RANCANG BANGUN DIGITAL *AUDIO SHARE* DAN *PLAYER* DENGAN KEAMANAN DATA *ADVANCED ENCRYPTION STANDARD* BERBASIS ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Yuyon Alif Masruri
125150201111034

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGESAHAN

RANCANG BANGUN DIGITAL AUDIO SHARE DAN PLAYER DENGAN KEAMANAN DATA ADVANCED ENCRYPTION STANDARD BERBASIS ANDROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Yuyon Alif Masruri
NIM. 125150201111034

Skripsi ini telah diuji dan dinyatakan lulus pada
09 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Eng. Herman Tolle, S.T, M.T	Aswin Suharsono, S.T, M.T
NIP. 19740823 200012 1 001	NIK. 201102 840919 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 09 Agustus 2016

Yuyon Alif Masruri

NIM. 125150201111034

KATA PENGANTAR

Dengan nama Allah SWT Yang Maha Pengasih dan Penyayang. Segala puji bagi -Nya karena atas rahmat dan hidayah-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul "RANCANG BANGUN *DIGITAL AUDIO SHARE* DAN *PLAYER* DENGAN KEAMANAN DATA *ADVANCED ENCRYPTION STANDARD* BERBASIS ANDROID". Shalawat serta salam atas junjungan besar kita Nabi Muhammad S.A.W. beserta keluarga dan para sahabat sekalian.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan baik lahir maupun batin selama penulisan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan rasa hormat dan terima kasih penulis kepada :

1. Ayah Imam Muslim, S.Ag, Ibu Sringatin dan seluruh keluarga besar atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta yang senantiasa tidak hentinya memanjatkan doa, mengajarkan kedisiplinan dan semangat demi terselesaikannya skripsi ini.
2. Bapak Dr. Eng. Herman Tolle, S.T, M.T dan Bapak Aswin Suharsono, S.T, M.T selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan bijaksana mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Drs. Mardji, M.T, dan Bapak Edy Santoso, S.Si, M.Kom selaku Dekan, Wakil Dekan 1, Wakil Dekan 2 dan Wakil Dekan 3 Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D dan Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Jurusan Teknik Informatika dan Kepala Prodi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Ibu Indriati, S.T, M.T selaku dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
6. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
7. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
8. Sahabat - sahabatku Angkatan 2012 Teknik Informatika Alpha, Kelas TIF-E tahun 2012, Teman Seperjuangan BIOS CLOUD 2015-2016 dan semua sahabat di luar sana, terimakasih atas segala bantuannya selama menempuh studi di Teknik Informatika Universitas Brawijaya. M. Faizal

Sukma Dika, S.Kom, Danu Akbar, S.Kom, Aldi Yudha, S.Kom, Nurrizka Ajeng Utami, S.Kom, Febry Agistya I, S.Kom, Nancy Ramadhani, S.Kom, Wahyu Argo Prabowo, S.Kom, Rezqi Usy Andini, S.Kom, Sifasani Habibie, S.Kom, Qoirul Kotimah, S.Kom, Angga Bagus Arbiyant, S.Kom, Faris Dinar, S.Kom, Bagus Kesuma Atmaja, S. Kom, Ilham Rolis, S.Kom, dan Tantri Dwi Yuniar, S.Pd, terima kasih dukungan dan semangat yang selalu diberikan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya.

9. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Hanya doa yang bisa penulis berikan semoga Allah SWT memberikan pahala serta balasan kebaikan yang berlipat ganda. Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan masih jauh dari sempurna. Untuk itu, saran dan kritik yang membangun sangat penulis harapkan. Semoga skripsi ini membawa manfaat bagi penulis maupun pihak lain yang menggunakan dalam penelitian selanjutnya.

Malang, 09 Agustus 2016

Yuyon Alif Masruri
yuyonallif@gmail.com

ABSTRAK

Yuyon Alif Masruri. 2016. : RANCANG BANGUN DIGITAL *AUDIO SHARE* DAN *PLAYER* DENGAN KEAMANAN DATA *ADVANCED ENCRYPTION STANDARD* BERBASIS ANDROID. Skripsi Program Studi Teknik Informatika, Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya.
Dosen Pembimbing : Dr. Eng. Herman Tolle, S.T, M.T dan Aswin Suharsono, S.T, M.T

Teknologi *digital* audio memudahkan masyarakat mengakses musik maupun audio yang ingin dimiliki dan didengar. Namun dengan kemudahan tersebut, memungkinkan pelanggaran hak cipta dari karya seni terhadap masyarakat dalam menyebarluaskan teknologi *digital audio* yang berada dibawah perlindungan. Kemudahan penggunaan perangkat bergerak memungkinakan setiap pengguna bisa berbagi antar file kemana saja, tidak terlepas juga dengan file *digital audio* hanya dengan menggunakan jaringan internet. Oleh karena itu, perlu dikembangkan perangkat lunak perangkat bergerak yang memiliki keamanan data didalamnya. Dengan menggunakan teknologi kriptografi *AES,* diharapkan dapat memberikan keamanan dan kenyamanan dakam penggunaan perangkat lunak. Perancangan perangkat lunak Audiosafe menggunakan tahap perancangan fungsional berdasarkan hasil spesifikasi kebutuhan dan perancangan non fungsional atau keamanan data dengan merancang kinerja dari sisi kriptografi yaitu enkripsi dan dekripsi pada perangkat lunak berdasarkan 3 parameter yaitu waktu performansi kriptografi, keaslian data dan *security*. Implementasi dari perangkat lunak dikembangkan berdasarakan perancangan padah tahap sebelumnya dan diimplementasikan dalam bahasa pemrograman Java dengan development tool *Android Developer Tool* yaitu *Android Studio* dan diimplementasikan dalam *device smartphone Android.* Pengujian menggunakan teknik *Whitebox, Blackbox* dan Pengujian keamanan data. Dari hasil pengujian unit dapat disimpulkan bahwa algoritma dan unit perangkat lunak telah memenuhi kebutuhan fungsional. Hasil pengujian validasi dapat disimpulkan bahwa implementasi dan fungsionalitas memenuhi kebutuhan perangkat lunak. Pada pengujian waktu performansi kriptografi dapat disimpulkan bahwa performansi kriptografi menghasilkan waktu yang *tolerable* berdasarkan APDEX dengan syarat minimal ukuran data. Pengujian keaslian data menghasilkan kesimpulan bahwa ukuran dan konten *file* dapat kembali ke bentuk semula ketika mengalami proses dekripsi dari *file cipher digital audio* hasil enkripsi. Pengujian *security* dapat disimpulkan bahwa berdasarkan proses pada perangkat lunak, *file digital audio* tidak dapat diputar ketika belum diberikan *privilege* dari *user* pengunggah *digital audio.*

Kata kunci: *AES,* kriptografi*, digital audio,* perangkat lunak perangkat bergrerak*.*

ABSTRACT

Yuyon Alif Masruri. 2016. : *ANDROID DIGITAL BOOK SHARE AND PLAYER DEVELOPMENT WITH DATA SECURITY USING ADVANCED ENCRYPTION STANDARD.*

***Advisor* : Dr. Eng. Herman Tolle, S.T, M.T *and* Aswin Suharsono, S.T, M.T**

Digital audio technology makes people easier to access music and audio that would be possessed and be heard. However, with such ease to access music and video allows people to do the copyright infringement by means distribute music and audio illegally. As we know, mobile device has feature to share files anywhere include digital audio just by using internet. Because of many illegal spread of digital audio, people needs a software that has security to save data. By using AES cryptographic technology is expected to provide security and pleasantly use software in daily. Audiosafe software design used a functional stage to design it based on specification requirement result and non-functional design by designing cryptography technology, encryption and decryption. Encryption and decryption performance in Audiosafe has been implemented by using three parameters such as cryptography performance, authenticity of data, and security. The software implementation was developing based on design of the previous development and it implements with Java programming language and uses Android Developer Tools named Android Studio to make an android software in smartphone android. Software testing in this software is using whitebox, blackbox, and testing data security. The result of testing unit can be concluded that implementation and functionality has qualified of software requirement. In performance testing can be concluded that performance time of cryptography tolerable based on APDEX with a minimum data size. Authenticity of data testing has a conclusion that size and file content can be returned to its original when it processes decryption from file cipher digital in encryption audio result. Security testing can be concluded that audio file cannot be played when the user doesn't have privilege of digital audio uploader

Keyword: AES, cryptography, digital audio, mobile application.

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR GAMBAR xii
DAFTAR KODE SUMBER xiv
DAFTAR TABEL xv
DAFTAR LAMPIRAN xvii
PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan Masalah 3
1.6 Sistematika Pembahasan 4
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Kajian Pustaka 5
2.2 Dasar Teori 6
2.2.1 Media Player 6
2.2.2 *Library* Kriptografi AES dalam bahasa *Java* 6
2.2.3 JavaScript Object Notation 8
2.2.4 MP3 8
BAB 3 METODOLOGI 9
3.1 Studi Literatur 9
3.2 Analisis Kebutuhan Perangkat Lunak 10
3.3 Perancangan Perangkat Lunak 10
3.4 Implementasi Perangkat Lunak 11
3.5 Pengujian dan Analisis Hasil Perangkat Lunak 11

3.6 Penarikan Kesimpulan 11
BAB 4 SPESIFIKASI KEBUTUHAN DAN PERANCANGAN Perangkat lunak 12
4.1 Spesifikasi Kebutuhan Perangkat Lunak 12
4.1.1 Deskripsi Umum Perangkat Lunak 12
4.1.2 Defenisi, Istilah dan Singkatan 13
4.1.3 Aturan Penomoran 13
4.1.4 Fungsi Utama Perangkat Lunak 14
4.1.5 Spesifikasi Kebutuhan Perangkat Lunak 14
4.1.6 Detil Kebutuhan Fungsional 15
4.1.7 Detil Kebutuhan Non Fungsional 17
4.1.8 Interface Requirement Spesification 17
4.1.9 Analisa Aktor 18
4.1.10 Diagram Use Case 19
4.1.11 Skenario Use Case 20
4.2 Perancangan Perangkat Lunak 25
4.2.1 Perancangan Arsitektur Sistem 25
4.2.2 Perancangan Keamanan Data 26
4.2.3 Perancangan Aktifitas Sistem 29
4.2.4 Perancangan Interaksi Antar Objek 38
4.2.5 Perancangan Klas 48
4.2.6 Perancangan Basis Data 49
4.2.7 Perancangan Antarmuka 50
BAB 5 IMPLEMENTASI PERANGKAT LUNAK 56
5.1 Spesifikasi Sistem 56
5.1.1 Spesifikasi Perangkat Keras 56
5.1.2 Spesifikasi Perangkat Lunak 57
5.2 Batasan – Batasan Implementasi 57
5.3 Implementasi *Class* 58
5.4 Implementasi Kode Program 60
5.4.1 Implementasi Algoritma Mengunggah Audio 60
5.4.2 Implementasi Algoritma Memutar Audio 64
5.5 Implementasi Kriptografi 67

5.5.1 Implementasi Enkripsi *AES*-256 67
5.5.2 Implementasi Dekripsi *AES*-256 68
5.5 Implementasi Basis Data 70
5.6 Implementasi Antarmuka 72
5.6.1 Implementasi Antarmuka Melakukan *Login* 72
5.6.2 Implementasi Antarmuka Melakukan *Register* 72
5.6.3 Implementasi Antarmuka Mengunggah Audio 73
5.6.4 Implementasi Antarmuka Melihat Daftar Audio 73
5.6.5 Implementasi Antarmuka Mengunduh Audio 74
5.6.6 Implementasi Antarmuka Melihat Daftar Audio yang Terunduh 75
5.6.7 Implementasi Antarmuka Memutar Audio 75
5.6.8 Implementasi Antarmuka Menghapus Audio di *Server* 76
5.6.9 Implementasi Antarmuka Menghapus Audio di Daftar yang Terunduh 76
5.6.10 Implementasi Antarmuka Memberikan *Privilege Member* 77
5.6.11 Implementasi Antarmuka Melihat Profil 77
BAB 6 PENGUJIAN DAN ANALISIS 78
6.1 Perangkat Keras Pengujian 78
6.2 Rencana Pengujian 78
6.3 Pengujian Unit 79
6.3.1 Pengujian Unit Mengunggah Audio 79
6.3.2 Pengujian Unit Memutar Audio 82
6.4 Pengujian Validasi 85
6.4.1 Kasus Uji Pengujian Validasi 86
6.4.2 Hasil Dan Analisa Pengujian Validasi 90
6.5 Pengujian Keamanan Data 92
6.5.1 Pengujian Performasi Waktu Eksekusi Kriptografi 92
6.5.2 Pengujian Keaslian Data 96
6.5.3 Pengujian *Security* 99
6.6 Analisis Hasil Pengujian 100
BAB 7 KESIMPULAN DAN SARAN 102
7.1 Kesimpulan 102
7.2 Saran 103

DAFTAR PUSTAKA..... 104
LAMPIRAN 106

DAFTAR GAMBAR

Gambar 2.1 Diagram blok AES 7
Gambar 3.2 Diagram Metodologi Penelitian 9
Gambar 4.1 Deskripsi Umum Sistem 13
Gambar 4.2 *Use Case* Diagram AUDIOSAFE 19
Gambar 4.3 Perancangan Sistem 25
Gambar 4.4 *Diagram* Alir Perancangan Enkripsi Kriptografi *AES* 27
Gambar 4.5 *Diagram* Alir Perancangan Dekripsi Kriptografi *AES* 28
Gambar 4.6 *Activity Diagram* Melakukan Register 29
Gambar 4.7 Activity Diagram Melakukan Login 30
Gambar 4.8 *Activity Diagram* Mengunggah *Audio* 31
Gambar 4.9 *Activity Diagram* Melihat Daftar *Audio* 31
Gambar 4.10 *Activity Diagram* Mengunduh *Audio* 32
Gambar 4.11 *Activity Diagram* Melihat Daftar *Audio* yang Terunduh 33
Gambar 4.12 *Activity Diagram* Memutar *Audio* 34
Gambar 4.13 *Activity Diagram* Menghapus *Audio* di Server 35
Gambar 4.14 *Activity Diagram* Menghapus *Audio* di *Audio* yang Terunduh 36
Gambar 4.15 *Activity Diagram* Memberikan Privilege 37
Gambar 4.16 *Activity Diagram* Melihat Profil 37
Gambar 4.17 *Activity Diagram* Melakukan Logout 38
Gambar 4.18 *Sequence Diagram* Melakukan Register 39
Gambar 4.19 Sequence *Diagram* Melakukan Login 40
Gambar 4.20 *Sequence Diagram* Mengunggah *Audio* 40
Gambar 4.21 *Sequence Diagram* Melihat Daftar *Audio* 41
Gambar 4.22 *Sequence Diagram* Mengunduh *Audio* 42
Gambar 4.23 *Sequence Diagram* Melihat Daftar *Audio* yang Terunduh 43
Gambar 4.24 *Sequence Diagram* Memutar *Audio* 44
Gambar 4.25 *Sequence Diagram* Menghapus *Audio* di Server 45
Gambar 4.26 *Sequence Diagram* Menghapus *Audio* di *Audio* yang Terunduh 46
Gambar 4.27 *Sequence Diagram* Memberikan Privilege 46
Gambar 4.28 *Sequence Diagram* Melihat Profil 47

Gambar 4.29 *Sequence Diagram* Melakukan Logout 48
Gambar 4.31 Relasi Antar Tabel Database *Audiosafe* 50
Gambar 4.32 Navigation Map *Audio*safe 51
Gambar 4.33 Antarmuka Melakukan Login 51
Gambar 4.34 Antarmuka Melakukan Login 52
Gambar 4.35 Antarmuka Mengunggah *Audio* 52
Gambar 4.36 Antarmuka Melihat Daftar *Audio* 53
Gambar 4.37 Antarmuka Mengunduh *Audio* 53
Gambar 4.38 Antarmuka Melihat Daftar *Audio* Terunduh 53
Gambar 4.39 Antarmuka Memutar *Audio* 54
Gambar 4.40 Antarmuka Menghapus *Audio* di Server 54
Gambar 4.41 Antarmuka Menghapus *Audio* di Daftar yang Terunduh 55
Gambar 4.42 Antarmuka Memberikan Privilege 55
Gambar 4.43 Antarmuka Melihat Profil 55
Gambar 5.1 Antarmuka Melakukan *Login* 72
Gambar 5.2 Antarmuka Melakukan Register 73
Gambar 5.3 Antarmuka Mengunggah Audio 73
Gambar 5.4 Antarmuka Melihat Daftar Audio 74
Gambar 5.5 Antarmuka *Mengunduh Audio* 74
Gambar 5.6 Antarmuka Melihat Daftar Audio yang Terunduh 75
Gambar 5.7 Antarmuka Memutar Audio 75
Gambar 5.8 Antarmuka Menghapus Audio di Server 76
Gambar 5.9 Antarmuka Menghapus Audio di Daftar yang Terunduh 76
Gambar 5.10 Antarmuka Memberikan *Privilege Member* 77
Gambar 5.11 Antarmuka Melihat Profil 77
Gambar 6.1 *Flow graph* algoritma mengunggah audio 81
Gambar 6.2 *Flowgraph* algoritma memutar audio 84
Gambar 6.3 Grafik waktu proses enkripsi *AES* terhadap besar *file* 95
Gambar 6.4 Grafik waktu proses dekripsi *AES* terhadap besar *file* 96

DAFTAR KODE SUMBER

Kode Sumber 2.1 Import Kriptografi di JCE 7
Kode Sumber 2.2 Inisiasi kriptografi AES 8
Kode Sumber 5.1 Algoritma *Use case* Mengunggah Audio 60
Kode Sumber 5.2 Implementasi Mengunggah Audio 62
Kode Sumber 5.3 Algoritma *Use case* Memutar Audio 64
Kode Sumber 5.4 Implementasi Memutar Audio 65
Kode Sumber 5.5 Implementasi Enkripsi *AES-256* 67
Kode Sumber 5.6 Implementasi Dekripsi *AES*-256 68
Kode Sumber 6.1 *Pseudocode* Algoritma Mengunggah Audio 79
Kode Sumber 6.2 *Pseudocode* Algoritma Memutar Audio 82

DAFTAR TABEL

Tabel 4.1 Daftar Defenisi, Istilah dan Singkatan 13
Tabel 4.2 Aturan Penamaan dan Penomoran 13
Tabel 4.3 Kebutuhan Fungsional 15
Tabel 4.4 Kebutuhan Non Fungsional 17
Tabel 4.5 Interface Requirement Spesification 17
Tabel 4.6 Analisa Aktor 18
Tabel 4.7 Use Case Scenario Melakukan Regitrasi 20
Tabel 4.8 Use Case Scenario Melakukan Login 20
Tabel 4.9 Use Case Scenario Mengunggah *Audio* 21
Tabel 4.10 Use Case Scenario Melihat Daftar *Audio* 21
Tabel 4.11 Use Case Scenario Mengunduh *Audio* 22
Tabel 4.12 Use Case Scenario Melihat Daftar *Audio* yang Terunduh 22
Tabel 4.13 Use Case Scenario Memutar *Audio* 23
Tabel 4.14 Use Case Scenario Menghapus *Audio* di Server 23
Tabel 4.15 Use Case Scenario Menghapus *Audio* yang Terunduh 23
Tabel 4.16 Use Case Scenario Memberikan Privilege 24
Tabel 4.17 Use Case Scenario Melihat Profil 24
Tabel 4.18 Use Case Scenario Melakukan Logout 25
Tabel 4.19 Basisdata Tabel List*Audio* 49
Tabel 4.20 Basisdata Tabel *Member* 49
Tabel 4.21 Basisdata Tabel *Statusplay* 50
Tabel 5.1 Spesifikasi Perangkat Keras Mesin Pengembang 56
Tabel 5.2 Spesifikasi Perangkat Keras Mesin Instalasi 56
Tabel 5.3 Spesifikasi Perangkat Lunak Mesin Pengembang 57
Tabel 5.4 Spesifikasi Perangkat Lunak Mesin Instalasi 57
Tabel 5.5 Implementasi Klas 58
Tabel 5.6 Implementasi Tabel Members 71
Tabel 5.7 Implementasi Tabel Listaudio 71
Tabel 5.8 Implementasi Tabel Statusplay 71
Tabel 6.1 Kasus uji pengujian unit algoritma mengunggah audio 82

Tabel 6.2 Kasus uji pengujian unit algoritma memutar audio 85
Tabel 6.3 Kasus Uji Melakukan *Register* 86
Tabel 6.4 Kasus Uji Melakukan *Login* 86
Tabel 6.5 Kasus Uji Mengunggah Audio 86
Tabel 6.6 Kasus Uji Melihat Daftar Audio 87
Tabel 6.7 Kasus Uji Mengunduh Audio 87
Tabel 6.8 Kasus Uji Melihat Daftar Audio yang Terunduh 88
Tabel 6.9 Kasus Uji Memutar Audio 88
Tabel 6.10 Kasus Uji Menghapus Audio diServer 88
Tabel 6.11 Kasus Uji Menghapus Audio yang Terunduh 89
Tabel 6.12 Kasus Uji Memberikan Privilege 89
Tabel 6.13 Kasus Uji Melihat Profil 90
Tabel 6.14 Kasus Uji Melakukan *Logout* 90
Tabel 6.15 Hasil pengujian validasi 90
Tabel 6.16 Kasus uji Pengujian Performansi *AES* 92
Tabel 6.17 Skenario uji performansi kecepatan kriptografi 93
Tabel 6.18 Hasil pengujian performansi Enkripsi *AES* 94
Tabel 6.19 Hasil pengujian performansi Dekripsi *AES* 94
Tabel 6.20 Kasus uji dan data uji pengujian keaslian data 97
Tabel 6.21 Skenario uji pengujian Keaslian Data *AES* 97
Tabel 6.22 Hasil uji *AES* keaslian data *file size* 97
Tabel 6.23 Hasil uji *AES* keaslian data *file content checksum* 98
Tabel 6.24 Data uji pengujian security *AES* 99
Tabel 6.25 Skenario uji pengujian security *key* collision *AES* 99
Tabel 6.26 Hasil pengujian *security* 100

DAFTAR LAMPIRAN

LAMPIRAN A PERANCANGAN KELAS DIAGRAM 10606
A.1 Perancangan Klas Diagram Fungsi Login, Logout dan Main 10606
A.2 Perancangan Klas Diagram Fungsi Online Audio 10707
A.3 Perancangan Klas Diagram Fungsi Downloaded Audio 10808
A.4 Perancangan Klas Diagram Fungsi Profile 10909
A.5 Perancangan Klas Diagram Fungsi Privilege User 11010

PENDAHULUAN

Dalam bab ini akan dibahas mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan skripsi.

1.1 Latar Belakang

Indonesia memiliki keanekaragaman seni budaya. Salah satu jenis seni adalah musik. Musik adalah bunyi yang diatur menjadi pola yang dapat menyenangkan telinga atau mengkomunikasikan perasaan atau suasana hati. Musik mempunyai ritme, melodi, dan harmoni yang memberikan kedalaman dan memungkinkan penggunaan beberapa instrumen atau bunyi-bunyian (Oxford University, 2005). Pada awalnya, musik dimainkan secara langsung sebagai salah satu bentuk seni pertunjukan di dalam masyarakat, dinyanyikan dan didengarkan dalam aktivitas keseharian. Seiring dengan perkembangan zaman, teknologi memungkinkan musik untuk direkam dan dihadirkan kembali dalam bentuk suara tanpa perlu menghadirkan pemusik. Kehadiran teknologi rekaman suara tersebut menghasilkan bentuk yang dikenal dengan istilah *digital audio*. *Digital audio* merupakan penyajian dari suara asli yang berhasil direkam.

Dengan kemunculan teknologi *digital audio*, maka masyarakat dimudahkan untuk mengakses musik-musik yang ingin dimiliki. Tidak hanya musik, tetapi juga rekaman suara dari smartphone dapat dibuat dengan teknologi *digital audio*. Akan tetapi, persebaran musik dan rekaman suara dalam bentuk *digital audio* memungkinkan masyarakat mengakses dan menyebarluaskan hingga melanggar hak cipta dari karya seni musik tersebut. Hal tersebut dikenal dengan istilah pembajakan. Pembajakan telah banyak terjadi di Indonesia dan sangat memperihatinkan. Menurut data ASIRI, angka pembajakan tercatat sekitar 96% sejak tahun 2008 (Pramudyanto, 2013). Berdasarkan banyaknya kasus yang tercatat, dibutuhkan suatu keamanan penyebaran informasi berupa kriptografi. Kriptografi adalah salah satu cara untuk mengamankan data atau teknik yang prinsip utamanya adalah untuk melindungi keamanan informasi (Zhou & Tang, 2011). Dengan adanya kriptografi, maka kerahasiaan informasi akan tetap bisa terjaga tanpa dengan bisa mudah dicuri, dipastikan dikirimkan dengan benar, dan data dapat terjaga dari adanya pencurian data dari pihak ketiga.

Dengan kemudahan smartphone saat ini, setiap pengguna bisa berbagi antar file kemana saja, tidak terlepas juga dengan file *digital audio*. Pengguna mampu membagikan secara luas file *digital audio* hanya dengan menggunakan jaringan internet. Kemudahan penyebaran file *digital audio* membuat khawatir pencipta *digital audio* digital dikarenakan semakin luas penyebaran file tersebut berarti semakin marak pembajakan hak cipta audio. Untuk menanggulanginya diperlukan sistem audio player yang memiliki keamanan data sehingga mampu memberikan keamanan pada proses berbagi file *digital* audio tersebut. Setiap pengguna dibebaskan dalam melakukan pengunduhan *file digital audio* namun pada saat file

tersebut akan diputar, sistem tidak akan langsung memutarnya dikarenakan adanya keamanan data pada sistemnya.

Beberapa penelitian yang terkait konsep kriptografi *digital audio* dilakukan oleh Bismita Gadanayak, Chittaranjan Pradhan dan Neha Baranwal (2011) tentang Pengamanan Bagian MP3. Penelitian ini melakukan pengamanan bagian mp3 dengan menggunakan teknik enkripsi AES dan menghasilkan proses yang cepat dan lebih aman pada perangkat lunak music e-commerce (Gadanayak & Pradhan, 2011). Penelitian terkait selanjutnya adalah Muhammad Faizal Sukma Dika, Herman Tolle dan Mahendra Data (2015) tentang perangkat lunak *digital book reader* berbasis android dengan keamanan data. Penelitian ini mengamanakan proses client-server dengan menggunakan enkripsi RSA dan untuk file *digital book* berformat PDF dienkripsi dengan AES (Dika, Tolle, & Data, 2015).

Selain itu penelitian terkait adalah Noni Endriani (2014) tentang enkripsi pada perangkat lunak SMS berbasis Android. Penelitian ini melakukan kriptografi pada SMS dengan menggunakan enkripsi algoritma AES dan sudah diimplementasikan dalam perangkat bergerak (Endriani & Utami, 2014). Dan penelitian terakhir yaitu Voni Yuniati, Gani Indriyanta, dan Antonius Rachmat C (2009) tentang enkripsi dan dekripsi semua file dengan kriptografi. Penelitian ini menggunakan algoritma AES-256 pada semua jenis file yang diujinya, dan didapatkan hasil dari analisisnya bahwa waktu yang diperlukan untuk proses enkripsi dan dekripsi tidak sama, dikarenakan adanya pemakaian resource komputer (Yuniati, Indriyanta, & C, 2009).

Berdasarkan penelitian-penelitian yang telah dipaparkan di atas, kriptografi ternyata mampu diterapkan untuk mengamankan data berupa *digital audio*. Oleh karena itu, penulis merancang dan membangun sistem *digital audio* share dan player dengan kemanan data. Pada penelitian ini digunakan Kriptografi dengan algoritma AES (*Advanced Encryption Standard*) dan akan diimplementasikan pada perangkat bergerak (*mobile application*) untuk mengamankan data *digital audio* pada perangkat lunak *music player*, serta mengukur pengaruh pengimplementasian keamanan data dari segi waktu eksekusi kriptografi, perbandingan isi konten data enkripsi dan dekripsi (keaslian data), dan menguji tingkat *security* dengan cara dapat atau tidaknya dari persetujuan *user* untuk memutar satu musik digital tersebut terhadap perangkat lunak itu sendiri.

1.2 Rumusan Masalah

Berdasarkan permasalan yang diangkat pada bagian latar belakang, maka rumusan masalah dikhususkan pada:

1. Bagaimana rancangan dan hasil implementasi perangkat lunak pemutar dan berbagi *digital audio* dengan pengamanan data menggunakan kriptografi *Advanced Encryption Standard* (AES) berbasis perangkat bergerak Android?
2. Bagaimana kinerja perangkat lunak ditinjau dari sisi waktu eksekusi kriptografi, keaslian data dan *security* data?

1.3 Tujuan

Adapun tujuan umum yang dapat diperoleh dari penelitian ini adalah Memberikan sarana kepada pengguna untuk dapat memutar dan berbagi *digital audio* secara aman dengan menggunakan perangkat lunak yang menggunakan keamanan data *Advanced Encryption Standard*.

Adapun tujuan khusus yang dapat diperoleh dari penelitian ini adalah:

1. Mengimplementasi kemananan data menggunakan *Advanced Encrytion Standard* pada perangkat lunak perangkat bergerak.
2. Mengetahui pengaruh ditinjau dari sisi waktu eksekusi kriptografi, keaslian data kriptografi *Advanced Encryption Standard* dan *security user* terhadap kinerja sistem ini.

1.4 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah:

1. Memahami penerapan dan pengaruh Algoritma *AES* untuk melindungi keamanan data *digital audio* dalam penelitian ini.
2. Bagi pencipta musik, mengurangi tingkat pembajakan *digital audio* miliknya dengan penyebarluasan skala terbatas.
3. Bagi penikmat, meningkatkan tingkat kepercayaan dalam menghargai pencipta musik dengan akses musik yang memiliki keamanan data dan penyebarluasan skala yang terbatas tanpa takut akan pembajakan.

1.5 Batasan Masalah

Agar permasalahan yang dirumuskan lebih terfokus, maka perancangan dan implementasi perangkat lunak ini dibatasi dalam hal:

1. Implementasi Kriptografi menggunakan *Advanced Encryption Standard* 256 bit (AES-256) pada environtment *Java Cryptography* (*javax.crypto*) JDK 7.
2. *File digital audio* yang dienkripsi adalah *file digital audio* berformat MP3.
3. Proses pengujian dilakukan pada daerah yang ter-*cover* jaringan UMTS atau Wi-Fi.
4. Pengujian pengaruh kriptografi terhadap perangkat lunak, diukur pada aspek:
   i. Waktu Eksekusi Kriptografi, dengan paramater kecepatan eksekusi pengamanan data kriptografi.
   ii. Keaslian data, dengan paramater membandingkan konten data dan tingkat kesuksesan dari hasil enkripsi dan dekripsi kriptografi.
   iii. *Security*, dengan parameter dilakukan pengujian dapat atau tidaknya *digital audio* diputar ketika *privilege* kepada member diijinkan maupun ditolak.

1.6 Sistematika Pembahasan

Sistematika penulisan laporan tugas ini adalah:

BAB 1 PENDAHULUAN

Menguraikan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Menguraikan tentang kajian pustaka terkait dengan penelitian yang telah ada seperti penelitian tentang keamanan data, kriptografi, dan perangkat lunak perangkat bergerak. Dasar teori dan referensi yang diperlukan untuk mendasari pengembangan perangkat lunak sistem.

BAB 3 METODE PENELITIAN

Menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan tugas yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian serta analisis serta pengambilan kesimpulan dan saran.

BAB 4 PERANCANGAN SISTEM

Menjelaskan tentang perancangan secara detail mengenai perancangan sistem dalam pembuatan perangkat lunak. Uraian perancangan sistem ini meliputi perancangan data mengenai input dan output sistem, perancangan proses mengenai bagaimana sistem akan bekerja dengan proses-proses tertentu, maupun perancangan antarmuka dalam desain dan implementasi yang akan digunakan dalam pembuatan sistem ini.

BAB 5 IMPLEMENTASI SISTEM

Menjelaskan implementasi perangkat lunak, implementasi algoritma, dan omplementasi perancangan agar dapat memenuhi tujuan dari penelitian.

BAB 6 PENGUJIAN DAN ANALISIS

Menjelaskan proses dan hasil dari pengujian terhadap sistem yang telah dibangun dan memastikan bahwa program telah sesuai dengan perancangan dan disertai analisis.

BAB 7 PENUTUP

Membuat kesimpulan dari keseluruhan uraian bab-bab sebelumnya, serta saran-saran dari hasil yang diperoleh, yang diharapkan dapat bermanfaat dalam pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian pustaka berisi penelitian sebelumnya yang terkait dengan penelitian yang dilakukan oleh penulis. Kajian pustaka digunakan sarana pembanding dan pendukung penulis dalam melakukan penelitian.

Penelitian pertama, adalah penelitian yang berjudul "*Encryption on MP3 Compression".* Dalam penelitian tersebut dijelaskan bagaimana mengimplimentasikan enkripsi pada saat pengompresan MP3 dengan menggunakan algoritma enkripsi AES 128. Ditampilkan hasil dari perbadingan *waveform* dan hasil waktu eksekusi untuk pengenkripsian, juga dijelaskan bahwa penggunaan algortima *AES* menggunakan memori yang rendah dan hasil keamanan yang bagus. (Gadanayak & Pradhan, 2011). Dengan penelitian tersebut, dapat dijadikan landasan untuk penggunaan algoritma enksipsi yang digunakan dalam penelitian ini, dengan hasil yang dijelaskan dapat dijadikan acuan bahwa enkripsi *AES* cocok untuk pengimplementasian *digital audio*.

Penelitian kedua, berjudul "Rancang Bangun Digital Book Reader Berbasis Android dengan Kemanan Data *Rivest Shamir Adleman Encryption* dan *Advanced Encryption Standard*". Dalam penelitian tersebut dijelaskan bagaimana penggunaan algoritma RSA dan AES dalam pengamanan data *digital book*, cara kerja dan *code* implementasinya. RSA digunakan dalam mengamankan proses *client-server* dan AES digunakan dalam enkripsi *file digital book.* Dalam penelitian tersebut juga dijelaskan tentang hasil dari pengujian waktu eksekusi kriptografi, *security, recoverability* dan *authority* (Dika, Tolle, & Data, 2015). Dengan penelitian tersebut, dapat digunakan acuan dan referensi untuk parameter-parameter yang akan diteliti dalam penelitian ini dalam pengimplementasian algoritma *AES*.

Penelitian ketiga, berjudul tentang "*A Study of Encryption Algorithms AES, DES and RSA for Security*". Dalam penelitian ini membahas tentang pengomparasian algoritma enkripsi yaitu, *AES*, *DES* dan *RSA*. Dilakukan juga penelitian waktu dalam enkripsi dan dekrispsi berdasarkan file dengan *packet size* yang sama. Didapatkan hasil dari uji coba dan pengomparasian bahwa algoritma *AES* lebih baik dari *DES* dan *RSA*. (Mahajan & Sachdeva, 2013). Dengan hasil penelitian tersebut dapat dijadikan pertimbangan bahwa algortima *AES* merupakan algoritma yang cocok digunakan dalam penelitian ini.

Penelitian keempat, berjudul tentang "Enkripsi dan Dekripsi dengan Algoritma AES 256 untuk Semua Jenis File". Dalam penelitian ini membahas tentang konsep dan proses kerja enkripsi dan dekripsi AES, serta penerapannya dalam *file* menggunakan algoritma AES 256. Didapatkan hasil analisis dari waktu dan besar *file* enkripsi dan dekripsi serta kesimpulan bahwa waktu yang diperlukan untuk proses enkripsi tidak sama dengan waktu proses dekripsi yang dikarenakan adanya pemakaian *resource* komputer dan juga konsep ukuran file dari proses enkripsi

dan dekripsi (Yuniati, Indriyanta, & C, 2009). Dengan hasil penelitian tersebut dapat mempengaruhi *boundary* penerapan algoritma *AES* yang benar.

2.2 Dasar Teori

2.2.1 Media Player

Media Player merupakan *software* yang memainkan *digital audio*, video atau animasi di dalam komputer. *Media player* menyediakan beberapa fitur, diantaranya pengguna dapat mengatur koleksi multimedia, memutar lagu dan film *rip CD* lagu ke dalam format MP3 dan format *digital audio* lainya, *burn-ing CD*, mendengarkan radio internet, download kontern dari toko musik *online* dan *streaming* konten dari internet. (Encyclopedia, 2016)

Salah satu *media player* untuk melakukan tugas terutama untuk memutar *digital audio* adalah *Android Multimedia*, dengan *support* untuk memutar berbagai jenis tipe media sehingga dengan mudah diintegrasikan dalam perangkat lunak. *Android Multimedia* dapat memutar *file* dari local storage maupun dari internet.

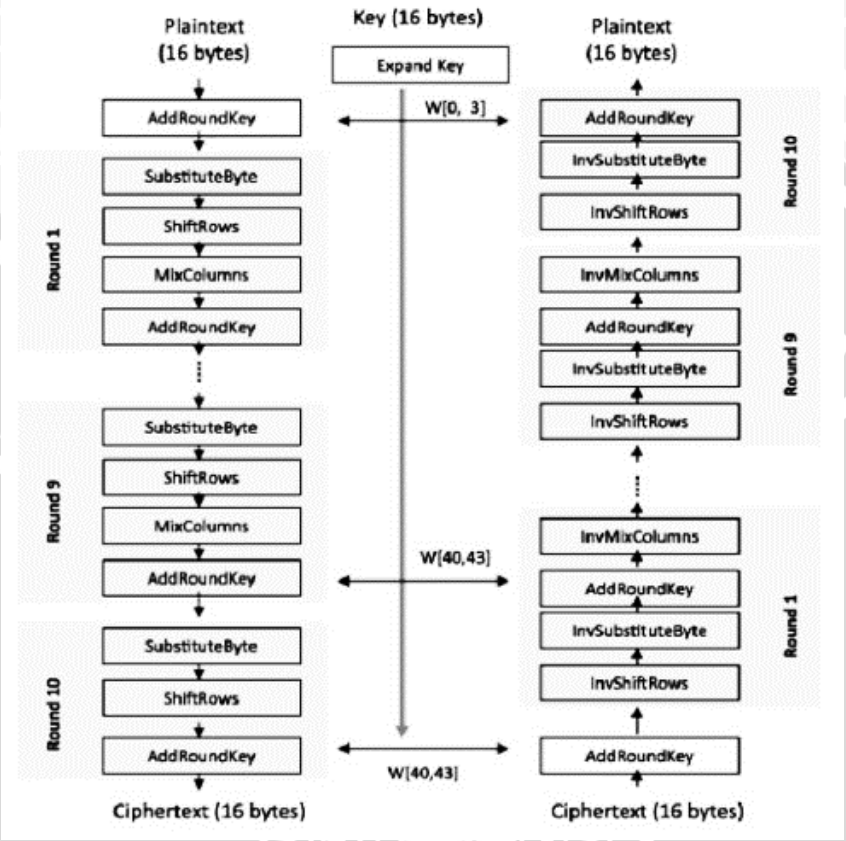
Dalam Sistem Operasi Android sudah disediakan *mediaplayer library*, yang memungkinkan untuk memutar dan mengendalikan seriap file audio di *background,* dengan menggunakan metode kontrol mediapplayer seperi play(), stop() dan seekto(). Pengembang hanya harus mengaplikasikannya dengan benar di dalam Komponen UI.

2.2.2 *Library* Kriptografi AES dalam bahasa *Java*

Advanced Encryption Standard (AES) merupakan penyempurnaan dari *Data Encryption Standard* (DES) yang bagian dari kode simetris dan pertama kali dipublikasikan oleh NIST (*National Institute of Standard and Technology*) pada tahun 2001 (Endriani & Utami, 2014, p. 2). AES menggunakan kunci simetris untuk melakukan proses enkripsi dan dekripsi. Ukuran kunci dalam algoritma AES bervariasi yaitu 128, 192, dan 256 bit. Perbedaan panjang kunci nantinya mempengaruhi jumlah perputaran yang akan diimplementasikan dalam AES.

AES mengimplementasikan beberapa operasi dalam proses algoritmanya, seperti SubBytes, Shiftrows, Mixcolumns, dan AddRoundKey. Proses enkripsi dimulai dari operasi *bitwise* XOR *plaintext* dan kunci. Kemudian pada langkah *subbytes*, setiap byte matriks *state* ai, j diganti dengan SubBytes S (ai, j) menggunakan 8-bit S-box (*substitution box*), yaitu Rijndael S-box. SubBytes digunakan pada putaran 1 sampai 10 dan diimplementasikan oleh tabel pencarian dengan 256 entri, yang masing-masing mempunyai lebar 8bit. Tabel pencarian diimplementasikan karena perhitungan invers perkalian dapat menerima sejumlah besar siklus untuk penyelesaian. Dikarenakan besar *cycles* untuk menghitung *inverse*, maka beberapa *cycles* tambahan dimasukkan pada *cycles*. Transformasi dilakukan sebanyak 16 kali putaran.

Proses Shiftrows selanjutnya mengimplementasikan sebagai *cyclic shift* dalam setiap barisnya. Kemudian pada tahap Mixcolumns empat *byte* setiap kolom dikombinasikan menggunakan *ivertible linear transfprmation*. Shiftrows dan Mixcolumns memberikan *diffusion* dalam cipher. Selanjutnya proses Addroundkey dilakukan kembali, dan ketika sampai pada sembilan *cycles* Mixcolumns tidak dieksekusi pada tahap akhir dan data enkripsi dihasilkan. Sedangkan dalam proses dekripsi merupakan kebalikan dari proses enkripsi. (Patel & Kanjariya, 2015)



Gambar 2.1 Diagram blok AES

Dalam *Java* sudah disediakan library untuk penggunaan kriptografi yang bernama *Java Cryptography Extenseion* (*JCE*) yang sudah diintegrasikan pada *Java Development Kit* (*JDK*)versi 1.4.x keatas. Sehingga tdak diperlukannya lagi dalam membangun dan membuat algoritma AES mulai dari awal. Lokasi *JCE* terletak pada paket *javax.crypto* Untuk pengimplimentasianya dapat dilakukan dengan memanggil fungsi kriptografi dari *JCE* dengan Kode Sumber 2.1:

Kode Sumber 2.1 Import Kriptografi di JCE

```
1  import javax.crypto.*;
2  import javax.crypto.spec.*;
```

dua baris Kode Sumber 2.1 menjelaskan untuk mengimport fungsi kriptografi dari *JCE* kedalam klas untuk pengimplementasiannya.

Kemudian dalam pengimplementasiannya, diperlukan inisiasi atau pemberian nilai awal algoritma apa yang akan digunakan untuk kriptografi yang

disediakan oleh *JCE*. Dalam penggunaan kriptografi AES maka inisiasi dilakukan seperti Kode Sumber 2.2.

Kode Sumber 2.2 Inisiasi kriptografi AES

```
1  Chiper chiper = Chiper.getInstace ("AES");
2  cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
```

Library AES dalam bahasa *Java* mencakup proses standar enkripsi dan dekripsi, dan penggunaan fungsi tipe kriptografi apa yang akan digunakan.

2.2.3 JavaScript Object Notation

JSON merupakan standar berbasis teks untuk pertukaran data. Format JSON dikenal ringan (berukuran kecil/*lightweight*), mudah dibaca dan ditulis, dan digeneralisasi maupun diparsing dalam perangkat lunak AJAX. Format ini dibuat berdasarkan bahasa pemrograman JavaScript (*Standard* ECMA-262 generasi ketiga, Desember 1999) menggunakan format teks yang independen dari bahasa pemrograman namun menggunakan kaidah penulisan yang dikenal luas seperti bahasa C (C, C++, C3, *Perl*, *Phython*, dan lain sebagainya). Hal tersebut membuat JSON mudah dan sangat ideal dalam pengiriman data . (Boci & Chen, 2012, p. 2).

JSON dibangun dalam dua struktur yaitu (El-Aziz & Kannan, 2014, p. 1):

1. Kumpulan pasangan nilai. Pada beberapa bahasa, hal ini dinyatak sebagai obyek, rekaman, struktur, kamus, table hash, daftar berkunci atau array asosiati
2. Daftar nilai terurutkan. Pada kebanyakan bahasa, hal ini dinyatakan sebagai *array*, *vector*, daftar, atau urutan.

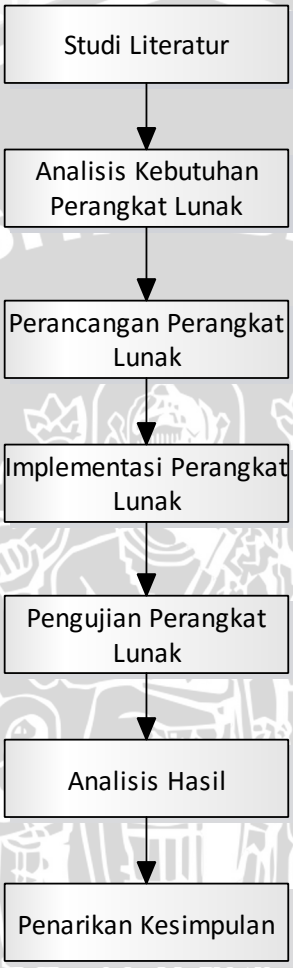
JSON lebih mudah dalam berkesinambungan dengan sisi client karena menggunakan *eval* ( ) pada JavaScript untuk mengubah data dari sisi server menjadi obyek JavaScript dan JSON lebih unggul dalam hal efisiensi transmisi data dengan kapasitas data yang semakin bertambah, dikarenakan format transmisi data di teks plain mempunyai aturan syntax dan format kebutuhan. (Boci & Chen, 2012, p. 2).

2.2.4 MP3

MP3 atau MPEG-1 audio layer 3 merupakan pengkodean dalam *digital audio* dengan format kompresi audio sesuai pendengaran manusia. MP3 adalah perbaikan dari teknologi sebelumnya yang kualitas teknologi setara kualitas CD dengan ukuran yang lebih kecil. Kompresi yang dilakukan oleh MP3 tidak mempertahankan bentuk asli dari sinyal input tetapi menghilangkan suara yang keberadaannya kurang atau tidak signifikan bagi sistem pendengaran manusia. Proses pembentukan format MP3 dilakukan dengan cara sinyal yang masuk dibagi menjadi beberapa blok dan ditransformasikan menjadi domain frekuensi. Domain tersebut disaring sehingga menghasilkan sinyal dengan range frekuensi signifikan bagi sistem pendngaran manusia. Setelah melakukan proses penyaringan, dilakukan proses kuantisasi dimana data yang terkumpul dalam proses penyaringan dijadikan satu dan menghasilkan suatu keluaran berbetntuk file MP3 (Gadanayak & Pradhan, 2011, p. 1).

BAB 3 METODOLOGI

Metodologi penelitian menjelaskan metode yang digunakan dalam perancangan dan pembangunan *Audio Share* dan *Player* dengan Keamanan Data *Advanced Encryption Standard* Berbasis Android. Tahapan metodologi penelitian dapat dilihat pada Gambar 3.1 dan akan dijelaskan di sub bab 3.



Gambar 3.2 Diagram Metodologi Penelitian

3.1 Studi Literatur

Dari penjelasan di atas dapat dikatakan bahwa metodologi penelitian memiliki cakupan lebih luas daripada metode. Metode sendiri dapat diartikan sebagai cara, prosedur, atau teknik untuk menjalankan sebuah proses secara logis, terurut, dan sistematik. Metode/teknik dapat berupa metode/teknik untuk pengumpulan data, untuk analisis data, atau algoritma untuk pemecahan masalah penelitian. Terkadang metode dibedakan dari teknik dengan pemahaman bahwa teknik itu lebih khusus dan operasional daripada metode. Dalam panduan penulisan ini pemilihan istilah tersebut diserahkan kepada penulis dan pembimbingnya. Yang

Dalam tahapan ini, penulis mempelajari literatur dari beberapa topik keilmuan yang berhubungan dengan penelitian yang dilakukan. Studi Literatur ini digunakan sebagai landasan teori untuk mempelajari dan menganalisis terkait dengan permasalahan yang ada guna untuk menunjang penelitian, diantaranya :

1. Rekayasa Perangkat Lunak berbasis Perangkat Bergerak
2. Pemrograman dengan menggunakan bahasa *Java*
3. Konsep kriptografi menggunakan algortima *AES*
4. *Library* dalam penggunaan ulang perangkat lunak

Studi literatur tentang topik diatas diperoleh dari berbagai macam sumber diantaranya buku katalog dan digital yang berisikan tentang teori dan materi, jurnal penelitian, dan dokumentasi projek-projek yang memiliki keterkaitan dengan penelitian.

3.2 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan dilakukan dengan menentukan kebutuhan apa saja yang dibutuhkan untuk merancang dan membangun perangkat lunak agar perangkat lunak yang dihasilkan berkualitas dan sesuai dengan tujuan dari penelitian. Berikut adalah kebutuhan yang dibutuhkan dalam pembuatan perangkat lunak *Digital* Audio *Share* dan *Player* dengan Keamanan Data *Advanced Encryption Standard* berbasis Android.

1. Spesifikasi Kebutuhan
   a. Kebutuhan Fungsional, merupakan identifikasi kebutuhan yang merupakan kebutuhan utama yang akan diimplementasikan untuk membangun fungsionalitas sistem, dimana salah satunya adalah perangkat lunak dapat memutar *digital audio* yang terenkripsi.
   b. Kebutuhan Non Fungsional, merupakan analisa kebutuhan yang merupakan kebutuhan pendukung meliputi waktu eksekusi kriptografi, keaslian data, *security* dan kebutuhan pendukung lainnya untuk melengkapi kebutuhan perangkat lunak
2. Kebutuhan Data, merupakan kebutuhan tentang data pendukung, dalam penelitian ini data merupakan file yang akan diamankan dalam bentuk enkripsi atau *cipherfile* yaitu, *Audio* berformat MP3

3.3 Perancangan Perangkat Lunak

Perancangan sistem dibuat dengan tujuan untuk menyusun langkah-langkah kerja secara menyeluruh dan dilaksanakan setelah proses analisis kebutuhan dengan maksud mempermudah proses implementasi dan pengujian. Perancangan perangkat lunak berdasarkan *object-oriented analysis* dan *object-oriented design* yaitu menggunakan pemodelan UML (*Unifed Modeling Languange*). Perancangan dimulai dari tahap perancangan struktur dasar kerangka kerja dari perangkat lunak. Dilanjutkan tahap perancangan yang melakukan identifikasi apa saja yang dilakukan oleh sistem dan siapa saja yang dimodelkan dalam *use case diagram*, serta identifikasi *class* dan *interface* yang dibutuhkan dimodelkan dalam *class*

diagram. Kemudian mengidentifikasi interaksi antar objek yang dimodelkan dalam *sequence diagram* yang menggambarkan interaksi antar objek yang disusun dalam urutan waktu. Selanjutnya tahap perancangan identifikasi antarmuka pengguna dengan sistem dengan dimodelkan membuat desain. Di tahap terakhir, dilakukan perancangan keamanan data menggunakan konsep kriptografi *Advanced Encryption Standard* yang akan diimplementasi di sistem.

3.4 Implementasi Perangkat Lunak

Pada tahap ini, implementasi perangkat lunak didasarkan pada hasil perancangan yang telah ditentukan dan identifikasi di tahap sebelumnya. Diawali dengan pemahaman perancangan dan pengimplementasian kebutuhan fungsional yang telah dirancang. Tahap selanjutnya yaitu mengimplementasi algoritma yang telah dirancang dan menggabungkan dengan implementasi algoritma kriptografi dengan menggunakan konsep *Advanced Encryption Standard*. Di tahap terakhir adalah implementasi dari antarmuka dari pengguna dan sistem. Implementasi perangkat lunak sendiri menggunakan bahasa pemrograman Android berbasis Java dan untuk servernya menggunakan PHP.

3.5 Pengujian dan Analisis Hasil Perangkat Lunak

Pada tahap pengujian dilakukan untuk mengetahui berapa tingkat performa dan kinerja dari perangkat lunak untuk pemenuhan spesifikasi kebutuhan dan tujuan yang diharapkan. Perangkat lunak akan diuji dengan pengujian *white-box* , pengujian *black-box* dan pengujian keamanan. Pengujian unit menguji kebutuhan/requirement system yang telah didefinisikan didalam pada tahap analisis kebutuhan. Pengujian unit adalah pengujian basis path testing dengan berdasarkan kebutuhan utama perangkat lunak. Pengujian *black box* adalah pengujian yang ditekankan pada perangkat lunak ini, salah satu metodenya adalah pengujian validasi. Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah memenuhi kebutuhan dan performa. Pengujian keamanan dilakukan dengan tiga parameter yaitu pengujian performansi waktu, pengujian keaslian data dan pengujian *security*. Tahap selanjutnya adalah analisis hasil pengujian. Analisis Hasil dilakukan untuk menganalisis hasil dari pengujian yang sudah dilakukan didalam perangkat lunak.

3.6 Penarikan Kesimpulan

Perangkat lunak yang dihasilkan adalah *Digital Audio Share* dan *Player* dengan Kemanan Data *Advanced Encryption Standard* berbasis Android, kemudian ditarik kesimpulan mengenai hasil apa yang didapatkan selama proses penelitian.

BAB 4 SPESIFIKASI KEBUTUHAN DAN PERANCANGAN PERANGKAT LUNAK

Dalam bab ini menjelaskan tentang pengumpulan kebutuhan yang dibutuhkan dalam pembangunan perangkat lunak. Dengan pengumpulan spesifikasi kebutuhan perangkat lunak yang didapatkan akan dijadikan acuan untuk perancangan perangkat lunak *digital audio player* dan *share* dengan keamanan data *Advanced Encryption Standard* berbasis Android. Dalam bab ini, akan dijelaskan juga perancangan dalam membuat perangkat lunak.

4.1 Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi Kebutuhan Perangkat Lunak (SKPL) bertujuan untuk membuat dan mendokumentasikan spesifikasi perangkat lunak yang dibutuhkan di dalam sistem. Dokumen ini menjadi panduan dalam pengembangan perangkat lunak dari setiap tahapan pengembangannya baik dari pengembang maupun pengguna.

4.1.1 Deskripsi Umum Perangkat Lunak

AUDIOSAFE adalah perangkat lunak untuk memainkan dan berbagi *digital audio* berbasis Android. Dengan fitur berupa berbagi *digital audio*, pengguna dapat membagikan file *digital audio* secara luas dengan pengamanan data dan kunci yang terpisah, dan juga fitur memainkan *digital audio*. AUDIOSAFE membantu pengguna untuk memainkan *digital audio* dengan mengunduh file enkripsi dengan kunci terpisah yang sudah dibagikan dan terhubung dari server.

Sehingga dapat digambarkan secara umum bahwa AUIDIOSAFE memiliki 4 fitur utama, yaitu *download audio, upload audio, play audio dan* memberi *privilege*. *Download audio* memberi fitur kepada *user* mengunduh audio yang sudah diunggah oleh *user* lain, *upload audio* berfungsi untuk *user* agar dapat mengunggah *digital audio* yang didalamnya terdapat proses enkripsi sebelum dikirimkan ke server, *play audio* memberi fitur kepada *user* untuk dapat memutar audio yang didalamnya mengandung proses dekripsi kepada file sehingga nantinya dapat diputar, dan memberi *privilege* berfungsi untuk *user* agar dapat memberi hak ijin kepada *user* lain untuk dapat mendapatkan kunci dalam membuka *file* enkripsi.

Penulis merancang dan membangun perangkat lunak *player* dan *share digital audio* yang sekaligus dapat melindungi distribusinya dengan meng-implementasikan keamanan digital agar file yang didapat dari server AUDIOSAFE tidak mudah dibajak.

Gambar 4.1 Deskripsi Umum Sistem

4.1.2 Defenisi, Istilah dan Singkatan

Tabel 4.1 Daftar Defenisi, Istilah dan Singkatan

Defenisi, Istilah dan Singkatan	Uraian
SKPL	Spesifikasi Kebutuhan Perangkat Lunak atau dapat disebut juga sebagai *Software Requirement Spesification* (SRS) yang merupakan panduan dalam pengembangan perangkat lunak dari setiap tahapan pengembangannya.
AES	*Advanced Encryption Standard,* merupakan algoritma standar enksipsi yang digunakan dalam penelitian.
AUDIOSAFE	Perangkat Lunak *Digital Audio Share* dan *Player* dengan keamanan data *Advanced Encryption* Standard berbasis Android yang akan penulis rancang dan bangun dalam penelitian.
Audio Digital	Penyajian teknologi rekaman suara dari suara asli yang terrekam.
MP3	MP3 atau MPEG-1 *audio* layer 3 merupakan pengkodean dalam *digital audio* dengan format kompresi *audio* sesuai pendengaran manusia.

4.1.3 Aturan Penomoran

Tabel 4.2 Aturan Penamaan dan Penomoran

Hal / Bagian	Uraian
Kebutuhan Fungsional	SRS-F-AUDIOSAFE-xx
Kebutuhan Non Fungsional	SRS-NF-AUDIOSAFE-xx
Kebutuhan Interface	IRS-AUDIOSAFE-xx
Use Case	UC-AUDIOSAFE-xxx
Class Diagram (Analisa)	CD-AUDIOSAFE-xxx

Skenasio Use Case	SUC-AUDIOSAFE-xxx
Sequence Diagram	SD-AUDIOSAFE-xxx
Desain Antarmuka	INTF-xxx

4.1.4 Fungsi Utama Perangkat Lunak

Fungsi utama dari perangkat lunak AUDIOSAFE adalah bagaimana sistem dapat memberikan fitur keamanan *digital audio* yang dibagikan dalam sistem sehingga tidak dapat dibajak oleh *end user*. Pengguna tidak dapat membagikan file *digital audio* selain di perangkat lunak AUDIOSAFE karena sudah diamankan dengan teknik keamanan data menggunakan algoritma keamanan *Advanced Encryption Standard*. Dan juga Pengguna dapat mengunduh dan memainkan *digital audio* dari sistem AUDIOSAFE, namun dengan kunci yang harus dimasukkan.

4.1.5 Spesifikasi Kebutuhan Perangkat Lunak

1. Sistem AUDIOSAFE menyediakan fungsi untuk registrasi kepada *user* baru yang ingin menggunakan perangkat lunak ini dengan menggunakan parameter setidaknya ada *username* dan *password*. **(SRS-F-AUDIOSAFE-01)**
2. Sistem AUDIOSAFE mampu menyediakan fungsi *login* untuk memvalidasi *user* yang akan masuk. Fungsi *login* memerlukan parameter *username* dan *password* yang sebelumnya diinputkan dalam proses registrasi (*lihat : SRS-F-AUDIOSAFE-01*). Validasi *login* akan dilanjutkan ke *activity home* jika inputan user sesuai dengan *database* yang ada, jika terdapat *error* pada validasinya, maka akan dimunculkan notifikasi error pada login. **(SRS-F-AUDIOSAFE-02)**
3. Sistem AUDIOSAFE mampu menyediakan fungsi *user* untuk mengunggah *digital audio* kedalam sistem dengan kunci keamanan yang diinputkan oleh user. **(SRS-F-AUDIOSAFE-03)**
4. Sistem AUDIOSAFE memfasilitasi *user* untuk melihat daftar-daftar *digital audio* yang disediakan perangkat lunak yang sebelumnya dibagikan oleh *user*. **(SRS-F-AUDIOSAFE-04)**
5. Sistem AUDIOSAFE mampu memfasilitasi *user* untuk mengunduh *digital audio* yang diinginkan dari perangkat lunak.**(SRS-F-AUDIOSAFE-05)**
6. Sistem AUDIOSAFE memfasilitasi *user* untuk melihat daftar-daftar *digital audio* yang sudah diunduh. **(SRS-F-AUDIOSAFE-06)**
7. Sistem AUDIOSAFE menyediakan fungsi untuk *user* dapat memutar koleksi *digital audio* yang telah *user* unduh. **(SRS-F-AUDIOSAFE-07)**
8. Sistem AUDIOSAFE mampu menyediakan fungsi *user* untuk menghapus daftar *digital audio* yang telah *user* unggah. **(SRS-F-AUDIOSAFE-08)**

9. Sistem AUDIOSAFE mampu menyediakan fungsi *user* untuk menghapus daftar *digital audio* yang telah *user* unduh di daftar yang terunduh. **(SRS-F-AUDIOSAFE-09)**
10. Sistem AUDIOSAFE memfasilitasi *user* untuk dapat memberikan *privilege* kepada *user* lain dalam mendapatkan kunci dan membuka enkripsi. **(SRS-F-AUDIOSAFE-10)**
11. Sistem AUDIOSAFE memfasilitasi *user* untuk dapat melihat profilnya. **(SRS-F-AUDIOSAFE-11)**
12. Sistem AUDIOSAFE memfasilitasi *user* untuk *logout* dari perangkat lunak. **(SRS-F-AUDIOSAFE-12)**
13. Sistem AUDIOSAFE menyediakan parameter kualitas waktu dalam pengamanan data. **(SRS-NF-AUDIOSAFE-01)**
14. Sistem AUDIOSAFE menyediakan parameter kualitas keaslian data. **(SRS-NF-AUDIOSAFE-03)**
15. Sistem AUDIOSAFE menyediakan parameter kualitas *security*. **(SRS-NF-AUDIOSAFE-02)**

4.1.6 Detil Kebutuhan Fungsional

Tabel 4.3 Kebutuhan Fungsional

No	Nomor *Software Requirement Spesification* (SRS)	Nama Fungsi	Deskripsi Kebutuhan
1.	SRS-F-AUDIOSAFE-01	Registrasi *User*	Sistem AUDIOSAFE menyediakan fungsi untuk registrasi kepada *user* baru yang ingin menggunakan perangkat lunak ini dengan menggunakan parameter setidaknya ada *username* dan *password*. *User* menginputkan *username* dan *password* kedalam form yang disediakan kemudian menekan tombol *register* untuk menjalankan proses registrasi. Notifikasi registrasi berhasil akan muncul dan langsung dilanjutkan ke *acitivy login* jika proses registrasi berhasil, dan akan ditampilkan juga notifikasi registrasi gagal. *Username* dari setiap *user* bersifat unik sehingga setiap *user* harus memiliki *username* yang berbeda.
2.	SRS-F-AUDIOSAFE-02	Melakukan *Login*	Sistem AUDIOSAFE mampu menyediakan fungsi *login* untuk memvalidasi *user* yang akan masuk. *User menginputkan username* dan *password* yang sebelumnya diinputkan dalam proses registrasi Validasi *login* akan dilanjutkan ke *activity home* jika inputan user sesuai dengan *database* yang ada, dan dapat menggunakan fitur-fitur yang perangkat lunak sediakan. Jika terdapat *error* pada validasinya

No	Nomor *Software Requirement Spesification* (SRS)	Nama Fungsi	Deskripsi Kebutuhan
			atau (gagal dalam *login*), maka akan dimunculkan notifikasi error pada login.
3.	SRS-F-AUDIOSAFE-03	Mengunggah *Audio*	Sistem AUDIOSAFE mampu menyediakan fungsi untuk *user* mengunggah *digital audio* yang diinginkan kedalam sistem. *User* juga diharuskan untuk menginput nama *digital audio* dan gambar untuk *thumbnail* dalam perangkat lunak. *User* juga memasukkan kunci keamanan yang akan disimpan dalam server.
4.	SRS-F-AUDIOSAFE-04	Melihat Daftar *Audio*	Sistem AUDIOSAFE memfasilitasi *user* untuk melihat daftar-daftar *digital audio* yang disediakan perangkat lunak yang sebelumnya dibagikan oleh *user*. Sistem menampilkan katalog *digital audio* yang siap untuk diunduh dalam tampilan *cardview*. Jika perangkat lunak tidak dapat memnghubungi server untuk mengambil daftar *digital audio*, maka akan ditampilkan notifikasi.
5.	SRS-F-AUDIOSAFE-05	Mengunduh *Audio*	Sistem AUDIOSAFE mampu memfasilitasi *user* untuk mengunduh *digital audio* yang diinginkan dari perangkat lunak dengan diberikan opsi tombol *download audio*.
6.	SRS-F-AUDIOSAFE-06	Melihat Daftar *Audio* yang Terunduh	Sistem AUDIOSAFE memfasilitasi *user* untuk melihat daftar-daftar *digital audio* yang sudah diunduh. File *digital audio* yang sudah diunduh akan masuk kedalam direktori lokal ponsel *user*, dan dalam pengenalan ekstensi AUDIOSAFE, sistem hanya akan mengenali file yang berekstensi ".aufe" sehingga tidak akan tercampur penegenalannya dengan ekstensi *audio* yang lain.
7.	SRS-F-AUDIOSAFE-07	Memutar *Audio*	Sistem AUDIOSAFE menyediakan fungsi untuk *user* dapat memutar koleksi *digital audio* yang telah *user* unduh. Fungsi memutar *digital audio* akan dijalankan ketika *user* sudah diberikan privilege untuk dapat memutar.
8.	SRS-F-AUDIOSAFE-08	Menghapus *Audio* di Server	Sistem AUDIOSAFE mampu menyediakan fungsi *user* untuk menghapus daftar *digital audio* yang telah *user* unggah dengan disediakannya opsi tombol *delete audio*.
9.	SRS-F-AUDIOSAFE-09	Menghapus *Audio* yang Terunduh	Sistem AUDIOSAFE mampu menyediakan fungsi *user* untuk menghapus daftar *digital audio* yang telah *user* unduh dengan disediakannya opsi tombol *delete audio*.
10.	SRS-F-AUDIOSAFE-10	Memberi *privilege*	Sistem AUDIOSAFE memfasilitasi *user* untuk dapat memberikan *privilege* kepada *user* lain

No	Nomor *Software Requirement Spesification* (SRS)	Nama Fungsi	Deskripsi Kebutuhan
			untuk membuka enkripsi dan memutar *audio* yang telah diunduh.
11.	SRS-F-AUDIOSAFE-11	Melihat profil	Sistem AUDIOSAFE memfasilitasi *user* untuk dapat melihat profilnya. Sistem menampilkan gambar profil dan *digital audio* yang *user* telah unggah.
12.	SRS-F-AUDIOSAFE-12	Melakukan *Logout*	Sistem AUDIOSAFE memfasilitasi *user* untuk *logout* dari perangkat lunak untuk perlindungan jika *user* mengingingnkan untuk membuka kembali perangkat lunak dengan adanya autentikasi.

4.1.7 Detil Kebutuhan Non Fungsional

Tabel 4.4 Kebutuhan Non Fungsional

No	Nomor *Software Requirement Spesification* (SRS)	Parameter Kualitas	Deskripsi Kebutuhan
1.	SRS-NF-AUDIOSAFE-01	Waktu Kriptografi	Kecepatan proses enkripsi dan dekripsi tidak melebihi dari 4 detik, didasarkan pada *tolerable respon time* indeks application respon, APDEX.
2.	SRS-NF-AUDIOSAFE-02	Keaslian data	Perbandingan konten data yang sama tanpa mengubah isi dari hasil enkripsi dan dekripsi kriptografi .
3.	SRS-NF-AUDIOSAFE-03	*Security*	Keamanan dalam pemutaran audio dimanajemen oleh *privilege* yang diberikan oleh user.

4.1.8 Interface Requirement Spesification

Tabel 4.5 Interface Requirement Spesification

No	Nomor IRS	Informasi yang Dibutuhkan	Tujuan	Frekuensi	Format
1.	IRS-AUDIOSAFE-01	*Username* dan *password*	Registrasi User	Sekali saat user ingin mendafat untuk dapat menikmati fitur-fitur AUDIOSAFE.	*Username* String, panjang maksimal 20 karakter dan bersifat unik (hanya satu username yang bisa dipakai). Password String, dengan panjang

No	Nomor IRS	Informasi yang Dibutuhkan	Tujuan	Frekuensi	Format
					minimum 8 karakter.
2.	IRS-AUDIOSAFE-02	*Username* dan *password*	Autentikasi Login User	Sekali saat user ingin masuk kedalam sistem untuk dapat menjalankan fitur-fitur AUDIOSAFE.	*Username* String, panjang maksimal 20 karakter dan bersifat unik (hanya satu username yang bisa dipakai). Password String, dengan panjang minimum 8 karakter.
3.	IRS-AUDIOSAFE-03	Key untuk membuka Enkripsi *Audio*	Membuka sistem keamanan *digital audio* untuk dapat diputar.	Setiap kali user akan memutar *digital audio*.	Key (String), berupa kata kunci dari user.
4.	IRS-AUDIOSAFE-04	Login Status	Autentikasi One Device One Login	Sekali saat user melakukan logout	Login Status (String) merupakan barisan String yang membuat server mengenali satu user sedang login sehingga jika ada user yang sama mencoba login server akan menolaknya

4.1.9 Analisa Aktor

Tabel 4.6 Analisa Aktor

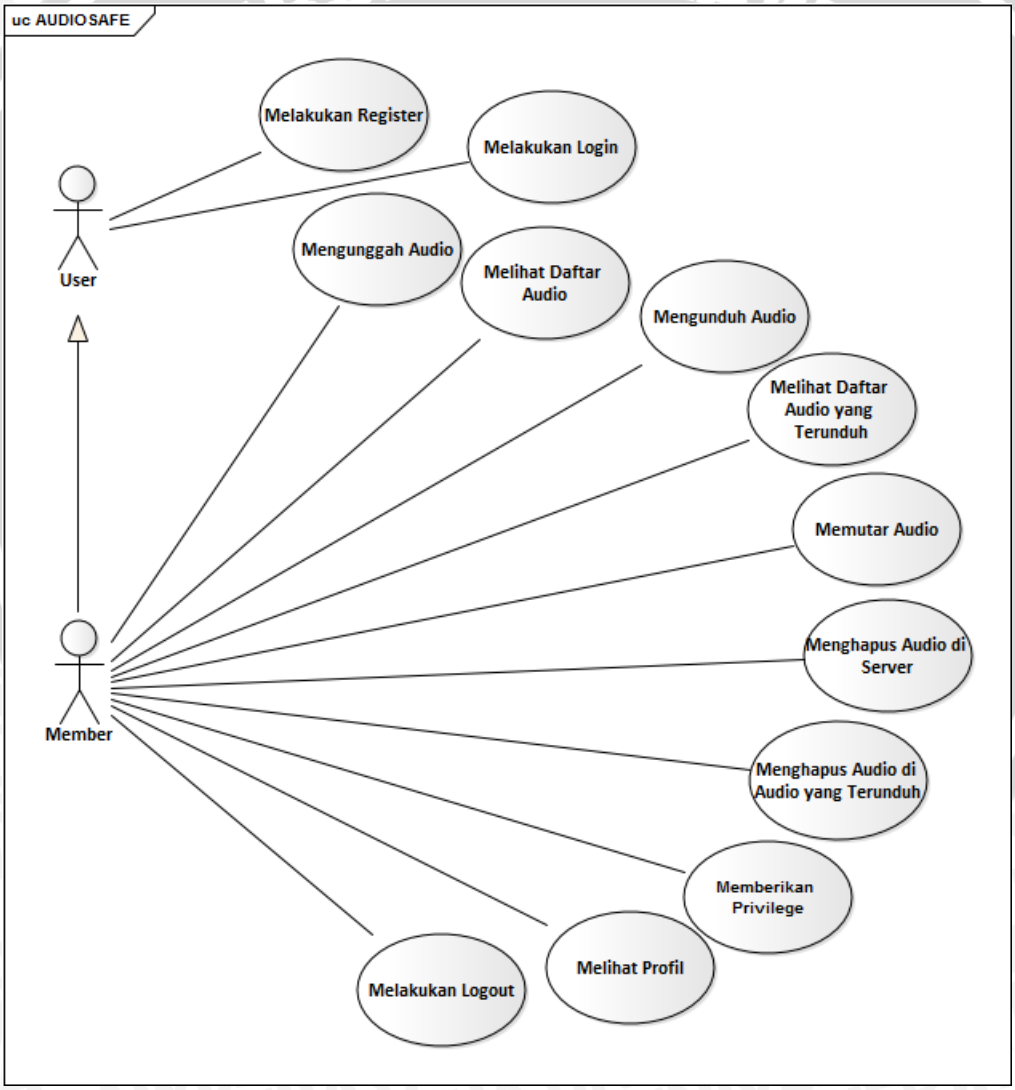
Kategori Pengguna	Tugas	Hak Akses ke Perangkat Lunak
User	Melakukan Registrasi, dan Melakukan Login ke dalam sistem.	SRS-F-AUDIOSAFE-01, SRS-F-AUDIOSAFE-02
Member	Mengunggah *Audio*, Melihat Daftar *Audio*, Mengunduh *Audio*, Melihat Daftar *Audio* yang Terunduh, Memutar *Audio*, Menghapus *Audio* di Server, Menghapus *Audio* di Daftar yang Terunduh, Memberikan *Privilege*,	SRS-F-AUDIOSAFE-03, SRS-F-AUDIOSAFE-04, SRS-F-AUDIOSAFE-05, SRS-F-AUDIOSAFE-06, SRS-F-AUDIOSAFE-07, SRS-F-AUDIOSAFE-08, SRS-F-AUDIOSAFE-09,

	Melihat Profil dan Melakukan Logout untuk keluar dari sistem	SRS-F-AUDIOSAFE-10, SRS-F-AUDIOSAFE-11, SRS-F-AUDIOSAFE-12.

4.1.10 Diagram Use Case

Use Case merupakan suatu diagram yang menggambarkan sejumlah *actor* dan hubungannya atau perilakunya terhadap fungsi yang disediakan sistem. *Use Case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat *user*) dan bukan bagaimana fungsi yang ada di dalam sistem.

Use Case pada Gambar 4.2 merupakan pemodelan yang digambarkan pada sis perangkat lunak AUDIOSAFE, yang mana didapatkan dari fungsionalitas dan *actor* dalam analisa spesifikasi kebutuhan. Didapatkan dua aktor yaitu *User* dan *Member* dan dua belas *use case*.



Gambar 4.2 *Use Case* Diagram AUDIOSAFE

4.1.11 Skenario Use Case

4.1.11.1 Spesifikasi *Use Case* : Melakukan Registrasi (SUC-AUDIOSAFE-01)

Tabel 4.7 Use Case Scenario Melakukan Regitrasi

Alur event untuk use case Melakukan Registrasi.	
Tujuan	Mendaftarkan pengguna baru untuk dapat menggunakan fitur-fitur/fungsi perangkat lunak AUDIOSAFE.
Aktor	*User.*
Pre-condition	*User* menekan tombol *register* pada activity *login* dan sudah masuk ke dalam activity *register.*
Main Flow	1. *User* meng*input*kan *username* dan *password* di kolom yang disediakan. 2. *User* menekan tombol *register*. 3. Sistem mengecek ketersediaan username apakah terdapat kesamaan atau tidak. 4. *Username* tidak terdapat duplikasi dan dapat digunakan dalam server. 5. Terdapat notifikasi dari server bahwa registrasi berhasil dan langusng diarahkan ke *activity login*.
Alternative Flow	1. Jika *username* terdapat memiliki kesamaan dengan *user* lain, maka sistem akan memberi notifikasi *error* "username sudah terdaftar".
Post Condition	*User* baru sudah terbuat dan *Member* dapat masuk dan menjalankan fitur-fitur AUDIOSAFE.

4.1.11.2 Spesifikasi *Use Case* : Melakukan Login (SUC-AUDIOSAFE-02)

Tabel 4.8 Use Case Scenario Melakukan Login

Alur event untuk use case Melakukan Login.	
Tujuan	Mengautentifikasi *user* untuk masuk ke dalam sistem dan menggunakan fitur-fitur AUDIOSAFE.
Aktor	*User.*
Pre-condition	*User* sudah masuk dalam *activity login* dan mempunyai akun.
Main Flow	1. *User* memasukkan *username* dan *password* ke dalam kolom yang disediakan. 2. *User* menekan tombol login. 3. Server memvalidasi *username* dan *password* yang dimasukkan valid atau tidak. 4. *Username* dan *password* valid kemudian akan diarahkan ke *activity* selanjutnya untuk dapat menggunakan seluruh fitur AUDIOSAFE.

Alternative Flow	1. Jika *username* dan *password* tidak valid, maka akan diberikan notifikasi *error* "username dan password salah" dan tetap akan ditahan di *activity login.*
Post Condition	*User* masuk menjadi *Member* dan dapat mengguakan fitur-fitur AUDIOSAFE.

4.1.11.3 Spesifikasi *Use Case* : Mengunggah *Audio* (SUC-AUDIOSAFE-03)

Tabel 4.9 Use Case Scenario Mengunggah *Audio*

Alur event untuk use case Mengunggah *Audio*.	
Tujuan	Mengunggah *digital audio* dari *user* terdaftar ke dalam server.
Aktor	*Member.*
Pre-condition	*Member* sudah masuk kedalam *activity* unggah *audio* setelah menekan tombol unggah *audio* di tab profil.
Main Flow	1. Memasukkan nama *digital audio*, gambar untuk *thumbnail*, kunci enksipsi dan memilih file *digital audio* dari lokal direktori *Member*. 2. *Member* menekan tombol unggah. 3. Sistem akan memberi notifikasi progress unggahan. 4. Sistem akan memberi notifikasi bahwa *audio* berhasil diunggah dan diarahkan ke *activity* profil.
Alternative Flow	1. Jika proses unggahan terganggu karena masalah jaringan maka akan muncul pesan jika proses unggah gagal dan user harus mengunggahnyanya mulai dari awal.
Post Condition	*Audio* sudah masuk ke dalam server AUDIOSAFE.

4.1.11.4 Spesifikasi *Use Case* : Melihat Daftar *Audio* (SUC-AUDIOSAFE-04)

Tabel 4.10 Use Case Scenario Melihat Daftar *Audio*

Alur event untuk use case Melihat Daftar *Audio*.	
Tujuan	Melihat daftar-daftar *digital audio* yang server sediakan.
Aktor	*Member.*
Pre-condition	Segala kondisi sebelum menuju *activity* melihat daftar *audio*.
Main Flow	1. *Member* menekan tab icon daftar *audio*. 2. Server mengirimkan daftar *audio* dari unggahan *member* lain . 3. Daftar *audio* ditampilkan dalam bentuk *cardview*.
Alternative Flow	1. Server tidak mengirimkan daftar apapun jika daftar *audio* unggahan *member* kosong.
Post Condition	*Member* melihat seluruh daftar *audio* unggahan *member* lain.

4.1.11.5 Spesifikasi *Use Case* : Mengunduh *Audio* (SUC-AUDIOSAFE-05)

Tabel 4.11 Use Case Scenario Mengunduh *Audio*

Alur event untuk use case Mengunduh *Audio*.	
Tujuan	Mengunduh *digital audio* dari server ke dalam lokal *storage device member.*
Aktor	*Member.*
Pre-condition	Memilih *audio* dari *activity* daftar *audio* dengan memilih salah satu item dari yang disediakan.
Main Flow	1. Sistem menampilkan *activity* unduh *audio* dan menampilkan nama *audio* dan thumbnail dari *audio*. 2. *Member* menekan tombol *download* . 3. Progress unduhan akan ditampilkan. 4. Notifikasi proses unduhan berhasil akan muncul ketika unduhan selesai.
Alternative Flow	1. Muncul notifikasi error "unduhan gagal" ketika terdapat gangguan dalam proses mengunduh.
Post Condition	*Member* mendapatkan *digital audio* unduhan kedalam *device*nya.

4.1.11.6 Spesifikasi *Use Case* : Melihat Daftar *Audio* yang Terunduh (SUC-AUDIOSAFE-06)

Tabel 4.12 Use Case Scenario Melihat Daftar *Audio* yang Terunduh

Alur event untuk use case Melilhat Daftar *Audio* yang Terunduh	
Tujuan	Melihat daftar-daftar *digital audio* yang sudah member unduh.
Aktor	*Member.*
Pre-condition	Segala kondisi sebelum menuju *activity* melihat daftar *audio* yang terunduh.
Main Flow	1. *Member* menekan tab icon daftar *audio* yang terunduh (siap dimainkan). 2. Sistem membaca data dari lokal direktori *device member*. 3. Daftar *audio* ditampilkan dalam bentuk *cardview*.
Alternative Flow	1. Tidak akan menampilkan daftar apapun jika daftar *audio* unduhan *member* kosong.
Post Condition	*Member* melihat seluruh daftar *digital audio* yang sudah *member* unduh.

4.1.11.7 Spesifikasi *Use Case* : Memutar *Audio* (SUC-AUDIOSAFE-07)

Tabel 4.13 Use Case Scenario Memutar *Audio*

Alur event untuk use case Memutar *Audio*.	
Tujuan	Memutar *digital audio* dari daftar unduhan *member*.
Aktor	*Member.*
Pre-condition	*Member* berada ditampilan *activity* daftar *audio* terunduh (siap dimainkan).
Main Flow	1. *Member* memilih item dari *audio* yang sudah didownload. 2. Akan diarahkan ke *activity detil audio*. 3. *Member* menekan tombol *play.* 4. Konten *audio* akan diputar.
Alternative Flow	1. Jika hak ijin belum diberikan dari pengunggah, maka akan audio tidak diputar.
Post Condition	*Member* dapat mendengarkan *digital audio* dari sistem.

4.1.11.8 Spesifikasi *Use Case* : Menghapus *Audio* di Server(SUC-AUDIOSAFE-08)

Tabel 4.14 Use Case Scenario Menghapus *Audio* di Server

Alur Event Untuk Use Case Menghapus *Audio*	
Tujuan	Menghapus *audio* m*ember* yang diunggah dari sistem.
Aktor	*Member.*
Pre-condition	*Member* memilih item *audio* dari *activity* profil.
Main Flow	1. Akan diarahkan ke *activity detil audio*. 2. *Member* menekan tombol *delete*. 3. Akan ditampilkan opsi konfirmasi, apakah benar untuk menghapus, jika *member* memilih iya maka *audio* akan dihapus dari sistem, jika tidak, maka fungsi akan dibatalkan.
Alternative Flow	1. Jika terdapat error dalam menghapus, maka ditampilakn notfiikasi error gagal menghapus.
Post Condition	*Audio* berhasil dihapus dari sistem.

4.1.11.9 Spesifikasi *Use Case* : Menghapus *Audio* yang Terunduh (SUC-AUDIOSAFE-09)

Tabel 4.15 Use Case Scenario Menghapus *Audio* yang Terunduh

Alur Event Untuk Use Case Menghapus *Audio* yang Terunduh	
Tujuan	Menghapus *digital audio* m*ember* yang diunduh dari sistem.
Aktor	*Member.*
Pre-condition	*Member* memilih item *audio* dari *activity* daftar *audio* yang terunduh.
Main Flow	1. Akan diarahkan ke *activity detil audio*.

	2. *Member* menekan tombol *delete.* 3. Akan ditampilkan opsi konfirmasi, apakah benar untuk menghapus, jika *member* memilih iya maka *audio* akan dihapus dari sistem, jika tidak, maka fungsi akan dibatalkan.
Alternative Flow	1. Jika terdapat error dalam menghapus, maka ditampilakn notfiikasi error gagal menghapus.
Post Condition	*Audio* berhasil dihapus dari sistem.

4.1.11.10 Spesifikasi *Use Case* : Memberikan Privilege (SUC-AUDIOSAFE-10)

Tabel 4.16 Use Case Scenario Memberikan Privilege

Alur event untuk use case Memberikan Key.	
Tujuan	Memberikan privilege kepada *member* lain.
Aktor	*Member.*
Pre-condition	*Member* masuk ke tab profil.
Main Flow	1. *Member* menekan tombol *privilege user* dan akan diarahkan ke *activity privilege user*. 2. *Member* menekan item *user* dalam list item *user* yang ingin diberikan kuncinya. 3. *Member* menekan tombol allow untuk pengijinan atau deny untuk penolakan 4. Sistem akan memberikan notifikasi bahwa *privilege* baru sudah diberikan.
Alternative Flow	-
Post Condition	*Privilege* baru diberikan kepada member lain.

4.1.11.11 Spesifikasi *Use Case* : Melihat Profil (SUC-AUDIOSAFE-11)

Tabel 4.17 Use Case Scenario Melihat Profil

Alur event untuk use case Melihat Profil.	
Tujuan	Melihat profil dari *member* dirinya sendiri.
Aktor	*Member.*
Pre-condition	Segala kondisi sebelum menuju ke *activity* profil.
Main Flow	1. *Member* menekan tab icon profil. 2. Server mengirimkan profil *member* dari server. 3. Data ditampilkan.
Alternative Flow	-
Post Condition	*Member* melihat seluruh data *member* dirinya sendiri.

4.1.11.12 Spesifikasi *Use Case* : Melakukan Logout (SUC-AUDIOSAFE-12)

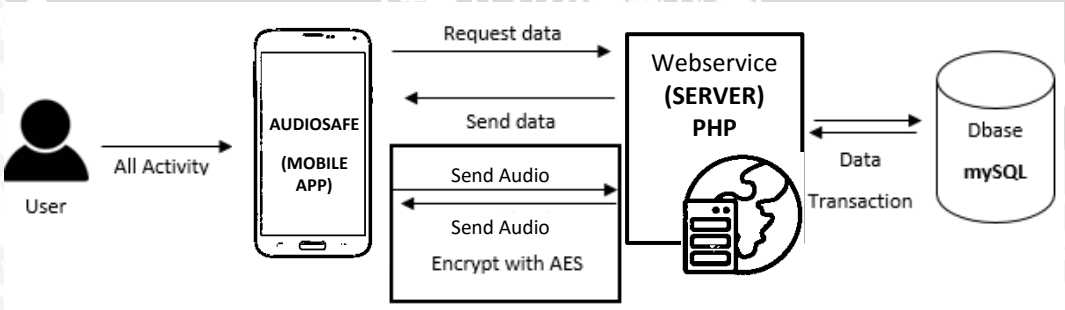
Tabel 4.18 Use Case Scenario Melakukan Logout

Alur event untuk use case Melakukan Logout.	
Tujuan	Keluar dari sistem AUDIOSAFE.
Aktor	*Member.*
Pre-condition	Segala kondisi sebelum menuju ke option menu.
Main Flow	1. *Member* menekan tombol option menu disebelah kanan atas. 2. Akan muncul *popout* menu. 3. Menekan pilihan logout.
Alternative Flow	-
Post Condition	Member sudah keluar dari sistem dan diarahkan ke *activiry login.*

4.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak dilakukan dengan membuat berbagai macam langkah perancangan, antara lain, perancangan aktifitas sistem, perancangan interaksi antar objek, perancangan klas, perancangan basis data, perancangan antar muka dan juga perancangan keamanan data. Pendekatan yang dilakukan untuk merancang perangkat lunak yaitu menggunakan pendekatan desain berorientasi objek, yang kemudian direpresentasikan kedalam *Unified Modeling Language* (UML). Perancangan aktifitas sistem, menggunakan permodelan bentuk *activity diagram*, perancangan interaksi objek yang dimodelkan dalam bentuk *sequence diagram*, dan perancangan klas yang dimodelkan dalam bentuk *class diagram.*

4.2.1 Perancangan Arsitektur Sistem



Gambar 4.3 Perancangan Sistem

Perancangan arsitektur sistem akan menjelaskan bagaimana tentang perancangan arsitektur sebuah sistem yang akan dibangun. *Audio*safe adalah perangkat lunak yang digunakan untuk berbagi dan memutar *digital audio* pada perangkat bergerak, hal itu dapat dikatakan *Audio*safe merupakan sebuah perangkat lunak. Tetapi *Audio*safe juga dapat dikatakan sebuah sistem karena

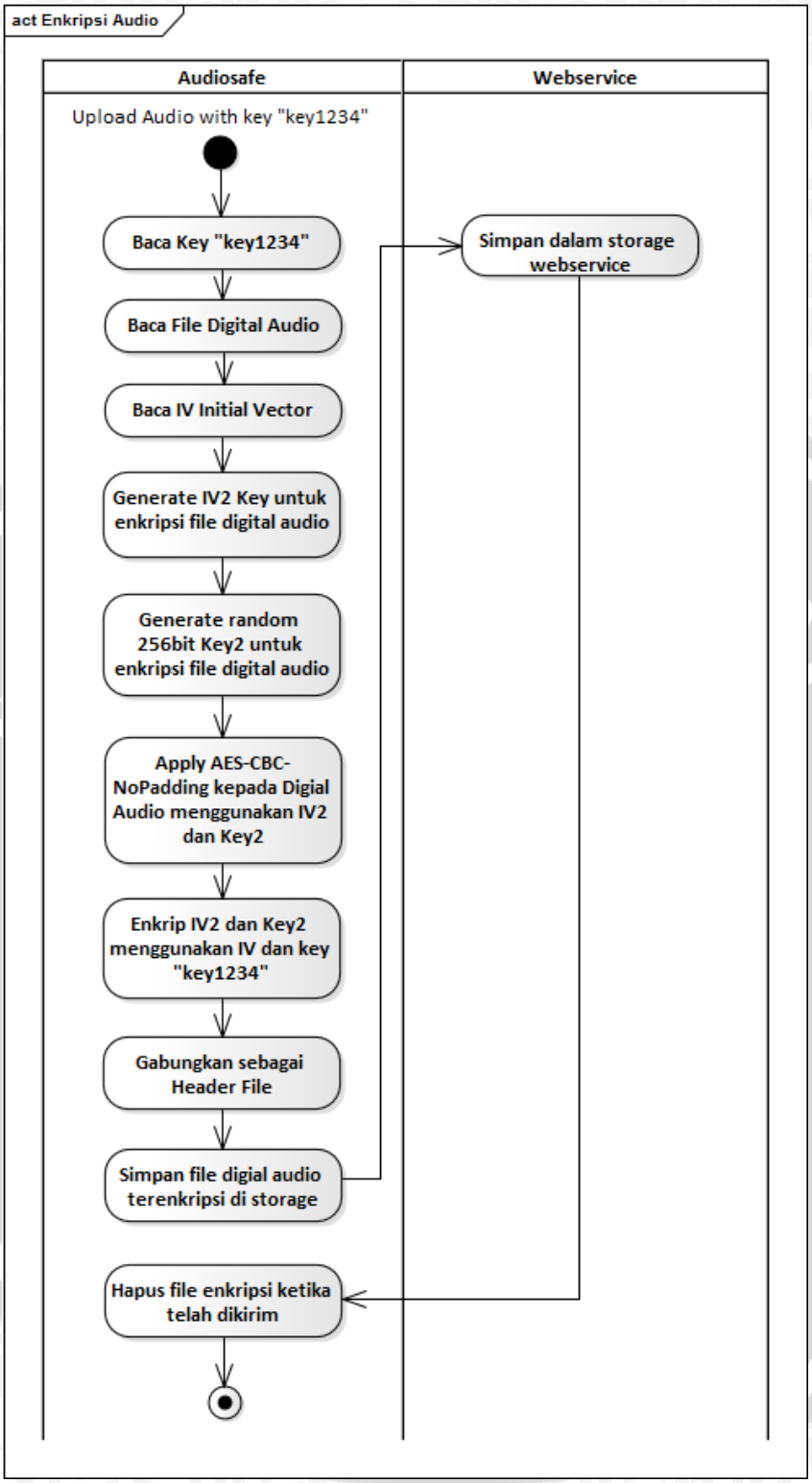
*Audio*safe secara satu kesatuan merupakan sistem yang dibangun dengan tujuan untuk mengamankan distribusi *digital audio* mulai dari server hingga sampai dengan aman ke pengguna melalui perangkat bergerak tersebut.

Perancangan *Audio*safe menggunakan konsep *native mobile application* pada perangkat mobile dengan sistem operasi android. Konsep native dipilih karena dapat mengunakan semua fitur yang dimiliki oleh perangkat smartphone dan menggunakan semua *resource* dalam sistem operasi. *Audio*safe pada android merupakan perangkat lunak yang dapat menjalankan semua aktifitas fungsionalitas *user.* Melalui perangkat lunak tersebut *user* secara keseluruhan dapat mengunduh, mengunggah dan berbagi *digital audio* yang dihubungkan dengan server. Sementara webservice (server) adalah sistem yang melayani *request user* melalui perangkat lunak android *Audio*safe, webservice berkomunikasi dengan *client* di perangkat bergerak menggunakan JSON dan database MySQL yang digunakan untuk menyimpan data. Dan untuk keamanan berbagi *digital audio* pada sistem ini, menggunakan keamanan data *Advanced Encryption Standard*.

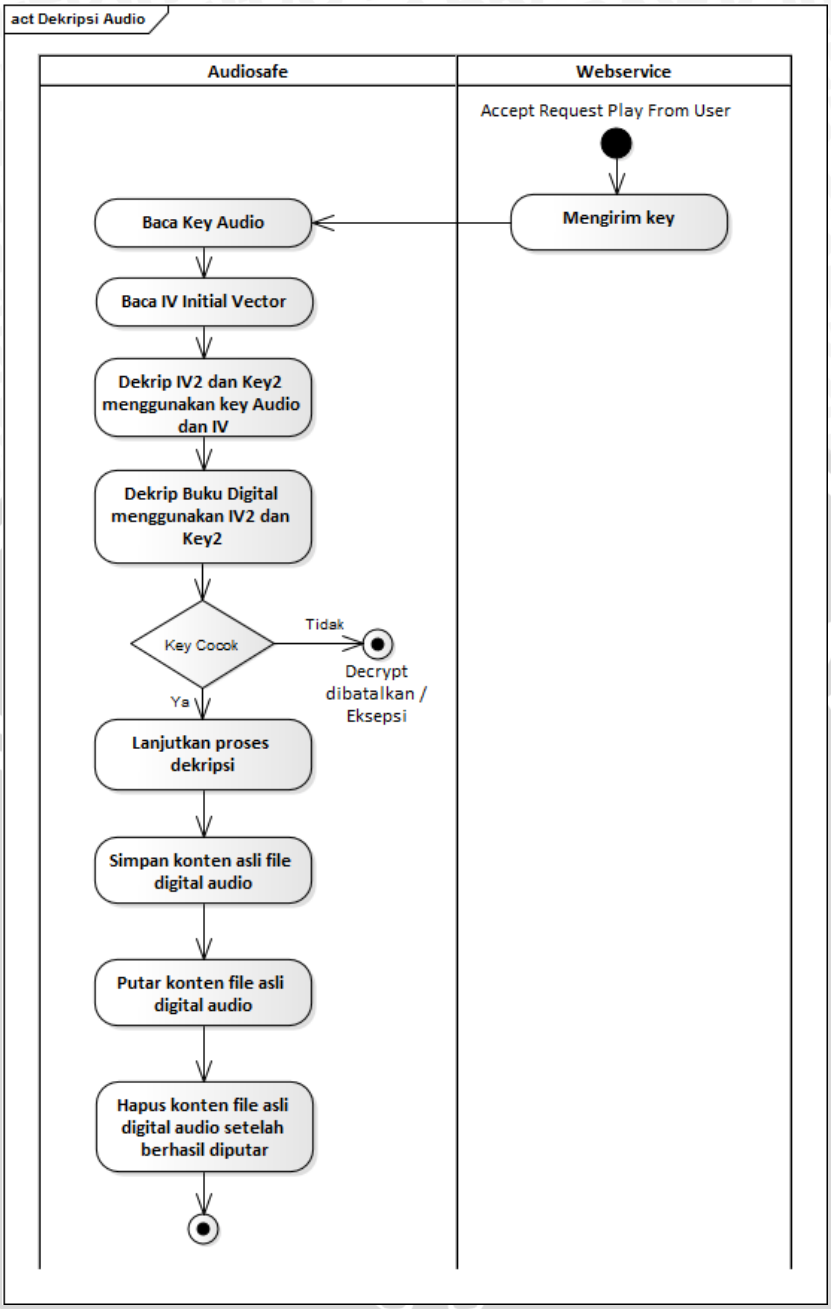
4.2.2 Perancangan Keamanan Data

Perancangan keamanan data menjelaskan bagaimana sistem akan merancang keamanan data untuk diterapkan melindungi *digital audio*. Keamanan data yang digunakan menggunakan algoritma *Advanced Encryption Standard* dengan CBC (*Chiper Block Chained*) *Mode*. *CBC Mode* menerapkan metode suatu blok dan blok lain saling terkait (chained). Saling terkait disini maksudnya adalah enkripsi dan dekripsi suatu blok data selalu melibatkan ciphertext (hasil enkripsi) blok sebelumnya. Agar blok-blok plaintext yang identik tidak menghasilkan blok-blok ciphertext yang identik pula, mode CBC (cipher block chaining) mengaitkan (chaining) satu blok dengan blok ciphertext sebelumnya dan menggunakan random initialization vector (IV) sebagai ciphertext blok ke-0. (Wicaksono, 2013).

Perancangan keamanan data ini digunakan untuk melindungi *digital audio*. Perancangan keamanan data *digital audio* tersebut menggunakan *AES*-256 bit, *CBC* Mode. *CBC* Mode dipilih karena mode ini menerapkan keamanan bertingkat dengan mengaitkan masing-masing blok yang akan dienkripsi dengan melakukan teknik *XOR* dengan blok sebelumnya. Jika terjadi kecacatan blok karena proses modifikasi, maka data tidak akan dapat dibuka. Berikut Gambar 4.4 dan 4.5 adalah *main flow* proses pengamanan *digital audio* dalam perangkat lunak *Audiosafe*.



Gambar 4.4 *Diagram* Alir Perancangan Enkripsi Kriptografi *AES*



Gambar 4.5 *Diagram* Alir Perancangan Dekripsi Kriptografi *AES*

Berikut merupakan alur kerja dari proses pengamanan *digital audio* pada perangkat lunak *Audio*safe :

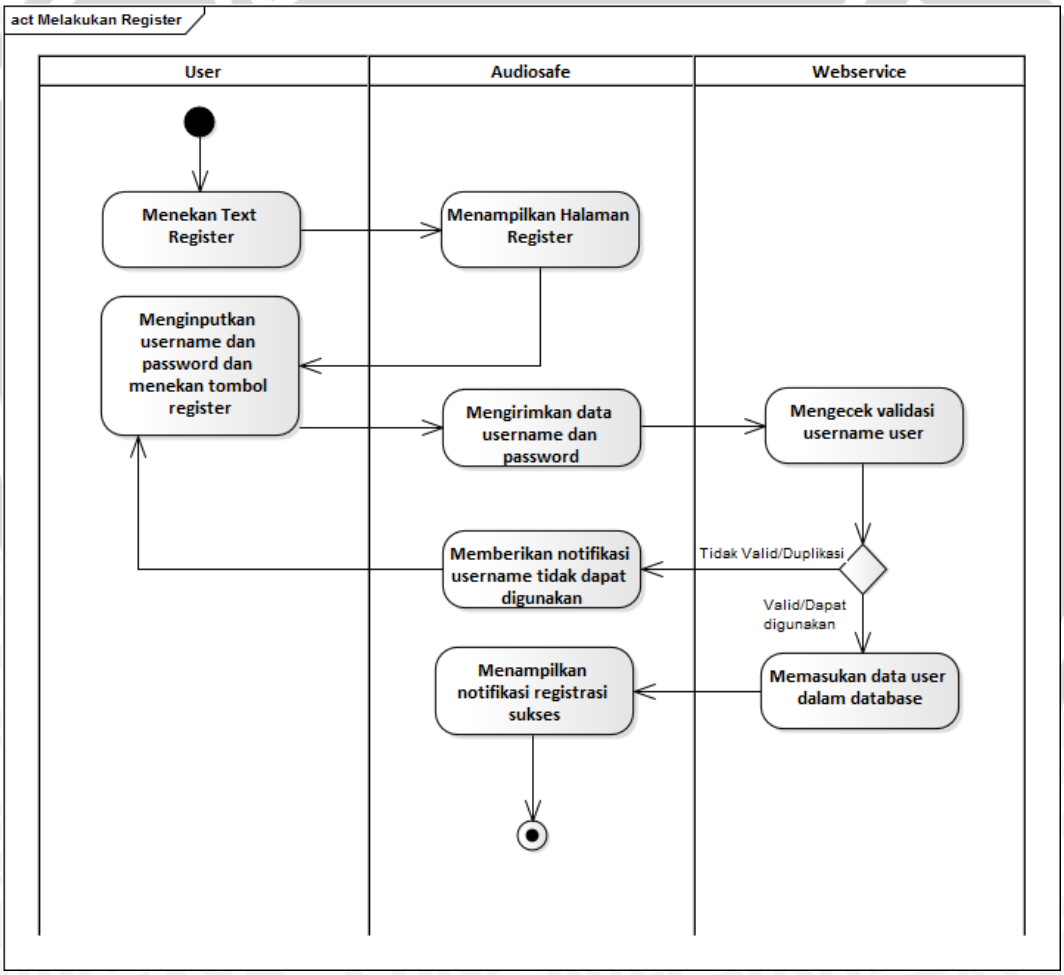
1. *User* mengunggah *digital audio* dengan key yang diinputkan, *digital audio* akan dienkrip dengan key dan kemudian dikirim ke *server,* kemudian server akan menerima *dan* selanjutnya key dan *file* hasil enkripsi tersimpan dalam *database*.
2. Saat *user* mengunduh *digital audio*, maka *server* akan menyediakan *file* hasil enkripsi dikirimkan menuju *device user*.

3. Ketika *user* ingin memutar *digital audio* tersebut, dilakukan proses dekripsi ketika *user* pengunduh sudah disetujui untuk diberikan *privilege allow* atau *approval* dari *user pengunggah*, *file* dalam *local storage* akan didekripsi dengan *key* yang diminta dari server secara otomatis, selanjutnya ketika dekripsi berhasil maka *file* dengan konten asli akan dikirimkan ke memori (RAM), file konten asli hasil dekripsi akan dihapus dengan rentang waktu 1ms setelah sudah berhasil dibaca melalui memori (RAM).

4.2.3 Perancangan Aktifitas Sistem

Perancangan aktifitas sistem menjelaskan bagaimana aktifitas sistem dengan penggunanya sehingga mendapat sebuah alur dan informasi yang jelas dan berguna bagi pengguna yang digambarkan dalam bentuk *activity diagram*. *Activity diagram* dibentuk dengan mereferensi *usecase scenario* pada tahap sebelumnya.

4.2.3.1 *Activity Diagram* Melakukan Register

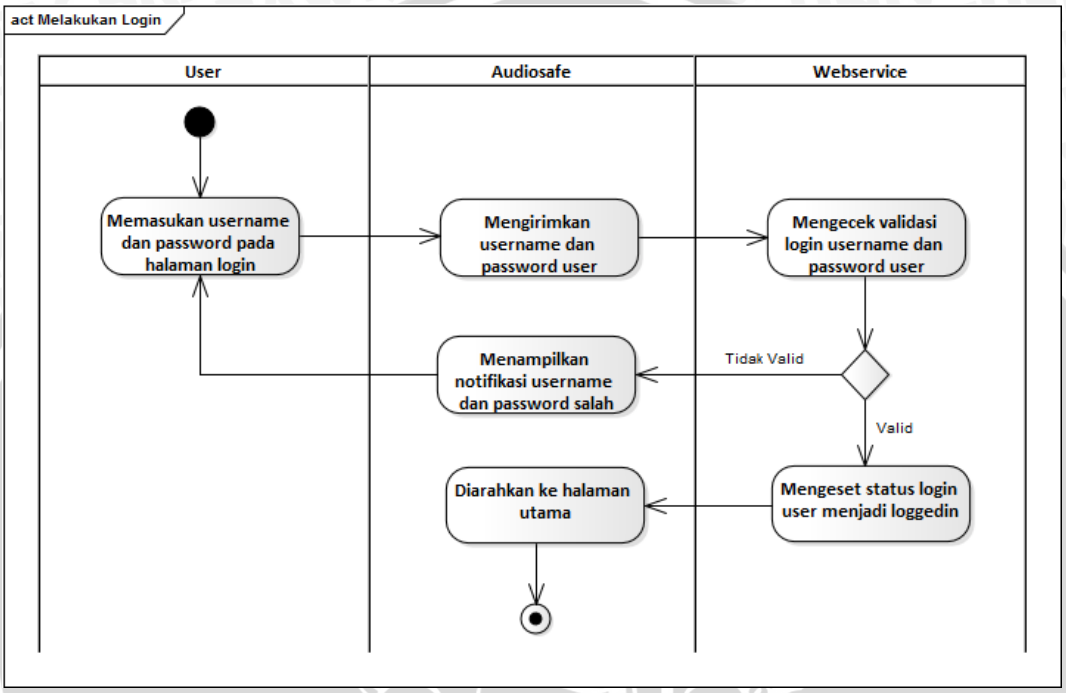


Gambar 4.6 *Activity Diagram* Melakukan Register

Dijelaskan pada Gambar 4.6, aktifiktas yang terjadi saat fungsi login pertama kali membuka aplikasi muncul, *user* akan menekan tombol register yang akan menampilkan halamannya. *User* akan memasukan parameter *username,*

password, dan gambar profilnya. Selanjutnya data tersebut akan dikirimkan ke *webservice* oleh *Audio*safe. Kemudian dilakukan pengecekan validasi *username* apakah mengandung duplikasi atau tidak, jika tidak mengandung duplikasi maka data tersebut akan dimasukkan kedalam *database*, dan selanjutnya akan ditampilkan notifikasi registrasi sukses. Jika mengandung duplikasi, maka akan ditampilkan oleh webservice, notifikasi bahwa *username* sudah digunakan.

4.2.3.2 *Activity Diagram* Melakukan Login

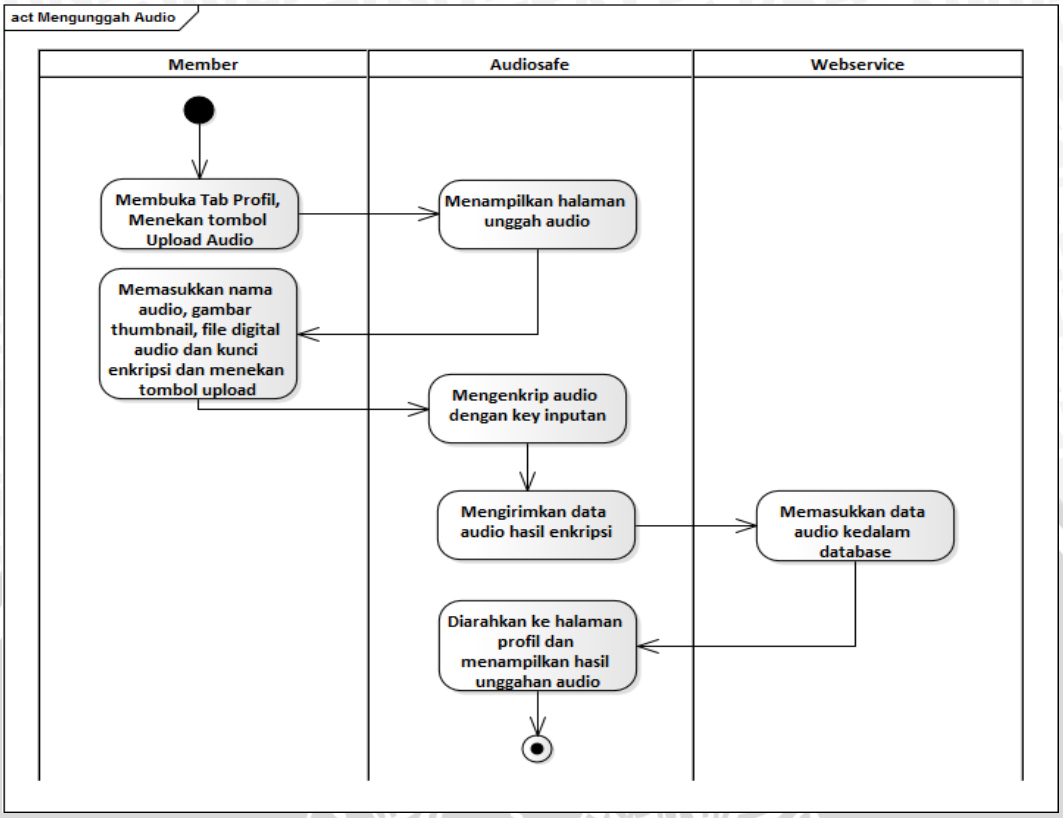


Gambar 4.7 Activity Diagram Melakukan Login

Dijelaskan pada Gambar 4.7, aktifitas melakukan *login*. Setelah *user* berhasil mendaftarkan akunnya, maka *user* diharuskan untuk melakukan *login* untuk dapat menjalankan fungsionalitas dari *Audio*safe. *User* memasukan parameter *username* dan *password* yang selanjutnya dikirimkan ke *webservice* oleh *Audio*safe untuk dicek validasinya apakah valid atau tidak. Jika valid, maka status *login* user di*database* diubah menjadi *loggedin*, hal ini dimaksudkan untuk mencegah satu akun *login* diberbagai macam *device*, selanjutnya akan diarahkan ke halaman utama. Jika tidak valid, maka akan ditampilkan notifikasi bahwa *username* dan *password* salah.

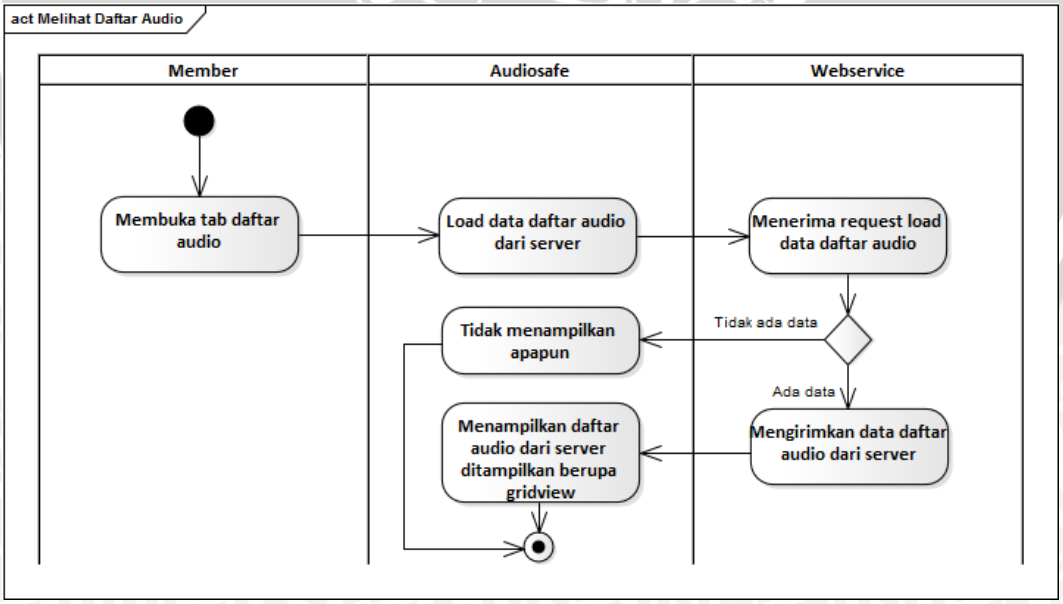
4.2.3.3 *Activity Diagram* Mengunggah *Audio*

Dijelaskan pada Gambar 4.8, alur aktifitas untuk mengunggah *digital audio*. *User* yang sudah *login* menjadi *member* membuka tab profil dan menekan tombol upload *audio* yang selanjutnya akan ditampilan halaman unggah *audio*. *Member* memasukan parameter nama *audio*, *coverart*, *file audio* dan kunci enkripsi yang diinginkan, yang selanjutnya *file audio akan dienkrip* dan akan langsung dikirimkan kepada *webservice* untuk dimasukkan database. Kemudian akan diarahkan kembali ke halaman profil dan menampilkan hasil unggahan *audio*.



Gambar 4.8 *Activity Diagram* Mengunggah *Audio*

4.2.3.4 *Activity Diagram* Melihat Daftar *Audio*

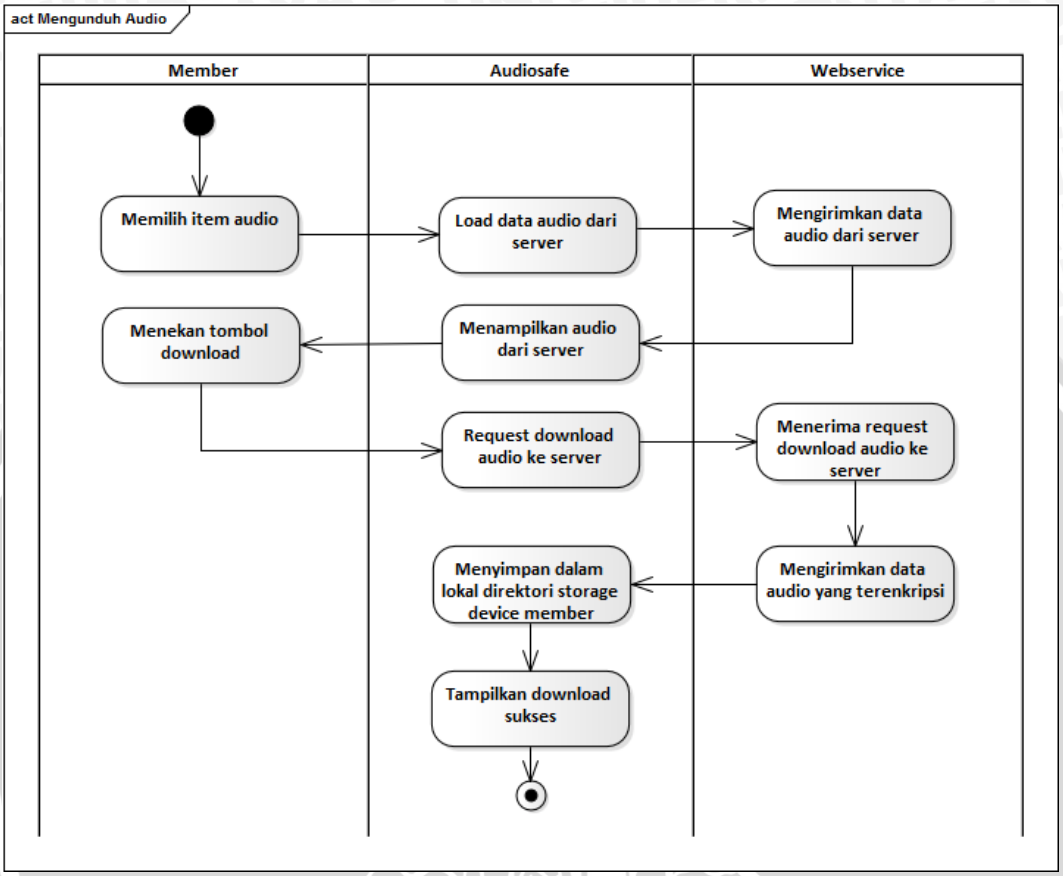


Gambar 4.9 *Activity Diagram* Melihat Daftar *Audio*

Dijelaskan pada Gambar 4.9, alur aktifitas melihat daftar *digital audio*. *Member* menekan tab daftar *audio* yang kemudian data akan diload oleh *Audio*safe dan

diterima requestnya oleh *webservice*. Jika tidak ada data, maka tidak akan ditampilakn apapun. Jika terdapat data *audio* maka akan disajikan dalam bentuk *cardview*.

4.2.3.5 *Activity Diagram* Mengunduh *Audio*

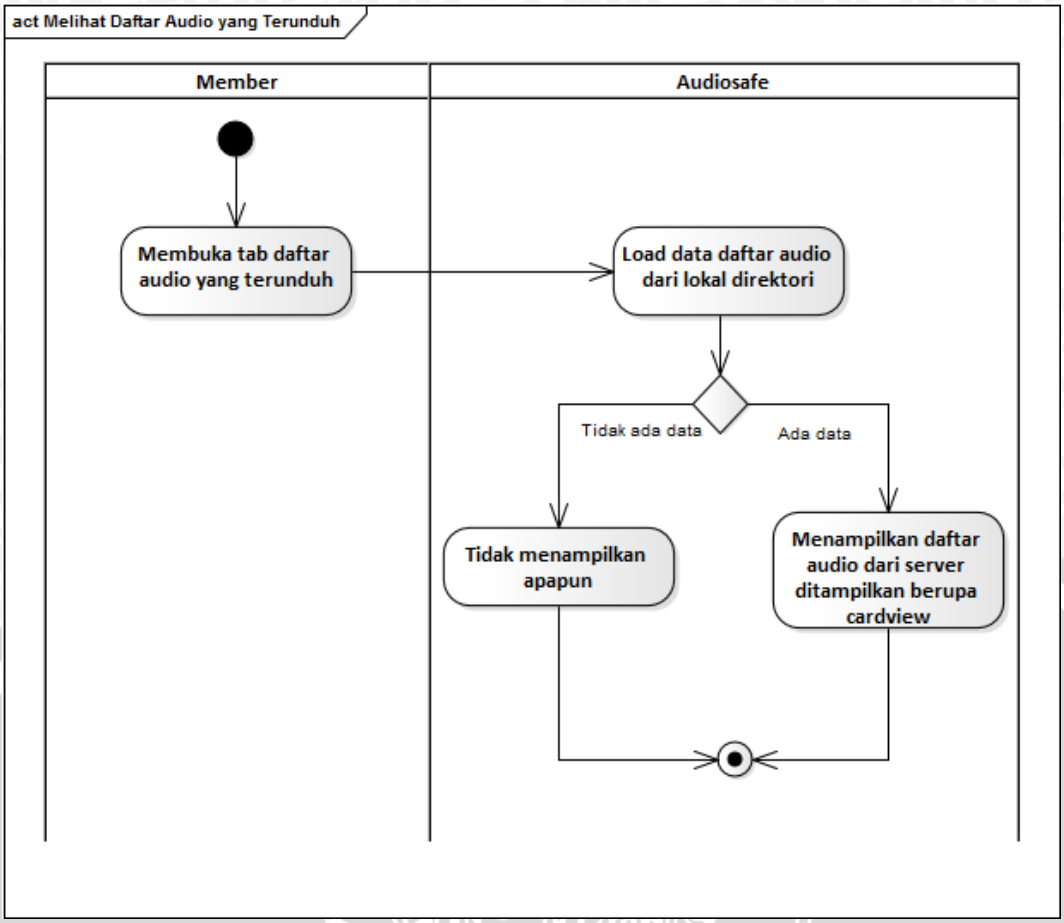


Gambar 4.10 *Activity Diagram* Mengunduh *Audio*

Dijelaskan pada Gambar 4.10, alur dari mengunduh *digital audio*. *Member* akan memilih item *audio* yang langsung ditampilkan detil *audio* dari hasil *load data*. Selanjutnya *member* menekan tombol download yang selanjutnya *direquest* kepada *webservice* untuk menguduh. *Webservice* akan menerima *request* dan *audio* yang akan dikirimkan akan di enkripsi dengan *key* dari *member* pengunggah *audio*. Setelah berhasil terenkripsi maka file akan dikirimkan ke lokal *storage device member*, dan menampilkan notifikasi berhasil mengunduh.

4.2.3.6 *Activity Diagram* Melihat Daftar *Audio* yang Terunduh

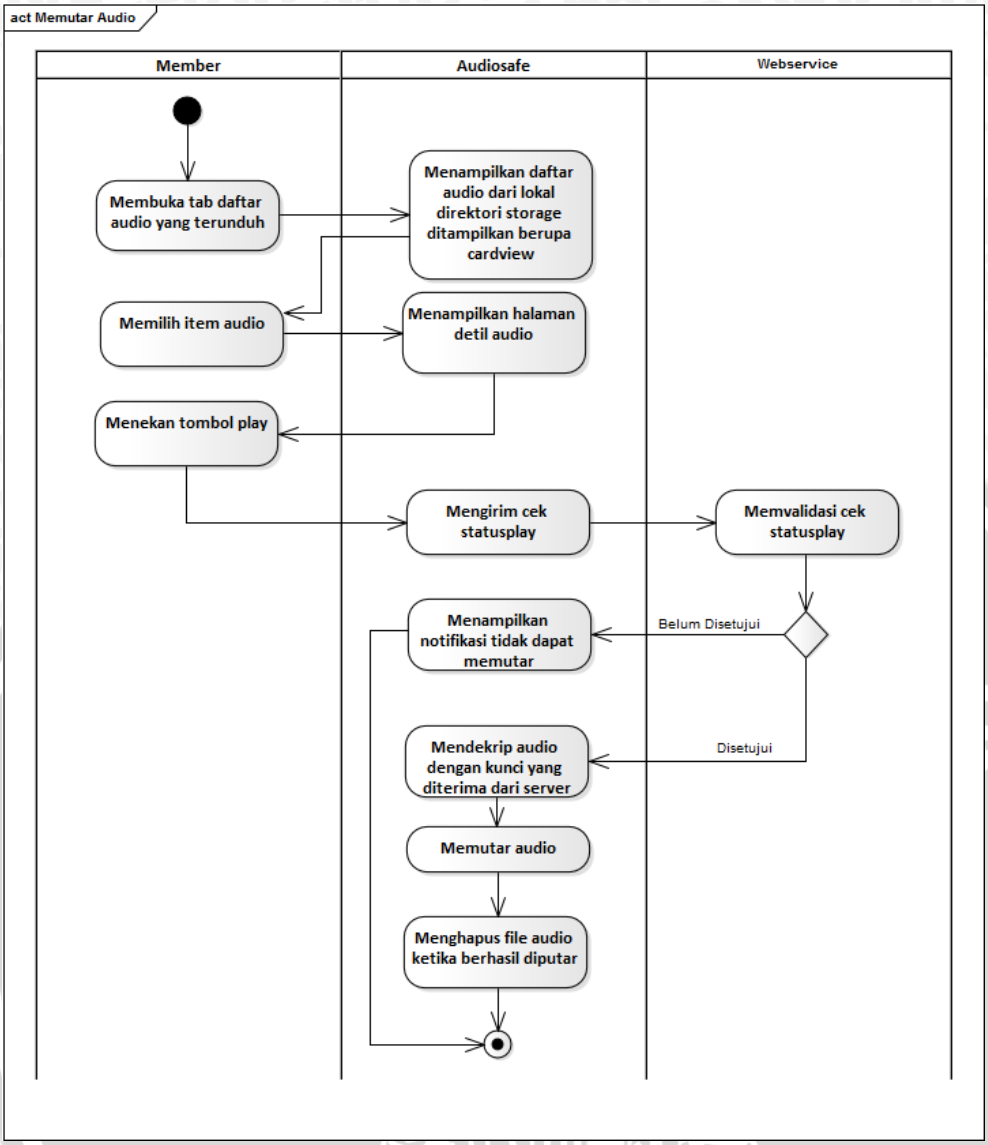
Dijelaskan pada Gambar 4.11, alur aktifitas melihat daftar *digital audio* yang sudah terunduh. *Member* menekan tab daftar *audio* yang terunduh yang kemudian data akan diload ke lokal *storage device member*. Jika tidak ada data, maka tidak akan ditampilakn apapun. Jika terdapat data *audio* maka akan disajikan dalam bentuk *cardview*.



Gambar 4.11 *Activity Diagram* Melihat Daftar *Audio* yang Terunduh

4.2.3.7 *Activity Diagram* Memutar *Audio*

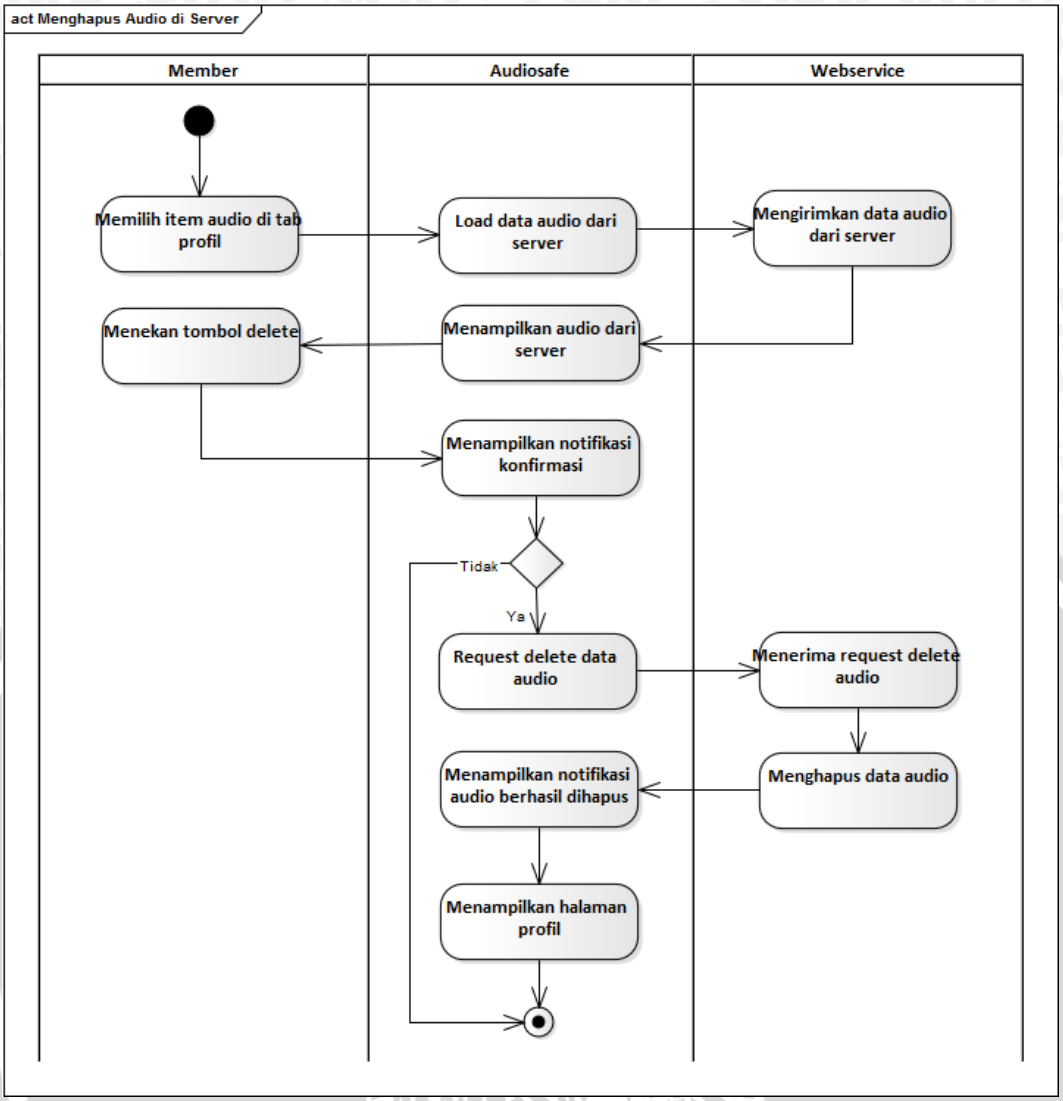
Dijelaskan pada Gambar 4.12, alur aktifitas dari memutar *digital audio*. *Member* membuka tab daftar *audio* yang terunduh kemudian akan ditampilkan daftar *audio* dari lokal *storage*. Selanjutnya *member* memilih item *audio* yang langsung ditampilkan detil *audio*, yang kemudian *member* menekan tombol play jika ingin memutarnya, jika statusplay sudah diijiinkan maka *file* akan didekrip dengan kunci yang dikirim server dan diputar, dan ketika belum disetujui maka akan menampilkan notifikasi.



Gambar 4.12 *Activity Diagram* Memutar *Audio*

4.2.3.8 *Activity Diagram* Menghapus *Audio* di Server

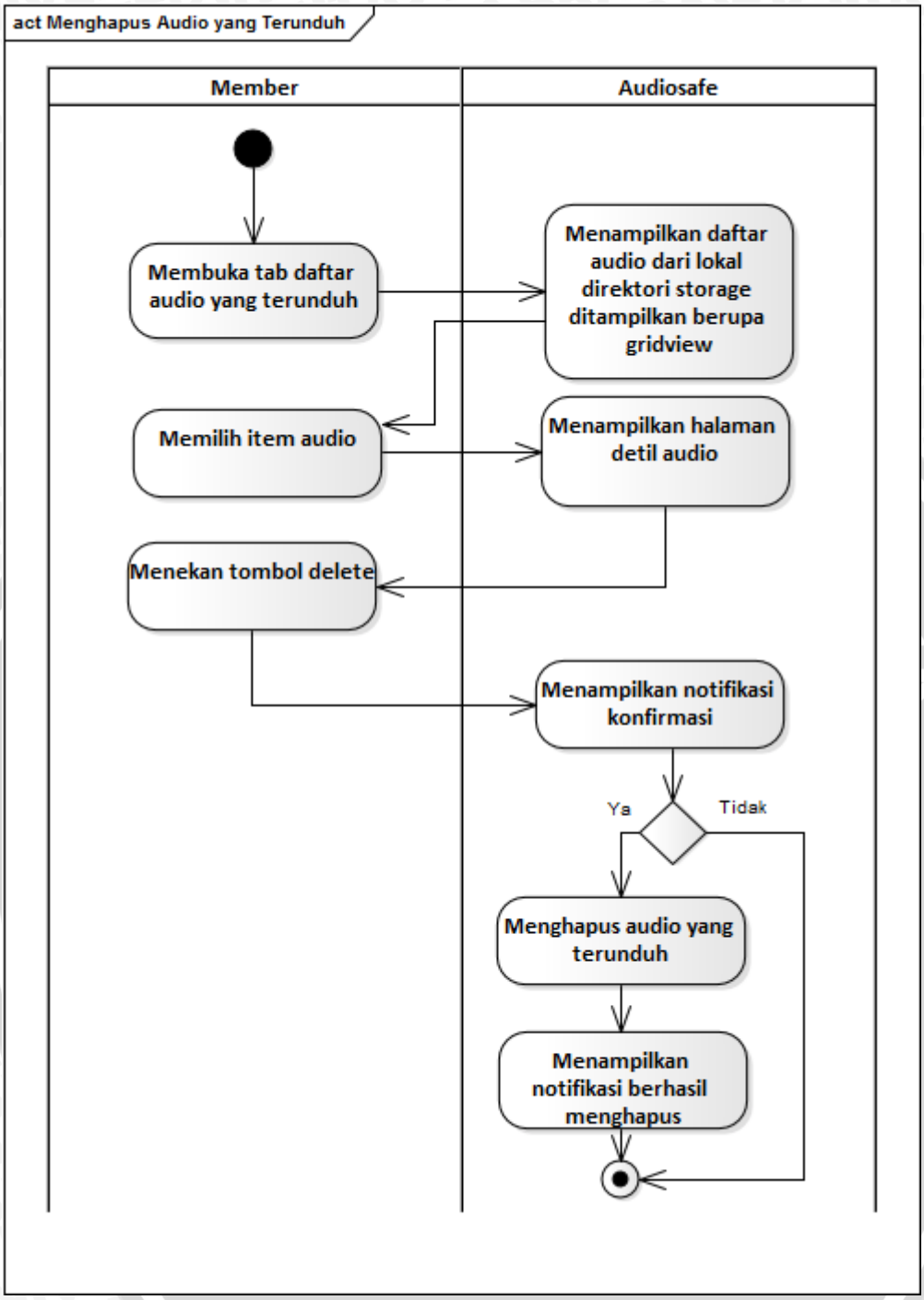
Dijelaskan pada Gambar 4.13, alur menghapus *digital audio* dari hasil unggahan *member* itu sendiri. *Member* menekan tab profil dan memiilh *audio* dari miliknya. Selanjutnya *member* menekan tombol *delete* untuk menghapus, tetapi sebelum dihapus akan dikonfirmasikan kembali apakah yakin dihapus atau tidak, jika benar, maka *Audio*safe akan me*request* untuk menghapus data di *database*, yang selanjutnya diarahkan kembali ke halaman profil. Jika tidak, maka tidak jadi untuk dihapus.



Gambar 4.13 *Activity Diagram* Menghapus *Audio* di Server

4.2.3.9 *Activity Diagram* Menghapus *Audio* di *Audio* yang Terunduh

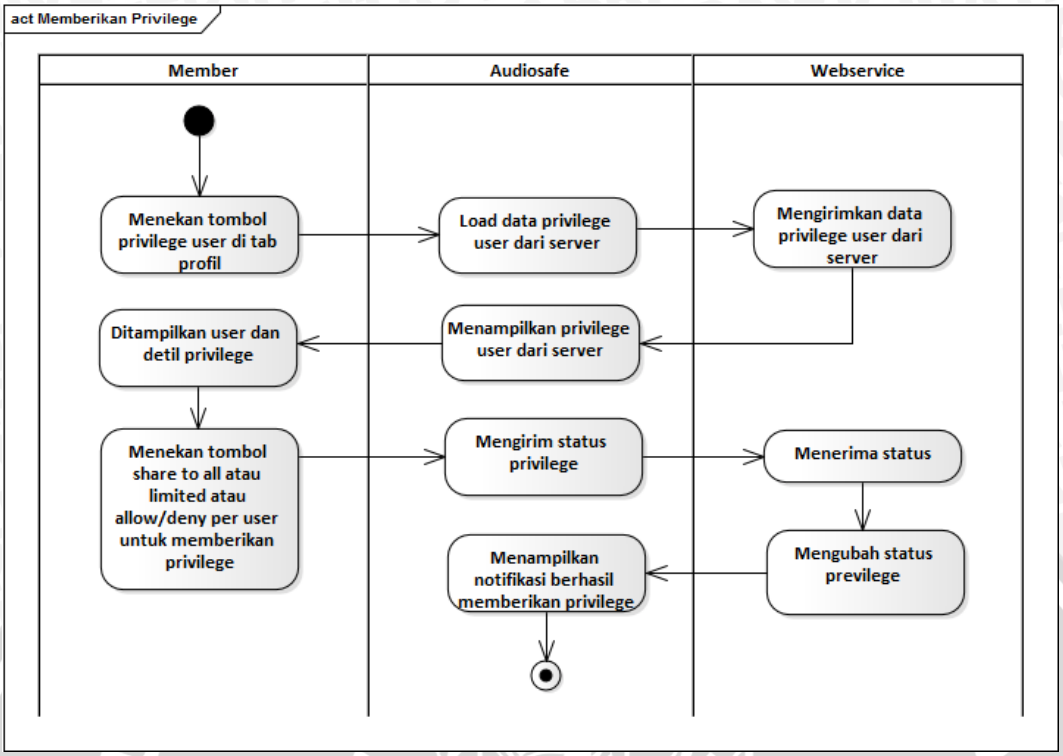
Dijelaskan pada Gambar 4.14, alur menghapus *digital audio* dari hasil unggahan *member* itu sendiri. *Member* menekan tab daftar *audio* yang terunduh dan memiilh *audio*. Selanjutnya *member* menekan tombol *delete* untuk menghapus, tetapi sebelum dihapus akan dikonfirmasikan kembali apakah yakin dihapus atau tidak, jika benar, maka *Audio*safe akan langsung menghapus data di lokal *storage*, yang selanjutnya ditampilkan notifikasi berhasil dihapus. Jika tidak, maka tidak jadi untuk dihapus.



Gambar 4.14 *Activity Diagram* Menghapus *Audio* di *Audio* yang Terunduh

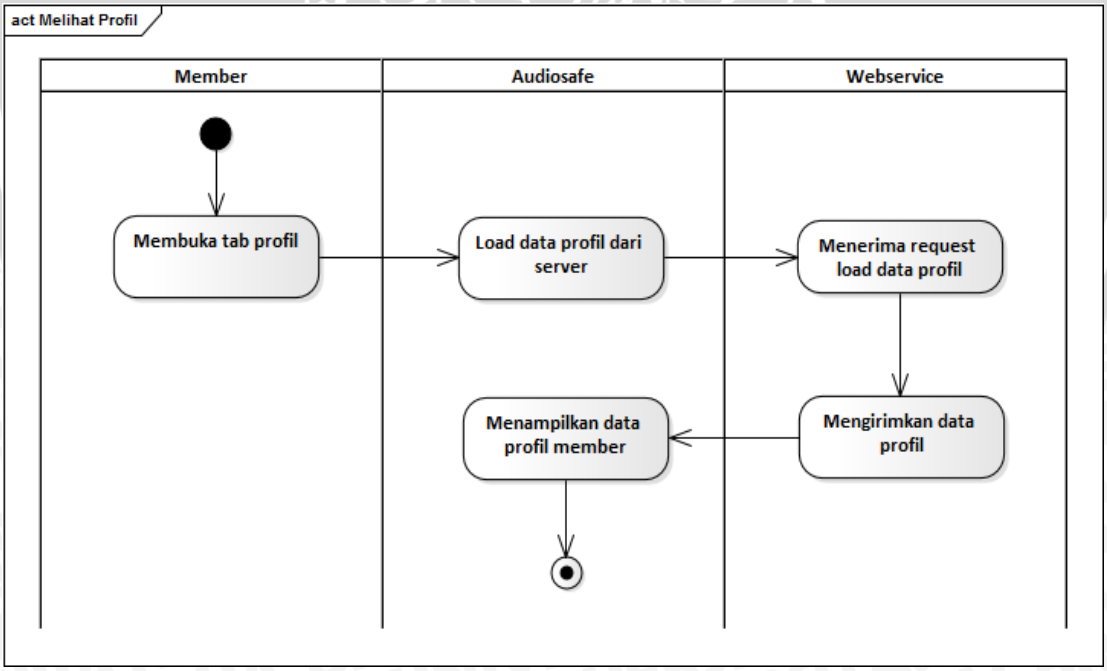
4.2.3.10 *Activity Diagram* Memberikan Privilege

Dijelaskan pada Gambar 4.15, alur aktifitas memberikan *privilege* kepada *member* penerima *previlege*. *Member* menekan tombol *previlege user* di tab profil. Langsung ditampilkan data dari hasil *load webservice*. *Member* menekan item user dan menekan tombol share to all atau limited atau allow/deny per user untuk memberikan *previlege* kepada *member* penerima hak ijin, dan *webservice* akan langsung mengubah *status* di *database*.



Gambar 4.15 *Activity Diagram* Memberikan Privilege

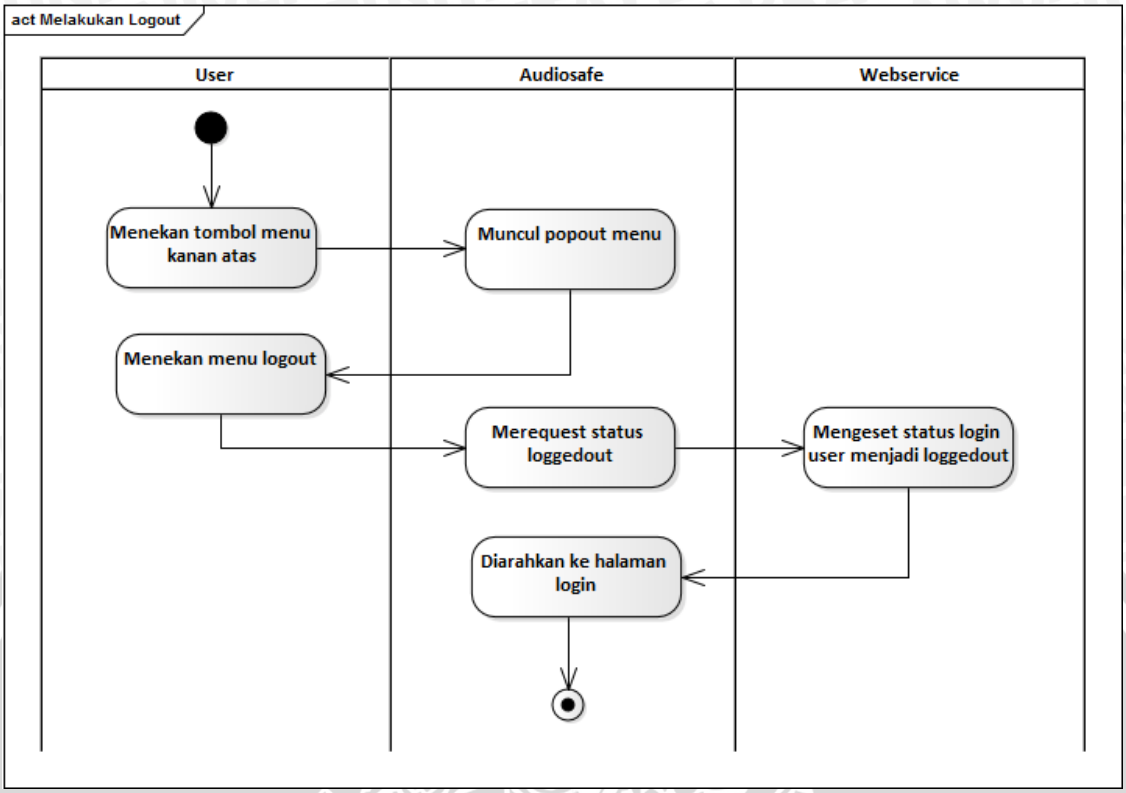
4.2.3.11 *Activity Diagram* Melihat Profil



Gambar 4.16 *Activity Diagram* Melihat Profil

Dijelaskan pada Gambar 4.16, alur aktifitas melihat profil *member*. *Member* menekan tab profil yang selanjutnya *Audio*safe me*request* data dari *webservice* yang selanjutnya akan ditampilkan seluruh data profil dan *digital audio* yang diunggah.

4.2.3.12 *Activity Diagram* Melakukan Logout



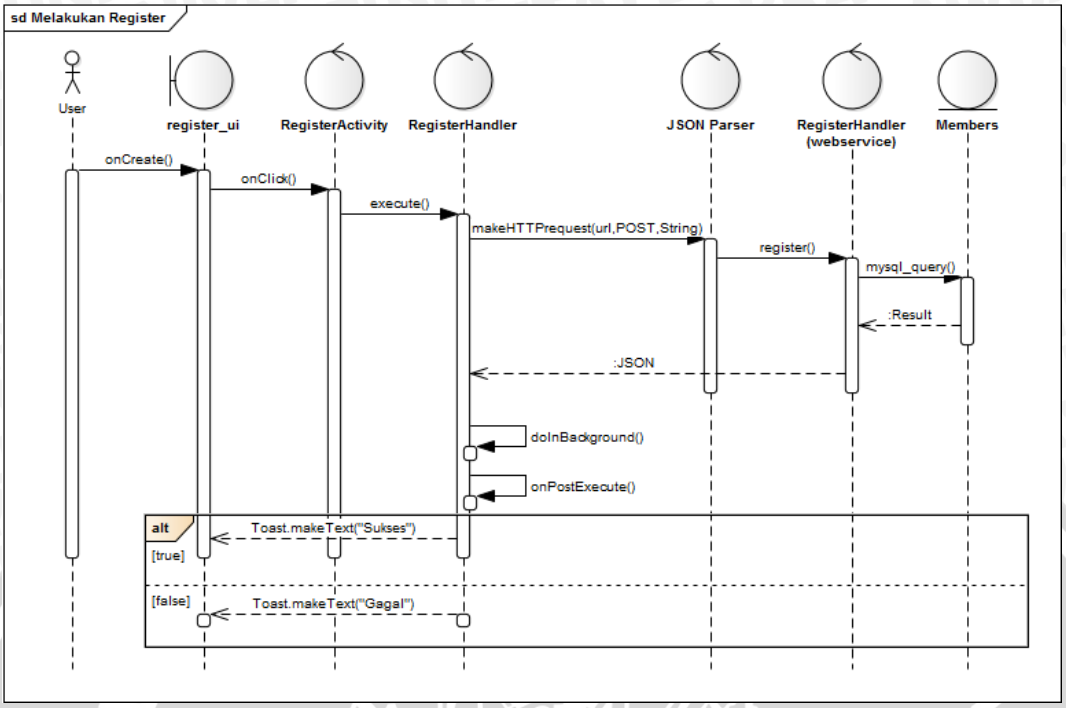
Gambar 4.17 *Activity Diagram* Melakukan Logout

Dijelaskan pada Gambar 4.17, alur aktifitas *member* ketika melakukan *logout* atau keluar dari layanan *Audio*safe. *Member* menekan tombol menu dipojok kanan atas, lalu ditampilkan menu. Saat *member* menekan tombol *logout*, maka *Audio*safe akan me*request* status menjadi ke *loggedout* dan langsung diarahkan ke halaman *login.*

4.2.4 Perancangan Interaksi Antar Objek

Perancangan interaksi objek menjelaskan bagaimana memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut meliputi pengguna, objek dan sebagainya dijelaskan berupa pesan yang digambarkan dalam *sequence diagram*. *Activity diagram* dibentuk dengan mereferensi *usecase scenario* dan bantuan alur aktifktas pada *activity diagram* pada tahap sebelumnya.

4.2.4.1 *Sequence Diagram* Melakukan Register



Gambar 4.18 *Sequence Diagram* Melakukan Register

Gambar 4.18 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi melakukan *register*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* memilih untuk membuka halaman registrasi atau `register_ui`. Tombol *register* ditekan ketika akan melakukan proses dan akan men*trigger* `onClick()` pada klas `RegisterActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas *RegisterHandler*, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `register()` pada klas `RegisterHandler` di *webservice*, yang selanjutnya akan mengolah data pada entitas *Members* dengan fungsi `mysql_query()`.

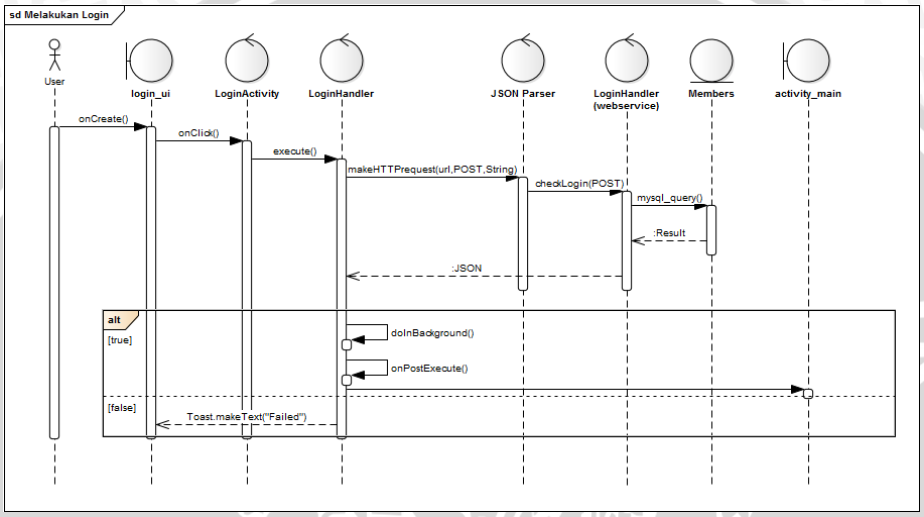
Selanjutnya hasilnya akan langsung dikirimkan ke klas *RegisterHandler* dalam bentuk JSON. Klas `RegisterHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()` dan `onPostExecute()`. Hasilnya akan ditampilkan melalui `register_ui` setelah memvalidasi apakah prosesnya berhasil atau tidak dengan menggunakan fungsi `toast.makeText()`.

4.2.4.2 *Sequence Diagram* Melakukan Login

Gambar 4.19 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi melakukan *login*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* membuka halaman *login* atau `login_ui`. Tombol *login* ditekan ketika akan melakukan proses dan akan men*trigger* `onClick()` pada klas `LoginActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas `LoginHandler`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung

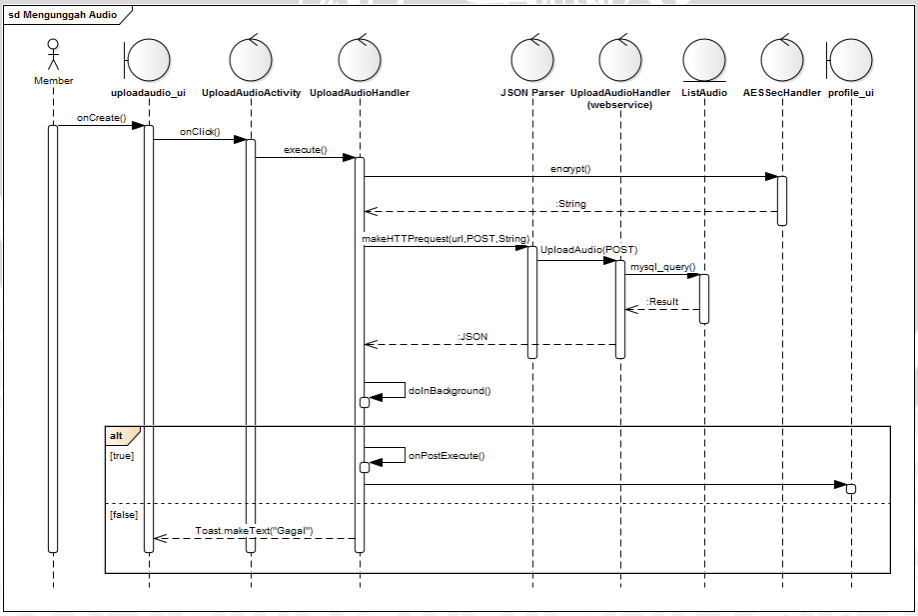
berkomunikasi dengan *webservice* dan men*trigger* fungsi `checkLogin()` pada klas `LoginHandler` di *webservice*, yang selanjutnya akan meminta data pada entitas *Members* dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke klas `LoginHandler` dalam bentuk JSON. Klas `LoginHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()` dan `onPostExecute()`. Hasilnya akan ditampilkan melalui diarahkan langsung ke `activity_main` ketika berhasil memvalidasi prosesnya, dan akan men*trigger* fungsi `toast.makeText()` jika tidak berhasil dalam proses validasinya.



Gambar 4.19 Sequence *Diagram* Melakukan Login

4.2.4.3 *Sequence Diagram* Mengunggah *Audio*

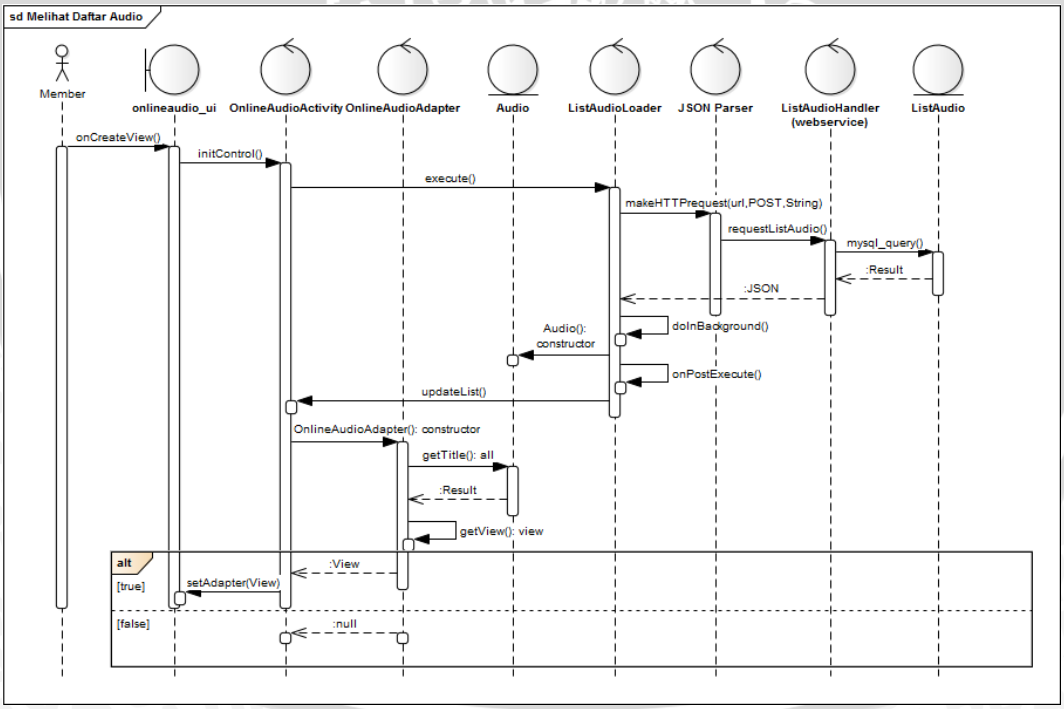


Gambar 4.20 *Sequence Diagram* Mengunggah *Audio*

Gambar 4.20 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi mengunggah *digital audio*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* membuka halaman unggah *audio* atau `uploadaudio_ui`. Tombol *upload* ditekan ketika akan melakukan proses pengunggahan dan akan men*trigger* `onClick()` pada klas `UploadAudioActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas `UploadAudioHandler`, yang kemudian langsung men*trigger* fungsi `encrypt()` dari AESSecHandler, yang selanjutnya langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `UploadAudio()` pada klas `UploadAudioHandler` di *webservice*, yang selanjutnya akan mengolah data pada entitas *ListAudio* dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke `klas UploadAudioHandler` dalam bentuk JSON. Klas `UploadHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()` dan `onPostExecute()`. Hasilnya akan ditampilkan melalui diarahkan langsung ke `profile_ui` ketika berhasil memvalidasi prosesnya, dan akan men*trigger* fungsi `toast.makeText()` jika tidak berhasil dalam proses validasinya.

4.2.4.4 *Sequence Diagram* Melihat Daftar *Audio*



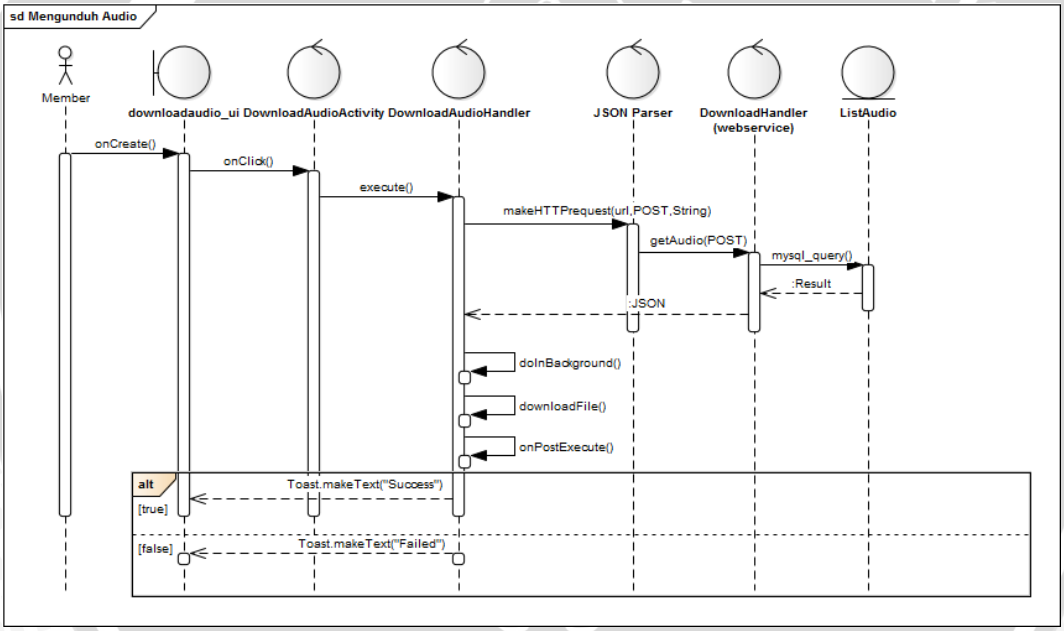
Gambar 4.21 *Sequence Diagram* Melihat Daftar *Audio*

Gambar 4.21 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi melihat daftar *digital audio* yang ada di *server Audio*safe. Sistem akan langsung men*trigger* fungsi `onCreateView()` ketika *user* membuka halaman daftar *audio* atau `listonlineaudio_ui`. Ketika `listonlineaudio_ui` dtampilkan, maka otomatis akan men*trigger* `initControl()` pada klas

`OnlineAudioActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas `ListAudioLoader`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `requestAudio()` pada klas `ListAudioHandler` di *webservice*, yang selanjutnya akan meminta data pada entitas `ListAudio` dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke klas `ListAudioLoader` dalam bentuk JSON. Klas `ListAudioLoader` akan mengolah hasil dari *webservice* melalui fungsi `doInBackground()`, *dan* fungsi itupula yang akan men*trigger* *constructor* entitas *Audio*. Selanjutnya fungsi `onPostExecute()` akan men*trigger* fungsi `updateList()` pada klas `OnlineAudioActivity` yang kemudian mentrigger klas `OnlilneAudioAdapter` untuk mengambil data *digital audio* dari entitas *Audio*. Hasilnya akan ditampilkan melalui bentuk *view* dengan klas `OnlineAudioActivity` untuk dapat menampilkan akan mentrigger fungsi `setAdapter()` dari klas `listonlineaudio_ui`.

4.2.4.5 *Sequence Diagram* Mengunduh *Audio*



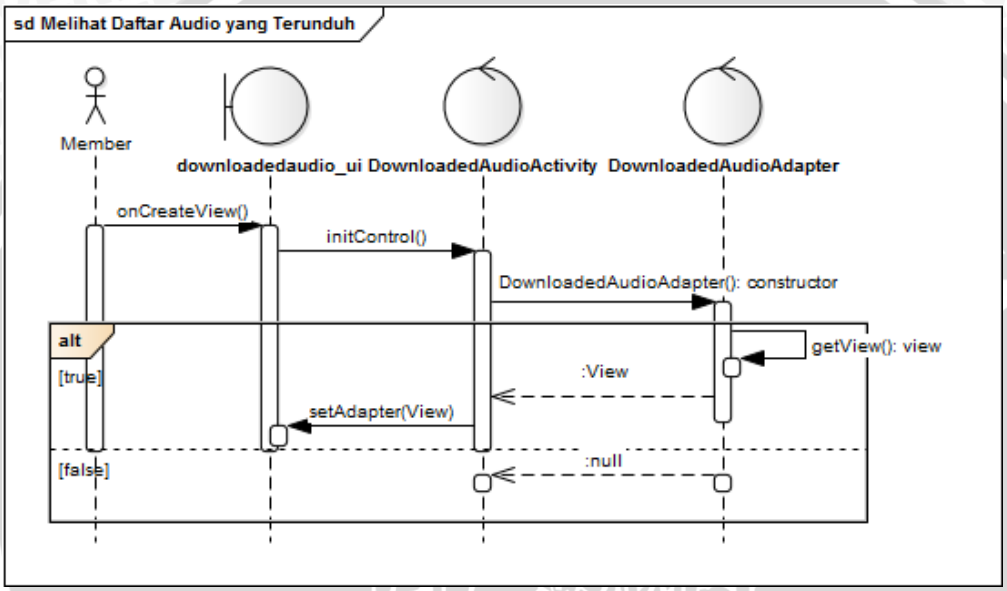
Gambar 4.22 *Sequence Diagram* Mengunduh *Audio*

Gambar 4.22 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi mengunduh *digital audio*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* membuka halaman unduh *audio* atau `downloadaudio_ui`. Tombol *download* ditekan ketika akan melakukan proses dan akan men*trigger* `onClick()` pada klas `DownloadAudioActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas `DownloadAudioHandler`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `getAudio()` pada klas

`DownloadHandler` di *webservice*, yang selanjutnya akan meminta data pada entitas `ListAudio` dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke klas `DownlaodAudioHandler` dalam bentuk JSON. Klas `DownloadAudioHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()`. Hasil JSON yang diolah akan digunakan untuk melakukan fungsi `downloadFile()` dan `onPostExecute()`. Hasilnya akan ditampilkan melalui diarahkan langsung ke `downloadaudio_ui` ketika berhasil memvalidasi prosesnya, dan akan men*trigger* fungsi `toast.makeText()` jika tidak berhasil dalam proses validasinya.

4.2.4.6 *Sequence Diagram* Melihat Daftar *Audio* yang Terunduh



Gambar 4.23 *Sequence Diagram* Melihat Daftar *Audio* yang Terunduh

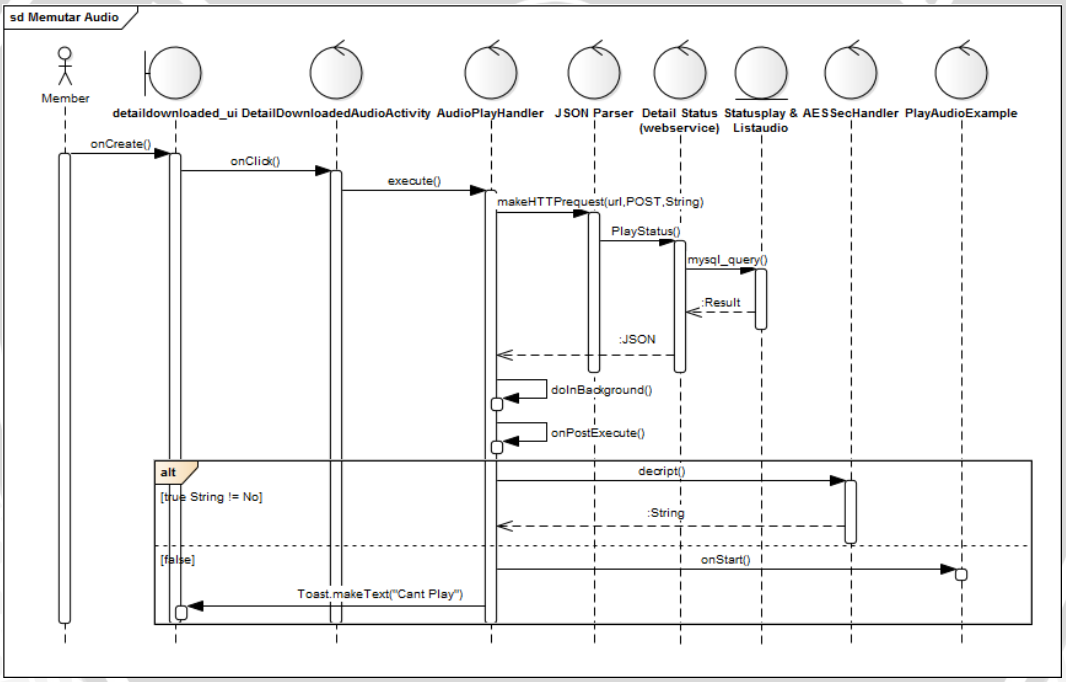
Gambar 4.23 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi melihat daftar *digital audio* yang terunduh. Sistem akan langsung men*trigger* fungsi `onCreateView()` ketika *user* membuka halaman unduh daftar *digital audio* yang terunduh atau `downloadedaudio_ui`. Ketika `downloadedaudio_ui` dibuka maka akan men*trigger* `initControl()` pada klas `DownloadedAudioActivity`. Selanjutnya `DownloadedAudioActivity` akan menjalankan konstruktor dari klas `DownloadedAudioAdapter` dengan fungsi `DownloadedAudioAdapter()` dan klas tersebut akan men*trigger* fungsi `getView()` untuk mengembalikan hasil dari prosesnya yaitu *View*. Hasil tersebut dikirimkan ke `DownloadedAudioActivity` untuk ditampilkan bentuk view melalui *trigger* fungsi `setAdapter()` dari klas `downloadedaudio_ui`.

4.2.4.7 *Sequence Diagram* Memutar *Audio*

Gambar 4.24 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi memutar *digital audio*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* membuka halaman *audio* yang siap diputar atau `detaildownloadedaudio_ui`. Tombol *play* ditekan ketika akan melakukan

proses dan akan men*trigger* `onClick()` pada klas `DetailDownloadedAudioActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas `AudioPlayHandler`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `PlayStatus()` pada klas `Detail status` di *webservice*, yang selanjutnya akan mengolah data pada entitas `Statusplay` dan `Listaudio` dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke klas `PlayAudioHandler` dalam bentuk JSON. Klas `AudioPlayHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()` dan `onPostExecute()`. Selanjutnya akan ditrigger fungsi `decrypt()` yang dimiliki oleh klas `AESSecHandler` dan akan mengembalikkan *String* status keberhasilan dari dekripsi. Selanjutnya akan di*trigger* `onStart()` ke *activity* `PlayAudioExample` ketika berhasil memvalidasi prosesnya, dan akan men*trigger* `fungsi toast.makeText()` jika tidak berhasil dalam proses validasinya.



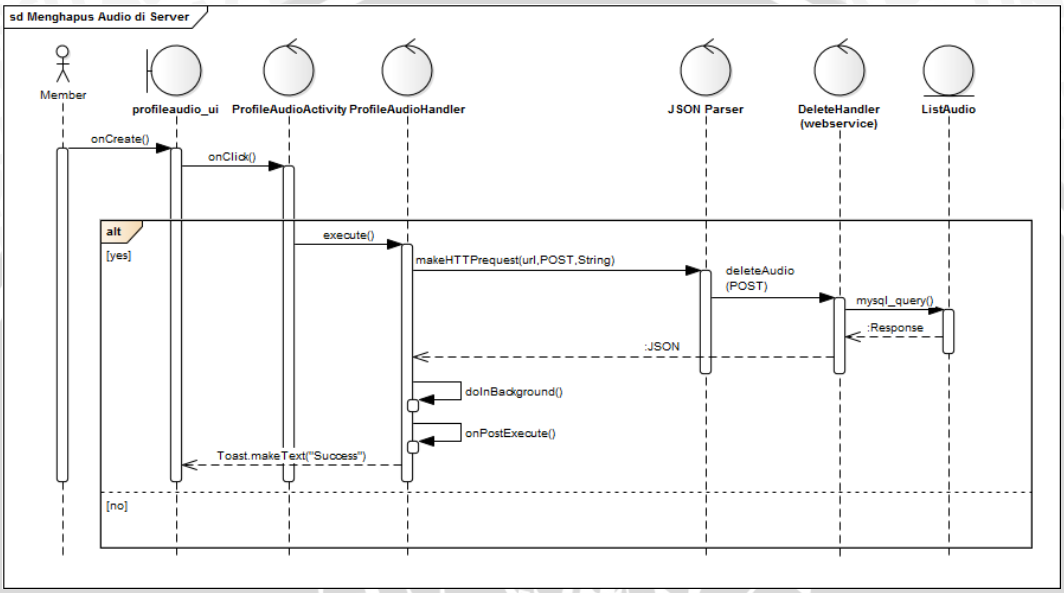
Gambar 4.24 *Sequence Diagram* Memutar *Audio*

4.2.4.8 *Sequence Diagram* Menghapus *Audio* di Server

Gambar 4.25 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi menghapus *digital audio* milik pengunggah yang berada di*server*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* memilih untuk membuka halaman *audio* pengunggah atau `profileaudio_ui`. Tombol *delete* ditekan ketika akan melakukan proses dan akan men*trigger* `onClick()` pada klas `ProfileAudioActivity`. Jika penghapusan dikehendaki maka akan ditrigger fungsi `execute()` yang dimiliki oleh klas `ProfileAudioHandler`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`.

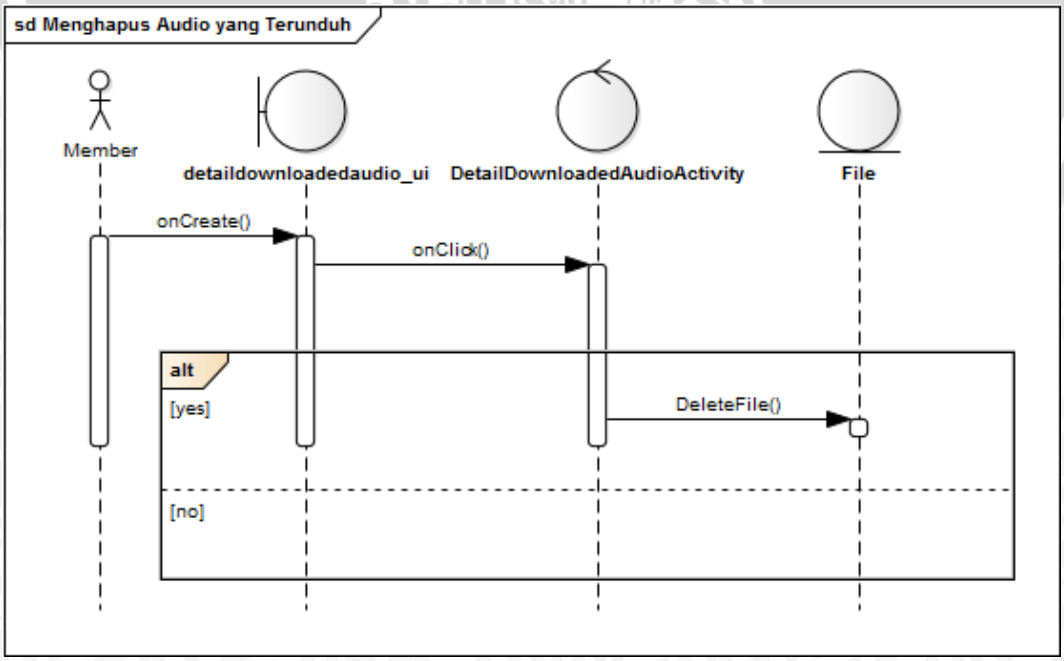
Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `deleteAudio()` pada klas `DeleteHandler` di *webservice*, yang selanjutnya akan mengolah data pada entitas `ListAudio` dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke klas `ProfileAudioHandler` dalam bentuk JSON. Klas `ProfileAudioHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()` dan `onPostExecute()`. Hasilnya akan ditampilkan melalui `profileaudio_ui` setelah memvalidasi dengan menggunakan fungsi `toast.makeText()`.



Gambar 4.25 *Sequence Diagram* Menghapus *Audio* di Server

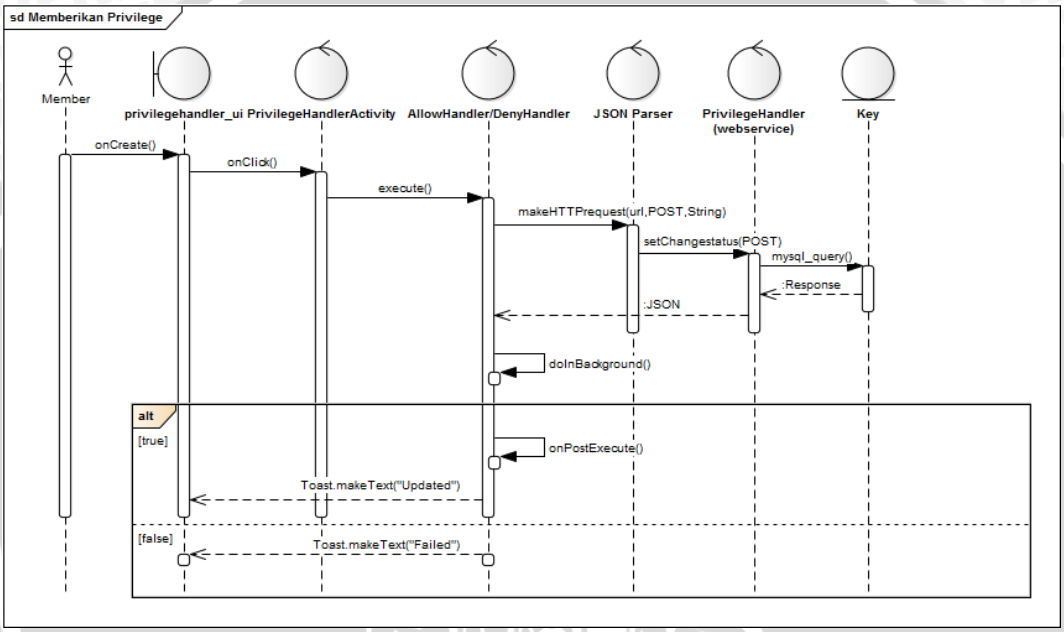
4.2.4.9 *Sequence Diagram* Menghapus *Audio* di *Audio* yang Terunduh



Gambar 4.26 *Sequence Diagram* Menghapus *Audio* di *Audio* yang Terunduh

Gambar 4.26 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi menghapus *digital audio* milik pengunggah yang berada di lokal *storage*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* memilih untuk membuka halaman *digital audio* yang terunduh atau `detaildownloadedaudio_ui`. Tombol *delete* ditekan ketika akan melakukan proses dan akan men*trigger* `onClick()` pada klas `DetailDownloadedAudioActivity`. Jika penghapusan dikehendaki maka akan ditrigger fungsi `DeleteFile()` pada entitas `File`.

4.2.4.10 *Sequence Diagram* Memberikan Privilege



Gambar 4.27 *Sequence Diagram* Memberikan Privilege

Gambar 4.27 adalah menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi memberikan *privilege*. Sistem akan langsung men*trigger* fungsi *on*`Create()` ketika *user* memilih untuk membuka halaman memberikan *privilege* atau `privilegehandler_ui`. Tombol *allow* atau *deny* ditekan ketika akan melakukan proses dan akan men*trigger* `onClick()` pada klas `PrivilegeHandlerActivity`. Selanjutnya akan ditrigger fungsi *execute()* yang dimiliki oleh klas `AllowHandler` atau `DenyHandler`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `setChangestatus()` pada klas `PrivilegeHandler` di `webservice`, yang selanjutnya akan mengolah data pada entitas *Key* dengan fungsi `mysql_query()`.

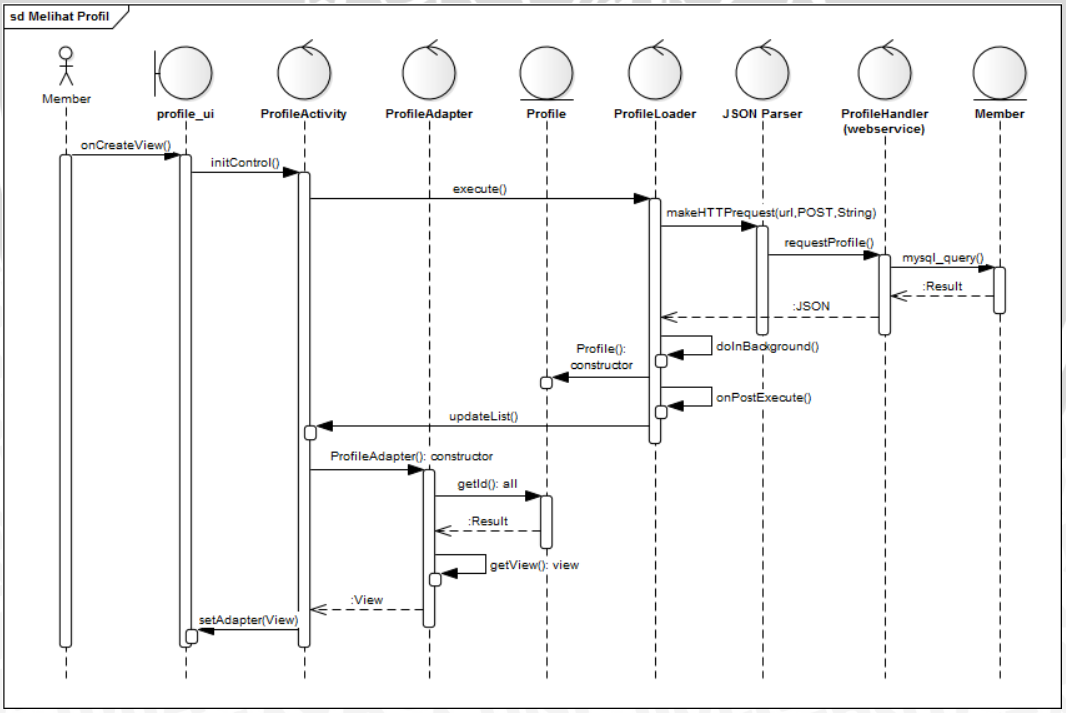
Selanjutnya hasilnya akan langsung dikirimkan ke klas `PrivilegeHandler` dalam bentuk JSON. Klas `AllowHandler` atau `DenyHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()` dan `onPostExecute()`. Hasilnya akan ditampilkan melalui `privilegehandler_ui` setelah memvalidasi

apakah prosesnya berhasil atau tidak dengan menggunakan fungsi `toast.makeText()`.

4.2.4.11 *Sequence Diagram* Melihat Profil

Gambar 4.28 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi melihat profil. Sistem akan langsung men*trigger* fungsi `onCreateView()` ketika *user* membuka halaman profil atau `profile_ui`. Ketika `profile_ui` dtampilkan, maka otomatis akan men*trigger* `initControl()` pada klas `ProfileActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas `ProfileLoader`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas *JSONParser*. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `requestProfile()` pada klas `ProfileHandler` di *webservice*, yang selanjutnya akan meminta data pada entitas `Members` dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke klas `ProfileHandler` dalam bentuk JSON. Klas *Profile* akan mengolah hasil dari *webservice* melalui fungsi `doInBackground()`, *dan* fungsi itupula yang akan men*trigger constructor* entitas `Profile`. Selanjutnya fungsi `onPostExecute()` akan men*trigger* fungsi *updateList()* pada klas `ProfileActivity` yang kemudian mentrigger klas `ProfileAdapter` untuk mengambil data *digital audio* dari entitas `Audio`. Hasilnya akan ditampilkan melalui bentuk *view* dengan klas `ProfileActivity` untuk dapat menampilkan akan mentrigger fungsi `setAdapter()` dari klas `profile_ui`.

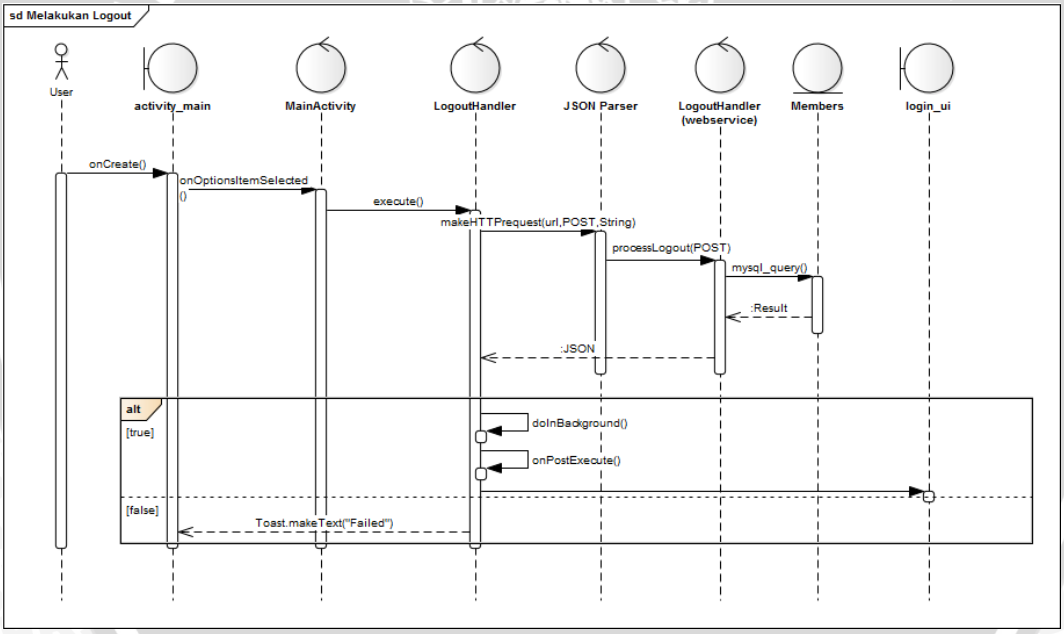


Gambar 4.28 *Sequence Diagram* Melihat Profil

4.2.4.12 *Sequence Diagram* Melakukan Logout

Gambar 4.29 menjelaskan interaksi antar objek dalam sistem untuk menjalalankan fungsi melakukan *login*. Sistem akan langsung men*trigger* fungsi `onCreate()` ketika *user* membuka halaman `activity_main`. Tombol *logout* ditekan ketika akan melakukan proses dan akan men*trigger* `OnOptionsItemSelected()` pada klas `MainActivity`. Selanjutnya akan ditrigger fungsi `execute()` yang dimiliki oleh klas `LogoutHandler`, yang kemudian langsung men*trigger* fungsi `makeHTTPrequest()` dari klas `JSONParser`. Klas `JSONParser` akan langsung berkomunikasi dengan *webservice* dan men*trigger* fungsi `processLogout()` pada klas `LogoutHandler` di *webservice*, yang selanjutnya akan mengolah data pada entitas *Members* dengan fungsi `mysql_query()`.

Selanjutnya hasilnya akan langsung dikirimkan ke klas `LoginHandler` dalam bentuk JSON. Klas `LogoutHandler` akan membaca hasil dari *webservice* melalui fungsi `doInBackground()` dan `onPostExecute()`. Hasilnya akan ditampilkan melalui diarahkan langsung ke `login_ui` ketika berhasil memvalidasi prosesnya, dan akan men*trigger* fungsi `toast.makeText()` jika tidak berhasil dalam proses validasinya.



Gambar 4.29 *Sequence Diagram* Melakukan Logout

4.2.5 Perancangan Klas

Perancangan Klas menjelaskan struktur sistem berdasarkan pemodelan dan hubungan antar klas-klas yang akan dibuat dalam membangun sistem. Didalam klas akan didefinisikan tentang atribut dan method yang akan digunakan untuk membangun sistem. Terlampir dalam Lampiran A Perancangan Klas Diagram

4.2.6 Perancangan Basis Data

Sistem menggunakan basis data untuk dapat menyimpan berbagai data terkait dengan sistem. *Audio*safe menggunakan basis data pada sisi server untuk menyimpan data yang berhubungan dengan fungsionalitas sistem yaitu tentang *list audio*, *member, key* dan *keystatus* yang digunakan oleh *Audio*safe pada perangkat bergerak. Tabel *listaudio*, berisi tentang segala informasi *audio* yang diunggah. Tabel *member*, berisi tentang segala informasi pengguna yang terdaftar. Dan tabel *statusplay* digunakan untuk memuat tentang seluruh informasi *privilege* dari *member* ke *member* lain.

Berikut ini adalah struktur tabel dan relasi antar tabel yang digunakan untuk menyimpan data dalam *database webservice Audio*safe :

Tabel 4.19 Basisdata Tabel List*Audio*

No	*Column*	*Type*	Keterangan
1.	*audioID*	*int*	Nomor identifikasi *audio*, yang digunakan sebagai identitas *audio* untuk transaksi antar *server-client* dan bersifat unik.
2.	*uploaderID*	*int*	Nomor identifikasi member, yang digunakan sebagai identitas member pengunggah audio yang merelasi memberID pada tabel members.
3.	*title*	*varchar*	Judul *audio* yang tersimpan dalam *server* yang nanti digunakan untuk ditampilkan dalam daftar *audio* pada *Audio*safe.
4.	*key*	*varchar*	Informasi *key* yang akan *user* gunakan untuk membuka enkripsi *file audio*.
5.	*coverpath*	*varchar*	Letak gambar cover *audio* yang nantinya akan ditampilkan pada *Audio*safe.
6.	*audiopath*	*varchar*	Letak *file audio* yang tersimpan dalam direktori lokal *server.*

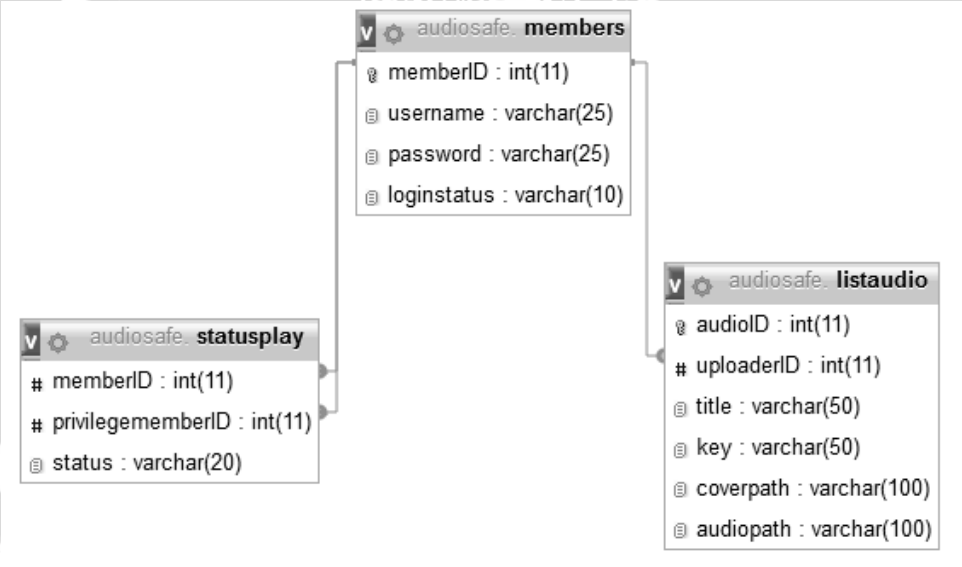
Tabel 4.20 Basisdata Tabel *Member*

No	*Column*	*Type*	Keterangan
1.	*memberID*	*int*	Nomor identifikasi member, yang digunakan sebagai identitas member untuk transaksi antar *server-client* dan bersifat unik.
2.	*username*	*varchar*	Informasi *username* yang terdaftar dalam sistem.
3.	*password*	*varchar*	Informasi *password* yang *user* gunakan yang terdaftar dalam sistem.
4.	*loginstatus*	*varchar*	Informasi status *login.* Informasi yang digunakan untuk mendukung fitur satu *login* akun di satu *device*. *Member* yang

			statusnya *loggedin* tidak dapat *login* di *device* lain sebelum member melakukan *logout* di *device* tersebut.

Tabel 4.21 Basisdata Tabel *Statusplay*

No	*Column*	*Type*	Keterangan
1.	*memberID*	*int*	Nomor identifikasi member, yang digunakan sebagai identitas member pemberi *privilege* yang merelasi memberID pada tabel members.
2.	*privilegememberID*	*int*	Nomor identifikasi member, yang digunakan sebagai identitas member penerima *privilege* yang merelasi memberID pada tabel members.
4.	*status*	*int*	Informasi status yang akan *user* gunakan untuk mengijinkan member dapat diijinkan memutar audio maupun tidak.



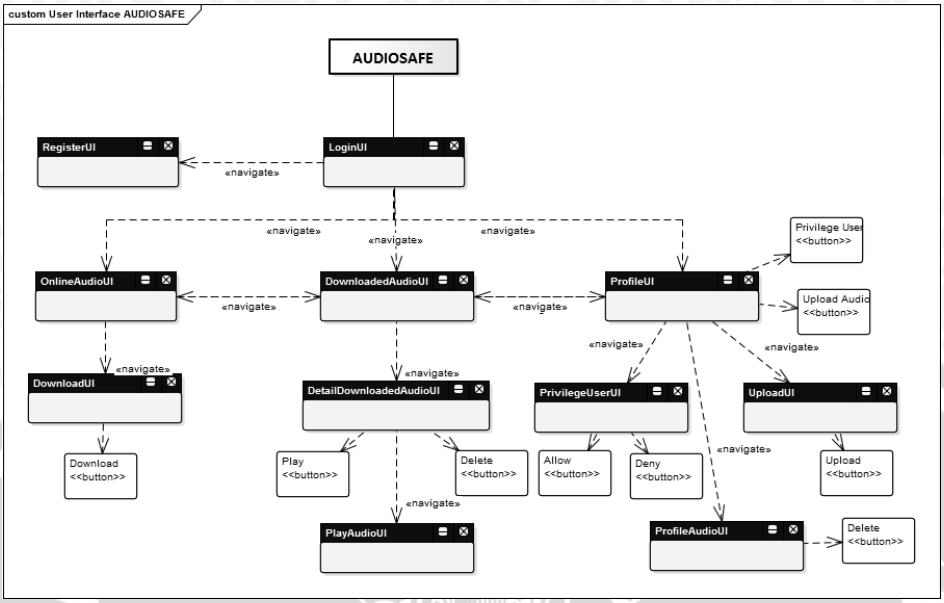
Gambar 4.30 Relasi Antar Tabel Database *Audiosafe*

Dalam gambar 4.31 tabel *members*, memberID berperan sebagai primary key dalam tabel. Tabel *listaudio, audioID* berperan sebagai primary key dalam tabel dan *uploaderID* berelasi dengan *memberID* pada tabel *members.* Tabel statuspplay *tabelmemberID dan privilegememberID* berelasi dengan *memberID* pada tabel *members* sehingga membuat menjadi *foreign key.*

4.2.7 Perancangan Antarmuka

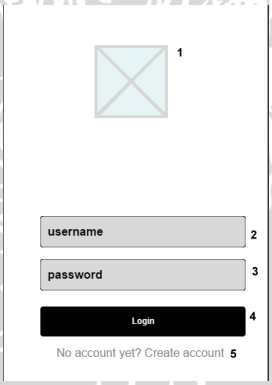
Perancangan antarmuka membahas tentang perancangan antarmuka yang akan ditampilkan kepada pengguna yang bertujuan untuk menjelankan fungsi yang disediakan oleh sistem. Dibawah ini merupakan *navigation map* untuk

menggambarkan seluruh halaman dan hubungannya, *content* dan menu dari perangkat lunak *Audio*safe. Berikut pada Gambar 4.31 adalah *navigation map* keseluruhan dari Audiosafe.



Gambar 4.31 Navigation Map *Audio*safe

4.2.7.1 Antarmuka Melakukan Login



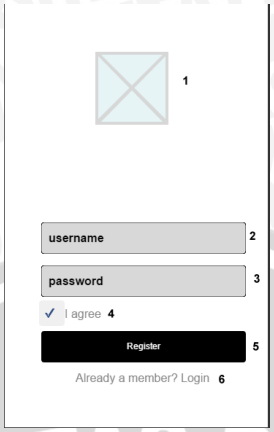
Komponen :

a. *Audio*safe Logo(1) : *ImageView*
b. *username*(2), *password*(3) : *EditText*
c. *Login* (4): *Button*
d. *Register* (5) : TextLink

Gambar 4.32 Antarmuka Melakukan Login

Gambar 4.32 menjelaskan antarmuka user saat *login*. Pada saat perangkat lunak dibuka dan user tidak tercatat melakukan login, maka *user* akan ditampilkan dengan halaman ini. Dalam halaman *login* terdapat dua form yang digunakan untuk memasukkan *username* dan *password* untuk *login*. Disediakan juga dua tombol yaitu tombol *login* dan *register,* tombol *login* digunakan untuk melakukan proses *login* ketika sudah memasukan *username* dan *password,* dan textlink register digunakan untuk *user* yang belum pernah mendaftar dalam sistem ini.

4.2.7.2 Antarmuka Melakukan Register



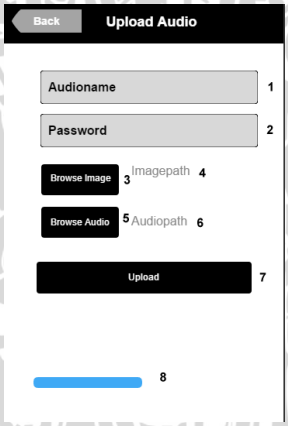
Komponen :

a. *Audio*safe Logo (1) : *ImageView*
b. *username* (2), *password* (3) : *EditText*
c. *Check agreement* (4) : *CheckButton*
d. *Register* (5) : *Button*
e. *Login (6) : TextLink*

Gambar 4.33 Antarmuka Melakukan Login

Gambar 4.33 menjelaskan antarmuka saat *user* menekan tombol *register*. *User* akan dihadapkan pada halaman ini, dimana *user* harus memasukan paramater *username*, *password*. Setelah terisi, disediakan tombol *register* untuk melakukan prosesnya.

4.2.7.3 Antarmuka Mengunggah *Audio*



Komponen :

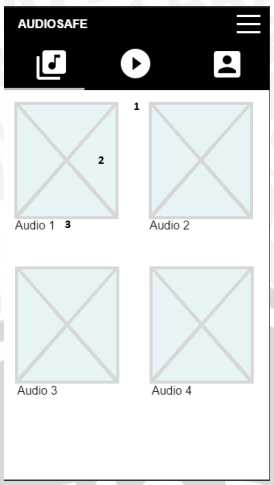
a. *Audio Name* (1), *Key*(2) : *EditText*
b. *Imagepath*(4), *Audiopath*(6) : *Textview*
c. *Browse Image* (3), *Browse Audio*(5), Upload (7) : *Button*
d. *Progessbar (8)*

Gambar 4.34 Antarmuka Mengunggah *Audio*

Gambar 4.34 menjelaskan antarmuka saat *member* menekan tombol *upload audio*. *Member* akan dihadapkan pada halaman ini, dimana *member* harus memasukan nama *audio*, *key,* tombol *browse image dan tombol browse audio* untuk mencari file didirektori yang digunakan dalam mengunggah *audio*. Setelah terisi, disediakan tombol *upload* untuk melakukan proses pengunggahan *audio*, dan akan muncul progressbar hasil dari progress proses pengunggahan.

4.2.7.4 Antarmuka Melihat Daftar *Audio*

Gambar 4.35 diatas, adalah tampilan dimana ketika *member* menekan tab daftar *digital audio* online dari server. Maka akan ditampilkan dalam bentuk *cardview,* dan ditampilkan *coverart* dan judul *audio* dalam setiap itemnya.

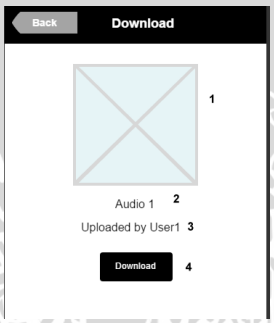


Komponen :

a. Cardview area (1) : *Cardview*
b. *Audio picture* (4) : *ImageView*
c. Judul *Audio* (3): *Textview*

Gambar 4.35 Antarmuka Melihat Daftar *Audio*

4.2.7.5 Antarmuka Mengunduh *Audio*



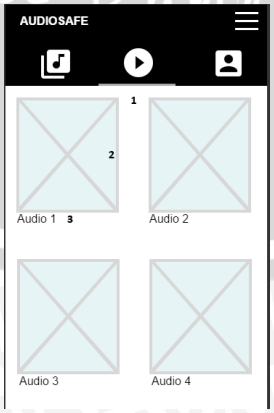
Komponen :

a. Gambar *Audio* (1) : *ImageView*
b. Judul *Audio*(2), Uploader(3) : *TextView*
c. *Download* (4) : *Button*

Gambar 4.36 Antarmuka Mengunduh *Audio*

Gambar 4.36 diatas, adalah tampilan dimana ketika *member* menekan item *audio* dari daftar *audio* online dari server. Maka akan ditampilkan halaman antarmuka dalam mengunduh *audio*. Dimana disajikan *coverart*, judul *audio dan pengunggah audio*, beserta tombol yaitu, *download* yang digunakan untuk mengunduh file *audio* dari sistem.

4.2.7.6 Antarmuka Melihat Daftar *Audio* yang Terunduh



Komponen :

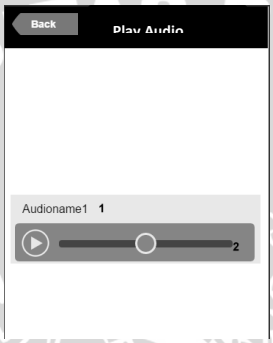
a. Cardview area (1) : *Cardview*
b. *Audio picture*(4) : *ImageView*
c. Judul *Audio* (3): *Textview*

Gambar 4.37 Antarmuka Melihat Daftar *Audio* Terunduh

Gambar 4.37 diatas, adalah tampilan dimana ketika *member* menekan tab daftar *digital audio* online dari daftar yang terunduh yang berada di lokal *storage member*. Maka akan ditampilkan dalam bentuk *cardview,* dan ditampilkan *coverart* dan judul *audio* dalam setiap itemnya.

4.2.7.7 Antarmuka Memutar *Audio*

Gambar 4.38 menjelaskan antarmuka yang ditampilkan ketika sudah memilih item *audio*, menekan tombol *play*, dan sudah mendapat ijin untuk memutar dan diberikan *key* untuk dekripsi *file*. Komponen penyusun halaman ini adalah activity *intent android multimedia* yang akan membaca file dan merendernya kedalam halaman ini. Dalam halaman ini akan disediakan navigasi playback, yang dapat berfungsi, *play, pause,* dan menavigasi dibagian mana *audio* bisa diputar.

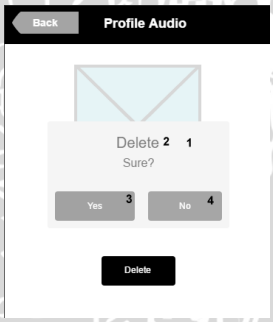


Komponen :

a. Judul *Audio* (1) : *TextView*
b. Playback Rendering Area (2)

Gambar 4.38 Antarmuka Memutar *Audio*

4.2.7.8 Antarmuka Menghapus *Audio* di Server



Komponen :

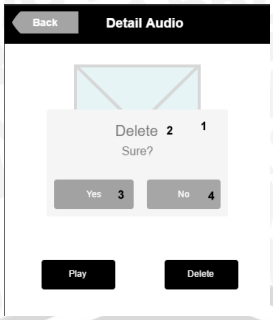
a. *Notification* (1) : Popup
b. *Delete* (2) : *TextView*
c. *Yes* (3), *No* (4) : *Button*

Gambar 4.39 Antarmuka Menghapus *Audio* di Server

Gambar 4.39 diatas, ditampilkan antarmuka dalam penghapusan *digital audio* di *server* dari *member* pengunggah *audio*. Ketika *member* menekan tombol *delete* maka akan ditampilkan notifikasi konfirmasi untuk menghapus, *yes* untuk melakukan proses penghapusan dan *no* untuk membatalkan.

4.2.7.9 Antarmuka Menghapus *Audio* di Daftar yang Terunduh

Gambar 4.40 diatas, ditampilkan antarmuka dalam penghapusan *digital audio* di daftar yang terunduh (dalam lokal *storage member*). Ketika *member* menekan tombol *delete* maka akan ditampilkan notifikasi konfirmasi untuk menghapus, *yes* untuk melakukan proses penghapusan dan *no* untuk membatalkan.

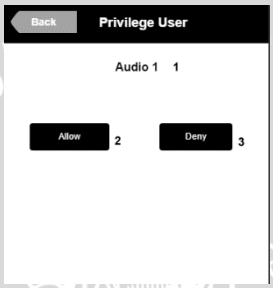


Komponen :

a. *Notification* (1) : Popup
b. *Delete* (2) : *TextView*
c. *Yes* (3), *No* (4) : *Button*

Gambar 4.40 Antarmuka Menghapus *Audio* di Daftar yang Terunduh

4.2.7.10 Antarmuka Memberikan Privilege



Komponen :

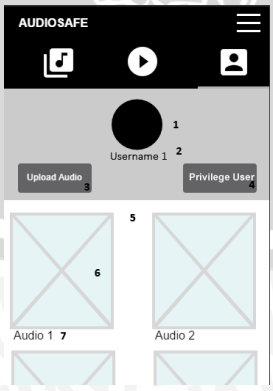
a. Judul *Audio* (1),
b. Allow (2), Deny (3) : *Button*

Gambar 4.41 Antarmuka Memberikan Privilege

Gambar 4.41 menjelaskan antarmuka yang disajikan ketika menekan tombol *privilege user* dalam tab profil. *Member* akan disuguhkan nama *user* penerima *privilege*. Disediakan tombol allow dan deny yang berfungsi untuk menyetujui maupun menolak dalam pemutaran *digital audio*.

4.2.7.11 Antarmuka Melihat Profil

Gambar 4.42 menjelaskan antarmuka yang disajikan ketika *member* menekan tab profil. Ditampilkan gambar profil dan nama dari *member*. Selain itu disediakan juga dua tombol yaitu, *upload audio* yang berfungsi untuk mengunggah *audio* dan *privilege user* yang digunakan untuk memberikan *privilege* kepada member lain.



Komponen :

a. Gambar Profil (1) : *ImageView*
b. *Username* (2) : *TextView*
c. *Upload Audio* (3), *Privilege User*(4) : *Button*
d. Cardview area (1) : *Cardview*
e. *Audio picture* (4) : *ImageView*

Gambar 4.42 Antarmuka Melihat Profil

BAB 5 IMPLEMENTASI PERANGKAT LUNAK

Dalam bab ini akan dibahas tentang implementasi perangkat lunak berdasarkan dari hasil analisa kebutuhan dan perancangan perangkat lunak yang telah dilakukan pada tahap sebelumnya. Implementasi akan dilakukan dalam beberapa tahap antara lain, dimulai dari mendefinisikan spesifikasi sistem pembangun perangkat lunak sekaligus spesifikasi sistem guna berjalannya perangkat lunak, selanjutnya adalah batasan batasan dalam proses implementasi, implementasi *Class* berdasarkan rancangan pada tahap rancangan, Implementasi kode program berdasarkan algroritma dari fungsional yang memenuhi kebutuhan perangkat lunak, dan pada tahap terakhir yaitu, implementasi pada antarmuka perangkat lunak.

5.1 Spesifikasi Sistem

Berdasarkan analisa kebutuhan dan perancangan perangkat lunak yang dirancang pada tahap sebelumnya, memberikan hasil yang digunakan dalam penentuan sistem untuk pengembangan, instalasi dan pengujian terkait perangkat lunak. Sistem operasi Android dipilih berdasarkan hasil dari analisa dan perancangan perangkat lunak yang diberikan, oleh karena itu spesifikasi sistem untuk pengembangan perangkat lunak harus mampu menjalankan *Android Developer Tool*.

5.1.1 Spesifikasi Perangkat Keras

Pengembangan perangkat lunak *Audiosafe* membutuhkan perangkat keras yang mampu menjalankan program *Android Developer Tool* dengan spesifikasi prosesor, memori, dan display yang akan disebutkan dalam Tabel 5.1

Tabel 5.1 Spesifikasi Perangkat Keras Mesin Pengembang

Nama Komponen	Spesifikasi
System Model	*Hewlett-Packard Compaq 421*
Processor	*Intel(R) Core(TM) 2 Duo T6670 @2.20 GHz*
Memory	*6 GB 10600 MHz DDR3*
Display	*ATI Mobility Radeon HD 4500 Series 512 MB*

Dalam melakukan proses instalasi sistem, dimana perangkat lunak di jalankan dan dilakukan pengujian terhadap syarat pemenuhan kebutuhan, maka digunakan perangkat bergerak berbasis *Android* dengan spesifikasi perangkat keras yang disebutkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Keras Mesin Instalasi

Nama Komponen	Spesifikasi
System Model	*Redmi Note 3 Pro*

Processor	*Hexa-core Max 1.8Ghz*
Memory	*16 GB ROM, 2 GB RAM*
Display	*Adreno(TM) 510 @600MHz*

5.1.2 Spesifikasi Perangkat Lunak

Dalam mengembangkan perangkat lunak *Audiosafe* digunakan perangkat lunak dengan spesifikasi yang dijelaskan pada Tabel 5.3.

Tabel 5.3 Spesifikasi Perangkat Lunak Mesin Pengembang

Nama Komponen	Spesifikasi
Sistem Operasi	*Windows 10 Pro 64-bit*
Bahasa Pemrograman	*Java*
Intergrated Development Environment	*Android Developer Tool*

Dalam proses penggunaan dan pengujian dari hasil pengembangan, perangkat lunak ini menggunakan perangkat bergerak dengan perangkat lunak yang dispesifikasikan pada Tabel 5.4.

Tabel 5.4 Spesifikasi Perangkat Lunak Mesin Instalasi

Nama Komponen	**Spesifikasi**
Operating Sistem	*Android 5.1.1 codename : Lollipop*
Platform	*AOSP*
SDK	22

5.2 Batasan – Batasan Implementasi

Dalam pengembangan perangkat lunak *Audiosafe* terdapat batasan-batasan dalam pengimplementasiannya, antara lain :

1. Perangkat lunak *Audiosafe* dirancang untuk di jalankan pada *smartphone Android* dalam lingkungan yang terhubung dengan internet untuk menggunakan fitur terkait komunikasi dari dan menuju *server*.
2. Komunikasi data antara perangkat lunak dengan *server* menggunakan bentuk komunikasi pertukaran data dengan format *JSON*.
3. Menggunakan minimal API v22
4. Penggunaan *webService* untuk komunikasi data dibangun dengan PHP.
5. Teknik keamanan data yang digunakan adalah *AES-256* dan *Environment* dalam proses implementasinya menggunakan *Java* Crypthographic.
6. Pembuatan *layout* antarmuka aplikasi menggunakan format *Extensible Markup Language*.

5.3 Implementasi *Class*

Class yang dirancang pada tahap sebelumnya direalisasikan pada *file Java* yang berekstensi *.java*. Implementasi klas-klas yang dirancang melalui tahap analisa kebutuhan dan perancangan perangkat lunak dijelaskan dalam Tabel 5.5.

Tabel 5.5 Implementasi Klas

Package	Nama File	Keterangan Klass
com.skripsi.yuyon.audiosafe.config	AESSecHandler.java	Adalah *class* yang menangani segala proses/ fungsi pengenkripsian dan pendekripsian *digital audio* yang diterima maupun yang akan dikirim *user* dan *server*.
	AndroidMultiPartEntity.java	Adalah *class* yang menangani fungsi pengiriman file sekaligus paramater yang dikirimkan ke *server* dan juga progress dari proses pengunggahan melalui http.
	AppConfig.java	Adalah *class* yang menangani proses penginisialisasian alamat server yang akan digunakan.
	BitmapRenderer.java	Adalah *class* yang menangani fungsi penampilan data gambar dari *webservice* yang akan ditampilkan di *Audiosafe* mengenai fungsi-fungsi *list* dan detil *digital audio*
	JSONParser.java	Adalah *class* yang menangani penerimaan data *JSON* dari *webservice* dan menerjemahkannya kedalam *JSON* Object yang nantinya dapat dibaca oleh masing-masing klas lain yang menggunakan fungsinya.
	SessionManagerjJava	Adalah *class* yang menangani session *login* dan *logout user*. Digunakan agar *user* hanya melakukan satu kali *login* dan tidak harus melakukan *login* setiap kali akan masuk ke dalam sistem *Audiosafe*.
	Stopwatch.java	Adalah *class* yang menangani proses pencatatan waktu yang akan digunakan dalam penghitungan enkrispi dan deksipsi dari AES.
com.skripsi.yuyon.audiosafe.downloadedaudio	DownloadedAudioActivity.java	Adalah *class* yang menampilkan daftar *digital audio* yang sudah diunduh, dengan memilih *digital audio* maka akan diarahkan ke detil audio.
	DownloadedAudioAdapter.java	Adalah *class* yang mengatur bagaimana menampilkan distribusi variabel konten *digital audio* yang sudah terunduh dan ditampilkan pada DownloadedAudioActivity.
	DetailDownloadedAudioActivity.java	Adalah *class* yang mengatur proses menampilkan detil *digital audio*, proses dalam verifikasi status *play*, *play* audio dalam pengaturan membuka *player* audio di Android dan *delete* audio yang sudah terunduh.

	DetailDownloadedAudioAdapter.java	Adalah *class* yang mengatur aktivitas menampilkan konten dan data *online dari* detil *digital audio* yang terunduh.
com.skripsi.yuyon.audiosafe.login	LoginActivity.java	Adalah *class* yang mengatur segala aktivitas *user* untuk mencoba masuk ke dalam sistem *Audiosafe.*
com.skripsi.yuyon.audiosafe.main	MainActivity.java	Adalah *class* yang mengatur tampilan awal dari perangkat lunak *Audiosafe* saat *member* berhasil melakukan *login* ke dalam sistem dan dapat memberikan fungsi *logout* pada *activity* ini.
	TabsAdapter.java	Adalah *class* yang mengatur tata letak tampilan mengenai *class* `MainActivity`, mengatur komponen halaman akan yang disajikan dalam bentuk fragment.
	Audio.java	Adalah *class* yang menyimpan variabel sementara sebuah entitas *digital audio* yang berisi informasi *digital audio* yang dikirimkan oleh *server* melalui *webservice*.
com.skripsi.yuyon.audiosafe.onlineaudio	OnlineAudioActivity.java	Adalah *class* yang mengatur aktivitas menampilkan data *online* daftar yang dikirimkan *webservice* ke dalam perangkat lunak *Audiosafe*.
	OnlineAudioAdapter.java	Adalah *class* yang mengatur bagaimana menampilkan distribusi variabel masing-masing konten terhadap tampilan pada `OnlineAudioActivity`.
	DownloadAudioActivity.java	Adalah *class* yang mengatur aktivitas *member* mulai dari awal menampilkan halaman *download* beserta detil *digital audio*, proses *pengunduhan* dan *post* kondisi setelah *digital audio* selesai diunduh.
	DownloadAudioAdapter.java	Adalah *class* yang mengatur bagaimana menampilkan distribusi variabel masing-masing konten terhadap tampilan pada `DownloadAudioActivity`.
com.skripsi.yuyon.audiosafe.onlineaudio	ProfileActivity.java	Adalah *class* menampilkan identifikasi *member* yang telah berhasil masuk kedalam sistem, menampilkan daftar *digital audio* yang berhasil diunggah dan memberikan fungsi untuk ke halaman *Upload* dan halaman memberikan *previllege* ke *member* lain.
	ProfileAdapter.java	Adalah *class* yang mengatur distribusi variabel konten *digital audio* dan data yang telah dikirim oleh websevice lalu menampilkan kedalam `ProfileActivity`.
	ProfileAudioActivity.java	Adalah *class* yang menampilkan detil *digital audio* yang sudah dipilih dari *class* `ProfileActivity` dan diberikan fungsi untuk menghapus *digital audio* miliknya dari *server*.

	ProfileAudioAdapter.java	Adalah *class* yang mengatur distribusi variabel data *digital audio* yang telah dikirim oleh websevice lalu menampilkan kedalam ProfileAudioActivity.
	UploadAudioActivity.java	Adalah *class* yang mengatur aktifitas *member* mulai dari menampilkan *form* untuk pengunggahan *digital audio*, proses *upload* dan *post* kondisi setelah *upload* audio selesai.
	PrivilegeUserActivity.java	Adalah *class* yang menampilkan setiap daftar *member* yang berhak untuk diberi hak dalam pemutaran audio.
	PrivilegeUserAdapter.java	Adalah *class* yang mengatur distribusi variabel data *member* dan statusnya yang dikirim oleh websevice lalu menampilkan kedalam PrivilegeUserActivity.
	PrivilegeUserHandlerActivity.java	Adalah *class* yang mengatur aktifitas *member* untuk memberikan hak ijin atau hak tolak dalam pemutaran *digital audio* miliknya.
	PrivilegeUserHandlerAdapter.java	Adalah *class* yang mengatur distribusi variabel data *member* dan status yang telah dikirim oleh websevice lalu menampilkan kedalam ProfileActivity.
	User.java	Adalah *class* yang menyimpan variabel sementara sebuah entitas user yang berisi informasi user yang dikirimkan oleh *server* melalui *webservice*.
com.skripsi.yuyon.audiosafe.register	RegisterActivity.java	Adalah *class* yang mengatur aktivitas pendaftaran pengguna baru yang akan menggunakan perangkat lunak dan ingin menjadi member.

5.4 Implementasi Kode Program

Pengimplementasian kode program didapatkan dari *class* dan algoritma yang disusun untuk memenuhi kebutuhan yang telah didapat pada proses analisa kebutuhan dan perancangan. Berikut adalah beberapa implementasi kode program dari perangkat lunak *Audiosafe*. Tidak semua implementasi kebutuhan dituliskan dalam tulisan ini.

5.4.1 Implementasi Algoritma Mengunggah Audio

Algoritma menggunggah audio digunakan untuk membaca parameter inputan dan *file* dari *member,*mengenkripsi dengan algoritma AES dan mengirim kepada *webservice*. Algoritma mengunggah audio akan dijalankan saat tombol *Upload* Audio di tekan. Berikut adalah Kode Sumber 5.1 Algoritma Mengunggah Audio

Kode Sumber 5.1 Algoritma *Use case* Mengunggah Audio

```
1  String responseString = null;
2  String audioname = audionameText.getText().toString();
3  String key = keyText.getText().toString();
4  String fileimagePath = tvimage.getText().toString();
5  String fileaudioPathtemp = tvaudio.getText().toString();
```

```
6   String fileaudioPath = "mnt/sdcard/Audiosafe/Audio/temp.aufe";
7
8   try :
9   aes = new AESSecHandler(key);
10
11  catch :
12  UnsupportedEncodingException e
13  GeneralSecurityException e
14
15  try :
16  Stopwatch timer = new Stopwarch();
17  timer= new Stopwatch();
18  timer.start();
19  aes.encrypt(fileaudioPathtemp, fileaudioPath);
20  timer.stop();
21
22  catch :
23  IOException e
24  GeneralSecurityException e
25
26  HttpClient httpclient = new DefaultHttpClient();
27  HttpPost httppost = new HttpPost(UPLOAD_AUDIO_URL);
28
29  try :
30  AndroidMultiPartEntity entity = new AndroidMultiPartEntity(
31      new AndroidMultiPartEntity.ProgressListener() {
32      public void transferred(long num) {
33      publishProgress((int) ((num / (float) totalSize) * 100));}});
34
35  File sourceaudioFile = new File(fileaudioPath);
36  File sourceimageFile = new File(fileimagePath);
37
38  entity.addPart("uploaderID", new StringBody(memberID));
39  entity.addPart("audioname", new StringBody(audioname));
40  entity.addPart("key", new StringBody(key));
41  entity.addPart("audio", new FileBody(sourceaudioFile));
42  entity.addPart("image", new FileBody(sourceimageFile));
43  totalSize = entity.getContentLength();
44  httppost.setEntity(entity);
45
46  HttpResponse response = httpclient.execute(httppost);
47  HttpEntity r_entity = response.getEntity();
48
49  int statusCode = response.getStatusLine().getStatusCode();
50  if (statusCode == 200) :
51  responseString = EntityUtils.toString(r_entity);
52  responseprocess=1;
53
54  else :
55  responseString = "Error occurred! Http Status Code: "+ statusCode;
56  responseprocess=0;
57
58  catch :
59  ClientProtocolException responseString = e.toString();
60  IOException responseString = e.toString();
61
62  end
```

Implementasi algoritma mengunggah audio kemudian ditulis dalam bahasa *Java* sebagai *platform* pengembangan. Kode sumber algoritma mengunggah audio dimasukan kedalam fungsi `doInBackground` yang merupakan fungsi `Override Class AsyncTask`, yang berguna untuk memaksimalkan fungsi *Thread* pada *Android*. Fungsi *doInBackground* dijalankan ketika *Button Upload*

berhasil menjalankan fungsi `OnClickListener` yang didalamnya mengandung fungsi `UploadFiletoServer`.

Kode Sumber 5.2 Implementasi Mengunggah Audio

```
@Override
protected String doInBackground(Void... params) {

String responseString = null;
String audioname = audionameText.getText().toString();
String key = keyText.getText().toString();
String fileimagePath = tvimage.getText().toString();
String fileaudioPathtemp = tvaudio.getText().toString();
String fileaudioPath = "mnt/sdcard/Audiosafe/Audio/temp.aufe";

try {
aes = new AESSecHandler(key);

} catch (UnsupportedEncodingException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (GeneralSecurityException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
try {

timer= new Stopwatch();
timer.start();
aes.encrypt(fileaudioPathtemp, fileaudioPath);
timer.stop();

} catch (IOException e) {

} catch (GeneralSecurityException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost(UPLOAD_AUDIO_URL);

try {
AndroidMultiPartEntity entity = new AndroidMultiPartEntity(
new AndroidMultiPartEntity.ProgressListener() {

@Override
public void transferred(long num) {
publishProgress((int) ((num / (float) totalSize) * 100));}});

File sourceaudioFile = new File(fileaudioPath);
File sourceimageFile = new File(fileimagePath);

// Adding file data to http body
entity.addPart("uploaderID", new StringBody(memberID));
entity.addPart("audioname", new StringBody(audioname));
entity.addPart("key", new StringBody(key));
entity.addPart("audio", new FileBody(sourceaudioFile));
entity.addPart("image", new FileBody(sourceimageFile));

totalSize = entity.getContentLength();
httppost.setEntity(entity);

// Making server call
HttpResponse response = httpclient.execute(httppost);
```

```
61  HttpEntity r_entity = response.getEntity();
62
63  int statusCode = response.getStatusLine().getStatusCode();
64  if (statusCode == 200) {
65  // Server response
66  responseString = EntityUtils.toString(r_entity);
67  responseprocess=1;
68  } else {
69  responseString = "Error occurred! Http Status Code: "
70  + statusCode;
71  responseprocess=0;
72  }
73
74  } catch (ClientProtocolException e) {
75  responseString = e.toString();
76  responseprocess=0;
77  } catch (IOException e) {
78  responseString = e.toString();
79  responseprocess=0;
80  }
81
82  return responseString;
83
84  }
```

Penjelasan hubungan implementasi *source code* dengan Algoritma yang telah dituliskan sebelumnya adalah :

1. Baris 4-9 adalah pengambilan nilai variabel *audioname, key, fileImagepath, fileaudiotemp* dan *fileaudiopath* yang berisi deklarasi letak direktori penumpanan file hasil enkripsi sementara sebelum diunggah.
2. Baris 11-20 adalah inisiasi objek *AESSecHandler* dengan memparsing *variabel key* hasil dari pengambilan nilai key sebelumnya. Baris 14-10 adalah eksespi jika terjadi kesalahan saat inisiasi objek *AESSecHandler* yang menangani proses kriptografi dalam sistem.
3. Baris 21-33 adalah proses mengenai enkripsi file *digital audio* yang sumber *input* dan *output* direktorinya telah diketahui sebelumnya melalui inisialisasi variabel. Baris 23 adalah inisiasi objek Stopwatch untuk mengukuri kecepatan eksekusi enkripsi yang nantinya digunakan sebagai salah satu parameter pengujian performansi, baris 25 adalah proses enkripsi yang sebelumnya dengan menggunakan parameter key yang sudah diinisialisasi.
4. Baris 35-36 adalah inisiasi objek HttpClient dan HttpPost dengan memparsing parameter url *webservice* yang sudah ditentukan.
5. Baris 38- 39 adalah inisiasi objek AndroidMultiPartEntitiy yang nantinya menyusun parameter *file* dan *string* untuk dikirimkan ke *webservice*.
6. Baris 40-44 adalah fungsi dari AndroidMultiPartEntity yang digunakan untuk membuat parameter *progress* dari pengiriman *file* ke *server*.
7. Baris 46-47 adalah inisialisasi File dari parameter path yang sudah diambil.
8. Baris 50-57 adalah baris penyusunan parameter dan *source* apa saja yang dikirimkan kepada server.

9. Baris 60-61 adalah baris eksekusi dari pengiriman parameter dan *source* yang sudah disusun, serta didapatkan *response* apa yang didapatkan dari server.
10. Baris 63-72 adalah respon code yang didapatkan dari server, jika statuscode =200 maka proses upload akan dilanjutkan ke proses penghapusan *file temporary* dalam post execute, jika tidak maka akan di*terminate* dan ditampilkan pesan error dalam *post execute*.
11. Baris 74-80 adalah eksepsi ketika pengiriman ke server mengalami error.

5.4.2 Implementasi Algoritma Memutar Audio

Algoritma memutar audio digunakan untuk membuka *file digital audio* dan menyiapkannya untuk dapat di putar dalam *device member*. *Member* melakukan inputan berubah menekan tombol *play* pada halaman *Detail Downloaded Audio* yang sebelumnya *digital audio* dipilih dari komponen *Cardview* halaman *Downloaded Audio*.

Kode Sumber 5.3 Algoritma *Use case* Memutar Audio

```
1  int response;
2  try:
3  List<NameValuePair> params1 = new ArrayList<NameValuePair>();
4  params1.add(new BasicNameValuePair("audioname", Filename));
5  params1.add(new BasicNameValuePair("memberID", memberID));
6  JSONObject json =
7  jsonParser.makeHttpRequest(VIEW_DETAIL_STATUS_URL,"POST", params1);
8
9  if (json.length() > 0) :
10 JSONArray eventObject = json.getJSONArray("list_event");
11 event = eventObject.getJSONObject(0);
12 status = event.getString("status");
13 if (Objects.equals(status, "allowed")):
14 response = 1;
15 String fileaudioPathtemp =
16 "mnt/sdcard/Audiosafe/Audio/"+Filename+".mp3";
17 try :
18 aes = new AESSecHandler(event.getString("key"));
19
20 catch :
21 UnsupportedEncodingException e
22 GeneralSecurityException e
23
24 try :
25 timer= new Stopwatch();
26 timer.start();
27 aes.decrypt(Filepath, fileaudioPathtemp);
28 timer.stop();
29
30 catch :
31 IOException e
32 GeneralSecurityException e
33
34 pDialog.dismiss();
35
36 Intent intent = new Intent(DetailDownloadedAudio.this,
37 PlayAudioExample.class);
38 i.putExtra("File", fileaudioPathtemp);
39 i.putExtra("Filename", Filename);
```

```
40  startActivity(intent);
41
42  else :
43  response = 0;
44
45  else :
46  "No data"
47
48  catch :
49  JSONException e
50
51  end
```

Implementasi algoritma memutar audio kemudian ditulis dalam bahasa *Java* sebagai *platform* pengembangan. Kode sumber algoritma memutar audio dimasukan ke dalam fungsi `doInBackground` yang merupakan fungsi `Override Class AsyncTask`, yang berguna untuk memaksimalkan fungsi *Thread* pada Android. Fungsi *doInBackground* dijalankan ketika *Button Play* berhasil menjalankan fungsi `OnClickListener` yang didalamnya mengandung fungsi `AudioPlayHandler`.

Kode Sumber 5.4 Implementasi Memutar Audio

```
1   protected Boolean doInBackground(Void... arg0) {
2   int response;
3   try{
4   List<NameValuePair> params1 = new ArrayList<NameValuePair>();
5   params1.add(new BasicNameValuePair("audioname", Filename));
6   params1.add(new BasicNameValuePair("memberID", memberID));
7   JSONObject json =
8   jsonParser.makeHttpRequest(VIEW_DETAIL_STATUS_URL,"POST", params1);
9
10  if (json.length() > 0) {
11  JSONArray eventObject = json.getJSONArray("list_event");
12  event = eventObject.getJSONObject(0);
13  status = event.getString("status");
14  if (Objects.equals(status, "allowed")){
15  response = 1;
16  String fileaudioPathtemp =
17  "mnt/sdcard/Audiosafe/Audio/"+Filename+".mp3";
18  try {
19  aes = new AESSecHandler(event.getString("key"));
20
21  } catch (UnsupportedEncodingException e) {
22  // TODO Auto-generated catch block
23  e.printStackTrace();
24  } catch (GeneralSecurityException e) {
25  // TODO Auto-generated catch block
26  e.printStackTrace();
27  }
28  try {
29  timer= new Stopwatch();
30  timer.start();
31  aes.decrypt(Filepath, fileaudioPathtemp);
32  timer.stop();
33  } catch (IOException e) {
34  } catch (GeneralSecurityException e) {
35  // TODO Auto-generated catch block
36  e.printStackTrace();
37  }
38
39  pDialog.dismiss();
```

```
40
41  Intent intent = new Intent(DetailDownloadedAudio.this,
42  PlayAudioExample.class);
43  i.putExtra("File", fileaudioPathtemp);
44  i.putExtra("Filename", Filename);
45  startActivity(intent);
46  }
47  else {
48  response = 0;
49  }
50  }
51  }
52  else {
53  return "No data";
54  }
55  catch (JSONException e) {
56  e.printStackTrace();
57  }
58  return null;
59  }
```

Penjelasan hubungan implementasi *source code* dengan Algoritma yang telah dituliskan sebelumnya adalah :

1. Baris 4-7 adalah proses penyusunan variabel untuk pengambilan detail status *digital audio*, dan mengirimkannya ke *server* menggunakan metode *JSON*.
2. Baris 9 adalah untuk mengecek balasan *server*.
3. Baris 11-12 adalah baris untuk menanggapi respon jika *server* memberi pesan nilai parameter dari status.
4. Baris 13-19 adalah respon jika ternyata *status* adalah allowed maka akan diinisialisasi variabel *fileaudioPathtemp* yang berisi lokasi direktori hasil temporary dekripsi, dan pada baris 17 menginisiasi *AESSecHandler* dengan parameter *key* yang didapat dari parameter *key* dalam JSON.
5. Baris 19-25 adalah eksepsi ketika insiasi object *AESSecHandler* gagal.
6. Barus 26-31 adalah baris pendekrepsian *file* yang parameter *file input* dan *output* sudah diinisialisasi sebelumnya. Pada baris 27 adalah inisiasi objek Stopwatch untuk mengukuri kecepatan eksekusi dekripsi yang nantinya digunakan sebagai salah satu parameter pengujian performansi, baris 29 adalah proses dekripsi yang sebelumnya dengan menggunakan parameter key yang sudah diinisialisasi.
7. Baris 31-35 adalah baris eksepsi jika terjadi kesalahan dalam proses enkripsi.
8. Baris 37-44 adalah baris ketika proses dekripsi sudah selesai maka prosesdialog akan di*terminate* dan *digital audio* hasil dekripsi akan dilempar ke *activity* ACTION_VIEW dalam hal ini adalah memutar *file* media dengan *player* dari *device* yang selanjutnya melakukan proses thread dalam *post execute* yang bertugas untuk menghapus *file* asli *digital audio* yang telah terdekripsi setelah *digital audio* tersebut dimuat dalam RAM. Hal ini dilakukan agar *file* asli hanya bertindak sebagai removable temporary dan tidak dapat *member* ambil.

9. Baris 45-47 adalah kondisi lain ketika status tidak sama dengan allowed maka akan menghasilkan insialisasi variable response =0.
10. Baris 45-47 adalah kondisi lain ketika hasil dari json tidak mempunyai data maka akan ditampilkan “no data”
11. Baris 55-57 adalah eksepsi ketika gagal dalam peengambilan nilai dari *webservice.*

5.5 Implementasi Kriptografi

Implementasi Kriptografi dalam perangkat lunak *Audiosafe* proses enkripsi dan dekripsi dilakukan dalam *Android,* menggunakan AES-256 dalam proses kriptografinya.

5.5.1 Implementasi Enkripsi *AES*-256

Kode Sumber 5.5 Implementasi Enkripsi *AES-256*

```
1   public void encrypt( InputStream in, OutputStream out)
2   throws IOException, GeneralSecurityException {
3       try {
4        byte[] text = null;
5        ivSpec1 = new IvParameterSpec(generateIv1());
6        aesKey1 = new SecretKeySpec(generateAESKey1(ivSpec1.getIV(),
7   password), CRYPT_ALG);
8          ivSpec2 = new IvParameterSpec(generateIV2());
9          aesKey2 = new SecretKeySpec(generateAESKey2(), CRYPT_ALG);
10                out.write("AUFE".getBytes("UTF-8"));    // Heading.
11                out.write(ivSpec1.getIV());      // Initialization
12  Vector.
13                text = new byte[BLOCK_SIZE + KEY_SIZE];
14                cipher.init(Cipher.ENCRYPT_MODE, aesKey1, ivSpec1);
15                cipher.update(ivSpec2.getIV(), 0, BLOCK_SIZE,
16  text);
17                cipher.doFinal(aesKey2.getEncoded(), 0, KEY_SIZE,
18  text, BLOCK_SIZE);
19                out.write(text);    // Crypted IV and key.
20                hmac.init(new SecretKeySpec(aesKey1.getEncoded(),
21  HMAC_ALG));
22                text = hmac.doFinal(text);
23                out.write(text);    // HMAC from previous
24  cyphertext.
25                cipher.init(Cipher.ENCRYPT_MODE, aesKey2, ivSpec2);
26                hmac.init(new SecretKeySpec(aesKey2.getEncoded(),
27  HMAC_ALG));
28                text = new byte[BLOCK_SIZE];
29                int len, last = 0;
30                while ((len = in.read(text)) > 0) {
31                      cipher.update(text, 0, BLOCK_SIZE, text);
32                      hmac.update(text);
33                      out.write(text);    // Crypted file data
34  block.
35                      last = len;
36                }
37                last &= 0x0f;
38                out.write(last);    // Last block size mod 16.
39                text = hmac.doFinal();
40                out.write(text);    // HMAC from previous
41  cyphertext.
42          } catch (InvalidKeyException e) {
43                throw new
44  GeneralSecurityException(JCE_EXCEPTION_MESSAGE, e);} }
```

Penjelasan dari implementasi kriptografi *AES*-256 Mode enkripsi di Kode Sumber 5.5 adalah sebagai berkut :

1. Baris 4-9 adalah kode sumber untuk mendeklarasikan berbagai variabel terkait proses enkripsi. Baris 4 mendeklarasikan variabel *text* yang bertipe *byte* yang nantinya digunakan untuk menulis *byte-byte* kedalam *outputstream*. Baris 5-6 adalah baris yang mendeklarasikan variabel *key* ivSpec1 (16 *byte* di *generate* secara random)dan Aeskey1 (32 *byte* di*generate* dengan penggabungan ivSpec1 dengan *key user*). Baris 8-9 adalah baris mendeklarasikan *key* ivSpec2 dan aesKey2 yang masing masing memiliki panjang 16 *byte* dan 32 *byte*. ivSpec2 dan aesKey2 digunakan untuk mengenkrip konten *digital audio*, lalu kedua *key* tersebut dituliskan kedalam *header* dengan dengan terlebih dahulu dienkrip menggunakan ivSpec1 dan ivSpec2.
2. Baris 10-11 adalah *source code* untuk menulis *byte header*, Baris 10 untuk menulis identifier, Baris 11 untuk menuliskan ivSpec1 yang digunakan decryptor untuk sebagai pelengkap *key* mereka.
3. Baris 14-18 adalah *byte* untuk mengenkripsi ivSpec2 dan aesKey2 lalu menuliskannya kedalam *header*
4. Baris 19-21 adalah *source code* untuk melakukan teknik *digital* signature menggunakan HmacSHA256, *byte* yang sebelumnya diisi oleh hasil enkripsi ivSpec2 dan aesKey2 akan dihasing untuk menghasilkan identifier unik, setelah didapatkan *byte* identifer sepanjang 32 *byte* dari algoritma hasing SHA256 maka *byte* itu akan dituliskan pula dalam *header*. Hal ini digunakan sebagai acuan jika *header* yang berisi ivSpec2 dan aesKey2 tidak mengalami alter data atau perubahan lainnya.
5. Baris 24-26 adalah *source code* untuk memulai proses enkripsi konten *digital audio*.
6. Baris 28-39 Adalah baris untuk membaca *file* inputstream sebanyak BLOCK_SIZE=16 *byte* secara berulang lalu mengenkripsinya dan menuliskan kedalam *file* baru sambil mengupdate identifier dari *byte* yang tercipta hingga panjang *byte* habis terbaca dan semuanya telah terenkripsi.

5.5.2 Implementasi Dekripsi *AES*-256

Kode Sumber 5.6 Implementasi Dekripsi *AES*-256

```
1   public void decrypt(long inSize, InputStream in, OutputStream out)
2           throws IOException, General Security Exception {
3           try {
4                   byte[] text = null, backup = null;
5                   long total = 3 + BLOCK_SIZE +
6   BLOCK_SIZE + KEY_SIZE + SHA_SIZE + 1 + SHA_SIZE;
7        text = new byte[3];
8       readBytes(in, text);   // Heading.
9                   if          (!new          String(text,          "UTF-
10  8").equalsIgnoreCase("AUFE")) {
11                      throw new IOException("Invalid file header");
12                  }
13                  text = new byte[BLOCK_SIZE];
14                  readBytes(in, text);       // Initialization Vector.
15                  ivSpec1 = new IvParameterSpec(text);
```

```
16              aesKey1                         =                       new
17  SecretKeySpec(generateAESKey1(ivSpec1.getIV(),          password),
18  CRYPT_ALG);
19              cipher.init(Cipher.DECRYPT_MODE, aesKey1, ivSpec1);
20              backup = new byte[BLOCK_SIZE + KEY_SIZE];
21              readBytes(in, backup);     // IV and key to decrypt
22  file contents.
23        text = cipher.doFinal(backup);
24        ivSpec2 = new IvParameterSpec(text, 0, BLOCK_SIZE);
25        aesKey2  =  new  SecretKeySpec(text,  BLOCK_SIZE,  KEY_SIZE,
26  CRYPT_ALG);
27        hmac.init(new          SecretKeySpec(aesKey1.getEncoded(),
28  HMAC_ALG));
29        backup = hmac.doFinal(backup);
30        text = new byte[SHA_SIZE];
31        readBytes(in, text);       // HMAC and authenticity test.
32        if (!Arrays.equals(backup, text)) {
33              throw new IOException("Message has been altered or
34  password incorrect");
35              }
36              total = inSize - total;    // Payload size.
37              if (total % BLOCK_SIZE != 0) {
38        throw new IOException("Input file is corrupt");
39              }
40              if (total == 0) {    // Hack: empty files won't enter
41  block-processing for-loop below.
42                    in.read();    // Skip last block size
43              }
44              cipher.init(Cipher.DECRYPT_MODE, aesKey2, ivSpec2);
45              hmac.init(new    SecretKeySpec(aesKey2.getEncoded(),
46  HMAC_ALG));
47              backup = new byte[BLOCK_SIZE];
48              text = new byte[BLOCK_SIZE];
49              for (int block = (int) (total / BLOCK_SIZE); block >
50  0; block--) {
51              int len = BLOCK_SIZE;
52              if (in.read(backup, 0, len) != len) {   //
53  Ciphertext block.
54              throw  new  IOException("Unexpected  end  of  file
55  contents");
56                    }
57        cipher.update(backup, 0, len, text);
58        hmac.update(backup, 0, len);
59        if (block == 1) {
60        int last = in.read();       // Last block size mod 16.
61        len = (last > 0 ? last : BLOCK_SIZE);
62                    }
63                    out.write(text, 0, len);
64              }
65        out.write(cipher.doFinal());
66        backup = hmac.doFinal();
67        text = new byte[SHA_SIZE];
68        readBytes(in, text);       // HMAC and authenticity test.
69              if (!Arrays.equals(backup, text)) {
70                    throw  new  IOException("Message  has  been
71  altered or password incorrect");
72              }
73        } catch (InvalidKeyException e) {
74              throw                                                 new
75  GeneralSecurityException(JCE_EXCEPTION_MESSAGE, e);
76        }     } }
```

Penjelasan dari Implementasi dari Kriptografi *AES*-256 mode dekripsi pada Kode Sumber 5.6 adalah sebagai berikut :

1. Baris 4-6 adalah baris untuk melakukan deklarasi variabel, *byte text* untuk menampung *byte* guna kepentingan membaca *file*, backup adalah variabel untuk menampung *byte* guna kepentingan identifikasi *byte* yang digunakan dalam *digital* signature. Long total adalah variabel yang mendeklarasikan panjang *byte* tambahan diluar *byte* konten enkripsi didalam *file* seperti *header* dan *digital* identifier.
2. Baris 8-12 adalah *source code* pembacaan 3 *byte* awal pengecekan *Header*.
3. Baris 14-19 adalah *source code* untuk membaca *byte* selanjutnya sejumlah BLOCK_SIZE=16. 16 *byte* selanjutnya adalah ivSpec1 yang sebelumnya pada proses enkripsi dituliskan sebagai *header*. Setelah pembacaan ivSpec1 dilakukan *generate* aesKey1 menggunakan kombinasi ivSpec1 dan *key user* dan dilakukan inisiasi untuk melakukan mode dekripsi menggunakan aesKey1 dan dan ivSpec1.
4. Baris 20-27 adalah *source code* untuk membaca block selanjutnya sejumlah [16+32] yaitu tempat dimana *chiper text* dari ivSpec2 dan aesKey2 berada. Setelah pembacaan *byte* direntang tersebut, *byte* tersebut di dekripsi dan dimasukan kedalam variabel ivSpec2 dan asKey2 sebenarnya.
5. Baris 28-35 adalah *source code* untuk memvalidasi kecocokan *digital* signature. Blok yang sebelumnya dibaca yang berisi aesKey2 dan ivSpec2 dihash-ing dan dibandingkan dengan hashing dari proses enkripsi yang tertulis dalam 32 blok selanjutnya. Jika tidak cocok diberikan eksepsi.
6. Baris 36-42 adalah proses pemotongan *byte*, untuk menemukan *byte* yang merupakan *byte* penyusun konten *digital audio* yang terenkripsi.
7. Baris 44-46 adalah *source code* memulai dekripsi menggunakan aesKey2 dan ivSpec 2 dan melakukan inisiasi pengecekan *digital* signature pada konten *chiper digital audio*.
8. Baris 49-65 adalah proses pendekripsian, *file* akan di baca setiap BLOCK_SIZE 16 *byte* dan didekripsi lalu di tuliskan melalui outputstrem sambil diupdate *digital signaturenya,* proses ini akan terus berulang hingga panjang *byte* telah habis.
9. Baris 68-74 adalah baris untuk mencocokan *digital* signature dari *byte* yang telah didekripsi dengan *digital* signature yang telah tersisipkan melalui proses enkripsi sebelumnya yang dilakukan kepada *file*. Jika tidak cocok diberikan eksepsi.

5.5 Implementasi Basis Data

Dalam implementasi penyimpanan data, seluruh proses ditangani menggunakan *database management system* MySQL. Fungsi yang dijalankan untuk implementasi penyimpanan data menggunakan *query* SQL. Pada *database* yang dirancang terdapat relasi antar Tabel antara lain, tabel *members* yang berelasi dengan tabel *listaudio* dalam identitas *uploader* dan tabel *members* yang berelasi dengan tabel *statusplay* dalam identitas *member* dan identitas *privilegemember*. Sehingga tabel yang diimplementasikan memiliki unsur

hubungan satu sama lain yang terindikasi dengan adanya *Foreign Key.* Berikut dalam tabel 5.6, 5.7 dan 5.8 adalah implementasi dari tabel-tabel yang sebelumnya dirancang pada Bab Perancangan.

Tabel 5.6 Implementasi Tabel Members

Column	Type	Null
memberID	int(11)	No
username	varchar(25)	No
password	varchar(25)	No
loginstatus	varchar(10)	No

Dalam tabel 5.6, memberID berperan sebagai primary key dalam tabel.

Tabel 5.7 Implementasi Tabel Listaudio

Column	Type	Null
audioID	Int(11)	No
uploaderID	Int(11)	No
title	varchar(50)	No
key	varchar(50)	No
coverpath	varchar(100)	No
audiopath	varchar(100)	No

Dalam tabel 5.7, *audioID* berperan sebagai primary key dalam tabel dan *uploaderID* berelasi dengan *memberID* pada tabel *members*.

Tabel 5.8 Implementasi Tabel Statusplay

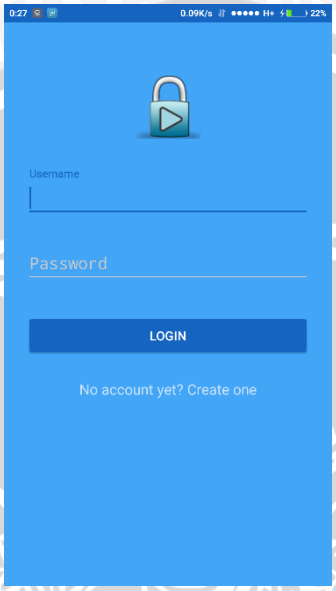
Column	Type	Null
memberID	Int(11)	No
privilegememberID	Int(11)	No
status	varchar(20)	No

Dalam tabel 5.8, *memberID dan privilegememberID* berelasi dengan *memberID* pada tabel *members* sehingga membuat menjadi *foreign key*.

5.6 Implementasi Antarmuka

Implementasi antarmuka perangkat lunak *Audiosafe* mengunakan *extended markup languange* (*XML*). Implementasi antarmuka didasarkan pada perancangan antarmuka yang dilakukan dalam bab perancangan pada tahap sebelumnya.

5.6.1 Implementasi Antarmuka Melakukan *Login*

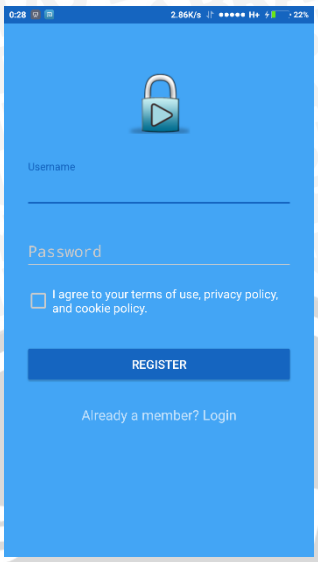


Gambar 5.1 Antarmuka Melakukan *Login*

Gambar 5.1 menjelaskan implementasi dari antarmuka melakukan *login* yang dirancang pada bab perancangan antarmuka. Komponen penyusun seperti *EditText*, *ImageView* dan -*Button* dalam perancangan telah diimplementasikan.

5.6.2 Implementasi Antarmuka Melakukan *Register*

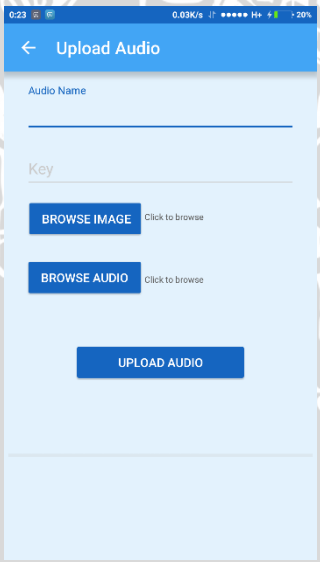
Gambar 5.2 menjelaskan implementasi dari antarmuka melakukan *register* yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti *EditText*, *ImageView, CheckButton* dan *Button* dalam perancangan telah diimplementasikan



Gambar 5.2 Antarmuka Melakukan Register

5.6.3 Implementasi Antarmuka Mengunggah Audio

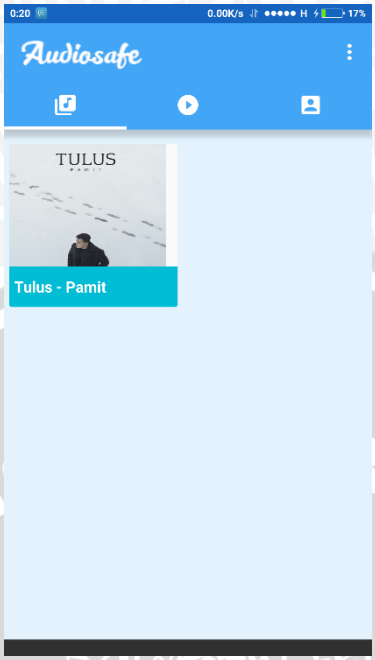
Gambar 5.3 adalah implementasi dari antarmuka mengunggah audio yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti *textView* dan *Button* yang disebutkan dalam perancangan telah diimplementasikan.



Gambar 5.3 Antarmuka Mengunggah Audio

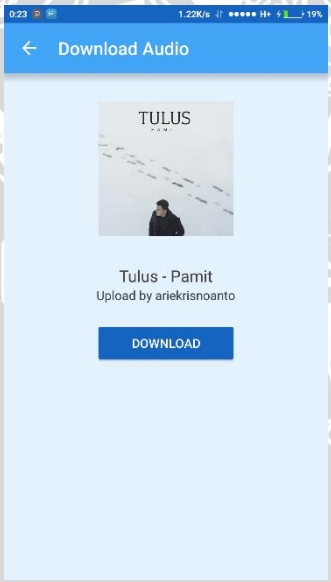
5.6.4 Implementasi Antarmuka Melihat Daftar Audio

Gambar 5.4 adalah implementasi dari antarmuka melihat daftar audio yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti teksView, ImageView dalam sebuah CardView seperti yang disebutkan dalam bab perancangan telah diimplementasikan.



Gambar 5.4 Antarmuka Melihat Daftar Audio

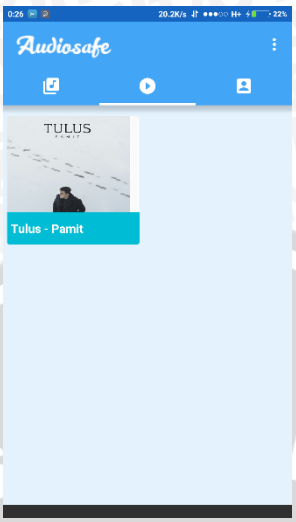
5.6.5 Implementasi Antarmuka Mengunduh Audio



Gambar 5.5 Antarmuka *Mengunduh Audio*

Gmbar 5.5 di halaman sebelumnya adalah implementasi dari antarmuka mengunduh audio yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti teksView, *ImageView* dan *button* seperti yang disebutkan dalam bab perancangan telah diimplementasikan

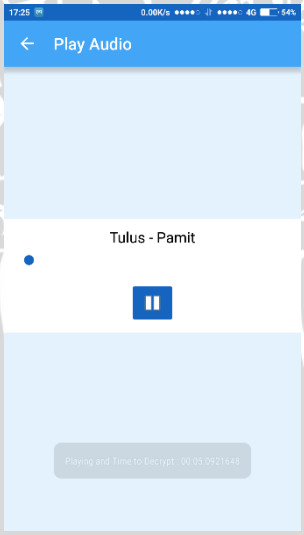
5.6.6 Implementasi Antarmuka Melihat Daftar Audio yang Terunduh



Gambar 5.6 Antarmuka Melihat Daftar Audio yang Terunduh

Gambar 5.6 adalah implementasi dari antarmuka melihat daftar audio yang terunduh yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti teksView, *ImageView* dalam sebuah CardView seperti yang disebutkan dalam bab perancangan telah diimplementasikan

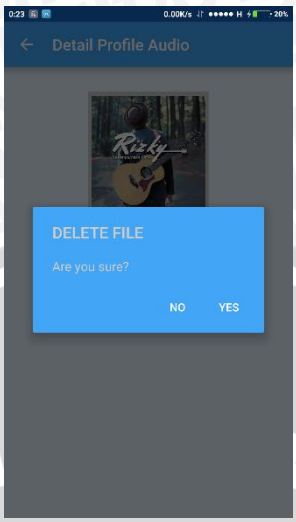
5.6.7 Implementasi Antarmuka Memutar Audio



Gambar 5.7 Antarmuka Memutar Audio

Gambar 5.7 di halaman sebelumnya adalah implementasi dari antarmuka memutar audio yang dirancang dalam bab perancangan antarmuka. Komponen penyusun *player* seperti yang disebutkan dalam bab perancangan telah diimplementasikan

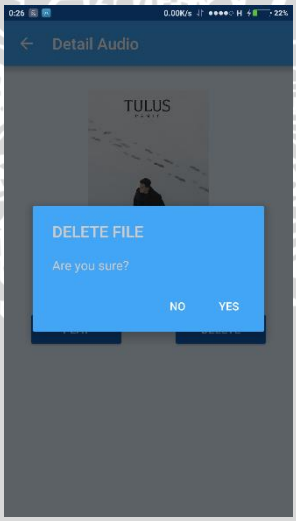
5.6.8 Implementasi Antarmuka Menghapus Audio di *Server*



Gambar 5.8 Antarmuka Menghapus Audio di Server

Gambar 5.8 adalah implementasi dari antarmuka menghapus audio di server yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti *alertdialog* seperti yang disebutkan dalam bab perancangan telah diimplementasikan

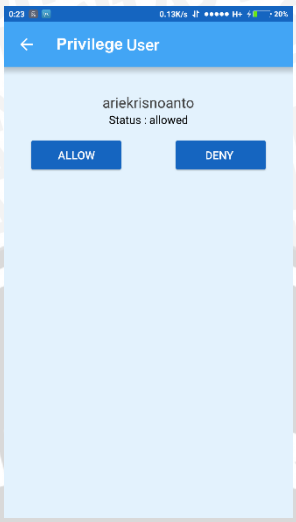
5.6.9 Implementasi Antarmuka Menghapus Audio di Daftar yang Terunduh



Gambar 5.9 Antarmuka Menghapus Audio di Daftar yang Terunduh

Gambar 5.9 adalah implementasi dari antarmuka menghapus audio di server yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti *alertdialog* seperti yang disebutkan dalam bab perancangan telah diimplementasikan.

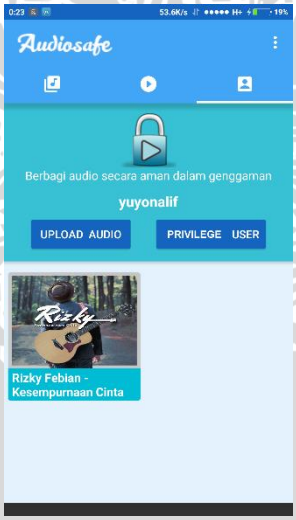
5.6.10Implementasi Antarmuka Memberikan *Privilege Member*



Gambar 5.10 Antarmuka Memberikan *Privilege Member*

Gambar 5.10 adalah implementasi dari antarmuka memberikan *privilege* kepada *member* di server yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti teksView, *ImageView* dan Button seperti yang disebutkan dalam bab perancangan telah diimplementasikan

5.6.11Implementasi Antarmuka Melihat Profil



Gambar 5.11 Antarmuka Melihat Profil

Gambar 5.11 adalah implementasi dari antarmuka melihat profil di *server* yang dirancang dalam bab perancangan antarmuka. Komponen penyusun seperti *alertdialog* seperti yang disebutkan dalam bab perancangan telah diimplementasikan.

BAB 6 PENGUJIAN DAN ANALISIS

Dalam bab ini akan dibahas tentang pengujian perangkat lunak dan pengujian mengenai keamanan data yang diterapkan dalam perangkat lunak, meliptui sisi performansi waktu, keaslian data, dan *security* yang telah diimplementasi pada tahap sebelumnya. Pengujian perangkat lunak dilakukan dengan 2 teknik yaitu *Whitebox* dan *Blackbox*. *Whitebox,* yaitu dengan menggunakan *Basis Path Testing* dalam algoritma pemenuhan kebutuhan, dan teknik *Blackbox* yaitu dengan menguji validasi fungsi kebutuhan fungsional.

Pengujian keamanan data di sisi performansi waktu menggunakan parameter kecepatan eksekusi proses kriptografi, di sisi keaslian data, digunakan parameter perbandingan ukuran data dan perbandingan konten data setelah mengalami proses enkripsi dan proses dekripsi, dan di sisi *security,* dilakukan pengujian dalam *privilege member* untuk menguji dapat atau tidaknya memutar *digital audio* dengan kondisi *privilege* yang diijinkan dan ditolak. Tahap selanjutnya adalah melakukan analisis atas keseluruhan hasil dari pengujian.

6.1 Perangkat Keras Pengujian

Perangkat keras yang diperlukan aplikasi *Audiosafe* ini adalah satu set komputer dengan, Spesifikasi :

1. *Processor*: Intel(R) Core(TM) 2 Duo T6670 @2.20 GHz
2. *System Model* : Hewlett-Packard Compaq 421
3. Memory : 6 GB 10600 MHz DDR3
4. *Display* : ATI Mobility Radeon HD 4500 Series 512 MB
5. *HDD:* 320 *GB*
6. Sistem Operasi : 64-*bit Windows 10 Operating System*

Sementara perangkat keras yang digunakan untuk mengukur *Audiosafe* pada perangkat bergerak adalah *Handheld Device* dengan spesifikasi :

1. System Model : Redmi Note 3 Pro
2. Processor : Hexa-core Max 1.8Ghz
3. Memory : 16 GB ROM, 2 GB RAM
4. Display : Adreno(TM) 510 @600MHz

6.2 Rencana Pengujian

1. Pengujian Unit : *Basis Path Testing* dengan dua algoritma sampel fitur utama berdasarkan kebutuhan utama perangkat lunak *Audiosafe.*
2. Pengujian Validasi : Pengujian kebutuhan/*requirement system* yang telah didefinisikan didalam spesifikasi kebutuhan perangkat lunak berdasarkan parameter terpenuhi atau tidaknya fungsi tersebut.

3. Pengujian Keamanan yang meliputi performansi waktu eksekusi kriptografi, keaslian data dan *security*.

6.3 Pengujian Unit

Pengembangan perangkat lunak berdasarkan pemrograman berorientasi objek menerapkan pengujian unit untuk suatu fungsi dari suatu klas. Pada pengujian unit untuk menguji perangkat lunak *Audiosafe* digunakan *Whitebox Testing* yang dilakukan dengan teknik *Basis Path Testing.* Pengujian dilakukan dengan memodelkan algoritma fungsi pada suatu *flow graph,* menentukan jumlah kompleksitas siklomatik, menentukan sebuah basis set dari jalur independen dan memberikan kasus uji pada setiap set yang telah ditentukan.

6.3.1 Pengujian Unit Mengunggah Audio

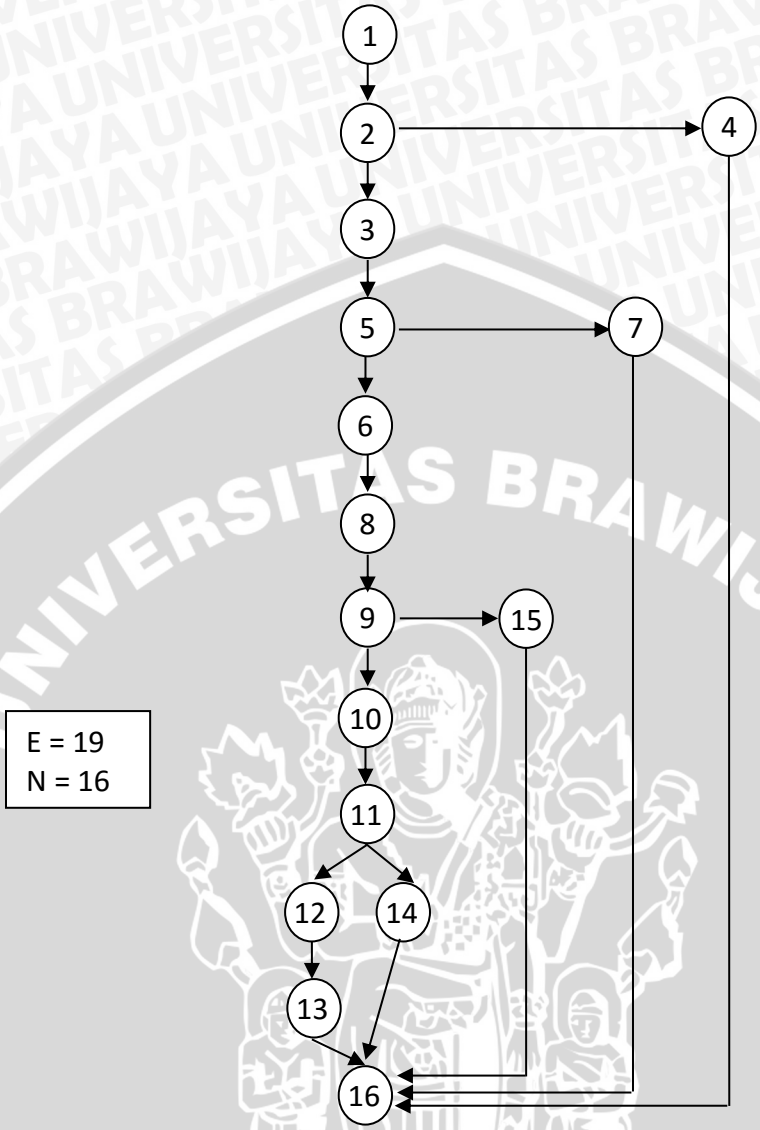
Kode Sumber 6.1 *Pseudocode* Algoritma Mengunggah Audio

```
1  String responseString = null;
2  String audioname = audionameText.getText().toString();
3  String key = keyText.getText().toString();
4  String fileimagePath = tvimage.getText().toString();
5  String fileaudioPathtemp = tvaudio.getText().toString();
6  String fileaudioPath = "mnt/sdcard/Audiosafe/Audio/temp.aufe";
7
8  try :
9  aes = new AESSecHandler(key);
10
11 catch :
12 UnsupportedEncodingException e
13 GeneralSecurityException e
14
15 try :
16 Stopwatch timer = new Stopwarch();
17 timer= new Stopwatch();
18 timer.start();
19 aes.encrypt(fileaudioPathtemp, fileaudioPath);
20 timer.stop();
21
22 catch :
23 IOException e
24 GeneralSecurityException e
25
26 HttpClient httpclient = new DefaultHttpClient();
27 HttpPost httppost = new HttpPost(UPLOAD_AUDIO_URL);
```

(1) (2) (3) (4) (5) (6) (7) (8)

```
28
29  try :                                                                    (9)
30  AndroidMultiPartEntity entity = new AndroidMultiPartEntity(
31      new AndroidMultiPartEntity.ProgressListener() {
32      public void transferred(long num) {                                  (10)
33      publishProgress((int) ((num / (float) totalSize) * 100));}});
34
35  File sourceaudioFile = new File(fileaudioPath);
36  File sourceimageFile = new File(fileimagePath);
37
38  entity.addPart("uploaderID", new StringBody(memberID));
39  entity.addPart("audioname", new StringBody(audioname));
40  entity.addPart("key", new StringBody(key));
41  entity.addPart("audio", new FileBody(sourceaudioFile));
42  entity.addPart("image", new FileBody(sourceimageFile));                  (11)
43  totalSize = entity.getContentLength();
44  httppost.setEntity(entity);
45
46  HttpResponse response = httpclient.execute(httppost);
47  HttpEntity r_entity = response.getEntity();
48
49  int statusCode = response.getStatusLine().getStatusCode();
50  if (statusCode == 200) :                                                 (12)
51  responseString = EntityUtils.toString(r_entity);                         (13)
52  responseprocess=1;
53
54  else :
55  responseString = "Error occurred! Http Status Code: "+ statusCode;       (14)
56  responseprocess=0;
57
58  catch :
59  ClientProtocolException responseString = e.toString();                   (15)
60  IOException responseString = e.toString();
61
62  end                                                                      (16)
```

Algoritma Mengunggah Audio dimodelkan ke dalam *flow graph* untuk menghitung kompleksitas siklomatis nantinya hasil dari perhitungan kompleksitas siklomatis akan menentukan banyaknya basis set / jalur independen pengujian dari algoritma Kode Sumber 6.1. Berikut Gambar 6.1 adalah konversi dari algoritma mengunggah audio menjadi *Flowgraph :*



Gambar 6.1 *Flow graph* algoritma mengunggah audio

Melalui *Flowgraph* tersebut dapat dihitung kompleksitas siklmatis dengan menggunakan rumus :

1. V(G) = 5 *regions*
2. V(G) = 19E – 16N + 2 = 5
3. V(G) = 4P + 1 = 5

Berdasarkan dari nilai kompleksitas siklomatis yang telah didapatkan dari perhitungan maka ditentukan 5 buah basis set dari jalur independen, yaitu :

1. Jalur 1 = 1,2,4,16
2. Jalur 2 = 1,2,3,5,7,16
3. Jalur 3 = 1,2,3,5,6,8,9,15,16
4. Jalur 4 = 1,2,3,5,6,8,9,10,11,14,16

5. Jalur 5 = 1,2,3,5,6,8,9,10,11,12,13,16

Tabel 6.1 Kasus uji pengujian unit algoritma mengunggah audio

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1.	Inisiasi objek *AES*SecHandler tidak memenuhi persyaratan	Muncul pesan *error*	Muncul pesan *error*
2.	Terjadi kesalahan pada proses enkripsi	Muncul pesan *error*	Muncul pesan *error*
3.	Proses enkripsi berhasil. Dan terjadi kesalahan pada saat pengiriman data ke server.	Muncul pesan *error*	Muncul pesan *error*
4.	Proses enkripsi berhasil. Tidak terjadi kesalahan pada saat pengiriman data ke server. Server mengirim response status *code* tidak sama dengan 200	Dimunculkan pesan *Error occurred!* disertai pesan status *code* dan menginisialisasi responseprocess=0 untuk diproses selanjutnya	Dimunculkan pesan *Error occurred!* disertai pesan status *code* dan menginisialisasi responseprocess=0 untuk diproses selanjutnya
5.	Proses enkripsi berhasil. Tidak terjadi kesalahan pada saat pengiriman data ke server. Server mengirim response status *code* sama dengan 200	Menginisialisasi responseprocess=1 untuk diproses selanjutnya	Menginisialisasi responseprocess=1 untuk diproses selanjutnya

6.3.2 Pengujian Unit Memutar Audio

Kode Sumber 6.2 *Pseudocode* Algoritma Memutar Audio

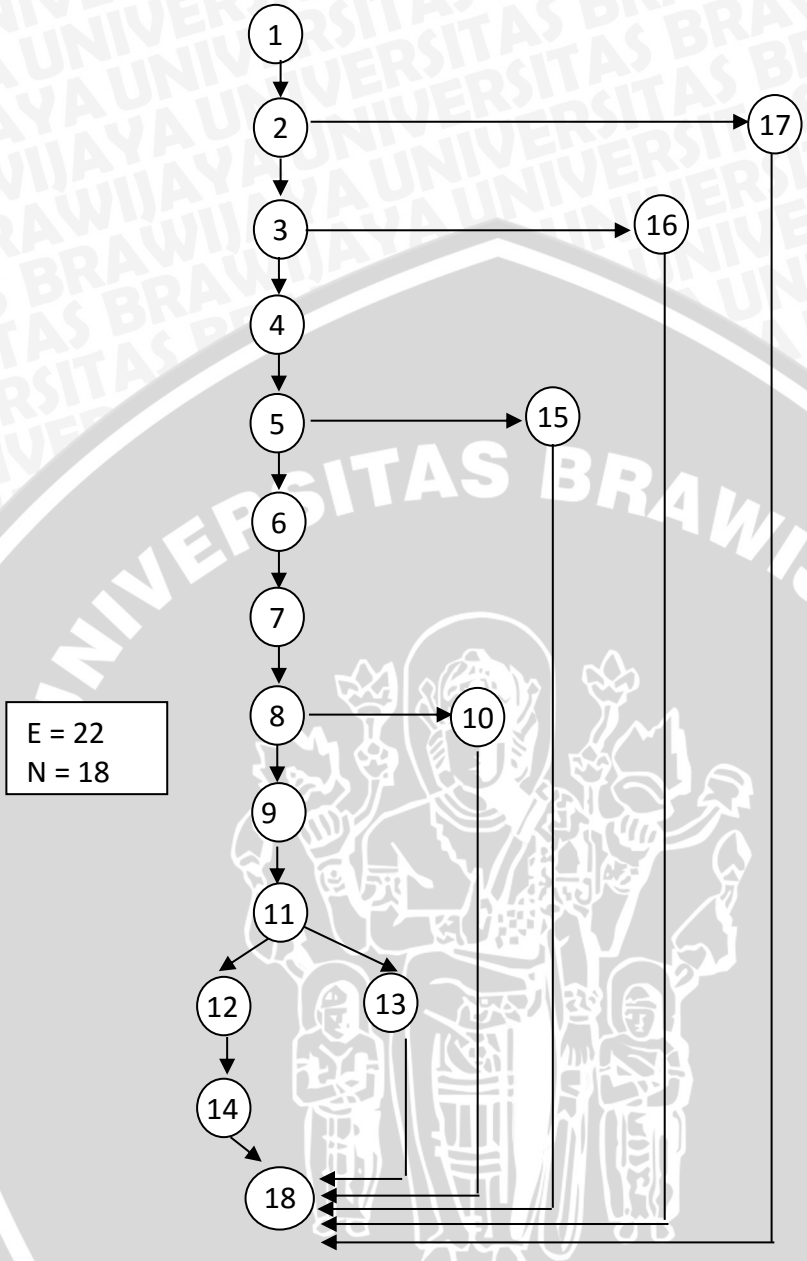
```
1   int response;
2   try:
3   List<NameValuePair> params1 = new ArrayList<NameValuePair>();
4   params1.add(new BasicNameValuePair("audioname", Filename));
5   params1.add(new BasicNameValuePair("memberID", memberID));
6   JSONObject                          json                          =
7   jsonParser.makeHttpRequest(VIEW_DETAIL_STATUS_URL,"POST",
    params1);
8   if (json.length() > 0) :
9   JSONArray eventObject = json.getJSONArray("list_event");
10  event = eventObject.getJSONObject(0);
11  status = event.getString("status");
12  if (Objects.equals(status, "allowed")):
13  response = 1;
14  String                          fileaudioPathtemp                  =
15  "mnt/sdcard/Audiosafe/Audio/"+Filename+".mp3";
16  try :
17  aes = new AESSecHandler(event.getString("key"));
18
```

1
2
3
4
5
6
7
8
9

```
19  catch :
20  UnsupportedEncodingException e
21  GeneralSecurityException e
22
23  try :
24  timer= new Stopwatch();
25  timer.start();
26  aes.decrypt(Filepath, fileaudioPathtemp);
27  timer.stop();
28
29  catch :
30  IOException e
31  GeneralSecurityException e
32
33  pDialog.dismiss();
34
35  Intent intent = new Intent();
36  intent.setAction(Intent.ACTION_VIEW);
37  File file =new File(fileaudioPathtemp);
38  intent.setDataAndType(Uri.fromFile(file),"audio/*");
39  startActivity(intent);
40
41  else :
42  response = 0;
43
44  else :
45  "No data"
46
47  catch :
48  JSONException e
49
50  end
51
```

10 11 12 13 14 15 16 17 18

Algoritma memutar audio kemudian dimodelkan ke dalam *Flowgraph* untuk menghitung kompleksitas siklomatis yang nantinya hasil dari perhitungan kompleksitas siklomatis akan menentukan banyaknya basis set / jalur independen pengujian dari algoritma Kode Sumber 6.2. Berikut Gambar 6.2 adalah konversi dari algoritma memutar audio menjadi *Flowgraph :*



Gambar 6.2 *Flowgraph* algoritma memutar audio

Melalui *Flowgraph* tersebut dapat dihitung kompleksitas siklomatis dengan menggunakan rumus :

1. V(G) = 6 *regions*
2. V(G) = 22E – 18N + 2 = 6
3. V(G) = 5P + 1 = 6

Berdasarkan dari nilai kompleksitas siklomatis yang telah didapatkan dari perhitungan maka ditentukan 6 buah basis set dari jalur independen, yaitu :

1. Jalur 1 = 1,2,17,18

2. Jalur 2 = 1,2,3,16,18
3. Jalur 3 = 1,2,3,4,5,15,18
4. Jalur 4 = 1,2,3,4,5,6,7,8,10,18
5. Jalur 5 = 1,2,3,4,5,6,7,8,9,11,13,18
6. Jalur 6 = 1,2,3,4,5,6,7,8,9,11,12,14,18

Tabel 6.2 Kasus uji pengujian unit algoritma memutar audio

Jalur	Kasus Uji	Hasil yang diharapkan	Hasil yang didapatkan
1.	Terjadi kesalahan saat merequest json ke server	Muncul pesan *error*	Muncul pesan *error*
2.	Proses request sudah berhasil diterima tetapi tidak mengirim data json apapun	Muncul pesan notifikasi "no data"	Muncul pesan notifikasi "no data"
3.	Status tidak sama dengan allowed dikirim dari server	Menginisialisasi response=0 untuk diproses pada tahap selanjutnya	Menginisialisasi response=0 untuk diproses pada tahap selanjutnya
4.	Inisiasi objek *AES*SecHandler tidak memenuhi persyaratan	Muncul pesan *error*	Muncul pesan *error*
5.	Proses Dekripsi mengalami kegagalan karena *digital audio* rusak atau didistribusikan antar *user* dengan ilegal	Muncul pesan *error*	Muncul pesan *error*
6.	Proses dekripsi *file* berhasil	Aplikasi me*render file digital audio* yang berhasil dienkrip dan mengirim kontennya untuk dapat diputar	Aplikasi me*render file digital audio* yang berhasil dienkrip dan mengirim kontennya untuk dapat diputar

6.4 Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun memenuhi dengan spesifikasi kebutuhan. Kebutuhan yang telah dijelaskan dalam spesfikasi kebutuhan perangkat lunak menjadi acuan untuk melakukan validitas kinerja dari kebutuhan tersebut. Pengujian menggunakan metode *Blackbox*, yaitu metode yang menguji validitas alur jalannya program dan kebutuhan fungsionalitas sistem tersebut.

6.4.1 Kasus Uji Pengujian Validasi

Tabel 6.3 Kasus Uji Melakukan *Register*

Nomor Kasus Uji	PV-*Audiosafe*-01
Nama Kasus Uji	Uji Melakukan *register*
Objek Uji	SRS-*Audiosafe*-01
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional melakukan register dalam perangkat lunak dapat dijalankan dan *user* dapat terdaftar dalam *database user server Audiosafe*
Prasyarat	*User* membuka perangkat lunak dan belum memiliki akun
Langkah - Langkah	1. Dalam halaman awal yaitu login, *user* menekan *textlink* register. 2. *User* diarahkan menuju aktifitas *register*. 3. *User* mengisikan *text* field, *username* dan *password*, dan mencentang checkbox 4. *User* menekan tombol *register*.
Hasil yang diharapkan	*User* berhasil mendaftarkan akun sesuai dengan inputan.

Tabel 6.4 Kasus Uji Melakukan *Login*

Nomor Kasus Uji	PV-*Audiosafe*-02
Nama Kasus Uji	Uji Melakukan *Login*
Objek Uji	SRS-*Audiosafe*-02
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional melakukan *login* dalam perangkat lunak dapat dijalankan dan *user* yang terdaftar dapat masuk kedalam sistem untuk melakukan fungsionalitas lainnya.
Prasyarat	*User* membuka perangkat lunak dan telah memiliki akun
Langkah – Langkah	1. Dalam halaman awal, u*ser* mengisikan *text* field, *username* dan *password*. 2. *User* menekan tombol *login*
Hasil yang diharapkan	*User* berhasil *login* dengan munculnya pesan *login* sukses dan diarahkan menuju aktifitas *main*.

Tabel 6.5 Kasus Uji Mengunggah Audio

Nomor Kasus Uji	PV-*Audiosafe*-03
Nama Kasus Uji	Uji Mengunggah Audio
Objek Uji	SRS-*Audiosafe*-03
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional mengunggah *digital audio* dalam perangkat lunak dapat dijalankan

	dan *member* dapat menyimpan dan melihat *digital audio* yang sudah berhasil diunggah ke server
Prasyarat	*Member* membuka Tab *Profile*
Langkah – Langkah	1. *User* membuka *tab profile* dan menekan tombol *upload audio..* 2. *Member* menginputkan *audioname, key, pathimage dan pathaudio*. 3. *Member* menekan tombol *upload*
Hasil yang diharapkan	*Digital audio* dapat tersimpan didalam server.

Tabel 6.6 Kasus Uji Melihat Daftar Audio

Nomor Kasus Uji	PV-*Audiosafe*-04
Nama Kasus Uji	Uji Melihat Daftar Audio
Objek Uji	SRS-*Audiosafe*-04
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional melihat daftar audio dalam perangkat lunak dapat dijalankan dan *member* dapat melihat daftar audio dari *member* lain.
Prasyarat	*Member* membuka Tab *Online Audio*
Langkah – Langkah	1. *Member* membuka *tab online audio* pada sisi kiri tab activity *main.* 2. Perangkat lunak akan memuat *digital audio*yang ada dalam *server*, *member* menunggu hingga proses memuat selesai.
Hasil yang diharapkan	*Member* dapat melihat daftar *digital audio* yang dimiliki oleh seluruh *member* lain.

Tabel 6.7 Kasus Uji Mengunduh Audio

Nomor Kasus Uji	PV-*Audiosafe*-05
Nama Kasus Uji	Uji Mengunduh Audio
Objek Uji	SRS-*Audiosafe*-05
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional mengunduh audio dalam perangkat lunak dapat dijalankan dan *member* dapat mengunduh *digital audio* dari hasil pilihan.
Prasyarat	*Member* membuka *tab online audio*
Langkah – Langkah	*1.* *Member* memilih tab *online audio,* yang selanjutnya memilih salah satu item *digital audio*. 2. *Member* akan diarahkan ke halaman *download audio* dengan nama audio, cover dan pengunggah yang sesuai dengan pilihan. 3. *Member* menekan *button download.*
Hasil yang diharapkan	Digital audio diunduh dari *server* dan disimpan di *external storage* device Android*member*.

Tabel 6.8 Kasus Uji Melihat Daftar Audio yang Terunduh

Nomor Kasus Uji	PV-*Audiosafe*-06
Nama Kasus Uji	Uji Melihat Daftar Audio yang Terunduh
Objek Uji	SRS-*Audiosafe*-06
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional melihat *digital audio* yang telah *member* unduh dalam perangkat lunak dapat dijalankan dan *member* dapat melihat daftar *digital audio* yang telah di*download*
Prasyarat	*Member* membuka *tab downloaded audio*
Langkah – Langkah	1. *Member* membuka *tab downloaded audio* pada sisi tengah antara dua *tab* activity *main* 2. Perangkat lunak akan memuat *digital audio* yang ada dalam *external storage device member*, 3. *Member* menunggu hingga proses memuat *digital audio* selesai.
Hasil yang diharapkan	*Member* dapat melihat daftar *digital audio* yang ia miliki di *external storage device.*

Tabel 6.9 Kasus Uji Memutar Audio

Nomor Kasus Uji	PV-*Audiosafe*-07
Nama Kasus Uji	Uji Memutar Audio
Objek Uji	SRS-*Audiosafe*-07
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional memutar *digital audio* dalam perangkat lunakdapat dijalankan dan *member* dapat mendengarkan dari konten *digital audio* tersebut.
Prasyarat	*Member* membuka *tab downloaded audio*
Langkah - Langkah	1. *Member* memilih *digital audio* yang tersedia dalam halaman *downloaded audio* 2. *Member* menekan tombol play pada halaman *detail downloaded audio*.
Hasil yang diharapkan	*Member* akan diberikan *playback* dari *android multimedia* pada halaman tersbut untuk mendengarkan *digial audio*.

Tabel 6.10 Kasus Uji Menghapus Audio diServer

Nomor Kasus Uji	PV-*Audiosafe*-08
Nama Kasus Uji	Uji Menghapus Audio diServer
Objek Uji	SRS-*Audiosafe*-08
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional menghapus audio diserver dalam perangkat lunak dapat dijalankan dan *member* dapat menghilangkan *digital audio* yang mereka unggah ke *server*.

Prasyarat	*Member* membuka *tab profile*
Langkah - Langkah	1. *Member* memilih *digital audio* yang tampil dalam halaman *profile*. 2. *Member* menekan tombol *delete* pada halaman tersebut. 3. Akan muncul alert dialog untuk konfirmasi *penghapusan* dari perangkat lunak. *Yes* untuk menghapus, dan *No* untuk batal
Hasil yang diharapkan	*Digital Audio* berhasil dihapus dari *server* dan *file* dalam *server* juga terhapus.

Tabel 6.11 Kasus Uji Menghapus Audio yang Terunduh

Nomor Kasus Uji	PV-*Audiosafe*-09
Nama Kasus Uji	Uji Menghapus Audio yang sudah diunduh
Objek Uji	SRS-*Audiosafe*-09
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional menghapus *digital audio* yang sudah *member unduh* dalam perangkat lunak yang dapat dijalankan dan *member* dapat menghilangkan *digital audio* yang mereka unduh dari *server*
Prasyarat	*Member* membuka *tab downloaded audio*
Langkah - Langkah	1. Me*mber* memilih *digital audio* yang tampil dalam halaman *downloaded audio*. 2. *Member* menekan tombol *delete* pada halaman tersebut. 3. Akan muncul alert dialog untuk konfirmasi *penghapusan* dari perangkat lunak. *Yes* untuk menghapus, dan *No* untuk batal
Hasil yang diharapkan	*Digital Audio* berhasil dihapus dari *external storage device.*

Tabel 6.12 Kasus Uji Memberikan Privilege

Nomor Kasus Uji	PV-*Audiosafe*-10
Nama Kasus Uji	Uji Memberikan Privilege
Objek Uji	SRS-*Audiosafe*-10
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional memberikan privilege kepada *member* lain dalam perangkat lunak yang dapat dijalankan
Prasyarat	*Member* membuka *tab profile*
Langkah - Langkah	1. Me*mber* menekan tombol *privilege user* 2. *Member* memilih salah satu *member* yang ingin *diupdate* status *privilige*nya. 3. *Member* menekan tombol *allow* pada halaman tersebut untuk memberi ijin atau tombol deny untuk menolak ijin. 4. Akan muncul notifikasi bahwa status sudah diperbarui.

Hasil yang diharapkan	Status dalam *privilege* sudah diupdate.

Tabel 6.13 Kasus Uji Melihat Profil

Nomor Kasus Uji	PV-*Audiosafe*-11
Nama Kasus Uji	Uji Melihat profil
Objek Uji	SRS-*Audiosafe*-11
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional melihat profil *member* lain dalam perangkat lunak yang dapat dijalankan
Prasyarat	*Member* membuka *tab profile*
Langkah - Langkah	1. Me*mber* menekan tab *profile* pada tab ketiga. 2. Perangkat lunak akan memuat data profil *member* dan daftar audio apa saya yang berhasil diunggah.
Hasil yang diharapkan	Memuat profil dan seluruh *digital audio* yang telah dilakukan *member*.tersebut

Tabel 6.14 Kasus Uji Melakukan *Logout*

Nomor Kasus Uji	PV-*Audiosafe*-12
Nama Kasus Uji	Uji *Logout*
Objek Uji	SRS-*Audiosafe*-12
Tujuan	Pengujian dilakukan untuk memastikan jika fitur fungsional melakukan *logout* dalam aplikasi dapat dijalankan dan *user* dapat keluar dari layanan aplikasi.
Prasyarat	*Tab* apapun dalam *activity main*
Langkah – Langkah	1. Pada pojok kanan atas terdapat *option menu,* lalu menekan pilihan *logout*
Hasil yang diharapkan	*Member dapat* keluar dari *activity main* dan dilempar menuju halaman *login* kembali.

6.4.2 Hasil Dan Analisa Pengujian Validasi

Berdasarkan pada kasus uji pengujian validasi pada Tabel 6.3 sampai dengan Tabel 6.12 menghasilkan hasil pengujian yang ditampilkan pada Tabel berikut :

Tabel 6.15 Hasil pengujian validasi

No	Nomor Kasus Uji	Hasil yang diharapkan	Hasil yang diperoleh	Status Validasi
1	PV-*Audiosafe*-01	*User* berhasil mendaftarkan akun sesuai dengan inputan.	*User* berhasil mendaftarkan akun sesuai dengan inputan.	*Valid*
2	PV-*Audiosafe*-02	*User* berhasil *login* dengan munculnya	*User* berhasil *login* dengan munculnya	*Valid*

		pesan *login* sukses dan diarahkan menuju aktifitas *main*.	pesan *login* sukses dan diarahkan menuju aktifitas *main*.	
3	PV-*Audiosafe*-03	*Digital audio* dapat tersimpan didalam *server*.	*Digital audio* dapat tersimpan didalam *server*.	*Valid*
4	PV-*Audiosafe*-04	*Member* dapat melihat daftar *digital audio* yang dimiliki oleh seluruh member lain.	*Member* dapat melihat daftar *digital audio* yang dimiliki oleh seluruh member lain.	*Valid*
5	PV-*Audiosafe*-05	*Digital audio* diunduh dari server dan disimpan di *external storage device Android member*.	*Digital audio* diunduh dari server dan disimpan di *external storage device Android member*.	*Valid*
6	PV-*Audiosafe*-06	*Member* dapat melihat daftar *digital audio* yang *member* miliki di *external storage device*.	*Member* dapat melihat daftar *digital audio* yang *member* miliki di *external storage device*.	*Valid*
7	PV-*Audiosafe*-07	*Member* akan diberikan *playback* dari *android multimedia* pada halaman tersbut untuk mendengarkan *digital audio*	*Member* akan diberikan *playback* dari *android multimedia* pada halaman tersbut untuk mendengarkan *digital audio*	*Valid*
8	PV-*Audiosafe*-08	*Digital Audio* berhasil dihapus dari *server* dan *file* dalam *server* juga terhapus.	*Digital Audio* berhasil dihapus dari server dan *file* dalam *server* juga terhapus.	*Valid*
9	PV-*Audiosafe*-09	*Digital Audio* berhasil dihapus dari *external storage device*.	*Digital Audio* berhasil dihapus dari *external storage device*.	*Valid*
10	PV-*Audiosafe*-10	Status dalam *privilege* sudah diupdate.	Status dalam *privilege* sudah diupdate.	*Valid*
11	PV-*Audiosafe*-11	Memuat profil dan seluruh *digital audio* yang telah dilakukan member tersebut	Memuat profil dan seluruh *digital audio* yang telah dilakukan member tersebut	*Valid*
12	PV-*Audiosafe*-12	Member dapat keluar dari *activity main* dan dilempar menuju halaman *login* kembali.	Member dapat keluar dari *activity main* dan dilempar menuju halaman *login* kembali.	*Valid*

6.5 Pengujian Keamanan Data

Pengujian keamanan yang meliputi performansi waktu eksekusi proses enkrpsi dan dekripsi kriptografi, keaslian data dalam aplikasi dan sekuritas *member*.

6.5.1 Pengujian Performasi Waktu Eksekusi Kriptografi

6.5.1.1 Rencana Uji Performansi Waktu Eksekusi Kriptografi

Pengujian performansi waktu eksekusi merupakan pengujian yang dilakukan untuk menguji dan mengetahui tingkat performansi proses kriptografi dari sisi kecepatan proses enkripsi maupun dekripsi terhadap besarnya *file* yang terjadi pada proses kriptografi dalam perangkat lunak. Perangkat lunak *Audiosafe* menggunakan teknik kriptografi *AES*, yang akan diuji performansinya dengan kecepatan proses yang dibutuhkan perangkat lunak untuk melakukan proses enkripsi maupun dekripsi terhadap besarnya *file* yang akan akan di enkripsi maupun didekripsi.

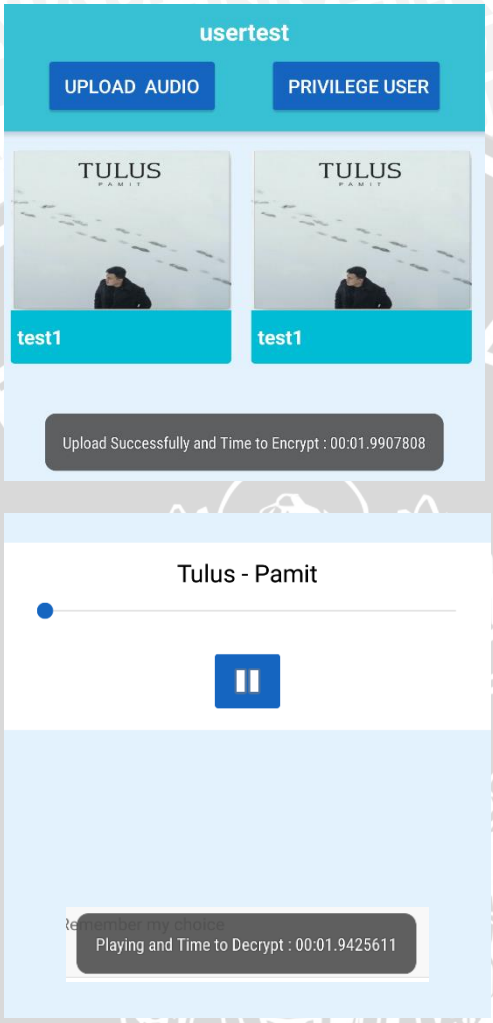
Pengukuran performansi waktu eksekusi dilakukan dengan mengukur numerikal dan trendline. Trendline merupakan garis yang dibuat melalui perhitungan secara statistik. (Widiyaningsih, 2012). Pengukuran kinerja waktu kriptografi juga didasarkan pada APDEX, yaitu index performa aplikasi yang ditentukan dalam tiga ukuran, yaitu *satisfiying response time, tolerable response time* dan *frustated response time.* (Apdex Alliance, 2007). Ukuran didasarkan pada T, waktu standar sebuah proses berjalan. Jika waktu respon proses antara 0-T maka dikatakan *satisfiying.* T-4T, *tolerable* dan diatas 4T adalah *frustated.* T yang digunakan dalam pengujian waktu eksekusi ini adalah detik. Detik adalah satuan waktu yang sangat kecil dan sering digunakan dalam operasi hitung. (Prihantoro, 2009) Sehingga 4 detik digunakan acuan batas *tolerable response time* yang digunakan dalam pengukuran ini.

Dibuatlah kasus uji untuk melakukan kedua teknik pengujian tesebut, dengan tujuan untuk mengukur performansi teknik kriptografi AES. Berikut dalam Tabel 6.16 dan Tabel 6.17 disajikan kasus uji dan skenario pengujian performansi kecepatan kriptografi.

Tabel 6.16 Kasus uji Pengujian Performansi *AES*

Nomor Kasus Uji	Nama Kasus Uji	Data Uji : Digital Audio Besarnya Data *(byte)*
Perf_*AES*_01	Pengujian Performansi *AES* Data 1	1.341.941
Perf_*AES*_02	Pengujian Performansi *AES* Data 2	2.484.942
Perf_*AES*_03	Pengujian Performansi *AES* Data 3	3.328.626
Perf_*AES*_04	Pengujian Performansi *AES* Data 4	4.309.423
Perf_*AES*_05	Pengujian Performansi *AES* Data 5	5.270.666
Perf_*AES*_06	Pengujian Performansi *AES* Data 6	6.386.812
Perf_*AES*_07	Pengujian Performansi *AES* Data 7	7.771.590
Perf_*AES*_08	Pengujian Performansi *AES* Data 8	8.731.432

Tabel 6.17 Skenario uji performansi kecepatan kriptografi

Nomor Kasus Uji	Perf_*AES*_01 s/d Perf_*AES*_08
Tools	*Library stopwatch.java.* Contoh tampilan tool pengujian terintegrasi dalam *mobile* : 
Data Pengujian	Data Uji Perf_*AES*_01 s/d Data uji Perf_*AES*_08
Langkah Pengujian	Proses Enkripsi *AES* : 1. Data dibaca (*Digital audio*/*String request*) 2. *Key inputan* dibaca 3. Mulai pencatatan waktu 4. Enkrip digital audio 5. Hentikan pencatatan waktu, tampilkan. Proses Dekripsi *AES*: 1. Data terenkrip dibaca (*Digital audio*/*String request*) 2. *Key audio* dari *server* dibaca 3. Mulai pencatatan waktu 4. Dekrip data (*Digital audio*/ *String request*) 5. Hentikan pencatatan waktu, tampilkan.

Expected Value	*Tolerable response process time* : 4 detik *(*Berdasarkan indeks *user experience* yang dibentuk oleh APDEX)

6.5.1.2 Hasil dan Analisa Pengujian Performansi Waktu Eksekusi Kriptografi

Berdasarkan pada kasus uji pengujian performansi yang dituliskan sebelumnya didapatkan hasil pengujian performansi proses kriptografi *AES* pada Tabel 6.18 dan Tabel 6.19.

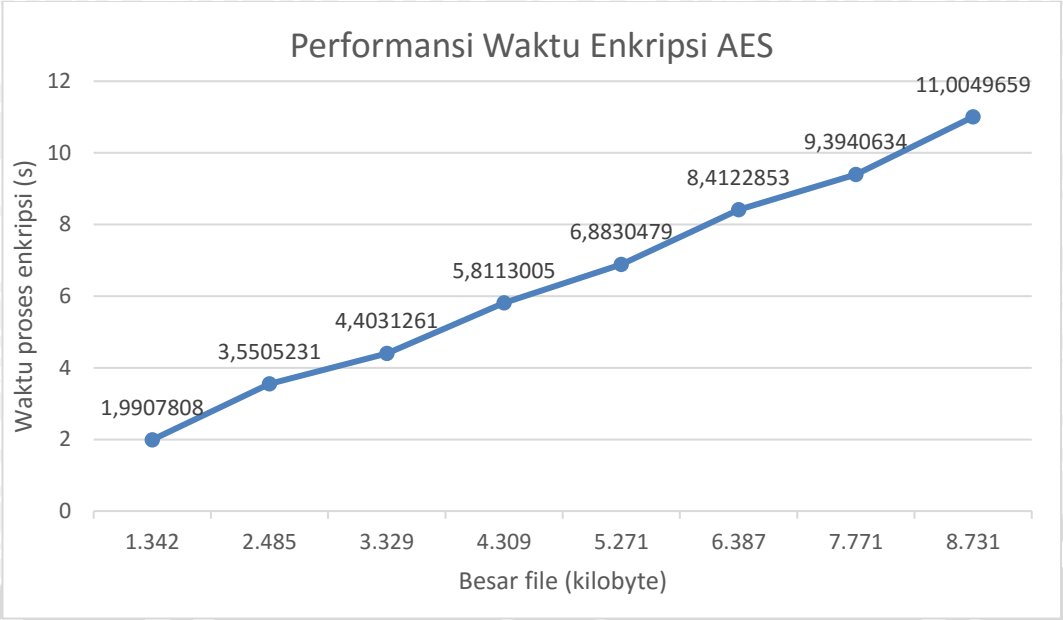
Tabel 6.18 Hasil pengujian performansi Enkripsi *AES*

No	Nomor Kasus Uji	Besar File (Kilo*bytes)*	Waktu proses enkripsi *(s)*
1	Perf_*AES*_01	1.342	01,9907808
2	Perf_*AES*_02	2.485	03,5505231
3	Perf_*AES*_03	3.329	04,4031261
4	Perf_*AES*_04	4.309	05,8113005
5	Perf_*AES*_05	5.271	06,8830479
6	Perf_*AES*_06	6.387	08,4122853
7	Perf_*AES*_07	7.771	09,3940634
8	Perf_*AES*_08	8.731	11,0049659

Tabel 6.19 Hasil pengujian performansi Dekripsi *AES*

No	Nomor Kasus Uji	Besar File (Kilo*ytes)*	Waktu proses dekripsi *(s)*
1	Perf_*AES*_01	1.342	01,9425611
2	Perf_*AES*_02	2.485	02,8530258
3	Perf_*AES*_03	3.329	04,6409801
4	Perf_*AES*_04	4.309	05,9836158
5	Perf_*AES*_05	5.271	06,6836633
6	Perf_*AES*_06	6.387	08,5483975
7	Perf_*AES*_07	7.771	09,9532429
8	Perf_*AES*_08	8.731	10,6805434

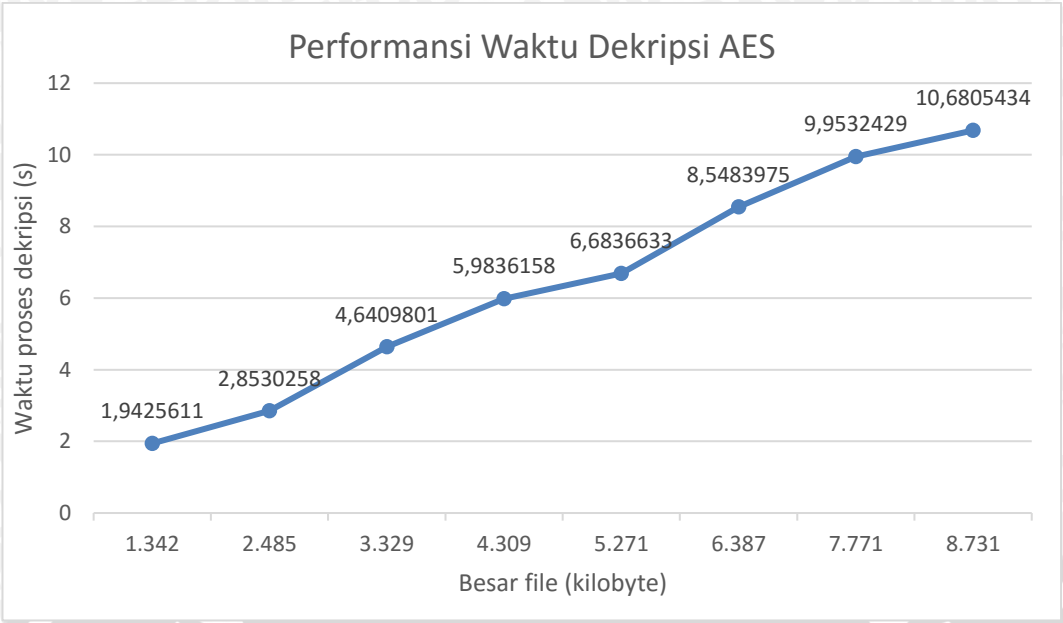
Berdasarkan tabel hasil pengujian performansi kriptografi *AES* didapatkan bentuk grafik pengaruh besar *file* atau besar data terhadap performa kedua teknk kriptografi tersebut



Gambar 6.3 Grafik waktu proses enkripsi *AES* terhadap besar *file*

Proses Enkripsi dilakukan pada perangkat lunak Audiosafe dalam *device*. Waktu proses enkripsi pada sample bernilai maksimal pada data ke 8 saat besar *file* 8.731 K*iloByte* atau sekitar 8,7 *MegaByte* yang bernilai hingga 11,004 detik. Dan waktu tercepat di antara sampel pemrosesan enkripsi berada pada data sample pertama yang bernilai 1.342 K*iloByte* yaitu bernilai 1,99 detik. Pada kasus uji 1-8, performansi waktu atau kecepatan enkripsi mengalami peningkatan sejalan dengan peningkatan ukuran data. Trendline dari pengaruh besar data terhadap waktu proses eksekusi enkripsi berbentuk pola linear yang terus naik. Dengan demikian dapat disimpulkan jika besar data sangat mempengaruhi performansi cepat atau lambatnya proses enkripsi. Jika waktu proses enkripsi yang didapatkan melalui sample tersebut terhadap expected value yaitu 4 detik, maka proses dekripsi data hingga 8,7 M*egaByte* adalah sesuatu yang tidak relevan dan performansinya sangatlah buruk jika mengacu pada index APDEX (Application Performance Index).

Diharapkan waktu proses enkrpsi terjadi seminimal mungkin, rentang performansi yang masih tolerable terjadi pada sample/ kasus uji antara 1 hingga 3 yang memberikan respon time/ proses enkripsi pada waktu 4.4 detik, dan dapat ditarik kesimpulan jika *Audiosafe* lebih baik menggunakan *audio digital* yang berukuran tidak lebih atau sama dengan kasus uji 3 dengan ukuran kurang lebih 3 Mega*Byte* untuk mendapatkan performa yang baik dari proses enkripsi yang diimplementasikan pada perangkat lunak ini.



Gambar 6.4 Grafik waktu proses dekripsi *AES* terhadap besar *file*

Proses Dekripsi juga dilakukan dalam perangkat lunak Audiosafe dalam *device*. Waktu proses dekripsi pada kasus uji bernilai maksimal pada data ke 8 saat besar *file* 8.731 K*iloByte* atau sekitar 8,7 M*egaByte*, bernilai hingga 10,68 detik, dan waktu tercepat dalam proses dekripsi berada pada data ke 1 yang bernilai 1.342 K*iloByte* yang hanya membutuhkan sekitar 1,942 detik. Pada kasus uji 1 hingga 8 terjadi peningkatan waktu dekripsi sejalan dengan besarnya data yang di dekripsi. Trendline dari pengaruh besar *file* terhadap waktu proses dekripsi tampak membentuk garis linear yang terus naik. Dengan demikian dapat disimpulkan jika besar *file chiper* sangat mempengaruhi performansi dari proses dekripsi, semakin besar atau kecilnya ukuran *file chiper* akan mempengaruhi cepat/lambatnya proses dekripsi. Jika waktu proses dekripsi yang didapatkan melalui sample tersebut terhadap expected value yaitu 4 detik, maka proses dekripsi data hingga 8,7 M*egaByte* adalah sesuatu yang tidak relevan dan performansinya sangatlah buruk jika mengacu pada index APDEX (Application Performance Index).

Diharapkan waktu proses dekripsi terjadi seminimal mungkin, rentang performansi yang masih tolerable terjadi pada sample/ kasus uji antara 1 hingga 3 yang memberikan respon time/ proses eksekusi dekripsi pada waktu 4.64 detik, dan dapat ditarik kesimpulan jika *Audiosafe* lebih baik menggunakan *digital audio* yang berukuran tidak lebih atau serupa dengan kasus uji 3 dengan ukuran kurang lebih 3 *MegaByte* untuk mendapatkan performa yang baik dari proses dekripsi yang diimplementasikan pada perangkat lunak ini.

6.5.2 Pengujian Keaslian Data

6.5.2.1 Rencana Uji Pengujian Keaslian Data

Pengujian keaslian data merupakan pengujian yang melakukan uji keaslian data pada objek uji yang mengalami proses kriptografi. Pengukuran dilakukan untuk mengetahui seberapa tingkat pengembalian data asli setelah dilakukan

teknik kriptografi, adakah data yang rusak atau data dapat dikembalikan kebentuk awal dengan tanpa terjadi perubahan konten di dalam objek uji tersebut.

Objek uji dari kriptografi *AES* adalah *file digital audio*, akan di ambil delapan sampel *digital audio*, akan dilakukan pengamatan terhadap MD5 signature masing-masing *file digital audio* pada tahap sebelum mengalami proses enkripsi, mengalami proses enkripsi dan setelah mengalami proses dekripsi. Diamati pula besarnya data seperti teknik yang dilakukan pada pengamatan terhadap MD5. Berikut pada Tabel 6.20 dan Tabel 6.21 disajikan kasus uji, data uji dan skenario pengujian dalam pengujian keaslian data.

Tabel 6.20 Kasus uji dan data uji pengujian keaslian data

Nomor Kasus Uji	Nama Kasus Uji	Objek Uji (*Digital Audio*)	Besarnya File (*byte*)
Rec_*AES*_01	Pengujian Keaslian Data *AES* 1	test1	1.341.941
Rec_*AES*_02	Pengujian Keaslian Data *AES* 2	test2	2.484.942
Rec_*AES*_03	Pengujian Keaslian Data *AES* 3	test3	3.328.626
Rec_*AES*_04	Pengujian Keaslian Data *AES* 4	test4	4.309.423
Rec_*AES*_05	Pengujian Keaslian Data *AES* 5	test5	5.270.666
Rec_*AES*_06	Pengujian Keaslian Data *AES* 6	test6	6.386.812
Rec_*AES*_07	Pengujian Keaslian Data *AES* 7	test7	7.771.590
Rec_*AES*_08	Pengujian Keaslian Data *AES* 8	test8	8.731.432

Tabel 6.21 Skenario uji pengujian Keaslian Data *AES*

Nomor Kasus Uji	Rec_*AES*_01 s/d Rec_*AES*_08
Tools	*Manual Check MD5 Checksum* dan *File Bytes*
Data Pengujian	Data uji Rec_*AES*_01 s/d Rec_*AES*_08
Langkah Pengujian	1. Baca *byte file* asli *digital audio* 2. Baca *MD5 checksum file* asli *digital audio* 3. Enkripsi *digital audio* menggunakan *Audiosafe* 4. Baca *byte file* hasil enkripsi data *digital audio* 5. Baca *MD5 checksum file* hasil enkripsi data *digital audio* 6. Dekripsi file hasil enkripsi data *digital audio* meng gunakan *Audiosafe* 7. Baca *byte file* hasil dekripsi data *digital audio* 8. Baca *MD5 checksum file* hasil dekripsi data *digital audio*
Expected Value	1. Besarnya *byte file* asli *digital audio* sebelum di enkripsi sama degan besarnya *file digital audio* setelah didekripsi dengan perangkat lunak. 2. Nilai MD5 *checksum digital audio* sebelum di enkripsi sama dengan MD5 *checksum digital audio* setelah didekripsi dengan perangkat lunak.

6.5.2.2 Hasil dan Analisa Pengujian Keaslian Data

Berdasarkan dari skenario pengujian keaslian data *AES* maka didapatkan hasil yang tercantum pada Tabel 6.22 yang memuat keaslian data *file size* dan Tabel 6.23 yang memuat keaslian data *file content checksum*.

Tabel 6.22 Hasil uji *AES* keaslian data *file size*

No	Nomor Kasus Uji	Besarnya File (*byte*)	Besarnya File Pasca Enkripsi (*byte*)	Besar File Pasca Dekripsi (*byte*)	Status
1	Rec_*AES*_01	1.341.941	1.342.084	1.341.941	*Valid*
2	Rec_*AES*_02	2.484.942	2.485.076	2.484.942	*Valid*

3	Rec_*AES*_03	3.328.626	3.328.772	3.328.626	*Valid*
4	Rec_*AES*_04	4.309.423	4.309.556	4.309.423	*Valid*
5	Rec_*AES*_05	5.270.666	5.270.804	5.270.666	*Valid*
6	Rec_*AES*_06	6.386.812	6.386.948	6.386.812	*Valid*
7	Rec_*AES*_07	7.771.590	7.771.732	7.771.590	*Valid*
8	Rec_*AES*_08	8.731.432	8.731.572	8.731.432	*Valid*

Tabel 6.23 Hasil uji *AES* keaslian data *file content checksum*

No	Nomor Kasus Uji	*Checksum* (i=*initial*, e=*encrypted*, d=*decrypted*)	Status
1	Rec_*AES*_01	ci = 2eb92c093393867f55b79a0904c75a81 ce= b7660766eab29906b871ac878d2d563f cd= 2eb92c093393867f55b79a0904c75a81	*Valid*
2	Rec_*AES*_02	ci = 12fc84d3feb3b0907a323786b8c5ec98 ce= e56aeec97d55c555e347715874df1b49 cd= 12fc84d3feb3b0907a323786b8c5ec98	*Valid*
3	Rec_*AES*_03	ci = 398a142737e3844d3c2216e20b6c44a1 ce= f68d073f29e8ebd55dac7cec2b9b860d cd= 398a142737e3844d3c2216e20b6c44a1	*Valid*
4	Rec_*AES*_04	ci = 216973dc69cd89c57af034d350c5d466 ce= 290da673085464a520cd7f56e08920ac cd= 216973dc69cd89c57af034d350c5d466	*Valid*
5	Rec_*AES*_05	ci = 4106b90ce50fba18ac508cffc031ec9b ce= 6283fe87e5f13e801102d01c54abb0e7 cd= 4106b90ce50fba18ac508cffc031ec9b	*Valid*
6	Rec_*AES*_06	ci= 6b979721f1331a3b5f1306c45e68180b ce= f3eec2e869eba900b0c805d8f9287174 cd= 6b979721f1331a3b5f1306c45e68180b	*Valid*
7	Rec_*AES*_07	ci= 7d74d98b3010a059110572e398eda217 ce= 3c3f7ad3706f078e87885f7380d8735a cd= 7d74d98b3010a059110572e398eda217	*Valid*
8	Rec_*AES*_08	ci= 21cdfbe666486246ddd1edfd8e3cb144 ce= 2c67b0f3fef12c068e65ee041e108d8f cd= 21cdfbe666486246ddd1edfd8e3cb144	*Valid*

Hasil dari pengujian keaslian data didasarkan pada keaslian data besarnya *file* dan konten *file* dengan menggunakan MD5 *Checksum* setelah didekripsi kembali dengan menggunakan perangkat lunak sehingga mendapatkan data yang sama dengan data awal sebelum *file* mengalami proses enkripsi. Melalui Tabel 6.22 di dapatkan status data kasus uji bernilai seluruhnya *valid* yang berarti besar *file* akan kembali seperti semula setelah mengalami proses dekripsi dari perangkat lunak. Tingkat keaslian data besar *file* dari delapan kasus uji bernilai hingga 100% yang berarti besar file yang terkena imbas dari enkripsi *AES*-256 dapat dikembalikan keukuran semula.

Sementara dari pengujian keaslian data yang didasarkan pada *file content checksum* pada Tabel 6.23 di atas, *checksum file* delapan kasus uji setelah mengalami proses dekripsi bernilai sama persis dengan *checksum file* asli sebelum mengalami proses enkripsi. *Checksum* digunakan untuk melihat apakah terjadi perubahan data terhadap suatu *file*, secara tidak langsung nilai *checksum* yang sama antara hasil dekripsi dan *file* asli membuktikan jika tidak ada konten *file* yang

berubah selama proses enkripsi. Tingkat keaslian data *file content checksum* dari delapan kasus uji bernilai hingga 100% yang berarti konten dari sampel *file* uji yang terkena imbas dari enkripsi *AES*-256 dapat dikembalikan kebentuk semula tanpa mengubah isi dari konten setelah dilakukan dekripsi.

6.5.3 Pengujian *Security*

6.5.3.1 Rencana Uji *Security*

Pengujian *Security* adalah pengujian untuk mengukur tingkat keamanan dari pemutaran audio yang dilakukan oleh *member*. Pengujian dilakukan untuk mengukur seberapa mampu perangkat lunak dalam mengirimkan kunci dan memutar audio ketika *privilege* kepada *member* lain diijinkan maupun ditolak dan secara tidak langsung mengukur seberapa efektif teknik *privilege member* dan teknik kriptografi untuk menekan penyebaran distribusi *digital audio* pada *member* yang tidak berwenang atau tidak diberi ijin untuk pemutaran.

Pengujian dilakukan dengan membuat sampel uji terdiri dari 2 *username,* 2 *digital audio* dan 2 kondisi status *privilege* yang disiapkan untuk 1 *username* penerima *privilege* tersebut, lalu dilakukan percobaan pemutaran *digital audio* secara kombinatorial. Hasil yang dharapkan adalah jika *privilege* diberikan status *allow* maka *file digital audio* dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak. Hasil yang diharapkan juga jika diberikan status *deny* maka perangkat lunak tidak akan melakukan proses dekripsi kepada *file digital audio* dan mengeluarkan notifikasi.

Tabel 6.24 Data uji pengujian security *AES*

Nomor Kasus Uji	Nama Kasus Uji	*username* pemberi *privilege*	*username* penerima *privilege*	status *privilege*	nama audio
Sec_*AES*_01	Pengujian *Security AES* 1	usertest1	usertest2	*allow*	test1
Sec_*AES*_02	Pengujian *Security AES* 2	usertest1	usertest2	*deny*	test1
Sec_*AES*_03	Pengujian *Security AES* 3	usertest2	usertest2	*allow*	test2
Sec_*AES*_04	Pengujian *Security AES* 4	usertest2	usertest2	*deny*	test2

Tabel 6.25 Skenario uji pengujian security *key* collision *AES*

Nomor Kasus Uji	Sec_*AES*_01 s/d Sec_*AES*_04
Data Pengujian	Data Uji Sec_*AES*_01 s/d Data uji Sec_*AES*_04
Langkah Pengujian	1. Menekan tombol play dalam halaman *detail downloaded audio* 2. Audiosafe mengirimkan request ke *webservice* 3. Server mengirimkan status *privilege* 4. Kumpulkan hasil kasus uji, dan bandingkan.
Expected Value	1. Diberikan status *allow* maka *file digital audio* dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.

	2. Diberikan status *deny* maka perangkat lunak tidak akan melakukan proses dekripsi kepada *file digital audio* dan mengeluarkan notifikasi.

6.5.3.2 Hasil dan Analisa Pengujian *Security*

Berdasarkan kasus uji pengujian security *AES*, penulis mencoba membandingkan status untuk memutar *digital audio* dalam kasus uji yang dibentuk melalui kombinasi *username, file digital audio* dan status *privilege* yang kemudian diterapkan dalam proses pemutaran *digital audio* dalam perangkat lunak Audiosafe untuk mengetahui tingkat sekuritas yang mungkin terjadi. Berikut adalah dalam Tabel 6.27 disajikan hasil dari kasus uji dan ekpetasi nilai yang dihasilkan valid atau tidak.

Tabel 6.26 Hasil pengujian *security*

No	Nomor Kasus Uji	*Expected Value*	*Result Value*	*Status Validasi*
1	Sec_*AES*_01	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*valid*
2	Sec_*AES*_02	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	*valid*
3	Sec_*AES*_03	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*File* digital audio dapat didekripsi dengan kunci yang dikirimkan oleh server dan dapat diputar dalam perangkat lunak.	*valid*
4	Sec_*AES*_04	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	Perangkat lunak tidak akan melakukan proses dekripsi kepada file digital audio dan mengeluarkan notifikasi	*valid*

Dari Tabel 6.27 pengujian *security* yang didasarkan pada *expected value* dan *result value*, status validitas pada empat kasus uji setelah mengalami proses pemutaran *digital audio* menunjukan semua valid. Tingkat *security* dari empat kasus uji bernilai hingga 100% valid yang berarti keamanan dalam mengukur seberapa efektif teknik *privilege member dan* teknik kriptografi untuk menekan penyebaran distribusi *digital audio* pada *member* yang tidak berwenang atau tidak diberi ijin untuk pemutaran benar-benar aman dalam penggunaan perangkat lunak ini.

6.6 Analisis Hasil Pengujian

Pengujian Unit dilakukan dengan membuat flowgraph pada sampel algoritma, menghitung kompleksitas siklamatis, membentuk basis set dari jalur independen, dan menguji berdasarkan unit tersebut. Pengujian dilakukan pada sampel

algoritma yaitu mengunggah audio dan memutar audio. Didapatkan bahwa dari hasil kasus uji, hasil yang diharapkan dari kasus uji sama dengan hasil yang didapatkan.

Pengujian Validasi dilakukan dengan membuat kasus uji berdasarkan kebutuhan fungsional yang diuji dalam validitas kinerja dan *item.* Kebutuhan fungsional yang sudah dispesifikasikan menghasilkan 12 fitur yang akan diuji. Berdasarkan hasiil dari pengujian validasi, maka semua memberikan hasil valid atau dengan kata lain, hasil yang diharapkan dalam kasus uji sama dengan apa yang didapatkan dalam pengujian.

Pengujian selanjutnya adalah menguji keamanan data berdasarkan 3 parameter yaitu, waktu performansi kriptografi, keaslian data dan *security*. Pengujian waktu performansi dilakukan ketika file mengalami proses enkripsi dan dekripsi untuk mengukur waktu yang diperlukan, pengujian keaslian data dilakukan untuk menguji keutuhan *file* saat mengalami proses dekripsi ketika sudah berhasil dienkripsi dan pengujian *security* dilakukan untuk menguji keefektifan teknik *privilege* dalam pengamanan pemutaran dan distribusi audio.

Berdasarkan hasil pengujian waktu performansi kriptografi, waktu yang diperlukan untuk menghasilkan waktu tolerable beradasarkan nilai APDEX dengan syarat kurang lebih file berukuran 3 MegaByte. Berdasarkan hasil pengujian untuk waktu dengan delapan sampel enkripsi pada file yang berukuran kurang lebih 3 MegaByte menghasilkan waktu 4,4 detik untuk mendapatkan performa waktu eksekusi dalam rentang tolerable jika mengacu pada APDEX. Demikian juga pada proses dekripsi dengan delapan sampel, untuk waktu yang dibutuhkan dalam file kurang lebih 3 MegaByte menghasilkan waktu 4,64 detik untuk mendapatkan performa waktu eksekusi dalam rentang tolerable jika mengacu pada APDEX.. Proses enkripsi dan dekripsi data lebih dari ukuran tersebut di dalam perangkat lunak mengakibatkan performa kecepatan waktu menjadi buruk atau frustated jika mengacu pada APDEX.

Pengujian keaslian data menunjukan hasil bahwa file yang mengalami proses enkripsi akan dapat kembali pada hasil yang sama saat mengalami proses dekripsi. Berdasarkan hasil pengujian dengan delapan sampel, tingkat keaslian data file dalam perangkat lunak semuanya dapat kembali seperti *file* asli dengan mencocokan ukuran data dan MD5 *Checksum*. Pengujian security menunjukan hasil bahwa berdasarkan proses pada perangkat lunak, file digital audio tidak dapat diputar ketika belum diberikan *privilege* dari *user* pengunggah digital audio. Berdasarkan hasil pengujian, tingkat security file dalam perangkat lunak semuanya valid ketika diberikan kasus uji kombinatorial yang berbeda.

Perangkat lunak juga masih mengalami kekurangan minor yang belum disebutkan dalam pengujian yaitu, perangkat lunak mengalami *forced closed* ketika audio unduhan berada pada device saat *user* pemilik audio tersebut login kedalam perangkat lunak.

BAB 7 KESIMPULAN DAN SARAN

Bab ini membahas tentang penarikan kesimpulan berdasarkan hasil dari penelitian yang dilakukan yang bertujuan untuk menjadi landasan pengetahuan penelitian selanjutnya dan saran yang bisa dijadikan untuk penyempurnaan penelitian selanjutnya.

7.1 Kesimpulan

Berdasarkan perancangan, implementasi dan hasil pengujian serta analisis yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut.

1. Perancangan perangkat lunak Audiosafe menggunakan tahap perancangan fungsional berdasarkan hasil spesifikasi kebutuhan yang menghasilkan 12 fitur dan perancangan non fungsional atau keamanan data dengan merancang kinerja dari sisi kriptografi yaitu enkripsi dan dekripsi pada perangkat lunak berdasarkan 3 parameter yaitu waktu performansi kriptografi, keaslian data dan *security*.
2. Implementasi dari perangkat lunak dikembangkan berdasarakan perancangan padah tahap sebelumnya dan diimplementasikan dalam bahasa pemrograman Java dengan development tool *Android Developer Tool* yaitu *Android Studio* dan diimplementasikan dalam *device smartphone Android.* Dalam implementasi perangkat lunak menggunakan jenis pendekatan *Native Apps* karena proses dalam kriptografi membutuhkan resource yang besar, dan *native apps* dipilih karena dapat menggunakan semua *resource* dari sistem operasi tersebut.
3. Pengujian dilakukan dengan beberapa tahap yaitu, pengujian unit (*Whitebox*), pengujian validasi (*Blackbox*) dan pengujian pada keamanan data. Pengujian keamanan data menggunakan 3 parameter yaitu waktu performansi kriptografi, keaslian data dan *security*.
4. Pengujian Unit (*Whitebox*) menghasilkan hasil jika fungsionalitas perangkat lunak disisi pengujian unit algoritma fungsi telah memenuhi hasil yang diharapkan. Pengujian Validasi (*Blackbox*) menghasilkan jika fungsionalitas perangkat lunak telah memenuhi kebutuhan yang diinginkan atau *valid* pada spesifikasi kebutuhan.
5. Pengujian waktu performansi kriptografi AES yang diimplementasikan dalam perangkat lunak menghasilkan waktu tolerable beradasarkan nilai APDEX dengan syarat kurang lebih *file* berukuran 3 MegaByte. Berdasarkan hasil pengujian untuk waktu enkripsi pada *file* yang berukuran kurang lebih 3 MegaByte menghasilkan waktu 4,4 detik untuk mendapatkan performa waktu eksekusi dalam rentang tolerable jika mengacu pada APDEX. Demikian juga pada proses dekripsi, untuk waktu yang dibutuhkan dalam *file* kurang lebih 3 MegaByte menghasilkan waktu 4,64 detik untuk mendapatkan performa waktu eksekusi dalam rentang tolerable jika mengacu pada APDEX.. Proses

enkripsi dan dekripsi data lebih dari ukuran tersebut di dalam perangkat lunak mengakibatkan performa kecepatan waktu menjadi buruk atau *frustated* jika mengacu pada APDEX.

6. Pengujian keaslian data menunjukan hasil bahwa file yang mengalami proses enkripsi akan dapat kembali pada hasil yang sama saat mengalami proses dekripsi. Berdasarkan hasil pengujian, tingkat keaslian data file dalam perangkat lunak bernilai 100%.
7. Pengujian *security* menunjukan hasil bahwa berdasarkan proses pada perangkat lunak, *file digital audio* tidak dapat diputar ketika belum diberikan *privilege* dari *user* pengunggah *digital audio*. Berdasarkan hasil pengujian, tingkat *security* file dalam perangkat lunak bernilai 100%.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan perangkat lunak ini bagi penelitian selanjutnya antara lain :

1. Implementasi perangkat lunak Audiosafe hanya dapat dijalankan pada versi SDK 21 keatas , diharapkan pada penelitian selanjutnya dapat diimplementasi juga pada versi sebelumnya.
2. Sistem operasi yang diimplementasikan apda perangkat lunak ini adalah Android, diharapkan implementasi dalam penelitian ini juga dilakukan pada sistem operasi lain seperti *IOS,* atau *Windows Phone* untuk mengukur performansi waktu eksekusi kriptografi, keaslian data dan *security* data pada *environment* berbeda.
3. Pada penelitian ini, *key* yang diberikan oleh server ke *device,* dikirimkan tampa melalui pengamanan data sehingga dapat menyebabkan *snipping* data pada pihak ketiga dan dapat melakukan dekripsi bebas pada *file.* Diharapkan pada penelitian selanjutnya dapat meneliti pengamanan data pada *key* dan data yang mengalami proses transaksi antara *server* dan *device*.

DAFTAR PUSTAKA

Apdex Alliance, I. (2007). *Application Performance Index - Apdex Technical Spesification.* Apdex Aliance, Inc.

Boci, L., & Chen, Y. (2012). Comparison Between JSON and XML in Application on AJAX. *International Conference on Conference Science and Service System*.

Dika, M. F., Tolle, H., & Data, M. (2015). Rancang Bangun Digital Book Reader Berbasis Android dengan Keamanan Data Rhivest Shamir Adleman dan Advandced Encryption Standard.

El-Aziz, A. A., & Kannan, A. (2014). JSON Encryption. *International Conference on Computer Communication and Informatics (ICCCI)*.

Encyclopedia. (2016). Diambil kembali dari PCMAG: http://www.pcmag.com/encyclopedia/term/46725/media-player

Endriani, N., & Utami, E. (2014). Implementasi Algoritma Enkripsi AES pada Aplikasi SMS (Short Message Service) Berbasis Android.

Gadanayak, B., & Pradhan, C. (2011). Encryption on MP3 Compression. *MES Journal of Technology and Management* , 86-89.

Mahajan, P., & Sachdeva, A. (2013). A Study of Encryption Algorithms AES, DES and RSA for. *Global Journal of Computer Science and Technology Network, Web & Security*.

Oxford University, P. (2005). *OXFORD Ensiklopedi Pelajar, terjemahan PT Widyadara.* Jakarta: PT Widyadara.

Patel, R., & Kanjariya, S. (2015). Design of Parallel Advanced Encryption Standard (AES) Algorithm. *International Journal of Research in Computer and Communication Technology*, 219-222.

Pramudyanto, A. B. (2013). Media Baru dan Peluang Counter-Hegemony atas Dominasi Logika Industri Musik (Studi Kasus Perkmbangan Netlabel di Indonesia). *Jurnal ILMU KOMUNIKASI*, 63-82.

Prihantoro, S. (2009, Januari). *Operasi Hitung.* Diambil kembali dari https://sonyprihantoro.files.wordpress.com/2009/01/operasi-hitung-satuan-waktu.pdf

Wicaksono, R. (2013, 3 13). *Padding Oracle Attack*. Diambil kembali dari http://www.ilmuhacking.com/cryptography/padding-oracle-attack/

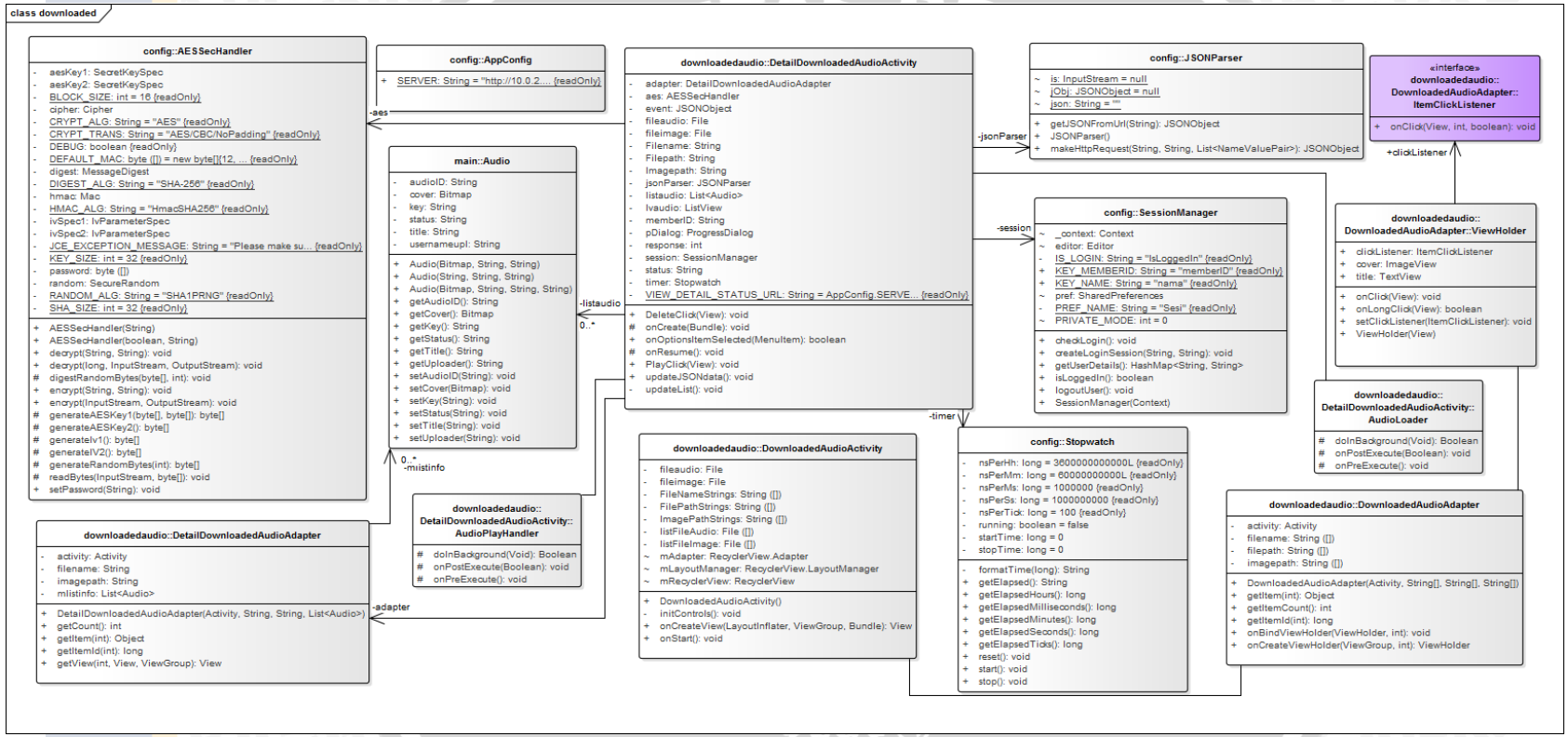
Widiyaningsih, S. (2012, November). *Cara Menambahkan Garis Trendline pada Chart/Grafik Excel*. Diambil kembali dari Cara Semua Cara: http://www.cara.aimyaya.com/2012/11/cara-menambahkan-garis-trendline-pada.html

Yuniati, V., Indriyanta, G., & C, A. R. (2009). Enkripsi dan Dekripsi dengan Algoritma AES 256 untuk Semua Jenis File. *Jurnal Informatika*, 22-31.

Zhou, X., & Tang, X. (2011). Reasearch and Implementation of RSA Algorithm for Encryption and Decryption. *The 6th International Forum on Strategic Technology*.

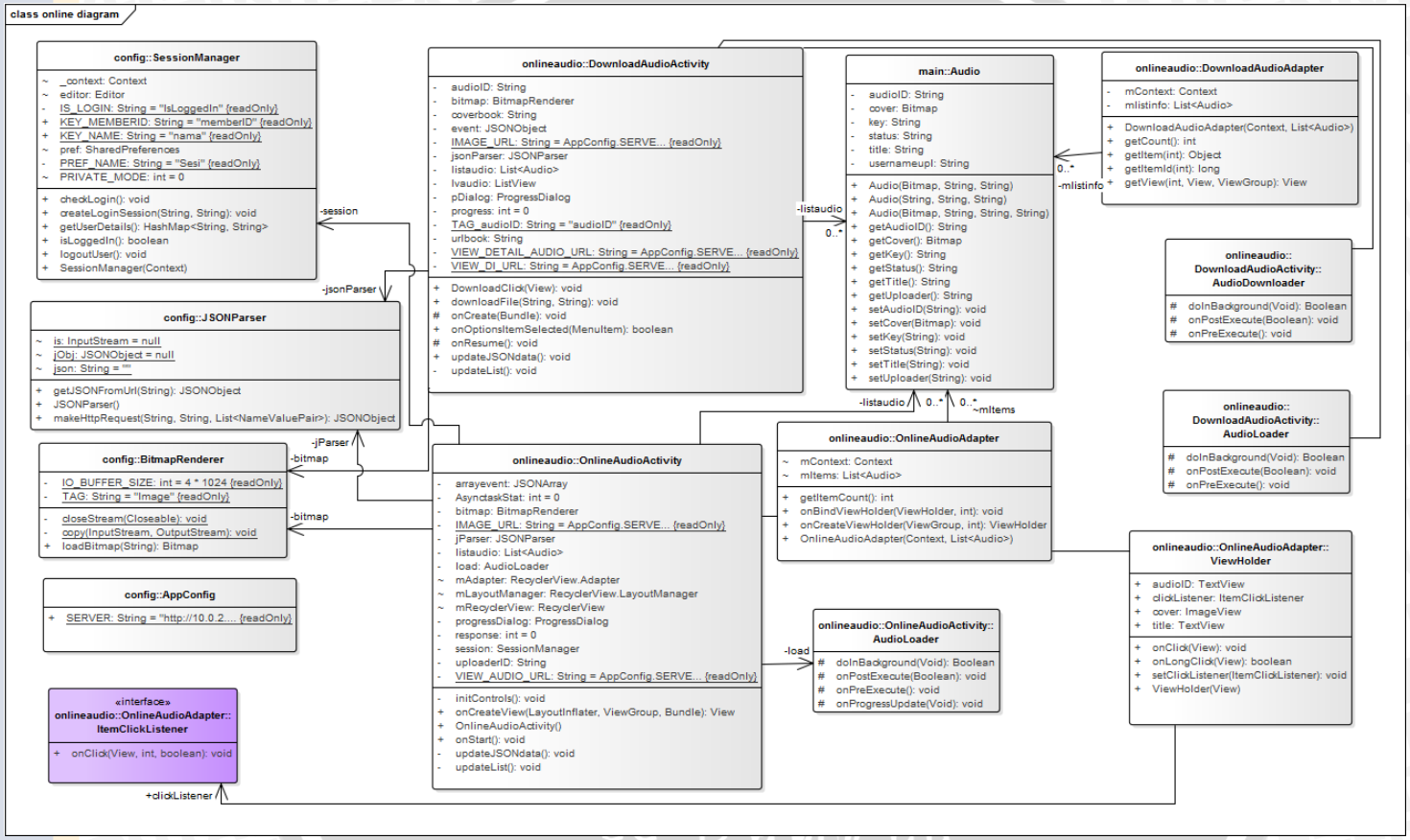
LAMPIRAN A PERANCANGAN KELAS DIAGRAM

A.1 Perancangan Klas Diagram Fungsi Login, Logout dan Main



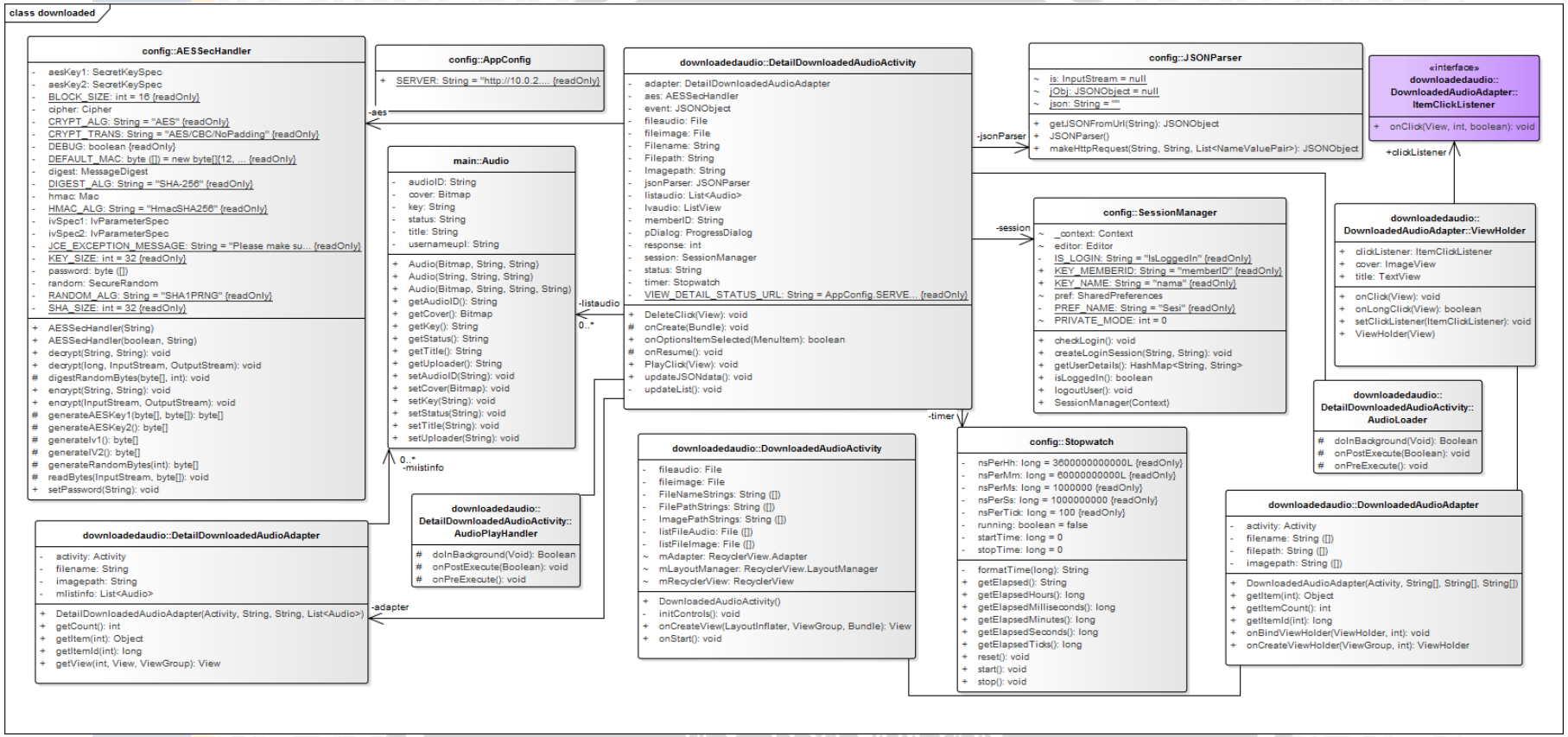
Gambar Lampiran 1 Klas Diagram Fungsi Login, Logout dan Main

A.2 Perancangan Klas Diagram Fungsi Online Audio



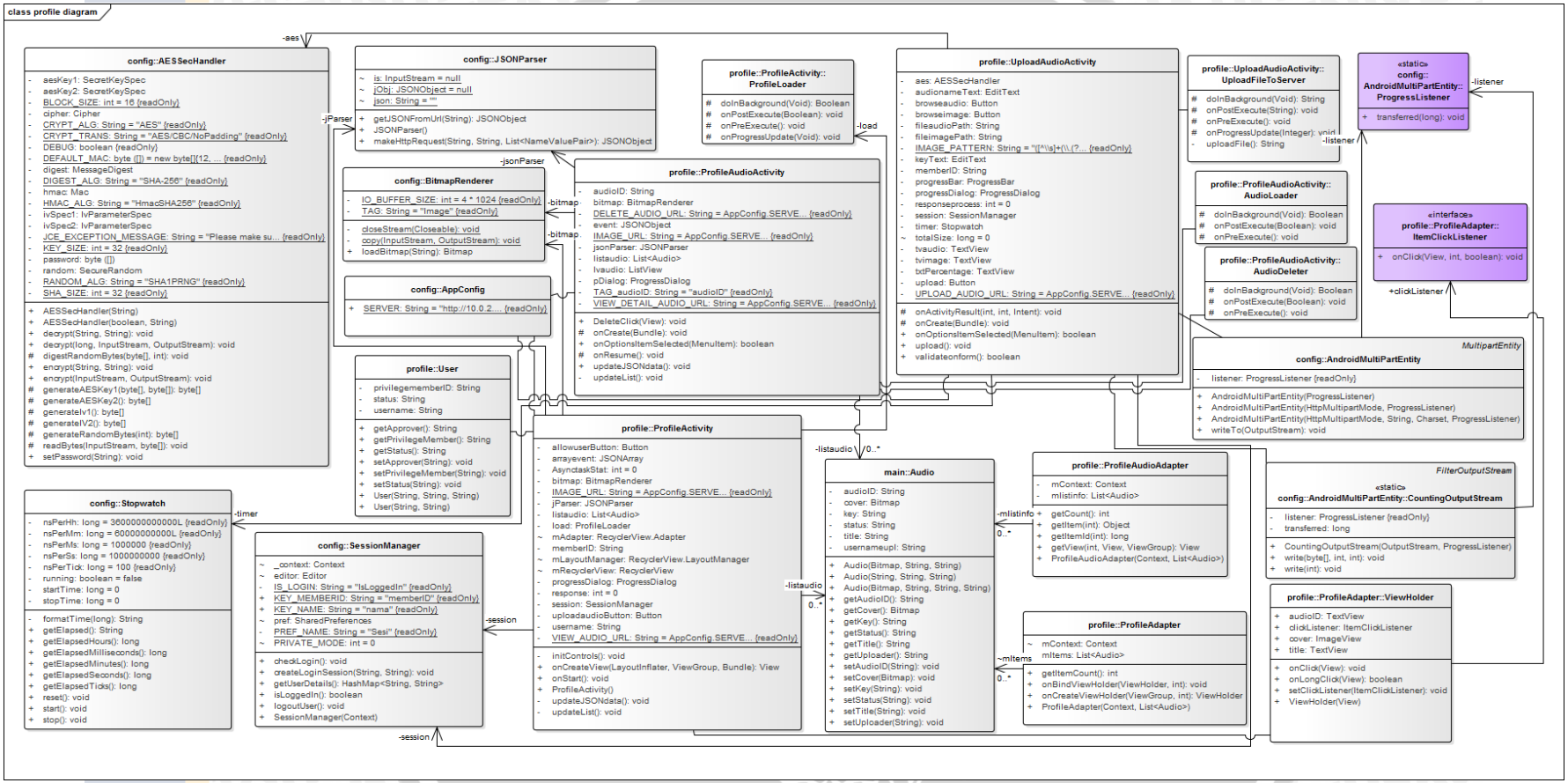
Gambar Lampiran 2 Klas Diagram Fungsi Online Audio

A.3 Perancangan Klas Diagram Fungsi Downloaded Audio



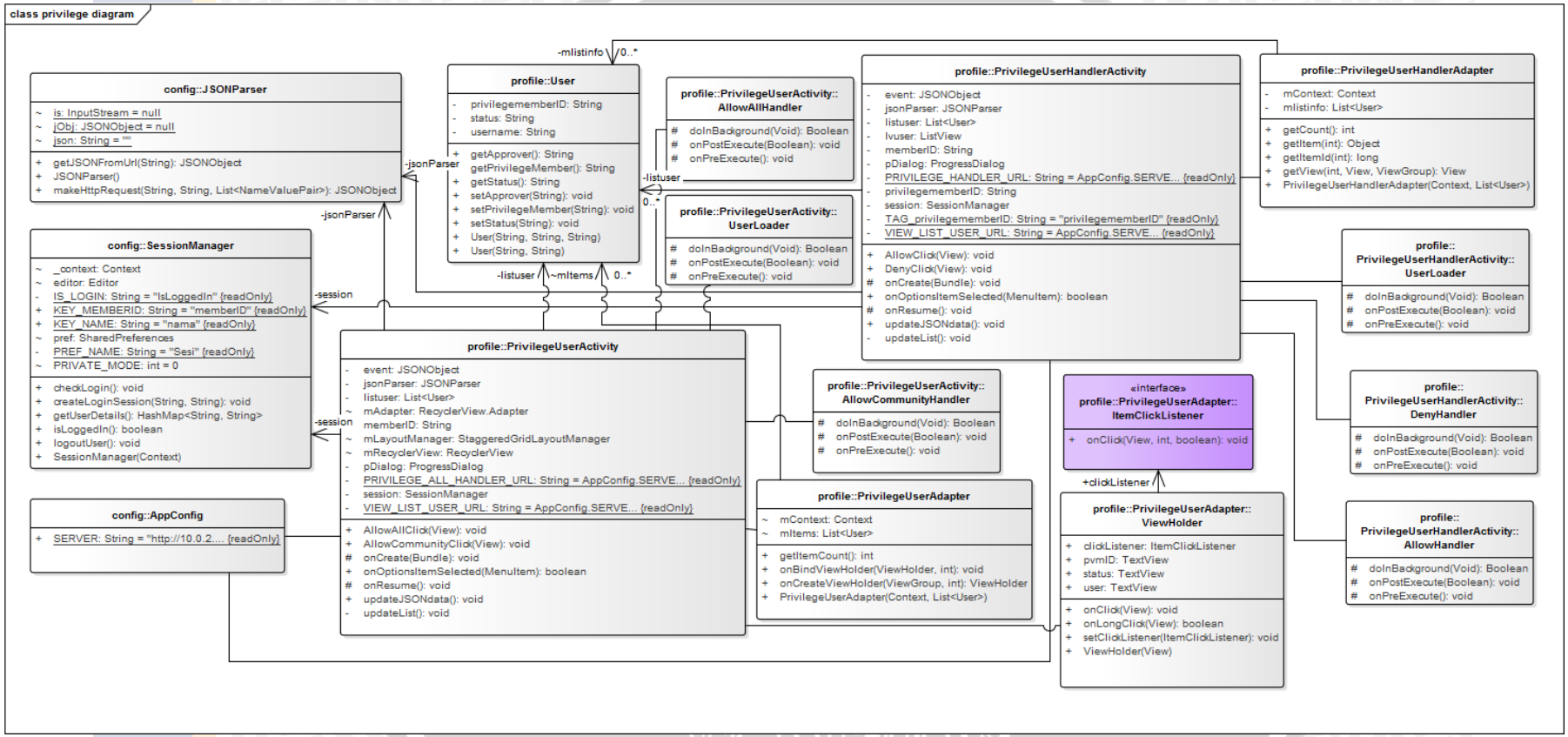
Gambar Lampiran 3 Klas Diagram Fungsi Downloaded Audio

A.4 Perancangan Klas Diagram Fungsi Profile



Gambar Lampiran 4 Klas Diagram Fungsi Profile

A.5 Perancangan Klas Diagram Fungsi Privilege User



Gambar Lampiran 5 Klas Diagram Fungsi Privilege User