BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metodologi

Metode penelitian yang akan digunakan dalam penelitian ini antara lain adalah sebagai berikut:

1. Studi Literatur

Pada tahap ini dilakukan studi mengenai referensi-referensi yang berkaitan dengan penelitian yang akan dilakukan penulis, seperti pembuatan aplikasi Android, metode *Hidden Markov Model*, proses ekstraksi *file* berbasis bunyi dengan MFCC, dan berbagai konsep pendukung lainnya. Referensi-referensi tersebut dapat berupa jurnal, artikel, buku, dan lain-lain.

2. Perancangan dan pembangunan aplikasi

Perancangan dan pembangunan aplikasi dibagi menjadi dua tahapan. Pertama adalah membuat aplikasi dapat memproses *file* berbasis bunyi. Pada tahapan ini aplikasi akan dapat membuka *file explorer* untuk mengakses *file* bertipe wav dan dapat melakukan ekstraksi pada *file* tersebut untuk mendapatkan MFCC yang akan digunakan pada HMM nantinya. Terakhir perancangan dan pembuatan metode *Hidden Markov Model* yang akan digunakan untuk mempelajari data yang sudah diekstraksi. Aplikasi dibuat dengan Android dengan dasar bahasa pemograman Java.

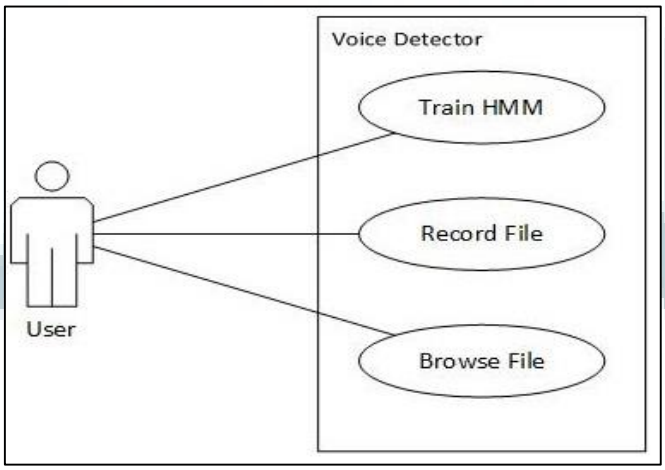
3. Uji Coba dan Evaluasi

Setelah aplikasi selesai dibuat, maka penulis akan melakukan uji coba pada aplikasi yang disertai dengan evaluasi terhadap hasil yang didapatkan dari uji coba yang dilakukan. Uji coba ini bertujuan untuk mencari *bug* yang masih ada pada aplikasi tersebut guna memperbaiki aplikasi tersebut.

3.2 Perancangan Sistem

3.2.1 *Use Case* Diagram

Use case diagram adalah gambaran *graphical* dari beberapa atau semua aktor dan interaksi di antara komponen-komponen yang menggambarkan sistem yang akan dibangun. *Use case* diagram digunakan untuk menggambarkan bagaimana pandangan orang luar atau aktor yang terlibat dari luar sistem. Diagram ini menggambarkan fungsionalitas sistem dan bagaimana sistem berinteraksi dengan bagian yang tidak ada di dalam sistem. Gambar di bawah ini adalah *use case* diagram dari aplikasi yang dibuat oleh penulis.



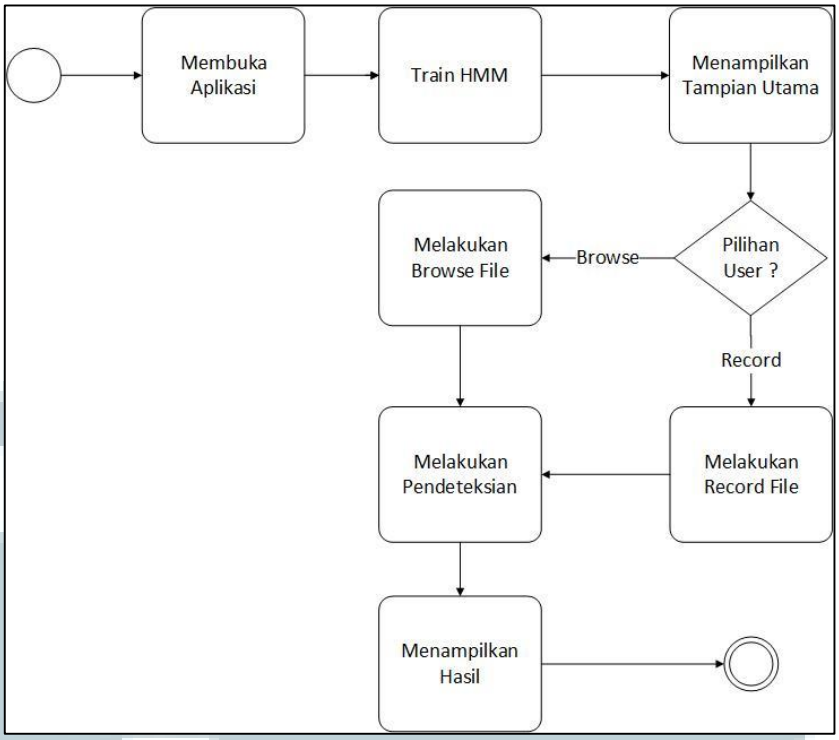
Gambar 3.1 *Use Case* Diagram

Pada diagram di atas terdapat satu *actor* saja yaitu *User*. *User* dapat melakukan tiga aktivitas, yaitu *train* HMM, *Record File*, dan *Browse File*. Untuk *train* HMM dilakukan pada saat *user* membuka aplikasi maka secara otomatis aplikasi akan membuka *file* train.txt yang berisi data MFCC yang sudah disediakan sebelumnya oleh peneliti. Setelah membaca isi dari *file* tersebut maka aplikasi akan melakukan proses pelatihan secara otomatis. Pada aktivitas *Record File user* dapat melakukan perekaman bunyi yang kemudian aplikasi dapat langsung mendeteksi bunyi pada *file* tersebut. Pada aktivitas Browse *File user* dapat memilih *file* yang sudah ada pada *device* sebelumnya untuk dideteksi. Setelah memilih *file* maka user dapat langsung melakukan pendeteksian menggunakan aplikasi.

3.2.2 *Activity* Diagram

Activity diagram merupakan sebuah diagram yang berisi tentang hal-hal yang menggambarkan langkah-langkah dan proses dari suatu sistem. *Activity* diagram yang dibuat oleh penulis mewakili bebearpa proses, proses-proses tersebut adalah proses merekam suara oleh *user*, proses membuka *file* oleh pengguna, dan proses *train* HMM.

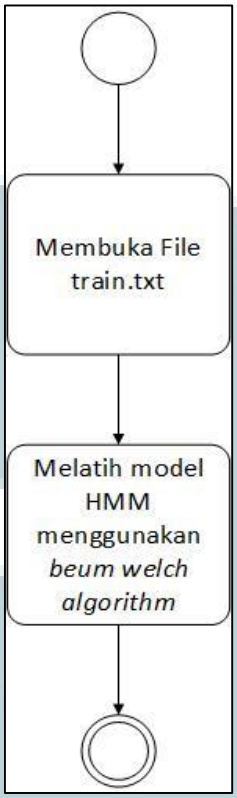
A. *Activity* Diagram Umum



Gambar 3.1 *Activity* Diagram Umum

Gambar di atas menunjukan bagaimana aktifitas yang ada pada aplikasi yang dibuat oleh penulis secara umum. Dimulai dari *user* membuka aplikasi kemudian aplikasi akan melakukan proses *training* HMM lalu aplikasi akan menampilkan tampilan utama dari aplikasi. Setelah itu *user* dapat memilih untuk melakukan *browse file* atau melakukan *record file* yang akan dilanjutkan dengan proses pendeteksian terhadap *file* yang sudah direkam atau yang sudah dipilih dan terakhir aplikasi akan menampilkan hasil dari pendeteksian terhadap *file* tersebut.

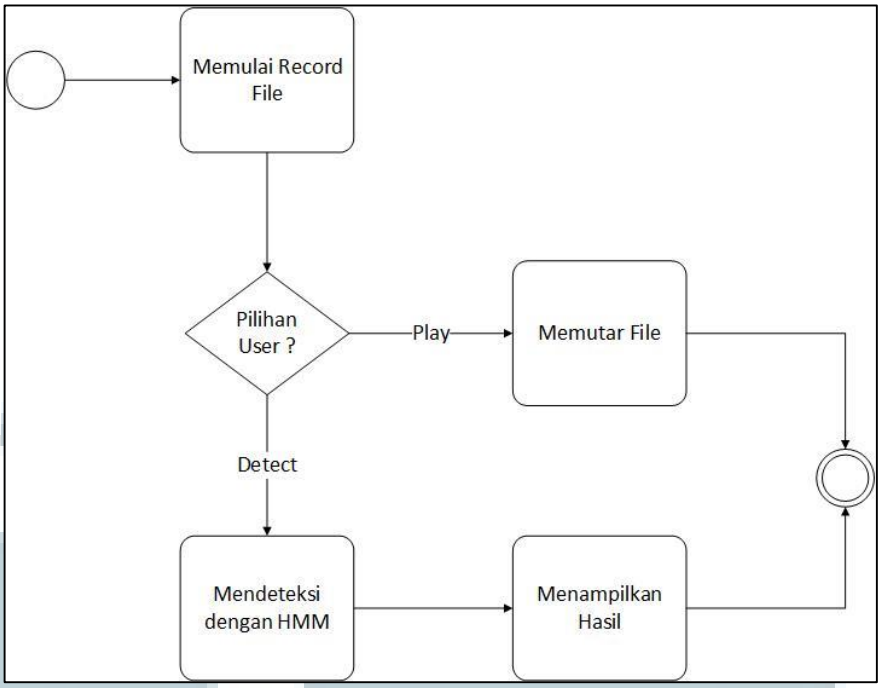
B. *Activity* Diagram Proses *Train* HMM



Gambar 3.2 *Activity* Diagram *train* HMM

Proses ini dimulai dengan membuka *file* yang berisi data-data MFCC dari *file-file* yang sudah disimpan sebelumnya untuk digunakan pada *training* aplikasi yang digunakan. Kemudian *train* dilakukan oleh aplikasi dengan menggunakan Baum Welch Algorithm. Proses *training* ini tidak dilihat oleh *user*, dikarenakan aplikasi secara otomatis akan melakukan proses *training* ketika aplikasi pertama kali dijalankan oleh *user* dan proses ini dijalankan sebelum memasuki *activity* utama.

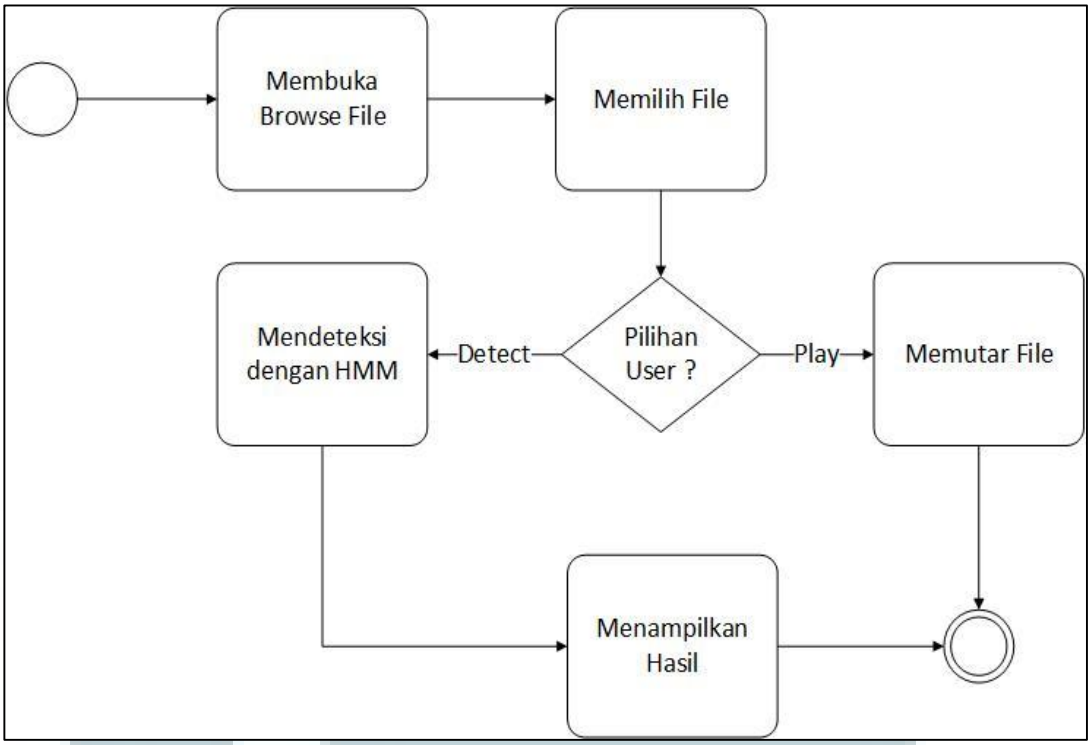
C. *Activity* Diagram Proses *Record file*



Gambar 3.3 *Activity* Diagram *record file*

Proses ini dimulai dengan melakukan perekaman terhadap suara ketukan yang ingin dideteksi, kemudian setelah merekam maka *user* bisa langsung melakukan pendeteksian atau memutar terlebih dahulu *file* yang sudah tersimpan. Jika melakukan pendeteksian, maka akan ditampilkan hasil dari pendeteksian tersebut pada aplikasi. Untuk bagian *record file*, penulis membuat agar aplikasi ini akan mengganti rekaman yang dibuat dengan rekaman terbaru. Jadi jika *user* melakukan dua kali rekaman, maka rekaman yang akan bisa diputar atau dideteksi adalah rekaman kedua yang direkam oleh *user*.

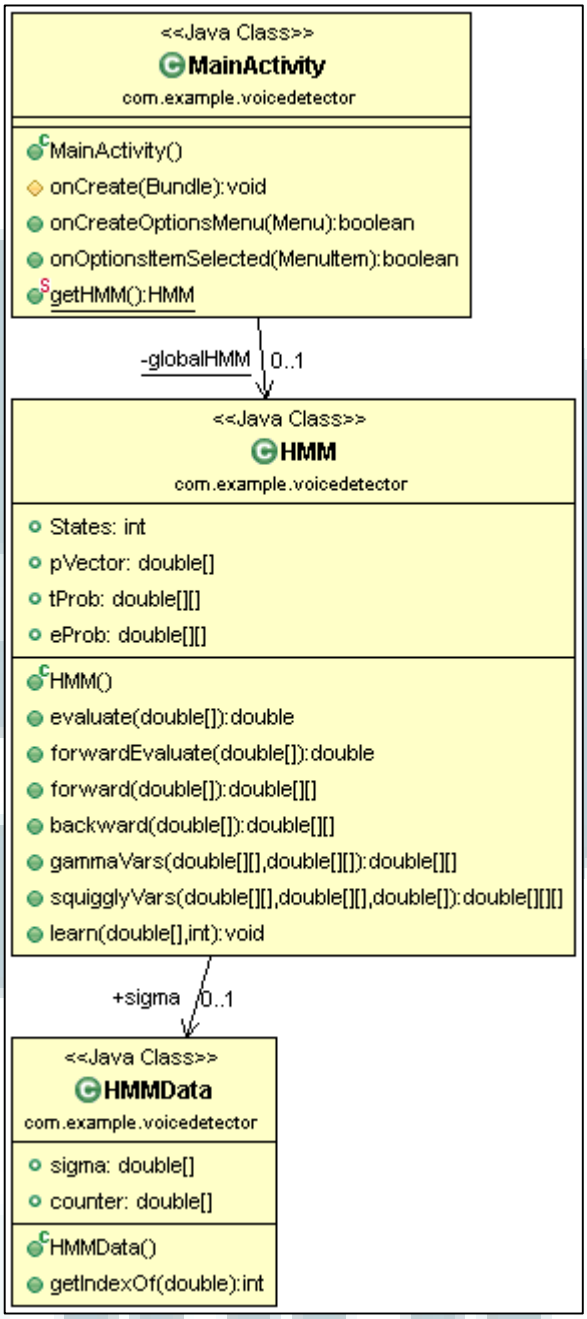
D. *Activity* Diagram Proses *Browse* File



Gambar 3.4 *Activity* Diagram *browse file*

Proses ini dimulai dengan melakukan *browse file* pada *device* yang digunakan untuk memilih *file* mana yang akan digunakan untuk proses pendeteksian kematangan semangka. Setelah memilih *file* yang diinginkan maka *user* dapat memutar *file* tersebut terlebih dahulu untuk memastikan bahwa *file* yang dipilih sudah benar. *User* dapat mennyudahi proses dengan menekan tombol *back* atau *exit* dari aplikasi atau *user* dapat melanjutkan proses dengan melakukan pendeteksian menggunakan metode HMM yang kemudian akan ditampilkan hasil dari pendeteksian tersebut pada aplikasi.

3.2.3 *Class* Diagram



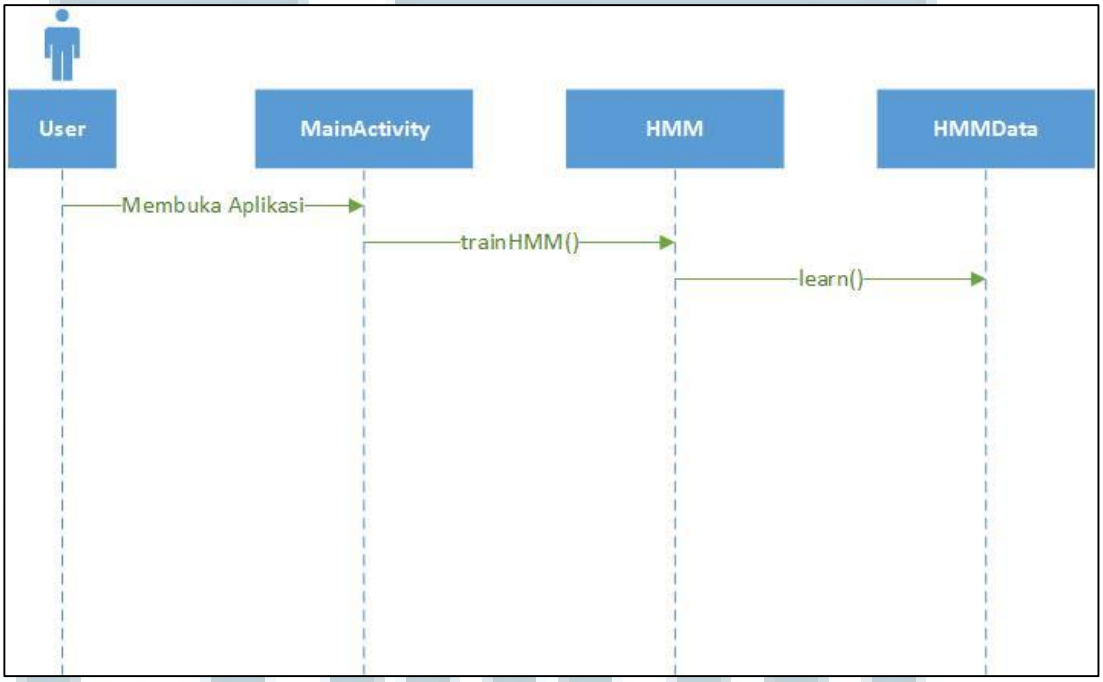
Gambar 3.5 *Class* Diagram

Gambar di atas adalah *class diagram* dari Hidden Markov Model yang akan digunakan pada aplikasi ini. *Class* HMMData adalah *class* yang akan digunakan untuk menyimpan data yang diperlukan untuk menghitung HMM nantinya.

3.2.4 *Sequence* Diagram

Sequence diagram untuk proses penggunaan HMM pada aplikasi. Proses dimulai dengan *user* memilih *file* yang ingin dideteksi atau melakukan proses perekaman pada bunyi yang ingin dideteksi. Kemudian aplikasi akan melakukan konversi *file* menjadi *byte* dan *double* pada *object audio*, lalu melakukan per hitungan MFCC dengan *library* (MFCC by Aldebaro Klautau) MFCC yang tersedia dan kemudian melakukan pendeteksian dengan menggunakan metode *evaluate* pada *object* HMM.

A. *Sequence* Diagram Proses *Train* HMM

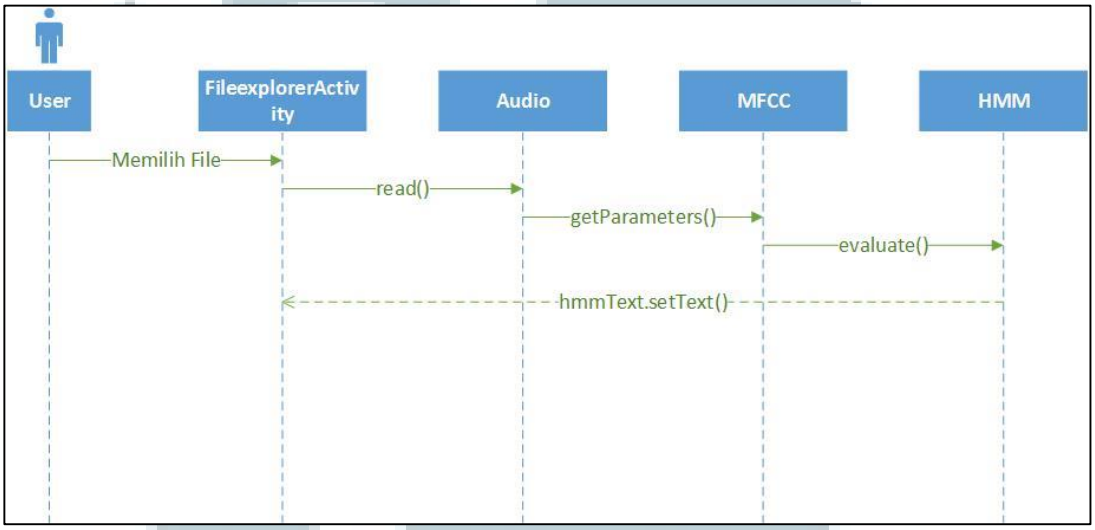


Gambar 3.6 *Sequence* Diagram *Train* HMM

Gambar di atas menunjukkan *sequence* diagram untuk proses *train* HMM pada aplikasi. Proses dimulai ketika *user* membuka aplikasi. Saat aplikasi terbuka, aplikasi akan menjalankan fungsi trainHMM() yang akan melakukan semua yang diperlukan untuk *train* HMM. Hal tersebut adalah membuka dan membaca data *file*

train.txt yang ada pada *device*, membuat *object* HMM, dan kemudian memanggil fungsi learn() yang ada pada HMM dengan parameter isi dari *file* train.txt yang sudah dibaca sebelumnya. Kemudian proses *train* HMM akan menggunakan HMMData sebagai *variable* yang berisi data-data HMM.

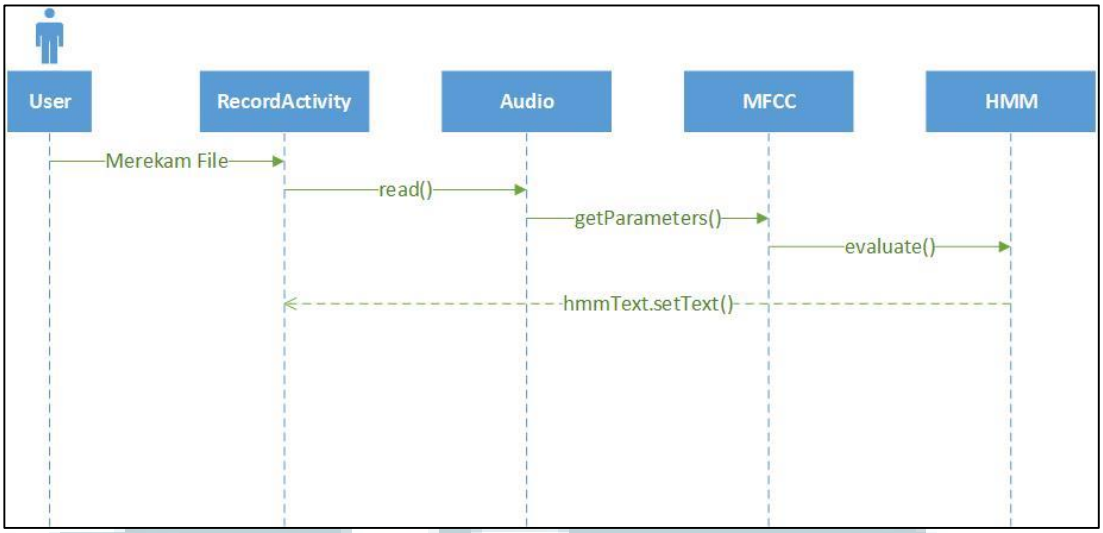
B. *Sequence* Diagram Proses *Browse File*



Gambar 3.7 *Sequence* Diagram *Browse File*

Gambar di atas menunjukkan *sequence* diagram untuk proses *browse file* pada aplikasi. Proses dimulai ketika *user* memilih *file* untuk di-*browse* pada FileexplorerActivity dan kemudian melakukan pendeteksian HMM setelah memilih *file* yang tersedia pada *device*. Proses pendeteksian dilakukan dengan mengubah *file* menjadi bentuk *double* dengan *object audio*, kemudian memasukkan ke dalam *library* (MFCC by Aldebaro Klautau) MFCC yang digunakan untuk menjadi deret angka MFCC, dan pada akhirnya menggunakan *object* HMM untuk melakukan pendeteksian pada deretan MFCC yang sudah dihasilkan.

C. *Sequence* Diagram Proses *Record File*



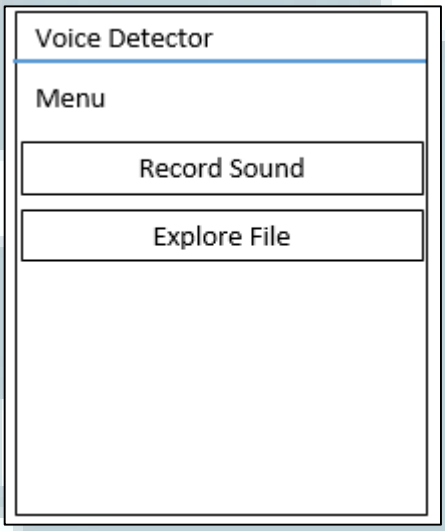
Gambar 3.8 *Sequence* Diagram *Record File*

Gambar di atas menunjukkan *sequence* diagram untuk proses *record file* pada aplikasi. Proses dimulai dengan *user* melakukan proses perekaman terhadap bunyi yang ingin dideteksi dengan melakukan ketukan sebanyak tiga kali pada buah semangka yang ingin dideteksi dan aplikasi akan merekam bunyi yang dihasilkan. Proses yang terjadi selebihnya mirip dengan proses *browse file*, yaitu *user* langsung dapat melakukan pendeteksian yang dilakukan oleh aplikasi dengan *object Audio*, *library* (MFCC by Aldebaro Klautau) MFCC, dan *object* HMM. Kemudian langsung mendapatkan hasilnya pada *EditText* yang disediakan.

3.4 Desain Antarmuka

Pada bagian ini, ditampilkan beberapa sketsa antarmuka sistem dari sisi pengguna.

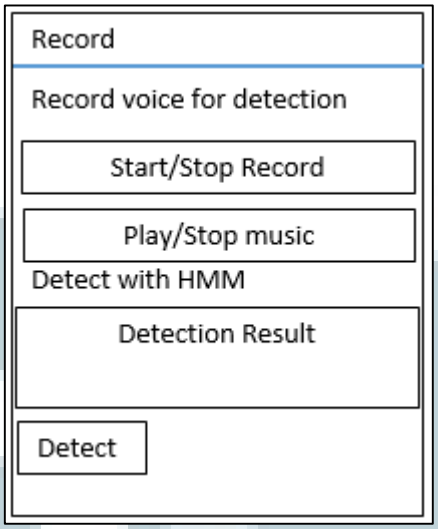
3.4.1 *Activity* Utama



Gambar 3.9 Desain *activity* utama

Activity ini adalah *activity* yang pertama ditampilkan pada *user* ketika membuka aplikasi ini. Terdapat dua tombol yang tersedia untuk melakukan berbagai hal yang berkaitan dengan aplikasi. Tombol pertama berguna untuk membuka *activity* baru yang berguna untuk melakukan perekaman suara sekaligus melakukan pendeteksian *file* tersebut menggunakan metode HMM yang sudah dibuat pada aplikasi. Kemudian tombol kedua digunakan untuk melakukan *browse file* jika ingin melakukan pendeteksian menggunakan *file* yang sudah ada terlebih dahulu pada *device* yang digunakan.

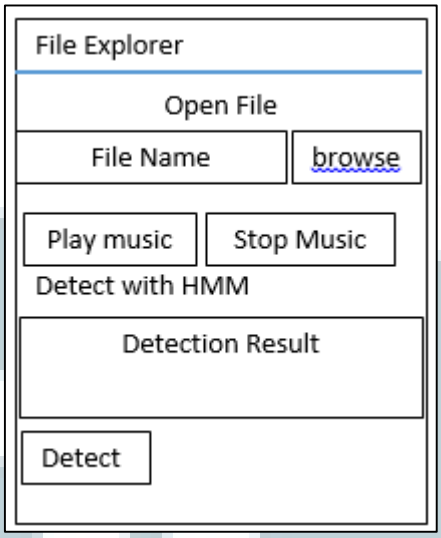
3.4.2 *Activity Record File*



Gambar 3.10 Desain *activity record file*

Activity ini dapat diakses dengan menekan tombol *Record Sound* yang ada pada *activity* utama. *Acitivity* ini berguna untuk melakukan perekaman *file* yang bisa dilaksanakan dengan menekan tombol *Start Record*. Jika sudah selesai melakukan perekaman, *user* tinggal menekan tombol yang sama untuk menyelesaikan perekaman. Kemudian dapat langsung melakukan pendeteksian dengan menekan tombol *Detect* dan aplikasi akan menampilkan hasil pendeteksian pada *view* yang disediakan.

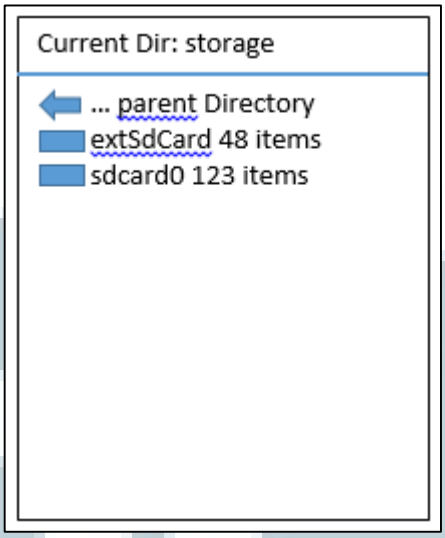
3.4.3 *Activity File Explorer*



Gambar 3.11 Desain *activity file explorer*

Activity ini dapat diakses ketika *user* menekan tombol *explore file* pada *activity* utama. *Activity* ini berguna untuk melakukan pendeteksian setelah *file* yang sudah ada sebelumnya dipilih oleh *user*. Tombol *browse* digunakan untuk memanggil *activity* yang digunakan untuk melakukan eksplorasi pada *device* untuk memilih *file* yang digunakan untuk pengecekan, tombol *play music* digunakan untuk memutar *file* yang sudah dipilih, tombol *stop music* digunakan untuk memberhentikan *file* yang sudah diputar, dan tombol *detect* digunakan untuk melakukan pendeteksian pada *file* yang sudah dipilih.

3.4.4 *Activity Browse File*



Gambar 3.12 Desain *activity browse file*

Activity ini dapat digunakan dengan menekan tombol *browse* pada *activity file explorer* yang digunakan untuk memilih *file* yang akan digunakan untuk melakukan pendeteksian. Ketika *user* sudah memilih, maka akan kembali kepada *activity file explorer* dengan *file* yang nanti akan digunakan untuk pendeteksian pada *activity file explorer*.