menu touch* ditunjukkan pada Gambar 5.5.



Gambar 5.3. Tampilan halaman *menu shake*

Sumber: Implementasi



Gambar 5.4. Tampilan halaman *list beat*

Sumber: Implementasi

Gambar 5.5. Tampilan halaman *menu touch*

Sumber: Implementasi

5.6.6. Halaman *touch game*

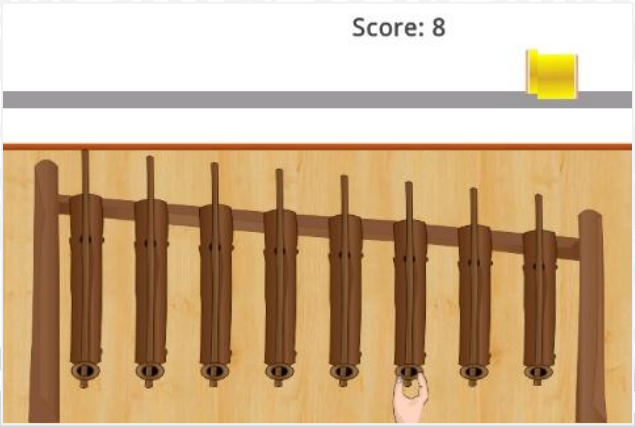
Halaman *touch game* merupakan halaman ketika bermain *beat* pada *mode touch*. Pada halaman ini terdapat delapan angklung dimana angklung paling kiri adalah angklung dengan nada do dan semakin ke kanan angklung nada semakin meninggi. Pada bagian angklung terdapat *hand cursor*, dimana *hand cursor* ini akan menunjukkan angklung yang sedang ditekan. Di bagian atas layar terdapat sistem *HUD* dimana di dalamnya terdapat panel untuk menampilkan *score* sementara. *Area hit* pada halaman *touch* game berupa bentuk persegi panjang dengan panjang 720 pixel dan tinggi bernilai 20 pixel dengan warna abu-abu. Kotak *beat* pada *touch game*, berbentuk persegi panjang bewarna kuning. Kotak *beat* akan muncul dari atas layar dan kemudian akan menghilang jika pemain menyentuh angklung dengan benar atau posisinya telah melebihi batas *area hit*. Tampilan halaman *touch* game ditunjukkan pada Gambar 5.6.

5.6.7. Halaman *record*

Halaman *record* terdiri dari dua tombol yaitu play dan record. Tombol play digunakan untuk menjalankan *demo file beat* dan tombol *record* digunakan untuk memulai perekaman *file beat*. Tampilan halaman *record* ditunjukkan pada Gambar 5.7.

5.6.8. Halaman *shake single player*

Ketika halaman *shake single player* muncul maka akan keluar sebuah gambar angklung satu nada. Gambar angklung satu nada akan membuat efek getar ketika *smartphone* digerakkan searah sumbu x. Nada yang dibunyikan pada saat pertama kali digerakkan adalah nada do. Untuk mengganti nada do ke nada lainnya, pemain harus menekan tombol keypad menu. Dengan demikian akan muncul sebuah kotak *dialog form select tone*. Setelah memilih salah satu nada maka kotak dialog akan menghilang dan nada telah berganti. Tampilan halaman *record* ditunjukkan pada Gambar 5.8.

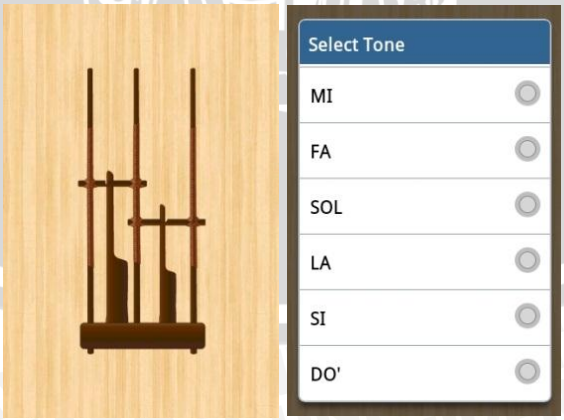


Gambar 5.6. Tampilan halaman *touch game*

Sumber: Implementasi

Gambar 5.7. Tampilan halaman *record*

Sumber: Implementasi



Gambar 5.8. Tampilan halaman *record*

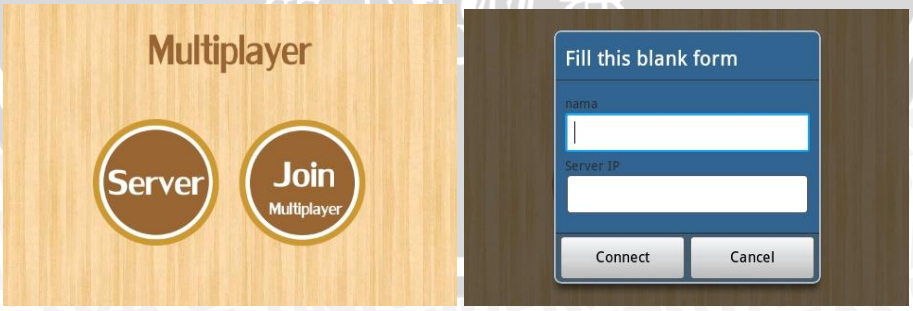
Sumber: Implementasi

5.6.9. Halaman *shake multiplayer menu*

Halaman *shake multiplayer menu* merupakan halaman yang digunakan untuk memilih peran *multiplayer*. Jika pemain ingin berperan sebagai *server* maka menekan tombol *server* dan jika pemain ingin berperan sebagai *client* maka menekan tombol join multiplayer. Ketika tombol *server* ditekan maka akan menuju halaman *list beat* dan ketika tombol *join multiplayer* ditekan maka akan muncul kotak dialog *form* pengoneksian ke *server*. Kotak dialog ini terdiri dari *form* nama dan *form ip server*. Jika tombol *connect* ditekan maka akan terkoneksi ke *server* dan menuju halaman *waiting server* dan jika tombol *cancel* ditekan maka akan menuju ke halaman *multiplayer menu*. Tampilan halaman *shake multiplayer menu* ditunjukkan pada Gambar 5.9.

5.6.10. Halaman *waiting client*

Halaman *waiting client* merupakan halaman yang digunakan untuk menjalankan proses pengoneksian dari *client* ke *server*. Halaman ini akan muncul pada pemain yang berperan sebagai *server* setelah memilih *file beat*. Ketika sebuah *client* terkoneksi ke *server* maka akan muncul sebuah pesan bahwa *client* dengan ip tertentu telah terkoneksi. Proses ini akan berhenti jika jumlah *client* telah mencapai tujuh atau pemain menekan tombol *stop*. Setelah proses ini selesai, maka halaman akan langsung diarahkan menuju halaman konfigurasi nada *client*. Tampilan halaman waiting client ditunjukkan pada Gambar 5.10.



Gambar 5.9. Tampilan halaman *multiplayer menu*

Sumber: Implementasi

Gambar 5.10. Tampilan halaman *waiting client*

Sumber: Implementasi

Gambar 5.11. Tampilan halaman konfigurasi nada *client*

Sumber: Implementasi

5.6.11. Halaman konfigurasi nada client

Halaman konfigurasi nada *client* merupakan halaman untuk menampilkan semua pemain yang terkoneksi ke *server* dan mengatur nada yang dimainkan oleh setiap pemain. Ketika pemain menekan salah satu *client* maka akan muncul sebuah kotak dialog untuk memilih nada yang dimainkan. Setelah nada dipilih, maka pemain akan menekan tombol *"Ok"* untuk menyimpan konfigurasi dan tombol cancel untuk membatalkan konfigurasi. Selain menyimpan konfigurasi nada, proses yang dijalankan setelah menekan tombol *"Ok"* adalah mengirim pesan dari *server* ke *client* dengan ip yang telah dipilih sebelumnya. Pesan yang dikirim adalah berupa judul *file beat* dan nada yang akan dimainkan oleh *clien*t. Setelah pengonfigurasian nada *client* telah dilakukan maka pemain harus menekan

tombol *keypad menu* untuk memunculkan kotak dialog memulai permainan. Tampilan halaman konfigurasi nada *client* ditunjukkan pada Gambar 5.11.

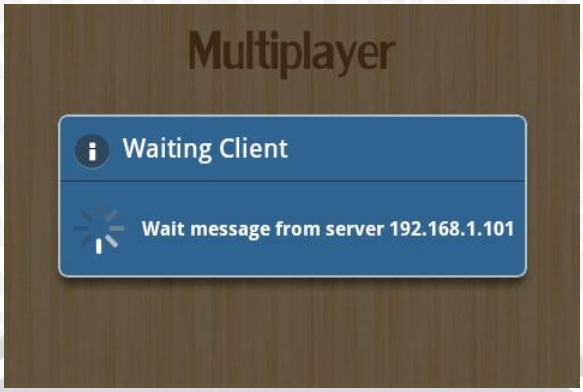
5.6.12. Halaman *waiting server*

Halaman *waiting server* merupakan halaman pada pemain *client* yang digunakan untuk menampilkan proses menunggu pesan dari *server*. Setelah pesan diterima oleh *client* maka akan dilakukan proses pengecekan *file beat*. Jika *file beat* ada pada *SD card* maka akan memunculkan kotak dialog memulai permainan dan jika tidak ada maka akan memunculkan kotak dialog untuk mengakhiri permainan *multiplayer*. Tampilan pada *waiting server* ditunjukkan pada Gambar 5.12.

5.6.13. Halaman *multiplayer game*

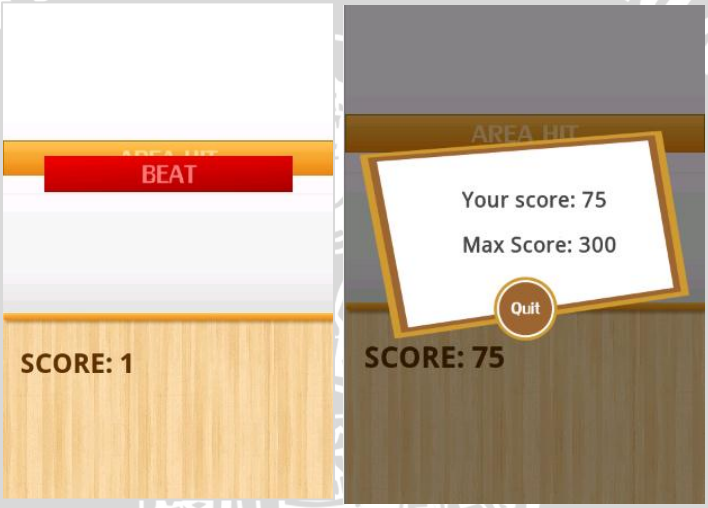
Halaman *multiplayer game* merupakan halaman permainan pada *multiplayer*. Ketika halaman pertama kali muncul, pemain yang berperan sebagai *client* akan muncul sebuah kotak dialog dengan tulisan *wait*. Sedangkan pemain yang berperan sebagai *server* akan muncul sebuah kotak dialog untuk memulai permainan. Pemain harus menekan tombol *yes* untuk memulai permainan. Ketika tombol *yes* ini ditekan, maka akan melakukan pengiriman pesan secara broadcast ke semua pemain. Dengan demikian semua pemain baik *server* maupun *client*, akan memulai permainan secara bersamaan.

Pada saat permainan dimulai akan musik pada file *beat* akan dimainkan dan akan muncul kotak *beat* dari atas layar. Ketika kotak *beat* menyentuh area hit maka pemain harus menggerakkan *smartphone* searah sumbu x untuk mendapatkan *score*. Di akhir permainan selesai, akan muncul sebuah kotak dialog yang menampilkan jumlah kotak *beat* yang benar. Tampilan halaman *multiplayer game* ditunjukkan pada Gambar 5.13.



Gambar 5.12. Tampilan halaman *waiting server*

Sumber: Implementasi



Gambar 5.13. Tampilan halaman *multiplayer game*

Sumber: Implementasi

5.6.14. Halaman daftar musik

Halaman daftar musik merupakan halaman untuk menampilkan semua file *audio* pada direktori *sdcard/Sarinande/Composer*. *File audio* ini selanjutnya akan digunakan sebagai *background* musik dalam membuat *file beat.* Pada daftar music ini terdapat tulisan *none* pada baris pertama. Ketika pemain memilih *none*, maka pemain tidak menggunakan *background* musik saat membuat *file beat.* Tampilan halaman daftar musik ditunjukkan pada Gambar 5.14.

5.6.15. Halaman *composer*

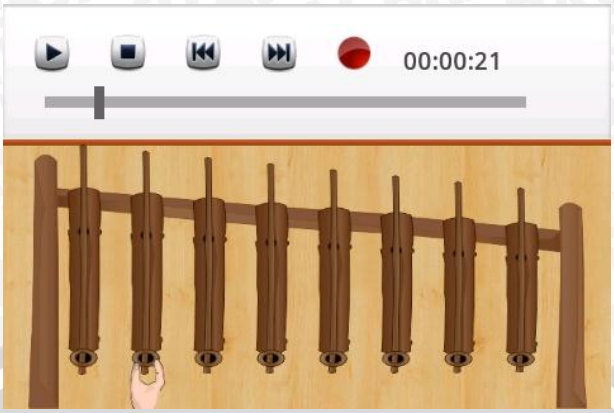
Halaman *composer* merupakan halaman yang digunakan untuk membuat *file beat.* Pada bagian atas layar *composer* terdapat panel yang terdiri dari beberapa tombol yang dapat mengatur *background* music. Panel ini tidak dapat digunakan ketika pemain telah memulai membuat *file beat.* Untuk memulai proses pembuatan *file beat,* pemain harus menekan *keypad menu* sehingga akan muncul kotak dialog memulai membuat *file beat.* Ketika tombol *yes* ditekan oleh pemain, maka proses telah dimulai dan gambar *microphone* pada panel berubah menjadi lingkaran berwarna merah. Setiap angklung yang disentuh akan disimpan oleh sistem. Ketika proses pembuatan telah selesai, maka pemain harus menekan *keypad menu* sehingga akan muncul kotak dialog untuk menyimpan *file beat.* Ketika tombol *yes* ditekan oleh pemain, maka akan menuju ke halaman *composer form*. Tampilan halaman *composer* ditunjukkan pada Gambar 5.15.

5.6.16. Halaman *form composer*

Halaman *form composer* merupakan halaman yang digunakan untuk memasukkan data-data yang diperlukan untuk membuat *file beat.* Ketika pertama kalai halaman *form composer* muncul akan keluar kotak dialog yang berisi *form title* dan *form author*. Setelah semuanya telah terisi pemain akan menekan tombol *save* untuk melakukan proses pembuatan *file beat.* Tampilan halaman *form composer* ditunjukkan pada Gambar 5.16.

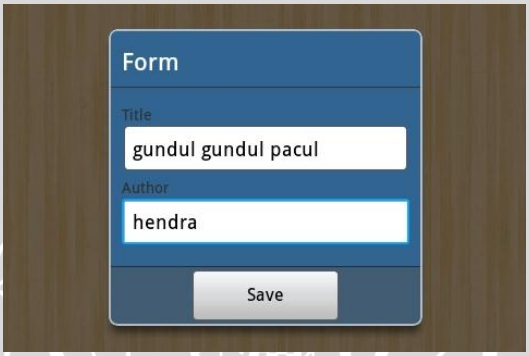
Gambar 5.14. Tampilan halaman daftar musik

Sumber: Implementasi



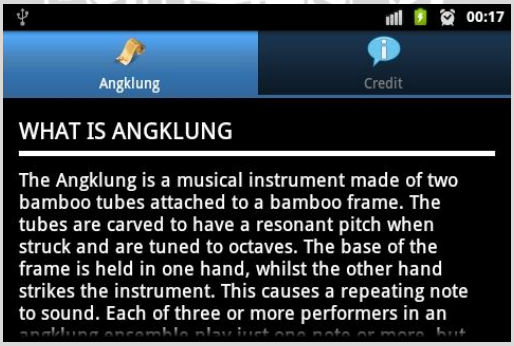
Gambar 5. 15. Tampilan halaman composer

Sumber: Implementasi



Gambar 5.16. Tampilan halaman *form composer*

Sumber: Implementasi



Gambar 5.17. Tampilan halaman *menu about*

Sumber: Implementasi

5.6.17. Halaman *menu about*

Halaman *menu about* merupakan halaman untuk menampilkan informasi mengenai angklung dan informasi mengenai pembuat *game* Sarinande Angklung.

Pada tampilan *menu about* dua *tab* yaitu *tab* angklung dan *tab credit*. *Tab* - angklung digunakan untuk melihat informasi mengenai angklung dan *tab credit* digunakan untuk melihat informasi pembuat *game*. Tampilan pada halaman *menu about* ditunjukkan pada Gambar 5.17.

5.6.18. Halaman *menu high score*

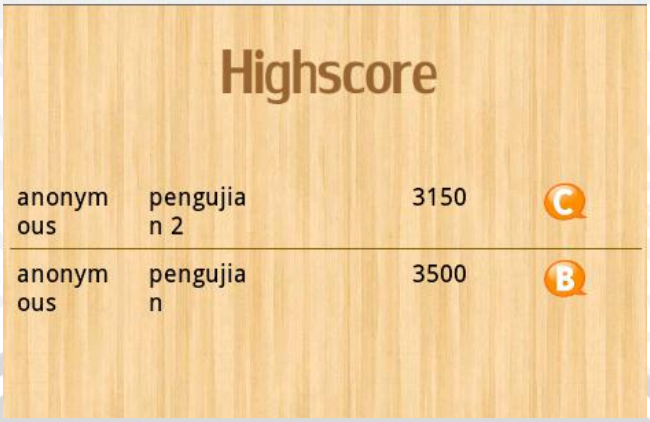
Halaman *menu high score* merupakan halaman yang digunakan untuk melihat *score* tertinggi dari setiap *beat* yang dimainkan. Dalam tampilan *high score,* terdapat empat kolom. Kolom pertama digunakan untuk menampilkan nama pemain, kolom kedua digunakan untuk menampilkan judul *beat,* kolom ketiga digunakan untuk menampilkan nilai yang diperoleh, dan kolom keempat digunakan untuk menampilkan *grade* yang diperoleh. Tampilan halaman *menu high score* ditunjukkan pada Gambar 5.18.

5.6.19. Halaman *menu setting*

Halaman *menu setting* merupakan halaman yang digunakan untuk melakukan konfigurasi terhadap sistem. Adapun yang dikonfigurasikan adalah pengaturan *volume* suara, *volume* musik, dan pengaktifan vibrasi. Tampilan halaman *menu setting* ditunjukkan pada Gambar 5.19.

5.6.20. Halaman menu *how to play*

Halaman *how to play* merupakan halaman yang digunakan untuk menunjukkan cara bermain Sarinande angklung. Cara bermain yang ditunjukkan adalah cara bermain *single player, multiplayer, touch game, record,* dan *composer*. Tampilan pada halaman menu *how to play* ditunjukkan pada Gambar 5.20.

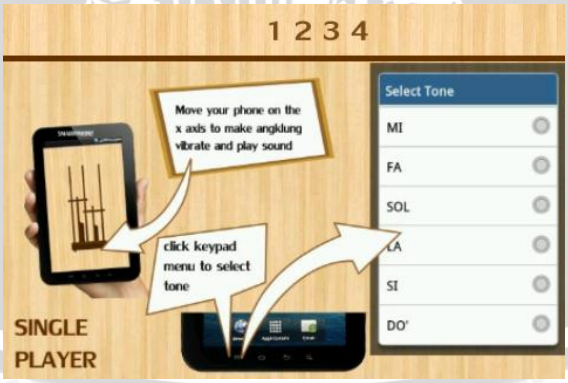


Gambar 5.18. Tampilan halaman *menu high score*

Sumber: Implementasi

Gambar 5.19. Tampilan halaman *menu setting*

Sumber: Implementasi



Gambar 5.20. Tampilan halaman menu *how to play*

Sumber: Implementasi