RANCANG BANGUN APLIKASI MOBILE INSTANT MESSAGING MENGGUNAKAN PROTOKOL XMPP

SKRIPSI

Oleh :
ACHMAD MUSTOFA
NIM. 07650071

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2014

RANCANG BANGUN APLIKASI MOBILE INSTANT MESSAGING MENGGUNAKAN PROTOKOL XMPP

SKRIPSI

Diajukan Kepada :
Dekan Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
ACHMAD MUSTOFA
NIM. 07650071

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI (UIN)
MAULANA MALIK IBRAHIM MALANG
2013

HALAMAN PERSETUJUAN

RANCANG BANGUN APLIKASI MOBILE INSTANT MESSAGING MENGGUNAKAN PROTOKOL XMPP

SKRIPSI

Oleh:
ACHMAD MUSTOFA
NIM. 07650071

Telah Diperiksa dan Disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Syahiduzzaman, M.Kom	**Yunifa Miftachul Arif, M.T**
NIP. 19700502 200501 1005	**NIP. 19830616 201101 1 004**

Tanggal 23 Januari 2014

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

SURAT PERNYATAAN
ORISINALITAS PENELITIAN

Saya yang bertanda tangan di bawah ini :

Nama : Achmad Mustofa

NIM : 07650071

Fakultas / Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Penelitian : RANCANG BANGUN APLIKASI MOBILE INSTANT MESSAGING MENGGUNAKAN PROTOKOL XMPP

Menyatakan dengan sebenar-benarnya bahwa hasil penelitian saya ini tidak terdapat unsur-unsur penjiplakan karya penelitian atau karya ilmiah yang pernah dilakukan atau dibuat oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata hasil penelitian ini terbukti terdapat unsur-unsur jiplakan, maka saya bersedia untuk mempertanggung jawabkan, serta diproses sesuai peraturan yang berlaku.

Malang, 23 Januari 2014
Penulis

Achmad Mustofa
NIM. 07650071

MOTTO

___.:.___

“Sebaik-baiknya manusai adalah yang memberi manfaat kepada orang lain.”

“Knowledge Belongs to Humanity”

____.:.____

PERSEMBAHAN

Yaa Allah.....

Terima kasih atas nikmat dan rahmat-Mu yang agung ini.
Hari ini hamba bahagia. Sebuah perjalanan panjang telah kau berikan secercah cahaya terang. Meskipun hari esok penuh teka-teki dan tanda tanya yang aku sendiri belum tahu pasti jawabanya. Di tengah malam aku bersujud, kupinta kepada-mu di saat aku kehilangan arah, kumohon petunjuk-mu. Aku sering tersandung, terjatuh, terluka dan terkadang harus kutelan antara keringat dan air mata. Namun aku tak pernah takut, aku takkan pernah menyerah karena aku tak mau kalah, Aku akan terus melangkah berusaha dan berdo'a tanpa mengenal putus asa.

Ku persembahkan karya tulis ini untuk
orang tua tercinta yang senantiasa bersujud dan bermunajat kepada Allah SWT untuk kebaikan dan kesuksesan putra tercintanya, serta senantiasa mendukung, memotivasi dan memberiku inspirasi
untuk terus berjuang. Karya tulis ini kupersembahkan sebagai jawaban atas kepercayaan yang telah berikan oleh kedua orang tuaku serta perwujudan bhaktiku kepada kedua orang tuaku

Spesial Thank's untuk teman-temanku tercinta serta teman-teman seperjuangan jurusan Teknik Informatika angkatan 2007 yang selalu

memberikan suntikan motivasi dan semangat dalam menjalani hidup

untuk tidak pantang menyerah

KATA PENGANTAR

Puji syukur alhamdulillah penulis panjatkan kehadirat Allah SWT atas berkat, rahmat, taufik dan hidayah-Nya, penyusunan skripsi yang berjudul "Rancang Bangun Aplikasi Mobile Instant Messaging Menggunakan Protokol XMPP" dapat diselesaikan dengan baik.

Penulis menyadari bahwa dalam proses penulisan skripsi ini banyak mengalami kendala, namun berkat bantuan, bimbingan, kerjasama dari berbagai pihak dan berkah dari Allah SWT sehingga kendala-kendala yang dihadapi tersebut dapat diatasi. Untuk itu penulis menyampaikan ucapan terima kasih dan penghargaan kepada Bapak Syahiduzzaman, M.Kom selaku pembimbing I dan Bapak Yunifa Miftachul Arif, M.T selaku pembimbing II yang telah dengan sabar, tekun, tulus dan ikhlas meluangkan waktu, tenaga dan pikiran memberikan bimbingan, motivasi, arahan, dan saran-saran yang sangat berharga kepada penulis selama menyusun skripsi.

Selanjutnya ucapan terima kasih penulis sampaikan pula kepada:

1. Dr. Cahyo Crysdian selaku Ketua Jurusan Teknik Informatika Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
2. Prof. DR. Sutiman Bambang Sumitro. SU.DSc, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
3. Teman-teman Jurusan Teknik Informatika angkatan 2007 dan teman-teman kuliah UIN Maulana Malik Ibrahim Malang maupun kampus lainnya.
4. Dan semua pihak yang telah membantu dalam menyelesaikan skripsi ini.

Penulis menyadari bahwa skripsi ini masih terdapat banyak kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun agar skripsi ini dapat lebih baik lagi. Akhir kata penulis berharap kerangka acuan skripsi ini dapat memberikan wawasan dan pengetahuan kepada para pembaca pada umumnya dan pada penulis pada khususnya. Amin Ya Rabbal Alamin.

Wassalamu'alaikum Wr. Wb.

Malang, 18 Juli 2014

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Achmad Mustofa. 2014. **Rancang Bangun Aplikasi Mobile Instant Messaging dengan Menggunakan protokol XMPP.**
Pembimbing : (I) Syahiduzzaman, M.Kom (II) Yunifa Miftachul Arif, M.T

Kata Kunci : Instant messenger, Civitas akademik, Media komunikasi, Real-time

Universitas merupakan salah satu lembaga pendidikan yang terdiri dari banyak elemen seperti mahasiswa, dosen, riset, pengabdian masyarakat, dll. Sebutan civitas akademik juga erat kaitannya dengan elemen universitas yang mana orang-orang di dalamnya menjadi suatu komponen penting yang tidak bisa dipisahkan. Arus pertukaran informasi yang begitu luas dan dalam jumlah yang besar dapat terjadi saat ada interaksi antara orang-orang yang terlibat pada sistem universitas. Interaksi dan komunikasi dosen dengan mahasiswa, mahasiswa dengan petugas administrasi, dosen dengan dosen, mahasiswa dengan mahasiswa, atau bahkan alumni. Walaupun pelaku interaksi merupakan komponen yang heterogen, namun semua masih tergolong dalam satu unit besar yaitu civitas akademik. Untuk memudahkan komunikasi antar civitas akademik, dapat digunakan teknologi media komunikasi yang bisa digunakan kapanpun dan di manapun karena cepatnya aliran informasi.

Penggunaan layanan internet instant messaging, dapat memudahkan proses interaksi antar civitas akademik karena sifatnya yang real-time, dan dapat digunakan bersama-sama dalam waktu yang bersamaan. Urgensi kebutuhan informasi dan komunikasi antar civitas akademika, kecepatan dan kemudahan akses media komunikasi, maka dalam penelitian ini akan dikembangkan sebuah layanan instan messenger untuk civitas akademik Univesitas Maulana Malik Ibrahim Malang pada jaringan komunikasi jabber. Tujuan yang ingin dicapai pada tugas akhir ini adalah membangun sebuah aplikasi instant messaging sebagai media komunikasi yang dapat digunakan secara bersama-sama, mahasiswa, dosen, dan alumni.

ABSTRACT

Achmad Mustofa. 2014. **Minimarket Search Application Using Methods Haversine Formula To Determine The Shortest Distance**.
Promotor : (I) Syahiduzzaman, M.Kom (II) Yunifa Miftachul Arif, M.T

University is an educational society that consist of many elements such as students, teachers, researches, community services, etc. academic community is also closely related with university element which the peoples inside become one inseparable important component. Large information flow exchange occured when there is interaction between the peoples inside university system. Interaction and communication between teachers and students, students and administration officers, teachers and teachers, students and students, or even alumni. Although interaction actor is a heterogent component, but all of them is still classified become a unity as academic community. In purpose to simplify communication between academic community, they are need communication technology which can be used everywhere and everytime because the rapid of flow information.

Internet instant messaging service can be used to help interaction process between academic community because it has real-time service, and also can be used together in one time. The research urgency is to accomplished information needs and communication between academic community, speed and simplicity of communication media, then need to develop a instant messenger service for academic community of Islamic State of University Maulana Malik Ibrahim Malang using Jabber communication network. The objective in this research is to build an instant messaging application as communication media which can be used together for students, teachers, and alumni.

Keywords: *Instant messenger, academic community, Communication media, Real-time*

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi dan komunikasi adalah produk media komunikasi yang dikembangkan dalam upaya memfungsikan komunikasi itu untuk menginformasikan *(to Inform)*, untuk mendidik *(to educate)*, untuk menghibur *(to entertain)* dan untuk mempengaruhi *(to influence)*. Pada gilirannya, derivasi multimedia yang berbasis ICT *(Information and Communication Technology)* telah mengubah dunia menjadi kecil dan seakan tanpa ruang dan waktu. Betapa segalanya dimudahkan dengan berbagai fasilitas komunikasi kontemporer. Tentu jika tidak ada sesuatu hal yang mengubah segalanya, dimasa depan perkembangan media komunikasi yang berbasis ICT atau TIK (teknologi informasi dan komunikasi) akan lebih pesat dan lebih maju menjawab setiap persoalan dalam komunikasi. Sebagai contoh, media informasi yang pada masa konvensional paling trendi disebarluaskan melalui media cetak seperti surat kabar atau buletin, kini sudah berkembang pesat melalui multimedia yang berbasis TIK seperti televisi berbayar atau bahkan melalui situs-situs internet yang bisa diakses kapan saja dan dimana saja jika mau.

Universitas merupakan salah satu lembaga pendidikan yang terdiri dari banyak elemen seperti mahasiswa, dosen, riset, pengabdian masyarakat, dan banyak lagi. Sebutan civitas akademik juga erat kaitannya dengan elemen universitas yang mana orang-orang di dalamnya menjadi suatu komponen penting yang tidak bisa dipisahkan.

Arus pertukaran informasi yang begitu luas dan dalam jumlah yang besar dapat terjadi saat ada interaksi antara orang-orang yang terlibat pada sistem universitas. Interaksi dan komunikasi dosen dengan mahasiswa, mahasiswa dengan petugas administrasi, dosen dengan dosen, mahasiswa dengan mahasiswa, atau bahkan alumni. Walaupun pelaku interaksi merupakan komponen yang heterogen, namun semua masih tergolong dalam satu unit besar yaitu civitas akademik. Untuk memudahkan komunikasi antar civitas akademik, dapat digunakan teknologi media komunikasi yang bisa digunakan kapanpun dan di manapun karena cepatnya aliran informasi. Penggunaan layanan internet seperti jejaring sosial, instant messaging, dapat memudahkan proses interaksi antar civitas akademik karena sifatnya yang real-time, dan dapat digunakan bersama-sama dalam waktu yang bersamaan.

Kebutuhan akan cepatnya mendapatkan informasi dari hasil komunikasi ini lah yang kemudian akan menjadi salah satu masalah mengapa perlu ada media komunikasi yang tepat untuk civitas akademik. Dalam penelitian oleh Junco dan Mastrodicasa terhadap 7704 mahasiswa universitas, ditemukan fakta bahwa 76% mahasiswa menggunakan instant messaging (sumber: Analisis Kepuasan Mahasiswa Terhadap Performa Nirkabel dan Pola Penggunaan Internet di Universitas oleh Pujianto Yugopuspito). Besarnya prosentase penggunaan Instant Messaging disini menunjukkan bahwa teknologi ini dapat diterima di lingkungan akademik.

Instant messager berguna untuk menunjang kehidupan akademik mahasiswa secara tidak langsung. Seperti, seorang profesor dan asisten pengajar menggunakan instant messaging untuk meningkatkan pembelajaran siswa di kelas universitas. Profesor Bob Burk dari universitas Carleton telah menggunakan MSN Messenger selama dua tahun terakhir untuk berkomunikasi dengan siswa-siswanya, ia telah membuat dirinya tersedia beberapa

malam antara 9 dan 11 PM karena pada jam ini adalah ketika siswa paling mungkin melakukan tugas sekolah (sumber: *Instant Messaging and Presence Technologies for College Campuses*, oleh Samir Chatterjee, Claremont Graduate University).

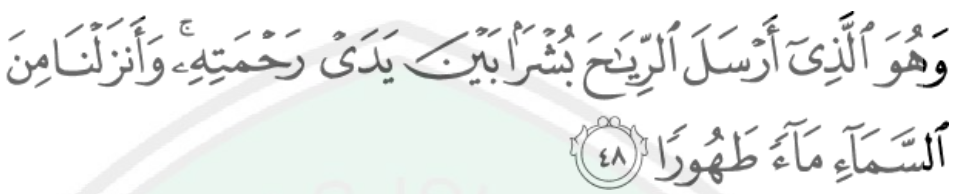
Contoh lain lagi adalah, sejumlah besar perpustakaan universitas telah mengembangkan referensi mereka sendiri untuk dapat diakses pada aplikasi instant messaging. Aplikasi instant messaging secara eksklusif menawarkan layanan referensi ini. Mereka memilih instant messaging karena mereka ingin bantuan instan, tidak ingin meninggalkan kantor/rumah, tidak ingin menjauh dari komputer mereka, dan menemukan itu mudah digunakan.

Instant messaging juga memiliki dampak kuat pada kehidupan sosial. Aplikasi ini dapat membantu mahasiswa mengembangkan dan memelihara hubungan sosial. Hubungan sosial akan memberikan mahasiswa pada *relationship, social support*, dan *friendship*, yang semuanya penting bagi pengembangan hubungan sosial mereka.

Dalam konteks dakwah, instant messaging yang digunakan pun tidak sekadar memfungsikan sebagian fungsi komunikasi seperti menginformasikan, mendidik dan mempengaruhi saja, tetapi juga mengoptimalisasikan upaya mengajak atau menyeru (*to invite/ to propagate*). Sehingga multimedia atau dalam konsepsi ilmu dakwah disebut wasilah, mengadopsi segala produk media komunikasi terutama multimedia berbasis teknologi informasi dan komunikasi sebagai media dakwah.

Namun ada hal yang mendasar yang perlu dicatat bahwa segala bentuk multimedia tersebut tidak mungkin berkembang dan dikembangkan tanpa ada sesuatu yang menjadi modal untuk berkomunikasi itu secara fundamental. Tentu sesuatu itu dalam pandangan

Islam tidak terjadi menjadi ada dengan sendirinya tetapi diadakan oleh yang maha mengadakan yaitu Allah SWT. Firman Allah dalam Q.S. Al-furqon ayat 48 :



"Dia-lah yang meniupkan angin (sebagai) pembawa kabar gembira dekat sebelum kedatangan Rahmat-Nya (hujan) dan kami turunkan dari langit air yang amat bersih".

Ayat-ayat dalam rangkaian surat Al-Furqon Asy-Syuaro dan An-naml adalah ayat-ayat dakwah para nabi yang didalamnya terdapat pula hal-hal yang berkenaan dengan media dakwah. Ini menjadi landasan teologis yang sangat ilmiah untuk mengklaim, menggugat atau meluruskan bahwa seharusnya media komunikasi yang berkembang saat ini terjadi karena keberadaan angin atau sebutlah gelombang elektromagnetik yang dapat menghantarkan resonansi suara dari suatu tempat ketempat yang lain. Al-qur'an menyatakan bahwa para nabi bertugas menyampaikan berita gembira dan peringatan kepada manusia. Untuk menyebarluaskan pesan ilahiyah itu, Allah menciptakan angin sebagai fasilitas atau media dakwah sebagaimana Nabi Sulaiman yang dapat menangkap resonansi berbagai suara binatang dan ketundukan angin kepadanya dengan ijin Allah. Di dalam Q.S. An-naml ayat 15 sampai dengan 44, Allah membelajarkan umat Islam dengan kisah sulaiman yang mandakwahkan ajaran tauhid mulai dengan menggunakan media lisan tulisan (surat) sampai media semacam 3G (dimasa sekarang) atau ICT dan bahkan belum sepadan melampaui itu. Lebih lagi dalam surat An-Naml ayat ke 40, Allah berfirman :

قَالَ ٱلَّذِى عِندَهُۥ عِلۡمٞ مِّنَ ٱلۡكِتَٰبِ أَنَا۠ ءَاتِيكَ بِهِۦ قَبۡلَ أَن يَرۡتَدَّ إِلَيۡكَ طَرۡفُكَۚ
فَلَمَّا رَءَاهُ مُسۡتَقِرًّا عِندَهُۥ قَالَ هَٰذَا مِن فَضۡلِ رَبِّي لِيَبۡلُوَنِيٓ ءَأَشۡكُرُ أَمۡ
أَكۡفُرُۖ وَمَن شَكَرَ فَإِنَّمَا يَشۡكُرُ لِنَفۡسِهِۦۖ وَمَن كَفَرَ فَإِنَّ رَبِّي غَنِيٌّ كَرِيمٞ ٤٠

"Berkatalah seorang yang mempunyai ilmu dari al-kitab, 'Aku akan membawa singgasana itu kepadamu sebelum matamu berkedip' maka tatkala Sulaiman melihat singgasana itu terletak dihadapannya, ia pun berkata: ' Ini termasuk karunia tuhanku untuk mencoba aku bersyukur atau mengingkari (akan nikmat-Nya). Dan barang siapa bersyukur maka sesungguhnya dia bersyukur untuk (kebaikan) dirinya sendiri dan barang siapa yang ingkar, maka sesungguhnya Tuhan-Ku maha kaya lagi maha Mulia".

Mengingat urgensi kebutuhan informasi dan komunikasi antar civitas akademika diatas, kecepatan dan kemudahan akses media komunikasi, serta potensi instant messaging dalam komunikasi *mobile*, maka dalam penelitian ini akan dikembangkan sebuah layanan instan messenger untuk civitas akademik yang berjalan pada platform Android. Tujuan yang ingin dicapai pada tugas akhir ini adalah membangun sebuah aplikasi instant messaging sebagai media komunikasi yang dapat digunakan secara bersama-sama, mahasiswa, dosen, dan alumni.

1.2 Rumusan Masalah

Bagaimana membangun sebuah aplikasi instant messaging berbasiskan platform android dengan memanfaatkan protokol komunikasi jabber yang diimplementasikan pada civitas kampus UIN Maulana Malik Ibrahim Malang.

1.3 Batasan Masalah

1. Aplikasi instant messenger berjalan pada platform android 2.2 sampai yang terbaru.
2. Sistem aplikasi messenger mendukung otentikasi dengan sistem manajemen user yang dibangun diluar sistem messenger.
3. Model dan desain aplikasi *groupchat* sesuai dengan XEP-0045 tentang *Multi-User Chat* (MUC).
4. Penggunaan dan pemanfaatan aplikasi untuk civitas akademik kampus UIN Maliki Malang.

1.4 Tujuan

Tujuan dari penelitian ini adalah membangun sebuah aplikasi instant messaging berbasiskan platform android dengan memanfaatkan protokol komunikasi jabber yang diimplementasikan pada civitas kampus UIN Maulana Malik Ibrahim Malang.

1.5 Manfaat

1. Komunikasi *near real-time*
2. Pengumuman dapat dipublikasikan secara efektif dan efisien
3. Memudahkan proses komunikasi civitas akademik kampus sehingga menunjang kegiatan akademik secara langsung.
4. Menciptakan proses komunikasi sebagai aktifitas sosial antara civitas akademik kampus.

5. Mempermudah koordinasi antar individu, antar karyawan ataupun antar anggota organisasi di kampus.

1.6 Sistematika Pembahasan

Penulisan skripsi ini tersusun dalam 5 (lima) bab dengan sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN

Pendahuluan, membahas tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penyusunan tugas akhir, metodologi, dan sistematika penyusunan skripsi.

BAB II DASAR TEORI

Kajian pustaka berisikan beberapa teori yang mendasari dalam penyusunan skripsi ini. Adapun yang dibahas dalam bab ini adalah dasar teori yang berkaitan dengan pembahasan tentang standar protokol *jabber*.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan mengenai analisis dan perancangan sistem aplikasi berbasis android.

BAB IV IMPLEMENTASI HASIL DAN PEMBAHASAN

Bab ini membahas tentang laporan skripsi berupa tahapan implementasi dan uji coba dari perancangan sistem serta analisis hasil yaitu implementasi tabel dan pembuatan program aplikasi.

BAB V PENUTUP

Bab ini berisi kesimpulan dan saran tentang hasil perancangan dan implementasi program.

BAB II

LANDASAN TEORI

2.1 Penggunaan Teknologi Informasi menurut Islam

Teknologi adalah salah satu hasil dari peradaban manusia. Itu karena manusia dianugerahi akal dan perasaan untuk selalu berkembang dan untuk meningkatkan mutu kehidupannya. Hal ini dibuktikan dengan adanya kehidupan yang terus meningkat tarafnya dari masa ke masa. Bukan hanya dalam satu bidang tertentu tapi dapat ditinjau dari berbagai aspek kehidupan. Sehingga mutu setiap individu meningkat dari masa ke masa. Segala sesuatu yang berhasil diciptakan manusia adalah untuk memudahkan proses berjalannya suatu kehidupan. Bukan hanya untuk jangka sementara tapi lebih pada orientasi yang berjangka waktu lama. Dan semuanya itu adalah hasil pemikiran manusia yang ingin menghasilkan kemudahan dalam proses hidupnya. Salah satu bukti yang dihasilkan adalah teknologi informatika. Teknologi informatika adalah salah satu penunjang kehidupan di era modern ini. Ada banyak manfaat yang dapat diambil dari penggunaannya. Seperti mencari lowongan pekerjaan, menyambung tali silaturahmi, serta mempermudah pekerjaan kelompok maupun individu. Sungguh suatu kemajuan yang signifikan bagi kehidupan manusia.

Namun sayangnya ada saja oknum-oknum yang menyalahgunakan teknologi untuk kepentingan dirinya yang kadang merugikan orang lain, seperti membajak data-data penting suatu serikat atau perkumpulan. Atau membuat orang malas untuk beranjak dari tempatnya untuk hanya sekedar menikmati dunia maya, padahal kadang belum tentu menghasilkan suatu perubahan bagi dirinya. Sudah tentu hal tersebut hanya menghabiskan waktu saja. Bahkan

ada juga yang sengaja memasukkan file-file yang tidak mendidik sehingga mengikis keimanan serta menjatuhkan adab anak bangsa. Sungguh pekerjaan yang sangat merugikan. Padahal Allah telah memerintahkan kita sebagai khalifah di bumi untuk merawat bumi dan memcari ilmu pengetahuan sebanyak-banyaknya. Agar kita menjadi kamalul insan yang memahami makna penciptaan kita di dunia ini. Bukan hanya sekedar untuk berfoya-foya saja dalam hidup ini. Sebagaimana dalam firman Allah swt yang berbunyi:

يٰمَعْشَرَ الْجِنِّ وَالْاِنْسِ اِنِ اسْتَطَعْتُمْ اَنْ تَنْفُذُوْا مِنْ اَقْطَارِ السَّمٰوٰتِ
وَالْاَرْضِ فَانْفُذُوْا ۗ لَا تَنْفُذُوْنَ اِلَّا بِسُلْطٰنٍ ۚ ﴿٣٣﴾

"Hai jama'ah jin dan manusia, jika kamu sanggup menembus (melintasi) penjuru langit dan bumi, Maka lintasilah, kamu tidak dapat menembusnya kecuali dengan kekuatan." (Ar-Rahman:33)

Kekuatan yang dimaksud di atas adalah dengan pengetahuan yang diberikan Allah kepada manusia untuk membuka jalan menuju kehidupan yang lebih baik. Semuanya itu adalah nikmat dari Allah kepada manusia agar menjadi khalifah yang baik di bumi, dalam artian menjadi wakil Allah untuk melindungi alam semesta dan seluruh isinya. Teknologi dalam Islam bukanlah tidak berdasarkan agama, karena ia tumbuh dan berkembang bersama Islam. Teknologi Islam mempunyai maksud untuk meberikan manusia pemahaman yang lebih mendalam terhadap kandungan dalam ayat-ayat Allah beserta rahasia yang tersirat di dalamnya, baik ayat qauliah maupun ayat kauniah melalui pemanfaatan daya fikir manusia secara totalitas. Agar manusia memanfaatkan potensi alam dan lingkungan dengan sebaik-baiknya. Teknologi dalam Islam diciptakan untuk membawa manfaat dan kemaslahatan

bersama bagi manusia yang sesuai dengan misi Islam yaitu *rahmatan lil 'alamin*. Yang selalu berusaha merujuk pada Al-Qur'an dan Sunnah rasul serta menghantarkan manusia kepada pemahaman, keyakinan yang lebih sempurna kepada kebenaran informasi , yang pada akhirnya dapat meningkatkan keimanan, ketakwaan kepada Allah, mengakui keagungan, kebesaran, dan ke Maha Kuasaan-Nya.

2.2 Instant Messenger

Instant Messenger atau dalam bahasa indonesia disebut dengan pesan instan merupakan sebuah teknologi internet yang memungkinkan para penggunanya untuk bisa mengirim pesan singkat secara langsung dalam sebuah jaringan internet. Para pengguna pesan singkat ini dapat langsung berkomunikasi pada waktu yang bersamaan atau *real-time* dengan menggunakan teks kepada pengguna lainnya yang sama-sama terhubung dalam sebuah jaringan.

Konsep awal teknologi instant messenger ini muncul pada awal-awal pengembangan sistem operasi UNIX dan jaringan internet. Pada penggunaannya, para pengguna yang sudah melakukan log ini dapat mengirimkan sebuah perintah berupa *<code>talk</code>*, *write*, *dan finger* untuk melihat siapa saja yang sudah masuk log dan akhirnya dapat mengirimkan pesan singkat kepada mereka-mereka yang sudah terhubung.

Istilah pesan singkat atau instant messenger saat ini umumnya merujuk pada sebuah teknologi yang dipopulerkan oleh *American Online* atau yang lebih dikenal dengan AOL. Kemudian istilah pesan singkat juga diikuti oleh *Yahoo!* dengan mengenalkan sebuah

jaringan yang disebut dengan *Yahoo! Messenger, Google,* dan *Microsoft* yang dikenal dengan nama *Windows Live Messenger*. Hal ini terus berlanjut hingga banyak perusahaan-perusahaan yang mengenalkan pesan singkat tersebut.

Pesan singkat tidak hanya berguna untuk berkomunikasi, dalam kegunaannya, instant messenger ini juga memberikan fasilitas antarmuka dalam berkomunikasi. Berikut adalah fungsi antar muka dari pesan singkat:

1. *Instans messages* : untuk mengirim pesan kepada teman yang sedang online atau sedang terhubung dengan jaringan internet secara bersamaan.
2. *Chat* : untuk menciptakan chat room atau ruangan mengobrol seperti halnya di dunia nyata dengan teman atau rekan kerja sehingga pembicaraan dapat berlangsung dengan baik.
3. Web *links* : untuk berbagi link mengenai website favorit atau website yang baik untuk dikunjungi.
4. Video : untuk mengirim serta menyaksikan video dan melakukan chatting secara face to face dengan teman. Sehingga cahtting pun tidak hanya dengan teks, akan tetapi dengan video call atau lainnya pun bisa berjalan.
5. *Images* : untuk melihat gambar yang ada teman Anda miliki.
6. *Files* : untuk berbagi file dengan mengirimkan file tersebut secara langsung kepada rekan atau teman.
7. *Talk* : berfungsi untuk pengguna agar bisa benar-benar berbicara dengan teman layaknya menggunakan telepon.
8. *Mobile capabilities* : untuk mengirim instan messenger melalui handphone.

2.3 Protokol XMPP/Jabber

Extensible Messaging and Presence Protocol (XMPP) adalah sebuah standar komunikasi *real-time* berbasis teks. Awalnya protokol ini lebih dikenal dengan nama Jabber, karena XMPP merupakan produk Jabber pada tahun 1999, kemudian diformulasikan oleh *XMPP Standard Foundation* (XSF) menjadi sebuah standar pada tahun 2002 (XMPP 1.0). Protokol ini sekarang biasa digunakan untuk beberapa hal berikut: *instant messaging*, *presence*, *multi-party chat, voice and video calls, collaboration, lightweight middleware*, *content syndication,* dan *generalized routing of XML data*.

Jaringan XMPP juga termasuk desain dan implementasi protokol TCP/IP dengan protokol XMPP ada pada layer *Application*. Pada dasarnya model TCP/IP adalah versi pemadatan model OSI yang terdiri dari empat layer *(Application, Transport, Internet, Network Access)*. Pada layer *application* ini, protokol XMPP mengintegrasikan berbagai macam aktifitas dan tugas, mendefinisikan protokol untuk komunikasi aplikasi node-node, mendefinisikan format pengalamatan antar host dan juga mengontrol spesifikasi user. Lapisan *internet* menjaga pengalamatan host dengan memberikan alamat IP dan menangani routing dari paket yang melalui beberapa jaringan. Lapisan *transport* mendefinisikan protokol untuk mengatur level service transmisi, menciptakan komunikasi *end-to-end*, menangani paket sequencing dan menjaga integritas data. Terakhir lapisan *network access* memonitor pertukaran data antara host dengan jaringan.

Dari awal pengembangan, komunitas developer Jabber mencoba untuk membuat standart IM dan menyarankan *interoperability* antar sistem IM. Usaha kooperatif ini sangat

kontras dengan perilaku dari provider IM lainnya yang menjaga agar sistem mereka tertutup dan terisolasi dari jaringan IM lain.

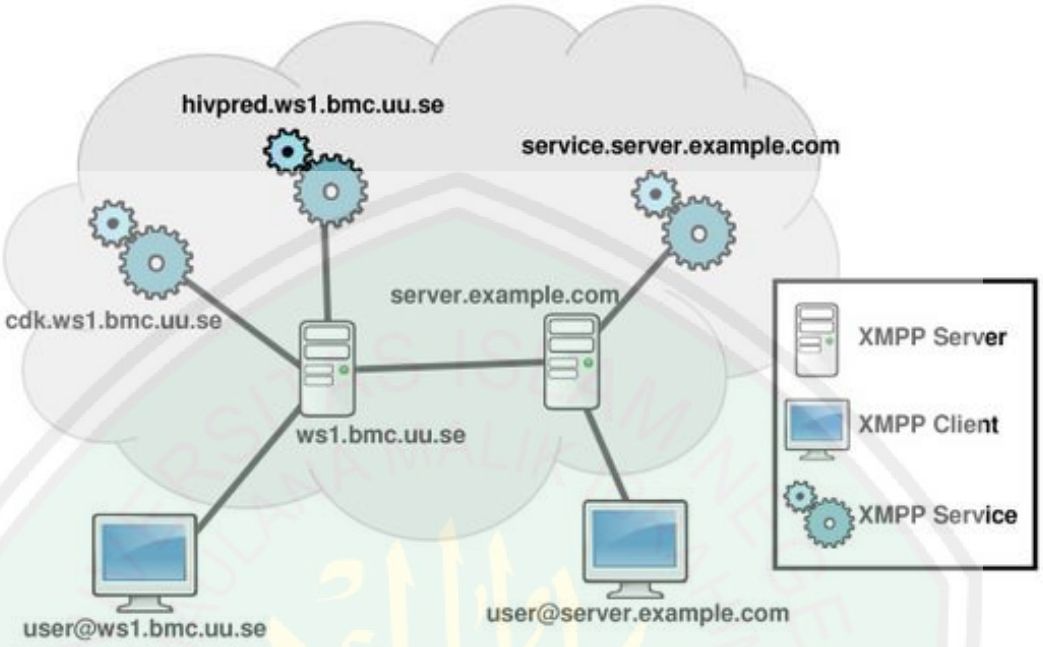
Sebagai usaha menjadikan Jabber sebagai protokol standar, pada bulan juni 2000, komunitas Jabber mempublikasikan protokol tersebut sebagai *Request for Comments* (RFC) kepada *Internet Engeneering Task Force* (IETF) sebagai bagian dari standar IMPP (Instant Messaging and Presence). tetapi IMPP ini tidak berjalan sukses. Pada bulan Mei 2001, Jabber Community dan Jabber Inc. membuat Jabber Software Foundation. Jabber Software Foundation adalah organisasi serupa dengan *Apache Foundation* yang keberadaannya adalah untuk menunjukkan dedikasinya terhadap dunia open source dan *interoperability* antar sistem IM.

Pada tahun 2002, *Internet Engineering Steering Group* (IESG) menyetujui formasi *Extensible Messaging and Presence Protocol Working Group* (XMPP) dengan *Internet Engineering Task Force* (IETF). Ruang lingkup *working group* adalah untuk mengeksplorasi dan dimana protokol tersebut digunakan, memodifikasi protokol yang sudah ada agar dapat memenuhi RFC 2799 seperti persyaratan yang ditentukan dalam spesifikasi *Common Presence and Instant Messaging* (CPIM). Fokus utama working group adalah membuat XML stream termasuk stream pada level security dan otentikasi, elemen data dan namespace yang dibutuhkan untuk mencapai dasar IM dan Presence. XMPP working group menerbitkan *XMPP Core Internet-Draft* sebagai dokumen yang menggambarkan fitur-fitur utama *Extensible Messaging and Presence Protocol*.

2.3.1 Desentralisasi Arsitektur

Jabber menggunakan arsitektur *client-server*. *Client* jabber dapat berkomunikasi dengan server jabber pada domain jabber mereka. Domain jabber memiliki keuntungan yaitu kemampuannya dalam memisahkan zona komunikasi, yang ditangani oleh server jabber yang berbeda, tidak seperti kebanyakan sistem IM lainnya yang menggunakan satu server terpusat untuk seluruh zona komunikasi. Pada jabber pesan dikirim oleh *client* ke server pengirim kemudian diteruskan ke server penerima baru kemudian disampaikan ke *client* penerima.

Jaringan terdesentralisasi dalam hal ini bagaimana sebuah server jabber dapat berkomunikasi dengan server jabber lainnya dan dapat diakses melalui internet. Masing-masing user terhubung pada home server, yang menerima informasi untuk mereka, selanjutnya server akan mentransfer data untuk kepemilikan user. Maka suatu domain dapat jalan pada server jabber. Masing-masing fungsi server bebas terhadap yang lainnya, dan dimaintain sendiri di dalam daftar user. User khusus diasosiasikan dengan server yang khusus pula, dan alamat jabber memiliki bentuk yang sama dengan alamat email. Jaringan yang terdistribusi ini menghasilkan sesuatu yang fleksibel, jaringan yang mampu terkontrol pada server yang memiliki skala yang lebih tinggi dibandingkan monolitik, tetapi dengan syarat bahwa layanan yang terpusat hanya dapat jalan pada vendor IM yang resmi.

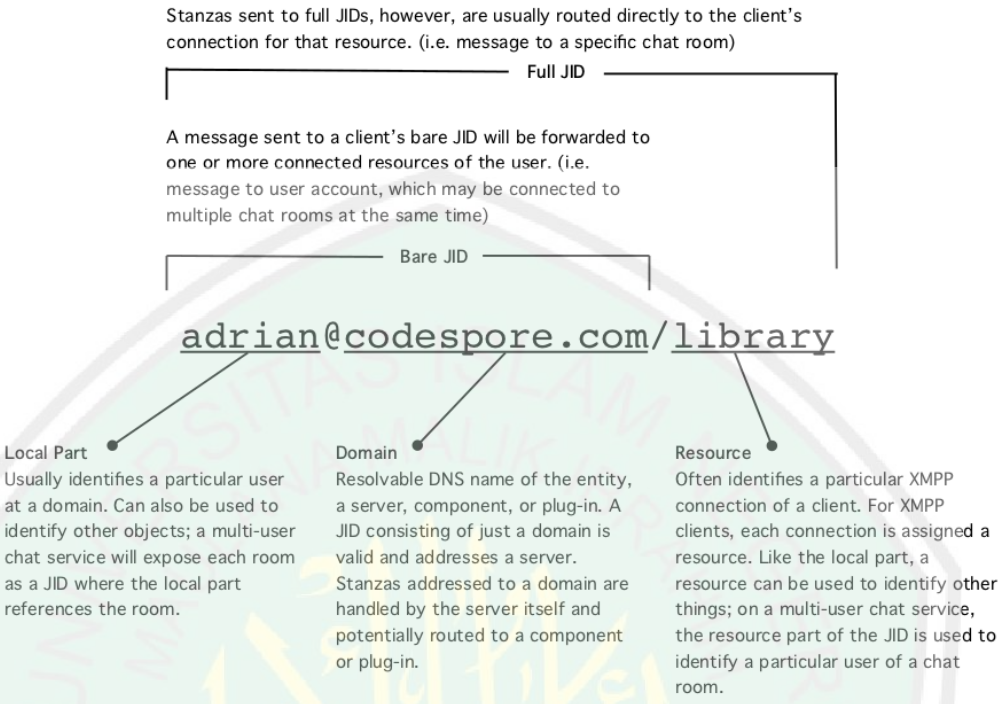


Gambar 2.1. Jaringan Jabber

sumber: http://wiki.xmpp.org

2.3.2 JID

Ada beberapa perbedaan entitas pada Jabber untuk dapat berkomunikasi dengan yang lainnya. Entitas ini dapat direpresentasikan berupa transport, groupchat room atau single jabber user. Tiap entitas pada jaringan jabber ini memiliki sebuah ID unik yaitu Jabber ID yang kemudian disingkat menjadi JID. JID telah digunakan secara internal dan eksternal untuk menyatakan kepemilikan atau routing informasi. JID ini dibentuk menjadi tiga bagian: *local part*, *domain* dan *resource*. Contoh JID adalah *mustofa@mimicreative.net/kantor*, dimana mustofa adalah *local part*, mimicreative.net merupakan *domain* dimana saya terhubung dan kantor adalah *resource* session dimana user sedang melakukan komunikasi.



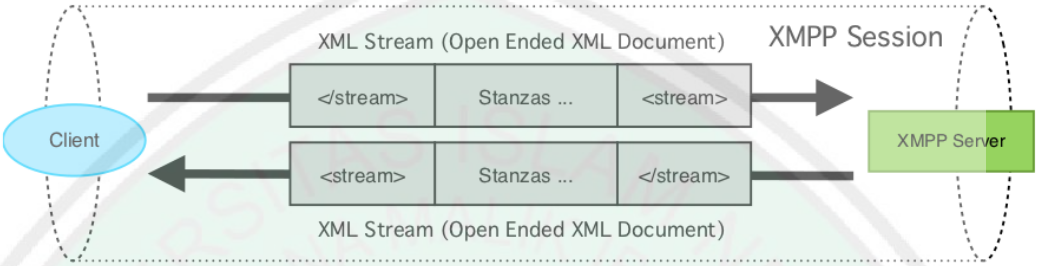
Gambar 2.2 Penulisan JID

sumber: http://wiki.xmpp.org

2.3.3 Core Protokol

Core protokol jabber didefinisikan pada RFC 3920. Pada dokumen ini, terdapat dua pokok bahasan yang dijelaskan, yaitu XML Stream dan XML Stanza. Seperti yang diketahui semua paket data yang ditransmisikan pada jaringan ini memiliki format XML. XML Stream yang hampir memenuhi isi dokumen mengenai bahasan (1) penggunaan protokol keamanan TLS (*Transport Layer Security*) pada jaringan jabber, (2) otentikasi menggunakan metode SASL (*Simple Authentication and Security Layer*) dan juga (3) mekanisme untuk *binding* sebuah resource untuk client koneksi yang terbentuk (*Resource Binding*). Setelah itu bahasan pada dokumen RFC 3920 akan beralih pada XML Stanza. Stanza ini merupakan salah satu dari child elemen XML Stream, seperti yang ditunjukkan pada gambar bawah.

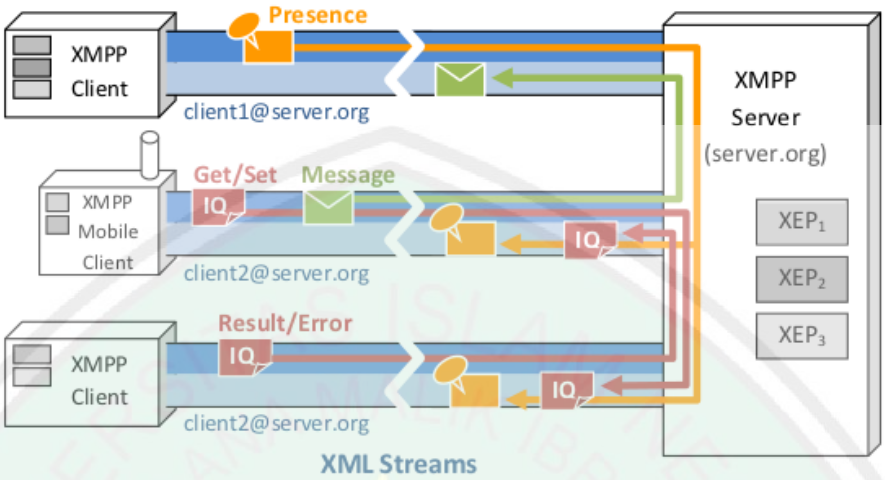
Pada XML Stanza terdapat tiga jenis aliran XML yang ditransmisikan, antara lain *message, presence,* dan *info query* atau lebih dikenal dengan IQ.



Gambar 2.3 XML Stream

sumber: http://wiki.xmpp.org

Jenis pertama (1) *message*, merupakan general paket jabber berisi informasi yang dikirim dari satu entitas ke entitas lainnya. Pengiriman paket ini bersifat *fire and forget*, artinya entitas pengirim tidak akan mendapatkan *result* laporan dari paket *message* yang terkirim. Selain itu juga *message* ini dikirimkan dari dan ke *one-to-one* atau *one-to-many* entitas. Jenis yang kedua adalah (2) *presence*, dikirimkan dengan tujuan *availability* kehadiran entitas yang terhubung dalam jaringan. Entitas dapat mengetahui status online atau *offline* dari setiap entitas lain karena adanya aliran data presence ini. Tidak seperti message, presence dikirimkan ke semua entitas (*broadcast*) yang sudah *subscribe* ke entitas tersebut. Terakhir, jenis yang ketiga adalah (3) *IQ*, digunakan untuk mekanisme *request-response* antar entitas dalam jaringan jabber. Mirip dengan metode GET dan POST pada protokol HTTP. Terdapat sebuah entitas yang mengirimkan *request* ke entitas lain, dan akan menerima *response* balasan dari entitas tersebut.

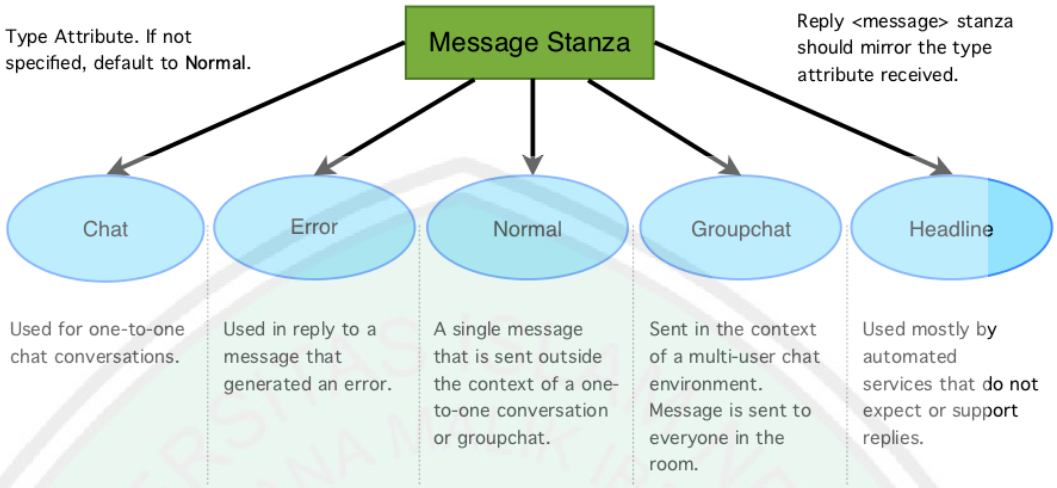


Gambar 2.4 XMPP Server

sumber: http://wiki.xmpp.org

2.3.3.1. Message

Jabber menggunakan stanza *message* untuk mengirim pesan. Pesan dapat dikirm antara jabber *client* dengan jabber server atau antara jabber server dengan server jabber yang lain. Stanza message sangat sederhana, Entitas *sender* mengirim stanza message ke entitas *recepient*. Secara *default* tidak ada *acknowledge* ketika *recepient* menerima stanza message. Jika pesan dikirim dan *recepient* dalam keadaan *offline* maka server berkewajiban menyimpan pesan tersebut dan mengirimkannya ketika *recepient* sudah dalam keadaan *online*/*available*. Proses seperti ini mengacu pada proses *store and foward*. Pada dasarnya format untuk stanza message adalah sebagai berikut :



Gambar 2.5 Message Stanza (Peter Saint-Andre, 2009)

```
<message id='' from='' to='' type='error|chat|groupchat'>
      <body></body>
</message>
```

- *to="" from=""* Mengidentifikasikan *sender* dan *recepient*. Format alamat jabber diatur dalam spesifikasi jabber. Atribut ini diperlukan untuk semua pesan.
- *id=""* Digunakan identifier yang unik pada pesan. *Client* dapat menggunakan id untuk mengidentifikasikan pesan jika pesan mengalami error. Atribut ini bersifat opsional.
- *type="error"* Mengindikasikan bahwa pesan adalah error message. Error ditunjukkan didalam sub elemen <error></error> di dalam elemen message.
- *type="chat"* Mengindikasikan bahwa pesan ditampilkan dalam sebuat *line-by-line* chat interface (*one-to-one* chat).

- *type="groupchat"* Mengindikasikan bahwa pesan ditampilkan dalam room chat interface.

Stanza message mempunyai dua set subelemen *(child element)* didalamnya. Pertama adalah *content* atau *payload* paket (*body* elemen dan *error* elemen) dan kedua adalah subelemen yang merupakan informasi metadata dari paket message tersebut (*x* elemen).

```
<body><body>
```

Subelemen ini membungkus isi pesan *(payload)* yang akan dikirimkan. Subelemen ini hanya diperbolehkan ada satu kali pada setiap paket stanza message dan juga harus berupa *plain-text*.

```
<x xlmns="jabber:x:"></x>
```

Subelemen ini digunakan untuk mengirim perintah antar *client* atau sebagai mekanisme tambahan. Setiap kali elemen ini digunakan, namespaces (xmlns) harus didefinisikan. Sebuah pesan dapat memiliki banyak elemen <*x*> ini. Sebagai contoh namespaces untuk *out-of-bond extension* dapat digunakan untuk mengirim file antar aplikasi.

```
<x xmlns="jabber:x:oob"></x>
<error type="nnn"></error>
```

Subelemen ini disertakan ketika atribut *type* dari pesan di set *error*. *Error* yang sebenarnya didefinisikan oleh atribut dengan *type="nnn"* yang menunjukkan jenis dari *error* tersebut.

- 302 – *redirect*
- 400 – *Bad Request*
- 401 – *Unauthorized*
- 402 – *Payment Reuired*
- 407 – *Registration Required*
- 408 – *Request Timeout*
- 409 – *Conflict*
- 500 – *Internal Sevrer Error*

Isi subelemen error adalah penjelasan teks untuk spesifik error yang sedang terjadi. Sebagai contoh, *bad request* akan memiliki format penulisan sebagai berikut :

```
<error type="400">Bad Request</error>
```

Contoh paket stanza message yang valid dikirim ke entitas/user tujuan adalah sebagai berikut:

```
<message
      from='mustofa@uin-malang.ac.id/kampus'
      to='haqqi@tomatech.mobi'>
      <body>Howdy</body>
</message>
```

2.3.3.2 Presence

Stanza presence ini bertanggung jawab terhadap dua hal dibawah ini, yaitu :

a. *Presence Update*, menginformasikan pengguna lain status presence yang sedang kita digunakan.
b. *Presence Subscription Management*, mengijinkan pengguna untuk mendaftarkan update presence dari pengguna lain dan mengatur siapa saja yang berhak mengetahui status presence-nya.

Dalam kedua peran tersebut server jabber bertindak sebagai penengah antara *presence information generator* dan *presence recepients*. Server tidak memiliki kewenangan untuk secara pasif mengatur rute dari *presence packet* namun secara aktif server berpartisipasi di dalam stanza presence untuk memastikan operasi dilakukan dengan benar.

Presence update menggunakan model pesan satu arah atau *one-way message*. *Client* mengirim stanza presence untuk memperbarui status kehadirannya kepada server, kemudian server meneruskan salinan dari paket tersebut kepada semua pihak yang terdaftar pada *presence subscription list* dari client pengirim. *Subscription list* tersebut dinamakan roster di dalam jabber, namun lebih umum dikenal dengan sebutan *buddy list*.

Entitas jabber dapat dapat meminta *subscribe* presence dari entitas/client lainnya. Proses *subscribing* tersebut adalah sebuah mekanisme/kesepakatan untuk mengetahui status kehadiran dari entitas yang satu dengan yang lain. Sebagai contoh, kita dapat meminta *subscribe* presence teman sehingga ketika teman tersebut online, kita akan mendapatkan notifikasi, tapi tidak sebaliknya.

Beberapa atribut stanza presence diasosiasikan sama dengan stanza message. Presence memiliki memiliki tipe atribut yang memiliki 7 state sebagai berikut:

1. *unavailable* : client tidak lama tersedia untuk berkomunikasi
2. *subscribe*: pengirim mengirimkan request untuk subscribe terhadap presence penerima
3. *subscribed*: pengirim yang telah diizinkan terhadap recipient untuk menerima presence mereka.
4. *unsubscribe* : subscription request yang telah ditolak atau subscription yang telah di cancel sebelumnya.
5. *probe*: request dari client yang presence saat ini
6. *error*: pesan kesalahan yang berlangsung berdasarkan pemrosesan atau menyediakan paket presence yang telah dikirim sebelumnya.

Server menggunakan *probe presence packet* untuk request spesifik entitas dari presence packet. Dalam hal ini entitas yang dimaksud adalah menentukan apakan entitas tersebut *available* atau *unavailable*. Entity Probe mengijinkan informasi presence untuk dikirimkan. Elemen request probe presence dikirim dengan menggunakan format dibawah ini

```
<presence type="probe">
```

Elemen-elemen dibawah ini digunakan di dalam elemen <*presence*>.

```
<status></status>
```

Elemen ini digunakan untuk menampilkan deskripsi status dari user yang dapat langsung dilihat oleh user lain. Misal, pengguna ingin menampilkan status yang

menunjukkan deskripsi dari apa yang sedang ia lakukan, *"i'm at lunch"* atau *"be back in 5 minutes"*

<priority><priority>

Elemen ini memberi prioritas dari presence pada satu entitas pengguna. Misal *smith@example.com* mungkin login dengan menggunakan *multiple resources (home computer, work dan work computer).* Elemen ini memberikan prioritas angka untuk setiap *resource*. Resource dengan angka yang tinggi adalah default *resource*. Semua pesan dan komunikasi akan diarahkan kepada *resource* yang mempunyai nilai prioritas paling tinggi.

Ketika prioritas *resource* paling tinggi tersebut menjadi *unavailable*, pesan dan komunikasi akan dikirim ke *resource* lainnya yang mempunyai nilai prioritas tertinggi kedua. Prioritas yang bernilai negatif menunjukkan bahwa *resource* tidak dapat digunakan untuk *direct* atau *immediate contact*.

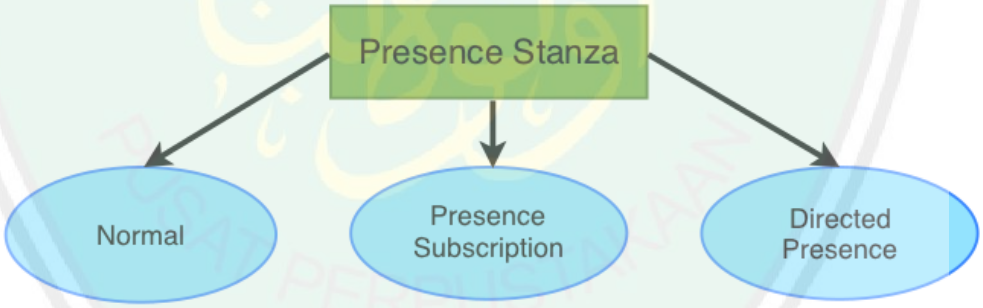
Elemen ini menunjukkan bagaimana status kehadiran seorang user kepada entitas/user lain.

- *show* : menggambarkan status yang tersedia dari entities atau resource yang spesifik. show memiliki empat nilai yang digunakan :
- *away* : temporarily away
- *chat* : bebas untuk chat
- *xa* : extended away
- *dnd* : do not disturb

- status : merupakan deskripsi bahasa natural yang bersifat opsional yang mendeskripsikan status yang tersedia.
- *priority* : bilangan integer bukan negatif yang menampilkan level prioritas pada resource yang terkoneksi, dengan 0 sebagai prioritas terendah.
- *error* : deskripsi pesan kesalahan

Contoh stanza presence adalah sebagai berikut :

```
<presence
      from='juliet@capulet.com/balcony'
      to='romeo@montague.net/orchard'>
      <show>away</show>
      <status>be right back</status>
      <priority>0</priority>
</presence>
```



Gambar 2.6 Presence Stanza (Peter Saint-Andre, 2009)

2.3.3.3 Info/Query

Meskipun secara garis besar trafik jabber terdiri dari message dan presence, sebagian besar pekerjaan mengimplementasikan *client* dan server adalah mengatur administrasi dan manajemen protokol yang mendukung message dan presence. Jabber melakukan tugas

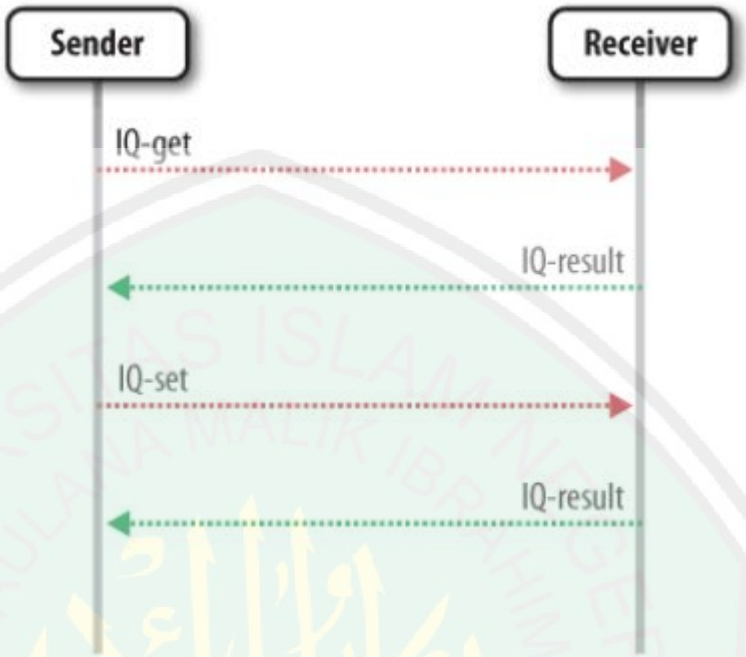
tersebut menggunakan *generic query protocol* yang disebut IQ. Gambar 2.8 menunjukkan gambaran cara kerja dari protokol IQ.

Tipe atribut pada protokol info/query memiliki 4 nilai yang dapat digunakan,

1. *get* : informasi request
2. *set* : menyediakan data yang dibutuhkan
3. *result* : respon terhadap get dan set request yang sukses
4. *error* : kesalahan yang terjadi dalam pemrosesan dan layanan get dan set request

Di dalam setiap IQ, sebuah namaspaces mendefinisikan tipe dari query yang akan dilakukan. Namespaces didefinisikan di dalam elemen query seperti yang ditunjukkan dibawah ini.

```
<query xmlns="*"/>
```



Gambar 2.7 Paket IQ (Peter Saint-Andre, 2009)

Sebagai contoh, *client* mengirim query-set dengan *client authentification* namespaces ke server untuk login.

```
<iq type="set" to="mustofa@uin-malang.ac.id">
      <query xmlns="jabber:iq:auth"></query>
</iq>
```

Format protokol IQ yang digunakan pada jabber dapat dirumuskan seperti yang ditunjukkan dibawah ini.

```
<iq
      type='set|get|result|error'
      to='handler_jid'
      from='originator_jid'
      id='unique'>
      <query xmlns='iq extension namespace'>
      <query_field1/>
      <query_field2/>
      </query>

</iq>
```

Protokol IQ ini sangat penting jika ingin membangun server berdasarkan kebijakan keamanan sistem yang harus dipenuhi oleh *client*. Jika keamanan client telah terpenuhi maka harus mendukung pula terhadap keamanan pada sisi server.

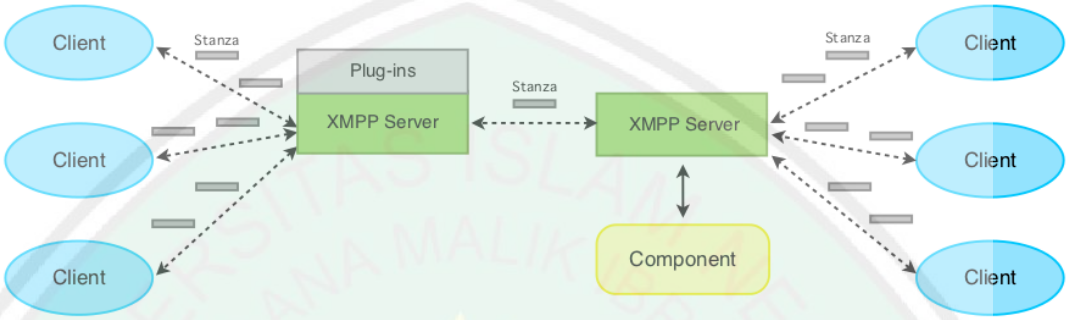
Gambar 2.8 Tipe IQ (Peter Saint-Andre, 2009)

2.3.4 Server Jabber

Pada dasarnya server jabber merupakan software aplikasi sebagai bentuk implementasi core protokol jabber. Secara fungsional, software ini memiliki tiga peranan besar:

- Mengkomunikasikan entitas satu dengan entitas lain pada jaringan jabber

- Berkomunikasi dengan server jabber yang lain
- Mengkoordinasikan beragam komponen yang diasosiasikan dengan server



Gambar 2.9 Komponen backend (Peter Saint-Andre, 2009)

Server jabber didesain modular, dengan paket kode internal yang khusus sehingga dapat menangani fungsionalitasnya seperti *registrasi*, otentikasi, *presence*, *contact list*, penyimpanan pesan yang berstatus *offline* dan sebagainya. Selain itu server jabber dapat dikembangkan dengan komponen eksternal yang memungkinkan administrator server untuk men-*suplemen* server pusat dengan layanan tambahan semacam gerbang untuk sistem aplikasi lainnya.

ejabberd

ejabberd adalah sebuah aplikasi server jaringan jabber yang ditulis dengan bahasa erlang. *ejabberd* dapat berjalan pada beberapa sistem operasi *Unix-like* seperti Mac OS X, GNU/Linux, FreeBSD, NetBSD, OpenBSD, dan OpenSolaris. Selain itu, *ejabberd* juga dapat berjalan pada sistem operasi Microsoft Windows. Nama *ejabberd* sendiri sebuah singkatan dari erlang jabber *daemon* dan ditulis dalam huruf kecil saja, seperti perangkat lunak *daemon* pada umumnya.

Aplikasi server ini didistribusikan dibawah ketentuan dari GNU General Public License. Alexey Scheppin dianugrahi sebagai *Erlang Developer of The Year* pada konferensi pengguna Erlang pada tahun 2006. Pada tahun 2009, aplikasi ini dinobatkan sebagai salah satu aplikasi open source yang paling populer yang ditulis dalam bahasa Erlang. O'Reilly memuji ejabberd untuk fitur skalabilitas dan *clustering* yang dimilikinya.

ejabberd memiliki sejumlah penyebaran dan penggunaan yang luas, seperti *IETF Groupchat Service, BBC Radio LiveText, Nokia Ovi, KDE,* dan juga pengembangan pada fitur *chat* pada facebook. Tahun 2009 *ejabberd* adalah server yang paling populer dari beberapa situs XMPP skala kecil yang terdaftar pada xmpp.org.

Dengan rilis utama setelah versi 2 (sebelumnya disebut ejabberd 3), skema versi diubah untuk mencerminkan tanggal rilis dengan "Tahun.Bulan-Revisi" (dimulai dengan 13.04-beta1). Hal ini juga mengumumkan bahwa pengembangan lebih lanjut akan dibagi menjadi 2 versi yaitu *ejabberd Community Server* dan *ejabberd Commercial Edition* yang menargetkan operator, website, service provider, perusahaan besar, universitas, perusahaan game, semuanya yang membutuhkan tingkat komitmen dan dukungan yang besar dari ProcessOne untuk menjalankan bisnis mereka.

ejabberd memiliki kesesuaian implementasi yang ketat dengan standarisasi protokol XMPP. Baik itu *core* XMPP maupun beberapa XEP populer. *ejabberd* memiliki antarmuka web yang dapat diterjemahkan ke dalam bahasa lain. Mendukung proses komputasi terdistribusi dengan menjalankan beberapa node yang saling terkoneksi, mendukung *runtime upgrade* dan menyediakan dukungan untuk *virtual host*. Terdapat beberapa alternatif sistem

manajemen database yang dapat digunakan, seperti PostreSQL, MySQL dan ODBC. Mendukung otentikasi LDAP, SSL/TLS, SASL dan STARTTLS.

ejabberd terdiri dari banyak modul yang dapat memberikan dukungan dan kemampuan tambahan seperti menyimpan pesan offline, menghubungkan dengan protokol *IRC*, atau informasi personal yang menggunakan vCards user (menyimpan vCards di LDAP atau database ODBC kompatibel mungkin dengan modul lain). Selain itu, beberapa modul dapat memberikan dukungan untuk ekstensi protokol XMPP, seperti *MUC*, *HTTP-polling*, *publish-subscribe*, dan pengumpulan statistik via XMPP. Dimulai dengan versi 2.0.0, *ejabberd* mendukungan transfer dengan *proxy65* yang memungkinkan entitas jaringan Jabber/XMPP di belakang NAT untuk berbagi file melalui SOCKS5 proxy. *ejabberd* dapat berkomunikasi dengan server XMPP lain dan dengan non-XMPP jaringan, menggunakan tipe khusus dari komponen XMPP disebut transportasi atau *gateway*.

2.3.6 Extensible

Huruf X dari XMPP merupakan kata extensible. Istilah extensible berkenaan dengan sifat *core* protokol jabber untuk membawa data (*payload*) yang telah didefinisikan untuk membangun berbagai macam sistem aplikasi yang berbeda. Dari sifat *extensible* ini kemudian terdapat sebuah istilah *extensions* yang berarti proses dan *payload* data yang dirancang untuk tujuan tertentu. Setiap orang dapat merancang atau membangun *extensions*-nya sendiri. Sehingga akan sangat banyak sekali *extensions* yang dapat ditemukan. Tapi terdapat juga *extensions* standar yang diatur oleh XSF (*Jabber Standart Foundations*). Extensions standar ini kemudian dikenal dengan istilah XEP (*Jabber Extension Protocol*).

Komunitas Jabber telah membuat banyak sekali extensions yang dapat dimanfaatkan untuk build sistem aplikasi. Sekitar 300-an XEP yang masuk dan dikoordinasi oleh XSF. Beberapa standar XEP untuk keperluan Instant Messenger akan dijelaskan pada bagian ini.

2.3.6.1 Multi User Chat (XEP-45)

Pada tahun 1999, komunitas jabber telah membuat sebuah basis sistem *text conferencing* yaitu *groupchat 1.0* (GC). Basis sistem ini merupakan minimal fitur untuk implementasi chat room. *Groupchat 1.0* ini merupakan implementasi yang dipakai pada *Internet Relay Chat (IRC)*. Spesifikasi Multi-User Chat (XEP-0045) ini merupakan spesifikasi *advanced features* sebagai pengembangan dari groupchat 1.0. Terdapat banyak sekali fitur yang ditambahkan pada spesifikasi ini, seperti *invitation*, *room moderation*, pembagian tipe room, dan *room discovery*.

Secara umum spesifikasi tidak merubah basis sistem groupchat 1.0 *(backwards-compatible)*, sehingga *baseline functionality* dari groupchat 1.0 ini masih dipakai oleh spesifikasi ini. Fungsionalitas groupchat 1.0 yang dimaksud adalah:

1. Setiap room memiliki identitas unik sebagai *"room JID"* yang memiliki format penulisan *room@service*, contoh *thewolfpack@conference.hangover.com*. *thewolfpack* adalah nama room sedangkan *conference.hangover.com* adalah alamat service dari Multi-User Chat.
2. Setiap partisipator didalam room teridentitas sebagai *"occupant JID"* dengan format penulisan *room@service/nickname*, contoh *thewolfpack@ conference.hangover.com/alex*. *alex* adalah room *nickname* dari user yang tergabung pada room.

3. User dapat masuk kedalam chat room (menjadi partisipator) dengan mengirim secara langsung sebuah presence kepada alamat *room@service/nickname*.
4. Partisipator dapat mengubah nicknamenya dengan mengirim informasi presence ke *room@service/newnickname*.
5. Sebuah message dalam multi-user chat memiliki tipe *groupchat* dan memiliki alamat *room@service*. Service kemudian akan mengirimkannya kepada seluruh partisipator.
6. Partisipator keluar dari room dengan mengirimkan presence tipe "unavailable" ke alamat *room@service/nickname*.

Sedangkan fitur tambahan yang ada pada spesifikasi XEP-0045 ini adalah sebagai berikut:

1. Native conversation logging.
2. Membolehkan user untuk *request* menjadi member pada *private* room.
3. Membolehkan partisipator untuk melihat occupant JID pada *non-anonymous* room.
4. Membolehkan moderator untuk melihat occupant JID pada *semi-anonymous* room.
5. Dapat mengatur hanya moderator saja yang dapat mengganti subjek sebuah room.
6. Membolehkan moderator untuk mengeluarkan (*kick*) *partisipator* dan *visitor* pada room.
7. Membolehkan moderator untuk mengeluarkan dan mencabut ijin bicara pada *moderated* room.
8. Membolehkan administrator untuk memberi dan mencabut ijin sebagai moderator room.
9. Membolehkan administrator untuk mengeluarkan user dari room *(ban)* dan juga mengatur daftar *ban* user.

10. Membolehkan administrator untuk memberikan dan menolak membership *privileges* dan juga mengatur daftar member pada *membe-only* room.
11. Membolehkan *owner* untuk mengatur konfigurasi sebuah room sebagai contoh memberikan limit member.
12. Membolehkan *owner* untuk menunjuk owner lain.
13. Membolehkan *owner* untuk memberikan dan mencabut ijin sebagai administrator dan juga mengatur daftar administrator.
14. Membolehkan owner untuk menutup room.

Pada dokumen spesifikasi ini, disebutkan juga beberapa *room type* yang didefinisikan seperti:

1. *public vs. hidden*
2. *persistent vs. temporary*
3. *password-protected vs. unsecured*
4. *members-only vs. open*
5. *moderated vs. unmoderated*
6. *non-anonymous vs. semi-anonymous*

Beberapa spesifikasi ini saya ambil dari dokumen manual XEP-0045 yang dikeluarkan IETF. Spesifikasi ini begitu komplek bahkan untuk pembuatan aplikasi *group chat*, tapi memang pada dasarnya spesifikasi ini tidak dibuat hanya untuk pembuatan aplikasi itu. Kita bisa membandingkan model yang ditawarkan spesifikasi ini dengan model yang ada pada spesifikasi *publish-subscribe* yang ada pada XEP-0060. Dari kedua model yang ditawarkan ini, model mana yang lebih cocok dengan aplikasi yang hendak kita buat.

Tidak hanya pembuatan group chat saja, saya pernah menemukan sebuah aplikasi chess yang juga memanfaatkan spesifikasi MUC untuk urusan bisnis aplikasinya. Yang saya maksud ini seperti *invite* lawan untuk bermain, lawan menyetujui undangan untuk bermain, melihat semua daftar permainan yang sedang berlangsung (room) dan juga sampai pada mekanisme user *viewer* (bukan pemain) yang hanya masuk untuk menonton jalannya permainan. Game-game model seperti ini tentunya kita sudah banyak melihat, bahkan kita pernah juga memainkannya sekarang ini, seperti game poker dari zinga, poin blank (PB), dll. Dan pada akhirnya kita bisa dengan mudah membuat aplikasi game yang serupa karena modelnya sudah ada dan terbentuk yaitu dengan model *Multi-User Chat* XEP-0045 ini.

Sebelum menyetujui untuk memilih spesifikasi ini, penting juga untuk mengetahui bahwa setiap pesan masuk pada room tidak-lah bersifat *durable*. Artinya adalah pesan masuk akan disampaikan pada user-user yang sedang *join* room sekarang. Penting juga mengetahui maksud kata *"join"* pada kalimat ini. Entitas/user harus memiliki status network *available* dan telah bergabung kedalam room, baru kemudian pesan masuk akan dapat dia terima, tapi jika kemudian koneksi terputus maka dia tidak akan mendapatkan lagi pesan sedang masuk sekarang ini. Baik tipe room publik maupun *member-only* memiliki mekanisme pesan seperti ini. Tidak ada mekanisme *queue* untuk pesan masuk agar pesan tersampaikan semua pada entitas yang sudah langganan ke group seperti yang ada pada protokol AMQP dan MQTT. Untuk beberapa permasalahan yang ada ini, MUC mendefinisikan *Discussion History* yang akan sangat membantu, tapi tentu saja cara ini merupakan *trick* untuk mengatasi permasalahan yang ada. Untuk alasan seperti ini jugalah, kita harus hati-hati dalam memilih protokol dengan spesifikasi *Multi-User Chat* ataukah protokol dengan spesifikasi publish-subscribe seperti XEP-0060, AMQP, MQTT, Apache Kafka, dll.

2.3.6.2 vCard user (XEP-0054)

vCard adalah sebuah standar yang digunakan secara luas untuk informasi personal user. Format vCard ini didefinisikan pada RFC 2426. Sedangkan extension ini adalah spesifikasi yang mendefinikan bagaimana menyimpan dan mengambil sebuah representasi XML vCard entitas/user dari server jabber. Penyimpanan dan pengambilan vCard dapat dilakukan dengan mengirim <*iq*/> tipe set (*storage*) atau get (*retrieval*) ke server entitas jabber. Aliran IQ memiliki child elemen <*vCard*/> dengan namespace 'vcard-temp'. <*vCard*/> elemen mengandung unsur-unsur vCard-XML yang didefinisikan oleh vCard-XML DTD.

```
<iq from='stpeter@jabber.org/roundabout' id='v1' type='get'>
      <vCard xmlns='vcard-temp' />
</iq>
```

Contoh ini digunakan untuk mengambil vCard pengguna (*sender*) dari server jabber. Untuk mengambil vCard dari entitas/user lain dilakukan dengan menambahkan atribut "to" pada <*iq*/> paket. Contoh keluaran yang akan didapat oleh *sender* adalah sebagai berikut:

```
<iq id='v1' to='tri@mimicreative.net/home' type='result'>
<vCard xmlns='vcard-temp'>
<FN>Achmad Mustofa</FN>
<N>
<FAMILY></FAMILY>
<GIVEN>Mustofa</GIVEN>
<MIDDLE/>
</N>
<NICKNAME>Tofa</NICKNAME>
<URL>http://goo-code.blogspot.com</URL>
<BDAY>1988-02-24</BDAY>
<ORG>
<ORGNAME>Mimicreative</ORGNAME>
<ORGUNIT/>
</ORG>
<TITLE>CTO</TITLE>
<ROLE></ROLE>
<TEL><WORK/><VOICE/><NUMBER>087859584883</NUMBER></TEL>
<TEL><WORK/><FAX/><NUMBER/></TEL>
<TEL><WORK/><MSG/><NUMBER/></TEL>
<ADR>
<WORK/>
<EXTADD></EXTADD>
<STREET>Jl. Soekarno hatta 29</STREET>
<LOCALITY>Malang</LOCALITY>
<REGION>Jatim</REGION>
<PCODE> 80202 </PCODE>
<CTRY>Indonesia</CTRY>
</ADR>
<TEL> <HOME /> <VOICE /> <NUMBER>087859584883</NUMBER></TEL>
<TEL><HOME/><FAX/><NUMBER/></TEL>
<TEL><HOME/><MSG/><NUMBER/></TEL>
<ADR>
<HOME/>
<EXTADD/>
<STREET/>
<LOCALITY>Malang</LOCALITY>
<REGION>Jatim</REGION>
<PCODE> 80209 </PCODE>
<CTRY>Indonesia</CTRY>
</ADR>
<EMAIL><INTERNET/><PREF/><USERID>goo.code@gmail.com</USERID></EMAIL>
<JABBERID>mustofa@jabber.com</JABBERID>
<DESC>
More information about me is located on my
personal website: http:goo-code.blogspot.com
</DESC>
</vCard>
</iq>
```

Jika tidak terdapat vCard kontak user, maka keluaran yang akan didapat sender adalah stanza IQ tipe error dengan elemen child <item-not-found>.

```
<iq id='v1' to='stpeter@jabber.org/roundabout' type='error'>
      <vCard xmlns='vcard-temp'/>
      <error type='cancel'>
      <item-not-found xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
      </error>
</iq>
```

Entitas/user bisa saja tidak memberikan informasi apapun terkait dirinya. Untuk kasus seperti ini keluaran yang akan didapatkan sender adalah seperti berikut.

```
<iq id='v1' to='stpeter@jabber.org/roundabout' type='result'>
      <vCard xmlns='vcard-temp'/>
</iq>
```

2.3.6.3 Stream Initiation (XEP-0095)

Spesifikasi ini memberikan metode untuk negosiasi *stream* konten, termasuk juga memberikan meta informasi dari aliran konten tersebut. Jabber adalah protokol yang memiliki basis messaging dan presence, sehingga kebutuhan untuk implementasi protokol-protokol generik akan sangat penting sekali. Spesifikasi ini dapat digunakan untuk menegosiasikan *stream* konten antara dua entitas pada jaringan jabber. *Stream* konten yang dimaksudkan disini dapat berupa *in-band*, atau mungkin juga *out-band*, data biner yang ditransfer pun dapat berupa *file*, *voice chat*, dan *video conferencing*. Proses spesifikasi ini dapat dijelaskan dari use case berikut ini:

1. *Sender* terlebih dahulu harus mengetahui bahwa receiver juga meng-implentasikan spesifikasi ini.
2. *Sender* kemudian menawarkan sebuah *stream initiation*.
3. *Receiver* menerima *stream initiation sender*.
4. *Sender* dan *receiver* kemudian menyiapkan sebuah stream yang telah disepakati melalui *stream initiation* ini.

Sebelum *sender* melakukan stream initiation, *sender* harus memastikan dulu bahwa entitas *receiver* juga dapat melakukan hal ini. Untuk melihat fitur dan spesifikasi yang dimiliki entitas lain dapat dilakukan dengan *Service Discovery (XEP-0030)*.

```
<iq
      type='get'
      to='receiver@jabber.org/resource'
      from='sender@jabber.org/resource'
      id='info' >
      <query xmlns='http://jabber.org/protocol/disco#info' />
</iq>
```

Entitas receiver harus memiliki namespace *http://jabber.org/protocol/si* yang merupakan representasi dari *stream initiation* XEP-0095.

```
<iq
      type='result'
      to='sender@jabber.org/resource'
      from= 'receiver@jabber.org/resource'
      id='info'>
<query
      xmlns='http://jabber.org/protocol/disco#info' >
...
<feature
      var = 'http://jabber.org/protocol/si' />
...
</query>

</iq>
```

Sender mulai melakukan negosiasi dengan mengirimkan sebuah paket <iq/> dengan tipe *set*, seperti pada contoh berikut:

```
<iq type ='set' id='offer1' to='receiver@jabber.org/resource'>
<si
      xmlns='http://jabber.org/protocol/si'
      id='a0'
      mime-type='text/plain'
      profile='http://jabber.org/protocol/si/profile/file-transfer'>
<file
      xmlns = 'http://jabber.org/protocol/si/profile/file-transfer'
      name='test.txt' size='1022'>
<desc>This is info about the file.</desc>
</file>
<feature xmlns='http://jabber.org/protocol/feature-neg'>
<x xmlns='jabber:x:data' type='form'>
<field var='stream - method ' type = ' list - single '>
<option>
      <value>http://jabber.org/protocol/bytestreams</value>
</option>
<option>
      <value>jabber:iq:oob </value >
</option >
<option>
      <value>http://jabber.org/protocol/ibb</value >
</option >
</field>
</x>
</feature>
</si>
</iq>
```

Pada poin ini, *receiver* dapat memperoleh informasi mekanisme *stream* yang dapat dilakukan *sender*. *Receiver* memilih salah satu mekanisme yang juga dapat receiver dukung untuk melakukan stream bytestreams.

```
<iq type='result' to='sender@jabber.org/resource' id='offer1'>
<si xmlns='http://jabber.org/protocol/si'>
<feature xmlns='http://jabber.org/protocol/feature-neg'>
<x xmlns='jabber:x:data' type='submit'>
<field var='stream-method'>
<value>http://jabber.org/protocol/bytestreams</value>
</field>
</x>
</feature>
</si>
</iq>
```

Jika tidak salah satu mekanisme yang dapat digunakan oleh *receiver* untuk melakukan stream, maka *receiver* akan mengirimkan <iq/> error sebagai berikut:

```
<iq type='error' to='sender@jabber.org/resource' id='offer1'>
      <error code='400' type='cancel'>
      <bad-request xmlns ='urn:ietf:params:xml:ns:xmpp-stanzas' />
      <no-valid-streams xmlns ='http://jabber.org/protocol/si'/>
      </error>
</iq>
```

Jika *receiver* menolak *stream initiation*, *reply* yang harus dikirimkan adalah *forbidden error* sebagai berikut:

```
<iq type='error' to='sender@jabber.org/resource' id='offer1' >
      <error code='403' type='cancel' >
      <forbidden xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
      <text xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'>
      Offer Declined
      </text>
      </error>
</iq>
```

2.3.6.4 SOCKS5 Bytestreams (XEP-0065)

SOCKS adalah protokol internet yang memfasilitasi jaringan routing paket antara aplikasi *client-server* melalui server proxy. SOCKS terdapat pada layer 5 dari model OSI Layer (lapisan perantara antara lapisan presentasi dan lapisan transport). Port 1080 adalah port yang biasa digunakan untuk server SOCKS. SOCKS menyediakan standar yang independen dari platform yang digunakan untuk mengakses proxy level sirkuit. Salah satu kemampuan penting SOCKS versi 5 adalah tambahan proses otentikasi dan password, serta memberikan layanan proxy terhadap layanan berbasis UDP, dengan pertama-tama melakukan koneksi TCP, dan kemudian menggunakannya untuk *relay* bagi data UDP.

SOCKS terdiri dari dua komponen, yaitu *SOCKS server* dan *SOCKS klien*. SOCKS server diimplementasikan pada leyer aplikasi, sedangkan SOCKS klien diimplementasikan diantara layer aplikasi dan layer transport. Kegunaan pokoknya adalah untuk bisa menyelenggarakan koneksi dari satu host pada satu sisi dari SOCKS server dengan host lain pada sisi yang lain dari SOCKS server, tanpa kedua host harus terhubung langsung dalam konteks TCP/IP.

Protokol jabber dirancang untuk mengirimkan potongan kecil stream XML antara entitas jaringan dan tidak dirancang untuk mengirimkan data biner. sehingga SOCKS5 dapat dijadikan sebagai *protokol generik* untuk *streaming* data biner antara dua entitas pada

jaringan jabber. Dengan spesifikasi XEP-0065 ini, dua entitas jabber dapat saling terhubung dan menyelenggarakan sebuah *bytestreams* untuk bertukar data biner seperti file dokumen, voice, video, dll. Ada dua skenario ditangani oleh spesifikasi ini:

1. Sebuah koneksi langsung antara dua entitas dapat diselenggarakan di mana Stream host adalah entitas requester (Direct connection).
2. Sebuah koneksi antara dua entitas yang termediasi, di mana Stream host itu adalah sebuah server proxy (Mediated connection).

Gambar 2.10 Mediated connection

Gambar 2.11 Direct connection

2.3.6.5 Receipts Message (XEP-0184)

Sesuai dengan *core* protokol jabber, bahwa stanza message bersifat *send and forget*. Message akan terkirim dan tidak mengembalikan laporan apapun ke *sender*. Core jabber tidak menspesifikan laporan ke *sender* bahwa stanza message yang dia kirim telah diterima pada tujuan. Dengan extension ini memberikan sebuah mekanisme apakah sebuah message yang telah dikirim oleh entitas sender sudah sampai pada entitas tujuannya *(delivered)*. Terdapat dua elemen yang didefinisikan pada spesifikasi ini, yang pertama adalah <*request*/> yang diikutsertakan pada setiap message entitas *sender*, kedua adalah <*received*/> yang dikirim oleh entitas *receiver*, menandakan sebagai ack bahwa message telah *delivered*.

Receipts message memiliki namespace *"urn:xmpp:receipts"* pada keluaran *Service Discovery*. Baik entitas sender maupun entitas *receiver* yang terhubung harus memiliki

namespace ini. Service discovery ini merupakan sebuah mekanisme bagi setiap entitas untuk mengetahui service atau fitur apa saja baik yang didukung oleh server maupun setiap entitas user.

```
<iq
      from='mustofa@mimicreative.net/kantor'
      id='disco1'
      to='himma@mimicreative.net/kantor'
      type='result'>
      <query xmlns='http://jabber.org/protocol/disco#info'>
      ...
      < feature var='urn:xmpp:receipts'/>
      ...
      </query>
</iq>
```

Contoh sebuah paket message dengan spesifikasi ini adalah sebagai berikut,

```
<message
      from = 'mustofa@mimicreative.net/kantor'
      id = '7650071'
      to='didik@uin.ac.id/kampus' >
      <body>Ngopi, nanti malem ditempat biasa.</body >
      <request xmlns='urn:xmpp:receipts' />
</message >
```

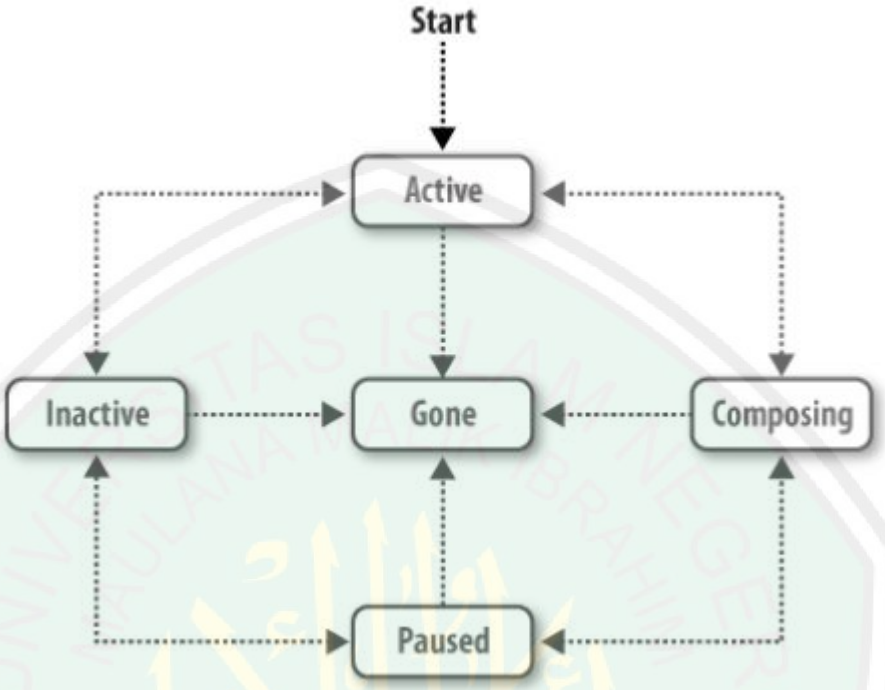
Sender harus memasukkan juga sebuah atribut id pada tiap message yang akan dia kirimkan. Tampak pada contoh diatas, id message tersebut angka random 0c7des. Untuk melakukan implementasi XEP-0184 adalah dengan menambahkan tag *request* pada paket message seperti yang tampak pada contoh diatas. Ketika entitas tujuan juga mengimplementasi spesifikasi ini, maka dia akan mengirimkan sebuah paket ack yang menandakan bahwa paket message telah diterimanya.

```
<message
      from='didik@uin.ac.id/kampus'
      id='698'
      to='mustofa@mimicreative.net/kantor'>
      <received xmlns='urn:xmpp:receipts' id = '7650071' />
</message>
```

2.3.6.5 Chat State Notifications (XEP-0085)

Pada dasarnya ekstensi ini menghasilkan beberapa informasi yang dapat membantu partner/user dalam sebuah sesi *conversation*. Informasi yang dimaksud adalah ketika user meninggalkan halaman chat *conversation*, atau ketika user sedang melakukan *typing*, atau mungkin juga membatalkan/menghapus semua kata yang seharusnya dia kirimkan. Ada lima *state* chat yang didefinisikan, yaitu:

1. <*active*/> : Definisi bahwa user mendukung ekstensi ini dan siap berpartisipasi terhadap chat session.
2. <*inactive*/> : Definisi bahwa user tidak lagi aktif berpartisipasi terhadap chat session.
3. <*gone*/> : Definisi bahwa user selesai melakukan session.
4. <*composing*/> : Definisi bahwa user sedang melakukan "typing".
5. <*paused*/> : Definisi bahwa user sebelumnya melakukan "typing" tapi kemudian berhenti (melakukannya).



Gambar 2.12 Chat state

Spesifikasi ini memiliki namespace *http://jabber.org/protocol/chatstates* pada keluaran Service Discovery. User maupun partner dalam sebuah sesi harus memiliki namespace ini. Hal ini dapat digunakan sebagai mekanisme cek dan ricek bahwa user entitas tujuan/lain juga mendukung extension ini.

```
<iq
      from='mustofa@mimicreative.net/kantor'
      id='disco1'
      to='himma@mimicreative.net/kantor'
      type='result'>
      <query xmlns='http://jabber.org/protocol/disco#info'>
      ...
      <feature var='http://jabber.org/protocol/chatstates'/>
      ...
      </query>
</iq>
```

Beberapa contoh sebuah sesi percakapan dengan *Chat State Notifications* adalah sebagai berikut,

```
<message
      from='mustofa@mimicreative.net/kantor'
      to='tri@mimicreative.net'
      type='chat'>
      <body>Howl, jadi kapan?</body>
      <active xmlns='http://jabber.org/protocol/chatstates'/>
</message >
```

Pada contoh, user *mustofa@mimicreative.net* memulai sebuah percakapan dengan mengirimkan paket <message/> dengan state active kepada *tri@mimicreative.net*. Jika partner percakapan yaitu *tri@mimicreative.net* juga support terhadap spesifikasi ini, maka contoh reply yang akan didapat adalah sebagai berikut:

```
<message
      from = 'tri@mimicreative.net/rumah'
      to = 'mustofa@mimicreative.net/kantor'
      type='chat'>
      <body>Nanti malem yah #BraveYourSelf</body>
      <active xmlns='http://jabber.org/protocol/chatstates'/>
</message>
```

Karena kedua entitas mendukung *chat state notifications*, maka keduanya kemudian dapat memperoleh state informasi terhadap sesi percakapan yang mereka lakukan.

```
<message
      from = 'mustofa@mimicreative.net/kantor'
      to='tri@mimicreative.net/rumah'
      type='chat'>
      <composing xmlns='http://jabber.org/protocol/chatstates'/>
</message>
```

2.3.6.6 Last Activity (XEP-0256)

Ekstensi ini mendefinisikan sebuah mekanisme untuk mengetahui kapan terakhir sebuah entitas user aktif (terkoneksi dan terautentikasi pada server). Spesifikasi ini memiliki namespace *jabber:iq:last*. Ketika user memulai sebuah sesi presence, maka paket berikut memberikan informasi bahwa kapan user tertentu terakhir online atau aktif (memiliki status mode *away* atau *xa*) pada jam dan waktu tertentu.

```
<presence from='haqqi@mimicreative.net'>
      <query xmlns='jabber:iq:last' seconds='86511'/>
</presence>
```

```
<presence from='himma@mimicreative.net'>
      <show>away</show>
      <query xmlns='jabber:iq:last' seconds='600'/>
</presence>
```

Pada contoh terdapat informasi yang memberitahukan bahwa user *haqqi@mimicreative.net* terakhir online pada 24 jam dan 111 detik yang lalu dan user *himma@mimicreative.net* aktif pada 10 menit yang lalu.

2.3.6.7 Delayed Delivery (XEP-0203)

Spesifikasi ini menyediakan timestamp informasi mengenai store data atau pesan *delay* pada server. Ketika entitas tujuan belum available untuk bertukar pesan, pesan yang masuk akan disimpan pada server (*delay*). Begitu entitas tujuan available, semua pesan *delay* akan terkirim padanya. Dengan memakai ekstensi ini, pesan *delay* yang masuk pada entitas tujuan akan terdapat tag <delay/> yang memuat informasi kapan entitas sender mengirim paket message ini.

```
<message from='' to='' type='chat'>
      <body>Hello</body>
      <delay xmlns='urn:xmpp:delay'
            from=''
            stamp='2002-09-10 T23:08:25Z'>
      Offline Storage
      </delay>
</message>
```

Receiving a Message Sent While Offline

```
<presence from='' to=''>
      <status>anon!</status>
      <show>xa</show>
      <priority>1</priority>
      <delay xmlns='urn:xmpp:delay'
            from=''
            stamp='2002-09-10 T23:41:07Z'/>
</presence>
```

Receiving the Last Presence Update of Another Entity

```
<message from='' to='' type='groupchat'>
      <body>Hola</body>
      <delay xmlns='urn:xmpp:delay'
            from=''
            stamp='2002-09-10 T23:05:37Z'/>
</message>
```

Receiving Cached Messages from a Conference Room

2.3.7 Keamanan Protokol

XMPP menggunakan dua macam metode untuk memperkuat otentikasi pada level XML *stream*. Ketika sebuah entitas telah siap dengan yang lainnya, maka hubungan antar entitas seolah-olah telah dibangun ketika user melakukan registrasi dengan server atau seorang administrator sebuah server terhadap server lainnya yang terpercaya, metode yang tersedia untuk autentikasi stream antara 2 entitas menggunakan XMPP yang telah beradaptasi dengan algoritma *Simple Authentication and Security Layer (SASL)*. Ketika tidak ada hubungan yang dapat dipercaya antara 2 server, beberapa level akan dibangun berdasarkan apa yang sudah ada pada *Domain Name Server (DNS)*, metode autentikasi digunakan dalam kasus ini adalah pada server dipasang protokol yang merupakan XMPP asli. Jika SASL digunakan untuk autentikasi server ke server, server harus tidak digunakan dialback. SASL dan metode autentikasi dialback digambarkan dalam sesi berikut ini. Jabber juga mendukung TLS (Transport Layer Security) yang merupakan protokol kembaran dari SSL, digunakan

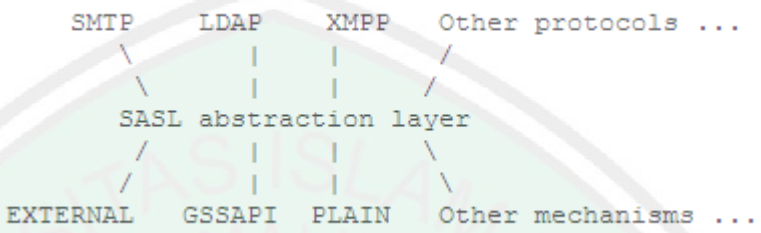
untuk menangani keamanan data yang ditransmisikan melalui jaringan Jabber. Integrasi Jabber dengan TLS ini diatur pada sebuah extensions yaitu XEP-0290.

1. SASL Authentication

SASL merupakan *framework* untuk aplikasi protokol yang mendukung sistem otentikasi. SASL digunakan untuk membuktikan ke server ketika melakukan otentikasi untuk membaca profile. SASL tidak menentukan teknologi yang digunakan untuk melakukan otentikasi, tapi SASL lebih mengarah ke proses negosiasi metode untuk melakukan otentikasi. Biasanya negosiasi SASL bekerja sebagai berikut, pertama permintaan otentikasi klien (mungkin secara implisit dengan menghubungkan ke server). Server merespon dengan daftar yang didukung mekanisme. Klien memilih salah satu mekanisme. Klien dan server kemudian melakukan pertukaran data, satu *round-trip* pada satu waktu, sampai otentikasi berhasil atau gagal. Setelah itu, klien dan server tahu lebih banyak tentang siapa yang berada di ujung lain dari saluran (Simon, 2012).

Menurut (Yenis, 2004) menyatakan bahwa SASL menyediakan metode umum untuk menambahkan otentikasi yang mendukung koneksi berbasis protokol. SASL *(Simple Authentication and Security Layer)* merupakan framework untuk menggabungkan otentikasi protokol dan data yang aman terhadap transportasi antara server dan klien yang terlepas dari protokol aplikasi. kerangka SASL memungkinkan baik aplikasi baru untuk menggunakan kembali protokol otentikasi yang ada pada mekanisme server, atau memungkinkan aplikasi lama untuk menggunakan protokol otentikasi baru tanpa memerlukan modifikasi ke SASL

library (Bartos, 2007). SASL secara konseptual kerangka kerja yang menyediakan lapisan abstraksi antara protokol dan mekanisme seperti yang diilustrasikan sebagai berikut diagram.



Gambar 2.13 Abstraksi otentikasi SASL

Langkah-langkah autentikasi SASL pada protokol Jabber adalah sebagai berikut :

1. *Client* mengawali stream pada server
2. Server merespon dengan sebuah tag stream yang telah dikirim ke *client*
3. Server menginformasikan pada *client* bahwa mekanisme autentikasi telah tersedia
4. *Client* memilih sebuah mekanisme autentikasi (initial respon)
5. Server mengirim *challenge* yang dienkodekan dengan base64 ke *client challenge* didekodekan
6. *Client* merespon *challenge*, respon didekodekan
7. Server mengirim *challenge* yang lain ke *client* dan selanjutnya *challenge* dikodekan
8. *Client* memberi respon terhadap *challenge*
9. Server menginformasikan kepada client apakah autentikasi sukses atau gagal
10. *Client* mengawali sebuah stream baru ke server

11. Server memberi respon dengan mengirim stream header ke *client*, dengan stream yang telah siap diautentikasi

2. Dialback Authentication

Di dalam XMPP termasuk sebuah metode level protokol untuk membuktikan bahwa koneksi antara 2 server dapat dipercaya (minimal seperti DNS yang dapat dipercaya). Metode ini disebut dialback dan hanya dapat digunakan dengan XML stream yang dideklarasikan berdasarkan namespace jabber:server. Tujuan dari protokol dialback adalah membuat spoofing protokol menjadi lebih sulit, dan selanjutnya akan lebih sulit pula untuk memalsukan bait-bait pada tag XML. Dialback tidak diharapkan sebagai mekanisme untuk mengamankan atau mengenkripsi stream antar server, tetapi hanya untuk membantu mencegah terjadinya spoofing pada server dan mengirim data-data yang salah yang ada pada server. Bagaimanapun, domain membutuhkan sistem keamanan yang lebih kokoh (robust) yang seharusnya tidak diperhitungkan hanya pada pilihan autentikasi ini saja.

2.4 Sistem Operasi Android

Android adalah sistem operasi yang berbasis linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti

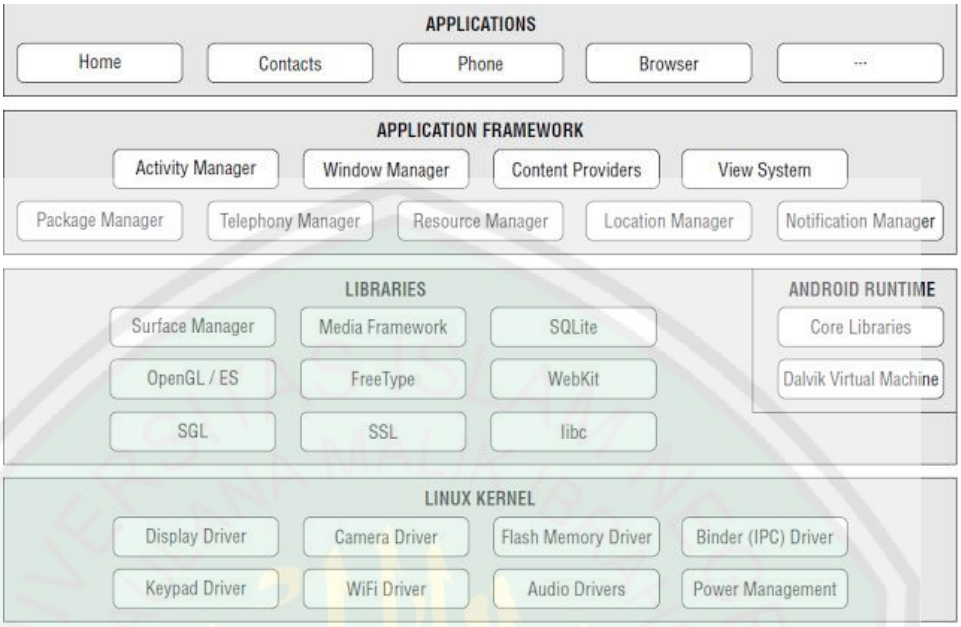
lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Fitur yang tersedia di android adalah:

1. Framework aplikasi, memungkinkan daur ulang dan penggantian komponen.
2. Browser terintegrasi berbasis engine Open Source WebKit yang juga digunakan di browser iPhone dan Nokia S60v3.
3. Rancangan handset. Platform disesuaikan dengan kebutuhan VGA (Video Graphics Adapter) yang lebih besar, library grafik 2D dan 3D yang berdasarkan pada spesifikasi OpenGL ES 1.0 serta layout smartphone yang tradisional.
4. Konektivitas. Android mendukung berbagai teknologi konektivitas seperti GSM (Global System for Mobile Communications)/EDGE (Enhanced Data rates for GSM Evolution), CDMA (Code Division Multiple Access), EV-DO (Evolution-Data Optimized), UMTS (Universal Mobile Telecommunications System), Bluetooth dan Wi-Fi (Wireless Fidelity).
5. Pesan. Android mendukung pengiriman pesan dalam bentuk SMS (Short Message Service) dan MMS (Multimedia Messaging Service).
6. Dukungan Java. Software yang ditulis dalam bahasa Java dapat dikompilasi dan akan dieksekusi pada mesin virtual Dalvik, yang 5merupakan implementasi dari VM (Virtual Machine) yang dirancang khusus untuk penggunaan perangkat bergerak.
7. Dukungan media. Android mendukung beberapa format audio/video seperti: H.263, H.264 (dalam kontainer 3GP atau MP4), MPEG-4 SP, AMR, AMR-WB

(dalam kontainer 3GP), AAC, HE-AAC (dalam kontainer MP4 atau 3GP), MP3, MIDI, OGG Vorbis, WAV, JPEG, PNG, GIF dan BMP.

8. Dukungan hardware tambahan. Android mendukung penggunaan kamera, layar sentuh, GPS (Global Positioning System), pengukur kecepatan, magnetometer, akselerasi 2D bit blits (dengan orientasi hardware, scaling, konversi format piksel) dan akselerasi grafis 3D.
9. Market. Mirip dengan App Store pada iPhone OS, Android Market adalah sebuah katalog aplikasi yang dapat di-download dan diinstal pada telepon seluler secara online, tanpa menggunakan PC (Personal Computer). Awalnya hanya aplikasi gratis saja yang didukung. Dan sejak tanggal 19 Februari 2009 aplikasi berbayar telah tersedia di Android Market untuk Amerika Serikat.
10. Multi-touch. Android memiliki dukungan bawaan untuk multi-touch yang tersedia pada handset terbaru seperti HTC Hero. Pada awalnya fitur tersebut dinonaktifkan pada level kernel (mungkin untuk menghindari pelanggaran paten terhadap teknologi layar sentuh Apple). Sejak Google merilis update untuk Nexus One dan berencana juga untuk merilis update untuk Motorola Droid yang memungkinkan multi-touch. Lingkungan pengembangan yang kaya, termasuk emulator, peralatan debugging, dan plugin untuk Eclipse IDE.



Gambar 2.14 Arsitektur android

sumber: http://wikipedia.com/android

Android berjalan pada kernel Linux dengan menggunakan berbagai macam *library*. Android ditulis dengan menggunakan bahasa C, Aplikasinya berjalan pada *application framework* yang dibangun dengan menggunakan Java dengan memanfaatkan Apache Harmony sebagai *compatible* java *library*-nya. Semua aplikasi Android berjalan pada *virtual machine* yang bernama Dalvik, dimana dalvik inilah yang bertugas untuk melakukan penterjemahan Java *Bytecode* menjadi Dalvik Dex Code (Dalvik-executable). Dalvik merupakan *virtual machine* yang menjadi layer/lapisan antara aplikasi dan sistem operasi. Dimana di dalam file aplikasi yang memiliki ekstensi .apk terdapat beberapa tipe file, diantaranya resource, *assets*, xml dan dex. File dex ini yang diprogram dengan menggunakan bahasa Java. File dex ini akan dijalankan oleh *Dalvik Virtual Machine* untuk melakukan berbagai macam aktifitas, mulai dari menampilkan User Interface, akses Internet,

menjalankan audio, memanggil kamera, dan sebagainya. Semua akses yang dilakukan oleh aplikasi tersebut harus melalui *Dalvik Virtual Machine* terlebih dahulu.

2.5 Black-Box testing

Black box testing adalah teknik pengujian yang menguji hanya berdasarkan kebutuhan dan spesifikasi. Black box testing juga disebut sebagai behavioral testing dan berfokus pada kebutuhan fungsi dari perangkat lunak. Proses umum yang terjadi pada black box testing yaitu:

1. Kebutuhan atau spesifikasi dianalisa terlebih dahulu.
2. Penentuan input valid terpilih berdasarkan spesifikasi untuk menentukan perangkat lunak berjalan dengan benar. Input yang tidak valid juga harus dipilih untuk memverifikasi bahwa perangkat lunak dapat mendeteksinya dan menanganinya dengan baik.
3. Penentuan output yang diharapkan sesuai dengan input yang telah dipilih
4. Pengujian dibuat dengan input yang telah dipilih
5. Pengujian dijalankan
6. Output yang sebenarnya dibandingkan dengan output yang diharapkan
7. Penentuan dibuat menyangkut perangkat lunak berfungsi sesuai dengan spesifikasi yang telah ditentukan.

BAB III

ANALISIS PERANCANGAN SISTEM

Pada bab ini akan dilakukan analisa kebutuhan sistem aplikasi dan perancangan sistem. Analisa sistem akan dilakukan pada kebutuhan sistem secara fungsional, kebutuhan sistem secara non-fungsional dan juga kebutuhan *device capabilities* untuk sistem aplikasi. Sedangkan perancangan sistem terbagi menjadi dua bagian yaitu perancangan tampilan atau user interface aplikasi dan perancangan proses sistem aplikasi.

3.1 Analisa Kebutuhan Sistem

Menurut Jogiyanto HM (1993), analisis sistem didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh kedalam bagian-bagian komponen dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikannya.

Berdasarkan latar belakang sebelumnya, pada bagian ini akan dipaparkan analisa kebutuhan sistem Instant Messaging yang akan dirancang. Bagian ini akan menjawab pertanyaan apa yang seharusnya diimplementasikan pada sistem dan bagaimana tujuan dari perancangan aplikasi ini dapat dicapai. Subbab pertama akan memaparkan analisa kebutuhan secara fungsional dari sistem aplikasi, kemudian diteruskan dengan analisa kebutuhan yang sifatnya non-fungsional dan terakhir pemaparan tentang kapabilitas *device* dan platform minimum yang dibutuhkan agar kebutuhan fungsional dan non-fungsional dapat diimplementasikan dengan baik. Pada setiap subbab terdapat ringkasan kebutuhan sistem disertai prioritas yang akan ditulis dalam format tabel.

3.2.1 Kebutuhan Fungsional

Secara fungsional aplikasi Instant Messaging harus memiliki kemampuan untuk mengirim sebuah konten dalam bentuk teks pesan (FR-1) dan juga data biner. Umumnya mobile platform dilengkapi juga sebuah kamera, *voice* dan video *recorder*. Pengguna smartphone sangat familiar dengan ketiga fitur ini. Aplikasi instant messaging yang akan dibangun harus bisa mengirim sebuah foto dari kamera mereka (FR-1.2), mengirim *voice* yang mereka rekam (FR-1.3) dan juga mengirim video setelah mereka rekam (FR-1.4). Selain itu, pengguna juga dapat memilih gambar dari *gallery* (FR-1.5) atau juga *browse* sebuah dokumen (FR-1.6) kemudian setuju untuk mengirimkannya.

Sistem messaging ini harus dapat mengambil otentikasi dari eksternal sistem yang dimilikinya (FR-2). Eksternal sistem yang dimaksud seperti sistem akademik atau sistem manajemen user yang dibuat diluar sistem aplikasi messaging. Hal ini dilakukan untuk mudah mengatur urusan bisnis aplikasi dengan user akun sistem. Berdasarkan skenario aplikasi bahwa pengguna aplikasi terbatas pada orang-orang tertentu saja (civitas kampus), maka dari itu semua user harus telah teregistrasi dan kemudian mengaktivasi akun miliknya *via* aplikasi (FR-2.2). User akun terbagi menjadi dua yaitu akun mahasiswa dan akun khusus untuk dosen dan staf (FR-2.3). Username yang digunakan untuk akun mahasiswa adalah nomer NIM mahasiswa tersebut, sedangkan untuk akun khusus merupakan username yang user sendiri pilih.

Penggunaan room untuk percakapan lebih dari dua user juga merupakan fitur yang harus ada aplikasi messaging (FR-3). Pada aplikasi yang akan dibangun, room chat tidak bebas dapat dibuat oleh user secara umum, hanya beberapa user pilihan yang dapat membuat room chat dalam hal ini akun khusus buat dosen dan staf karyawan kampus (FR-3.2). Hal ini dimaksudkan agar room chat dapat lebih terkontrol dan terkoordinasi

oleh sistem yang dibangun. Semua room chat yang sudah teregistrasi juga harus dapat dengan mudah ditemukan (*searchable*) oleh user lain (FR-3.3) dan kemudian setuju untuk bergabung. Sebuah room chat juga harus termoderasi oleh lebih dari seorang administrator (FR-3.4). Administrator sebuah room dapat memilih dan menyetujui member lain untuk menjadi juga administrator. Selain terdapat room chat yang sifatnya publik, terdapat juga room yang *private* (FR-3.5). Room *private* artinya room chat tersebut tidak *searchable* dan calon member harus dapat invite terlebih dahulu oleh member room tersebut. User yang dapat invite untuk bergabung ke sebuah room chat, dapat menolak ataupun menyetujui untuk bergabung (FR-3.6). Terdapat sebuah topik atau subyek dari masing-masing room (FR-3.7). Topik dapat dibuat oleh setiap member dan harus mudah terlihat oleh semua anggota room chat.

Setiap kontak harus terdapat profil user bersangkutan seperti jurusan, email, phone dan blog yang dapat diakses (FR-4). Kontak user pun harus dengan mudah ditemukan (*searchable*) (FR-4.2). Terdapat filter terhadap *field* jurusan dan tahun angkatan agar user dapat dengan mudah menemukan user-user lain yang mereka kenal. Selain itu juga semua kontak user yang sudah dimiliki pengguna aplikasi harus juga dapat diakses melalui kontak dari device android (FR-4.3).

FR-1	Mengirim pesan	High
FR-1.2	Mengirim sebuah foto dari kamera device	Medium
FR-1.3	Mengirim sebuah voice note	Medium
FR-1.4	Mengirim sebuah video yang telah direkam dari device	Medium
FR-1.5	Mengirim gambar dari image gallery	High
FR-1.6	Mengirim file dokumen	High
FR-2	Otentikasi eksternal system	High

FR-2.2	Registrasi dan aktivasi user akun	High
FR-2.3	User akun terbagi menjadi dua yaitu akun mahasiswa dan akun khusus untuk dosen dan staf	High
FR-3	Room chat	High
FR-3.2	Pembuatan room chat hanya terbatas user tertentu	High
FR-3.3	Room chat mudah ditemukan (*searchable*)	High
FR-3.4	Moderasi room oleh administrator	High
FR-3.5	Private room	Medium
FR-3.6	Konfirmasi user bergabung atau tidak dari sebuah room invite	High
FR-3.7	Topik untuk room chat	High
FR-4	Profil user kontak	High
FR-4.2	Kontak user mudah ditemukan (searchable)	High
FR-4.3	Semua kontak user pada aplikasi dapat juga ditemukan pada daftar kontak device android	High

Tabel 3.1. Kebutuhan fungsional

3.2.2 Kebutuhan Non-Fungsional

Ada beberapa kebutuhan non-fungsional dari sistem aplikasi yang dibangun. Kebutuhan non-fungsional dari sisi *user-experience* (human factor) yang harus diikuti, *design interface* aplikasi maupun beberapa kebutuhan fungsional yang sifatnya *additional* terhadap aplikasi. Pertama adalah standarisasi protokol (NF-1). Baik core protokol maupun extensions yang dipakai harus merupakan rekomendasi dan terdaftar pada XSF. Aplikasi harus bisa digunakan meskipun tidak dalam terkoneksi *(offline)* (NF-2). Setiap pesan user tersimpan dalam antrian ketika mereka *offline* dan akan terkirim kemudian ketika user kembali *online*. Semua data atau pesan yang masuk harus disimpan pada lokal *storage* yang aplikasi miliki (NF-2.2), agar dapat diakses walau user masih dalam keadaan *offline*. Aplikasi harus dapat menerima pesan baru meskipun device user sedang dalam keadaan *idle* (NF-5). Terdapat sebuah notifikasi yang memberitahukan user bahwa terdapat pesan baru yang dia terima (NF-6). Notifikasi harus diatur dengan baik

(*grouping*) agar user tidak merasa terganggu dengan banyaknya notifikasi yang masuk (NF-6.2). Memisahkan proses yang harusnya berjalan pada background *interface* aplikasi (NF-7) agar user dapat melakukan hal lain daripada menunggu proses selesai. Terdapat pemberitahuan *user typing* (NF-8) ketika *session* percakapan dimulai. Dan informasi pada setiap pesan harus terlihat jelas (NF-9).

NF-1	Standarisasi protokol	Medium
NF-2	Aplikasi bisa digunakan walaupun offline	High
NF-2.2	Terdapat mekanisme antrian untuk pesan yang belum terkirim karena offline	High
NF-3	Aplikasi harus dapat berjalan pada beberapa tipe hardware dengan sistem operasi Android 2.3 (Portability)	Medium
NF-4	Aplikasi ini harus dapat beroperasi terus menerus selama waktu yang diinginkan	High
NF-5	Pesan dapat masuk walaupun device dalam keadaan idle	High
NF-6	Terdapat notifikasi untuk pesan masuk	High
NF-6.2	Notifikasi diatur dengan baik (grouping)	Medium
NF-7	Proses background	High
NF-8	Pemberitahuan user typing	Medium
NF-9	Informasi status setiap pesan harus jelas (save, delay, deliver)	High

Tabel 3.2. Kebutuhan non-fungsional

3.2.3 *Device Capabilities*

Solusi aplikasi seperti instant messaging yang berbasiskan client-server atau peer-to-peer arsitektur, selalu membutuhkan *connectivity* secara konstan dari *wireless network* (DC-1). Kebutuhan device akan teknologi wireless mutakhir yang dapat menghemat bandwidth dan data rate (3G networks), akan dapat melancarkan transfer data-data biner dengan ukuran yang relatif besar (DC-1.2) seperti file dokumen (FR-1.5) dan video (FR-1.3).

Untuk mengirim sebuah foto yang diambil dari kamera device (lihat FR-1.1), maka harusnya terdapat sebuah kamera pada device yang dapat diakses melalui sebuah API yang telah disediakannya (DC-2.2). Begitu juga dengan mengirim sebuah voice note (lihat FR-1.2) dan video record (FR-1.3), device harus menyediakan audio recording API (DC-2.3) dan juga video recording API (DC-2.4) agar dapat diakses dan dimanfaatkan oleh aplikasi.

DC-1	Connectivity dari wireless network	High
DC-1.2	3G networks	Medium
DC-2	Mengirim media file	Medium
DC-2.2	Camera API	Medium
DC-2.3	Voice recording API	Medium
DC-2.4	Video recording API	Medium

Tabel 3.3. Device capabilities

3.2.4 *Server Capabilities*

Penggunaan protokol jabber sebagai *middleware* jaringan aplikasi, membutuhkan sebuah server dengan implementasi *core* protokol yang ketat dan mendukung beberapa standar XEP yang dibutuhkan oleh sistem aplikasi. Pada perancangan aplikasi messaging ini, beberapa XEP harus tersedia pada server yang tampil pada tabel berikut ini:

RFC-3920	XMPP Core	Spesifikasi core protokol jabber
RFC-3921	Instant Messaging and Presence	Spesifikasi dasar pengembangan protokol messaging dan presence
XEP-0045	Multi-User Chat	Spesifikasi protokol untuk group chat

XEP-0004	Data Forms	Spesifikasi bekerja dengan form pada protokol jabber
XEP-0012	Last Activity	Spesifikasi untuk mengetahui kapan waktu entitas terakhir terkoneksi
XEP-0030	Service Discovery	Spesifikasi untuk memperoleh informasi feature setiap entitas jaringan jabber
XEP-0054	vcard-temp	Spesifikasi untuk penerapan vCard user/entitas
XEP-0055	Jabber Search	Spesifikasi untuk mencari daftar kontak berdasar query field dari vCard
XEP-0095	Stream Initiation	Spesifikasi untuk negosiasi filetransfer
XEP-0065	SOCKS5 Bytestreams	Spesifikasi untuk mengadakan koneksi SOCKS5
XEP-0085	Chat State Notifications	Spesifikasi untuk mengadakan chat session dalam percakapan
XEP-0115	Entity Capabilities	Informasi feature dari tiap entitas jaringan jabber
XEP-0203	Delayed Delivery	Spesifikasi untuk memperoleh informasi waktu pada pesan delay
XEP-0184	Receipts Message	Spesifikasi untuk mekanisme pesan delivered

Tabel 3.4. Server capabilities

Pada sistem yang akan dibangun ini menggunakan aplikasi server ejabberd. Server jabber ini sudah terdapat banyak modul implementasi dari banyak spesifikasi XEP. Kesemua XEP yang harus terimplementasikan pada sistem, hanya tinggal diaktifkan saja tanpa harus membuatnya lagi.

3.3 Perancangan Sistem

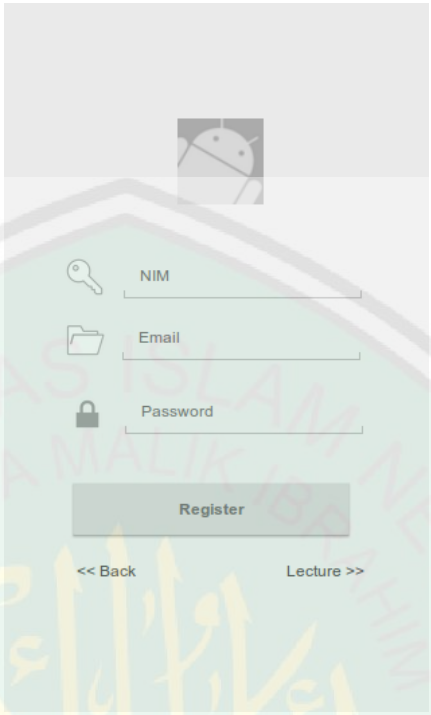
Perancangan sistem perangkat lunak terdiri dari perancangan *class diagram, activity diagram, flowcart diagram, sequence diagram*, dan perancangan antarmuka perangkat lunak. Perancangan sistem ini didapat dari hasil analisa kebutuhan yang kemudian dimodelkan dalam rancangan aplikasi seperti berikut:

3.3.1 Perancangan Antarmuka

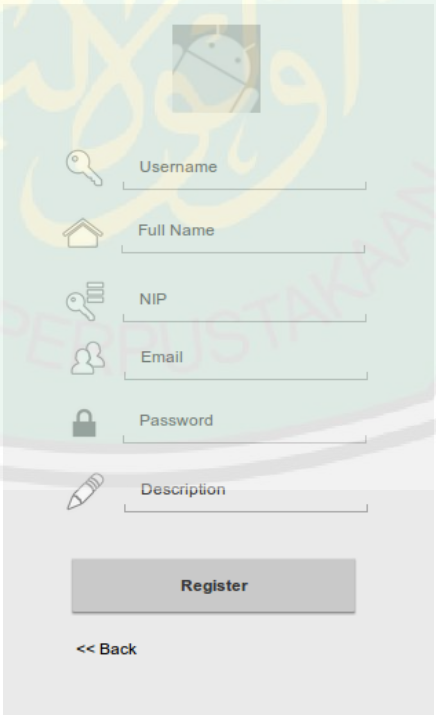
Layout aplikasi merupakan gambaran struktur program yang dijalankan. Perancangan antarmuka sangat diperlukan untuk mempermudah user dalam menggunakan aplikasi. Antarmuka pokok aplikasi instant messenger adalah halaman daftar kontak, halaman daftar room dan halaman chat baik itu chat user maupun chat room. Keempat halaman ini yang akan sering diakses oleh pengguna. Perancangan antarmuka aplikasi ini harus berfokus pada kemudahan akses pengguna untuk keempat halaman pokok ini. Berikut merupakan gambaran antarmuka aplikasi yang akan dibangun.

3.3.1.1 Login dan aktivasi akun

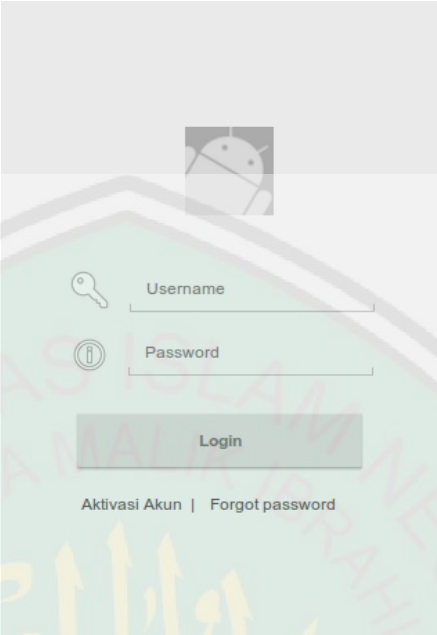
Halaman login berisi masukkan username dan password yang pengguna miliki. Halaman ini juga harus memiliki *link* ke halaman registrasi akun dan halaman forgot password. Terdapat dua halaman yang berbeda untuk registrasi user, yaitu halaman registrasi untuk mahasiswa dan halaman registrasi untuk dosen atau staff.



Gambar 3.1 Rancangan halaman registrasi mahasiswa



Gambar 3.2 Rancangan halaman registrasi dosen 1



Gambar 3.3 Rancangan halaman registrasi dosen 2

Pada halaman login berupa sebuah *form* yang terdiri dari kontrol text input untuk username (NIM untuk mahasiswa) dan text input untuk password user. Sedangkan untuk halaman registrasi juga terdapat sebuah *form* registrasi. Pada *form* registrasi mahasiswa semua kontrol berbentuk text input yaitu NIM, email dan password. Untuk *form* registrasi dosen juga berbentuk kontrol text input semua yaitu username, NIP, email, password dan keterangan.

3.3.1.2 Kontak user, room chat dan menu utama

Halaman daftar kontak dan room merupakan dua halaman yang pasti ada dalam aplikasi instant messenger. Kedua halaman ini akan sering diakses pengguna, maka dari itu perpindahan akses kedua halaman ini harus dibuat semudah mungkin. Perancangan antarmuka yang akan dilakukan adalah dengan membuatnya berbentuk *tab* yang terdiri dari halaman daftar kontak user, halaman daftar room dan halaman yang berisi menu

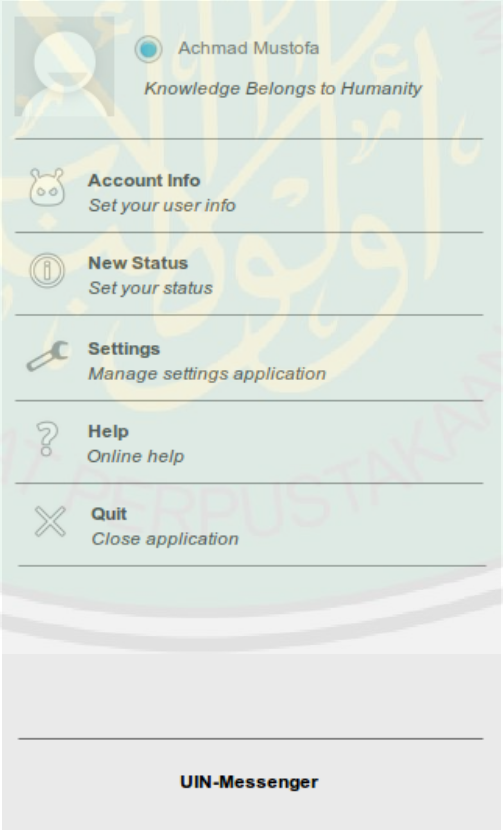
utama aplikasi. Perpindahan halaman harus dapat dilakukan dengan menggeser atau *swipe device* ke kanan atau ke kiri.

Gambar 3.4 Rancangan halaman kontak

Gambar 3.5 Rancangan halaman daftar room

Pada setiap user kontak (Gambar 3.4) menampilkan foto user, nama lengkap, isi percakapan terakhir dan status mode user. Percakapan terakhir disini juga dapat digantikan oleh status teks user. Status mode user yang tersedia adalah status *available*, *away*, *do not distrub* dan *offline*, yang masing-masing dari status mode terwakili oleh sebuah ikon.

Sama seperti halaman kontak user, setiap room chat (Gambar 3.5) menampilkan informasi seperti foto room, nama, topik atau isi percakapan terakhir dan juga status pengguna terhadap room bersangkutan. Status pengguna terhadap room seperti member room, administrator room dan pengguna sedang keadaan *offline* (belum *join*).

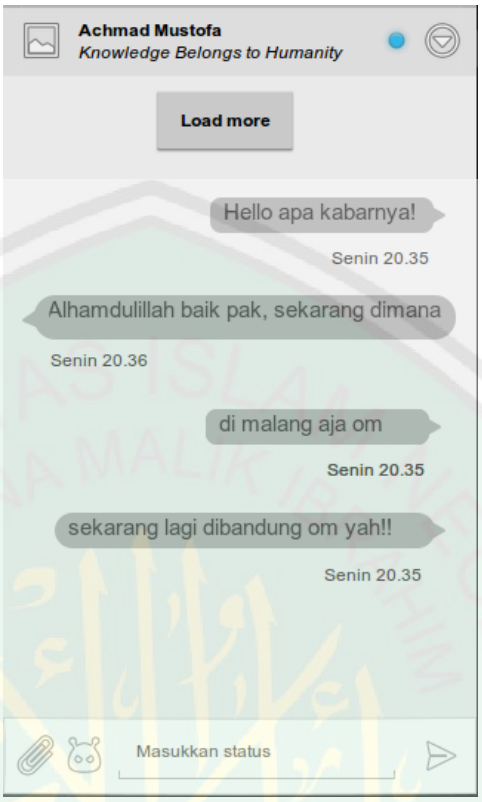


Gambar 3.6 Rancangan menu utama

Gambar 3.6 adalah perancangan halaman menu utama. Bagian atas terdapat informasi pengguna seperti foto, nama dan status teks pengguna. Menu utama aplikasi terdiri dari (1) menu *Account Info* yang akan mengantarkan pengguna menuju informasi profil yang pengguna miliki, (2) *New Status* untuk mengubah status teks dan status mode pengguna, (3) *Settings* berisi semua pengaturan aplikasi instant messanger, (4) *Help* yang akan mengantarkan pengguna ke situs dokumentasi aplikasi, dan (5) *Quit* untuk menutup aplikasi (bukan *logout*).

3.3.1.3 Halaman chat user

Halaman ini berisi konten percakapan antara pengguna dan user kontak. Konten percakapan bisa berupa teks, *voice note*, video, dan file dokumen. Perancangan antarmuka pada halaman ini akan terbagi menjadi 3 bagian utama, pertama adalah infrormasi user kontak yang akan terletak pada bagian atas halaman ini, kedua adalah konten percakapan user yang akan mendominasi halaman dan ketiga adalah masukkan pesan pengguna yang akan terletak pada bagian bawah halaman.



Gambar 3.7 Rancangan halaman chat kontak

Isi percakapan berbentuk balon yang memiliki orientasi ke kanan dan ke kiri. Balon kanan merupakan pesan yang dari pengguna sendiri, sedangkan balon kiri merupakan pesan user sebagai *partner* komunikasi. Ikon *attachment* akan berisi popup menu media untuk mengakses beberapa media seperti *image gallery* (FR-1.5), kamera (FR-1.2), *video recording* (FR-1.4) dan *file browser* (FR-1.6).

3.3.1.4 Halaman room chat

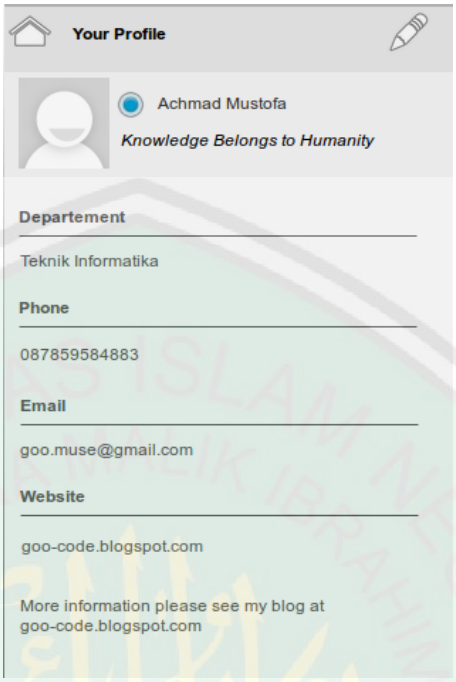
Halaman ini berisi konten percakapan room. Perancangan antarmuka pada halaman ini akan terbagi menjadi 3 bagian utama, pertama adalah infrormasi mengenai room yang akan terletak pada bagian atas halaman ini, kedua adalah isi atau percakapan

chat room yang akan mendominasi halaman dan ketiga adalah masukkan atau input pesan pengguna yang akan terletak pada bagian bawah halaman.

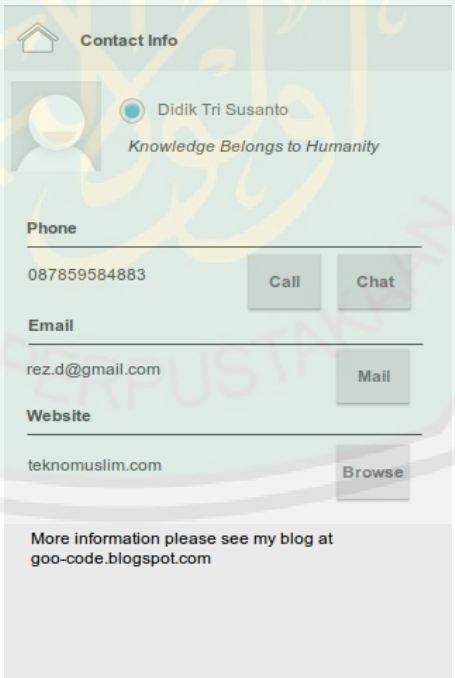
Gambar 3.8 Rancangan halaman chat room

3.3.1.5 Profil pengguna dan user kontak

Halaman profil pengguna berisi semua informasi mengenai pengguna aplikasi, mulai dari nama lengkap, foto, email, nomer telpon, website dan deskripsi singkat mengenai dirinya. Profil pengguna seperti yang ditunjukkan pada gambar. Halaman ini juga harus terdapat sebuah *button* yang akan menuju halaman perubahan profil. Pada perancangan antarmuka *button* ini terletak samping kanan atas pada bar aplikasi.



Gambar 3.9 Rancangan halaman profil pengguna



Gambar 3.10 Rancangan halaman info kontak user

Halaman profil user kontak berisi semua informasi mengenai user kontak, mulai dari nama lengkap, foto, email, nomer telpon, website dan juga deskripsi singkat

mengenai dirinya. Pada halaman ini juga harus memberikan kemudahan pengguna untuk mengirimkan pesan atau memanggil user *via* mail dan telpon.

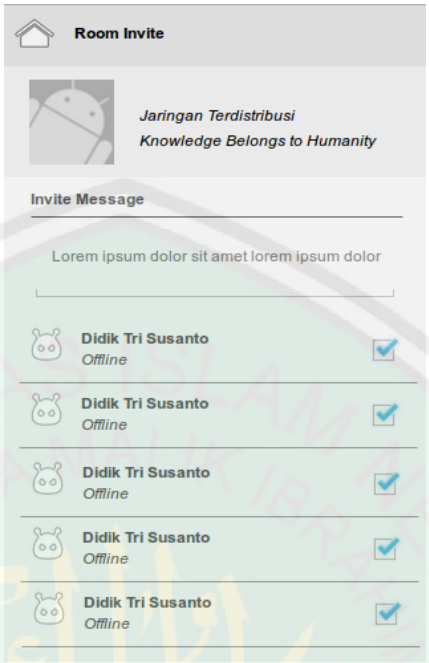
3.3.1.6 Form perubahan status pengguna

Halaman ini berisi form perubahan status oleh pengguna. Pada *form* ini terdapat status teks kontrol *text area* dan juga status mode berbentuk *radio button*. Status mode yang tersedia pada bagian ini yaitu status *available*, *away*, *status do not distrub*, dan status *offline*.

Gambar 3.11 Rancangan halaman edit status

3.3.1.7 Invite user

Halaman *invite* user berisi *form* untuk mengirimkan sebuah *invite* user untuk bergabung pada sebuah room. *Form* yang disediakan harus memungkinkan pengguna untuk mengundang ke lebih dari satu user kontak yang dia miliki.

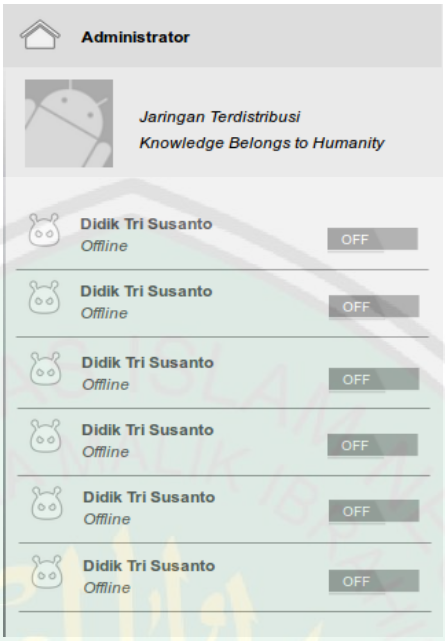


Gambar 3.12 Rancangan halaman room invite

Pada bagian atas terdapat informasi room seperti foto, nama dan deskripsi singkat room. Terdapat masukkan *input* alasan atau pesan singkat oleh pengguna kepada user *invite*. Kemudian pada bagian bawah berisi daftar semua kontak user yang dimiliki oleh pengguna. Pada setiap item kontak terdapat sebuah *checkbox* disamping kanan untuk seleksi user *invite*.

3.3.1.8 Form administrator room

Halaman ini berisi *form* untuk memberikan atau mencabut ijin sebagai administrator room. Pada setiap item member yang muncul menggunakan kontrol tombol *on/off* untuk seleksi member administrator. Pada bagian atas item member, terdapat informasi room foto, nama dan deskripsi dari room bersangkutan.



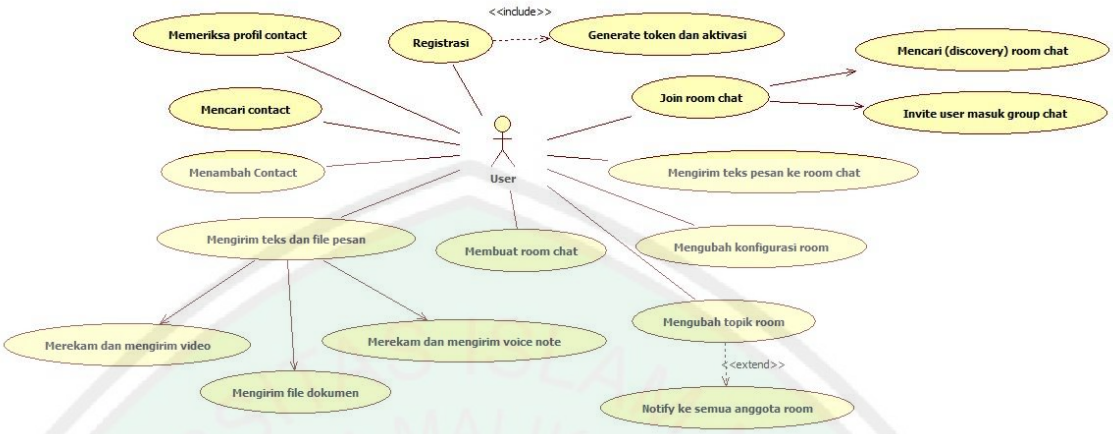
Gambar 3.13 Rancangan halaman edit admin room

3.3.2 Perancangan Proses

Perancangan proses dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan. Perancangan aplikasi berdasarkan *Obeject Oriented Analysis* dan *Object Oriented Design* menggunakan pemodelan UML (*Unified Modeling Language*). Perancangan proses dilakukan dengan mengidentifikasi kelas-kelas dan interface-interface yang dibutuhkan yang dimodelkan dalam class diagram. Beberapa aktivitas sistem akan ditampilkan berbentuk *activity diagram* dan beberapa lagi akan ditampilkan berbentuk sebuah *flowchart diagram*.

3.3.2.1 Use Case

Use case model merupakan pemodelan struktural yang mencerminkan fungsionalitas sistem dan merupakan kebutuhan fungsional yang digambarkan dari sudut pandang user pada sebuah sistem.

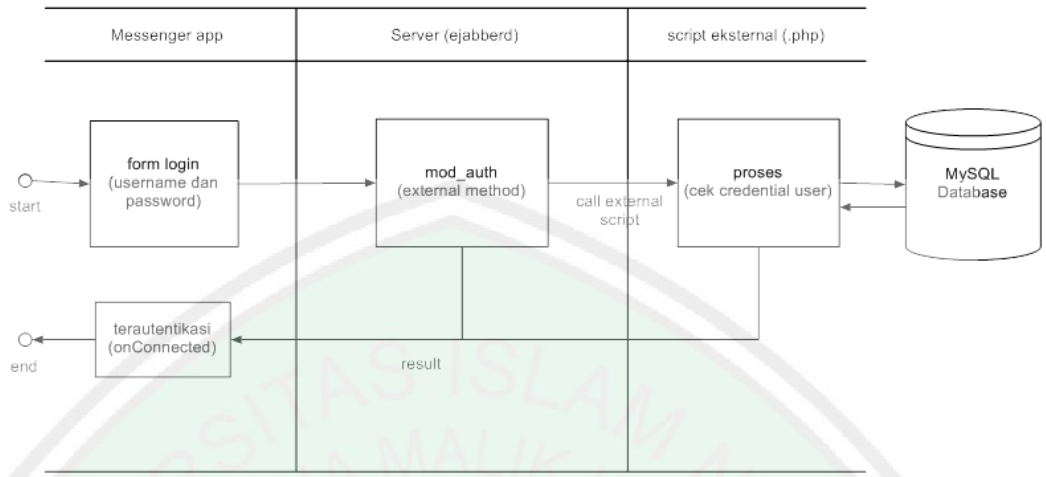


Gambar 3.14 Use case aplikasi

Use case ini merupakan daftar aktivitas yang dapat dilakukan oleh user terhadap sistem yang akan dibangun. Beberapa aktivitas seperti membuat room chat yang hanya bisa dilakukan oleh dosen atau karyawan (lihat analisa kebutuhan FR-3.2), mengubah konfigurasi room yang hanya bisa dilakukan jika user bersangkutan adalah admin room, dan lain sebagainya.

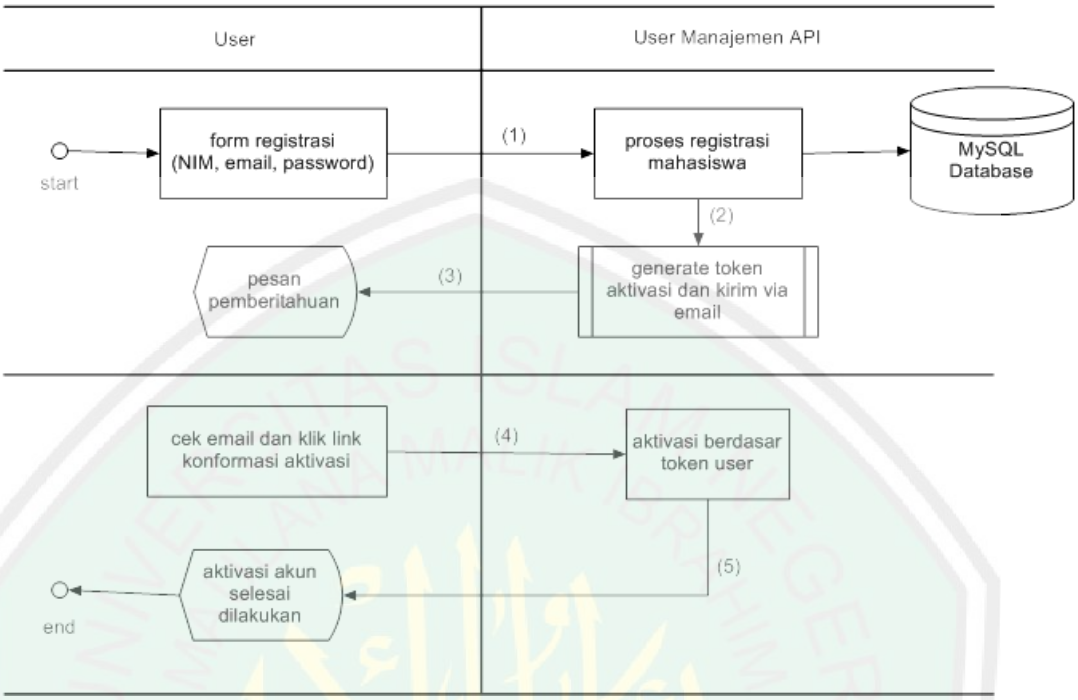
3.3.2.2 Activity Diagram

Activity diagram digunakan untuk memodelkan aspek dinamis dari sistem dan juga aliran kendali dari suatu operasi. *Activity diagram* secara esensial mirip dengan diagram alir (*flowchart*), memperlihatkan aliran kendali dari suatu aktivitas ke aktivitas lainnya.



Gambar 3.15 Activity diagram otentikasi user

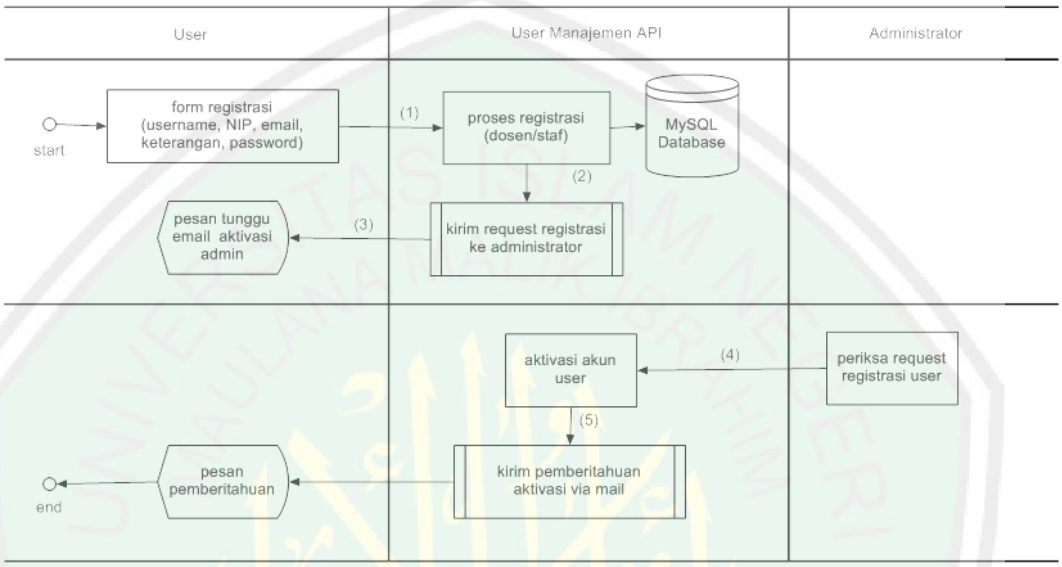
Diagram diatas merupakan diagram aktivitas untuk otentikasi user login. *Flow* yang diperlihatkan pada diagram dari aplikasi pengguna *(messenger apps)*, server ejabberd dan sebuah script eksternal. Pengguna memasukkan username dan password, kemudian submit mengikuti mekanisme otentikasi yang dimiliki oleh protokol jabber. Pada server ejabberd kemudian memanggil sebuah *script* php untuk memastikan apakah data *credential* yang pengguna masukkan telah terdaftar dalam database atau tidak. Hasilnya kemudian dikembalikan kepada pengguna aplikasi.



Gambar 3.16 Activity diagram registrasi mahasiswa

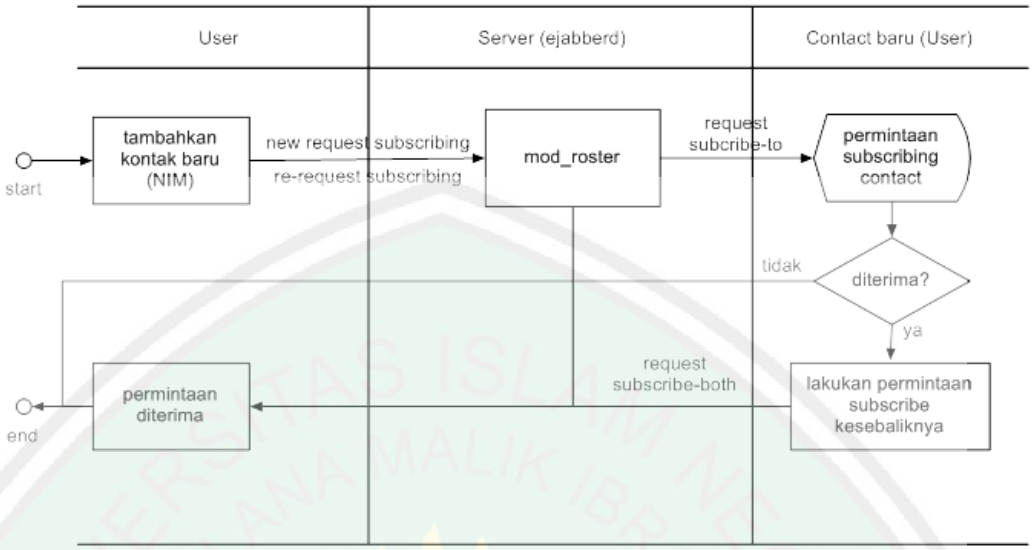
Diagram diatas merupakan diagram registrasi untuk mahasiswa. Terdapat dua aktivitas besar yang ditunjukkan, pertama adalah aktivitas registrasi dan kedua aktivitas aktivasi. Pada aktivitas yang pertama, user mahasiswa mengisi *form* registrasi yang hanya terdiri dari NIM, email dan password yang akan mereka gunakan. *Submit form* yang ditujukan kepada sebuah API via HTTP. Data mahasiswa akan dicatat dalam user database dengan aktivasi bernilai 0 yang berarti akun user belum teraktivasi. Proses kemudian menggenerate sebuah token untuk aktivasi user bersangkutan. Token kemudian dikirim kepada user via email. Aktivitas pertama berakhir dan dilanjutkan dengan aktivitas kedua untuk aktivasi akun user. User kemudian akan menemukan sebuah link konfirmasi dengan token untuk aktivasi akun yang user miliki. Link konfirmasi yang dihasilkan disini hanya berumur sehari. Jika user melebihi batas waktu yang ditentukan tersebut, maka user harus mengulangi proses dari awal. Setelah konfirmasi aktivasi akun

dilakukan, user kemudian dapat menggunakan aplikasi messenger yang telah dipersiapkan.



Gambar 3.17 Activity diagram registrasi dosen

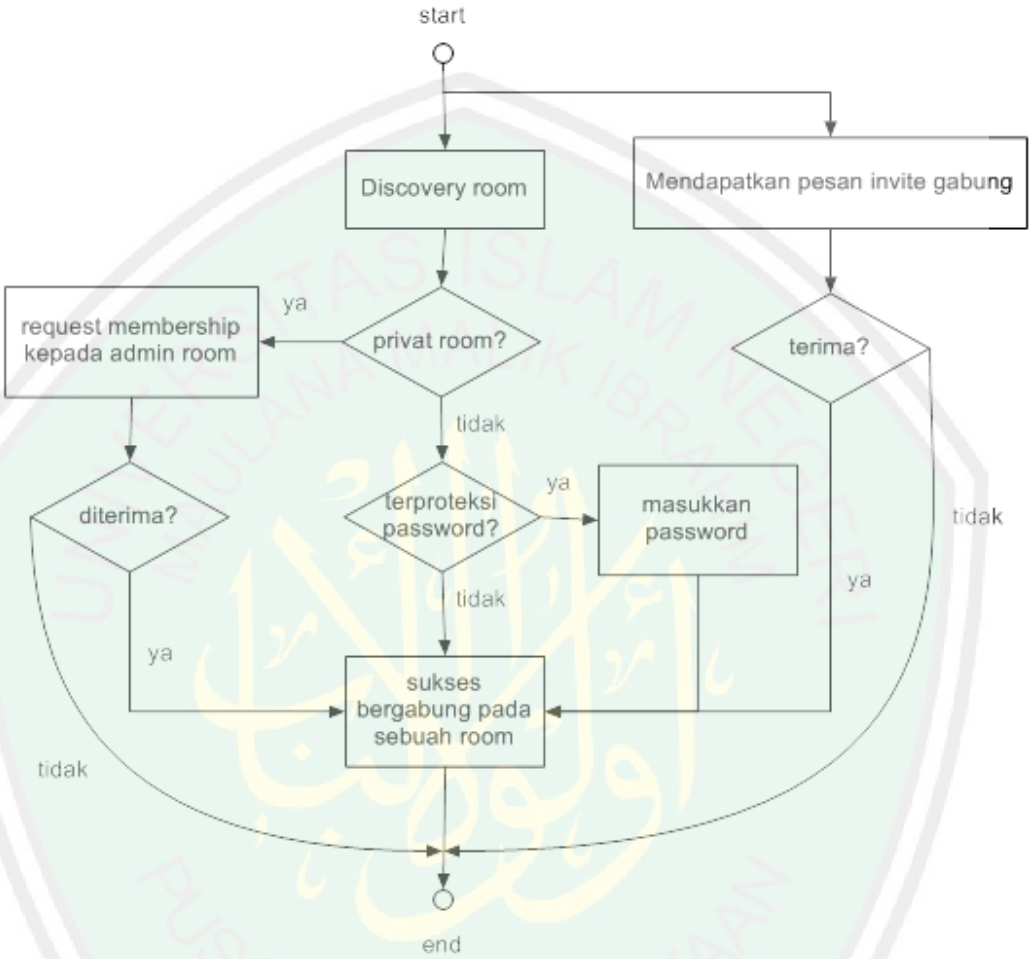
Diagram diatas merupakan diagram untuk registrasi dosen dan karyawan kampus. Terdapat dua aktivitas besar juga seperti pada registrasi mahasiswa. Perbedaan yang terlihat disini adalah pada aktivitas aktivasi akun. Jika pada mahasiswa memakai token untuk aktivasi, maka pada aktivasi dosen dilakukan oleh seorang administrator sistem. Administrator akan memeriksa permintaan user/dosen dengan melihat masukkan seperti NIP, email dan deskripsi dari user bersangkutan. Pada registrasi ini, user dapat memasukkan sendiri username yang akan dia gunakan pada aplikasi. Administrator kemudian dapat mengaktifkan akun user bersangkutan dan mengirimkan sebuah email sebagai pemberitahuan bahwa akun telah aktif dan dapat digunakan.



Gambar 3.18 Activity diagram subscribing user

Diagram ini merupakan diagram untuk aktivitas *subscribing* sebuah kontak baru. Proses diawali dengan memasukkan username kontak baru. Username untuk mahasiswa adalah NIM yang mereka miliki, sedangkan dosen dan staf memiliki username unik yang dapat mereka pilih sendiri. *Request subscribing* diteruskan kepada user tujuan dan menunggu apakah permintaan diterima ataukan tidak. Jika permintaan diterima, user tujuan juga secara otomatis mengirimkan permintaan *subscribe* kepada user asal yang kemudian akan diterima secara otomatis oleh user asal. Sehingga disini kedua user asal dan user tujuan saling dapat menerima *presence* dari user masing-masing.

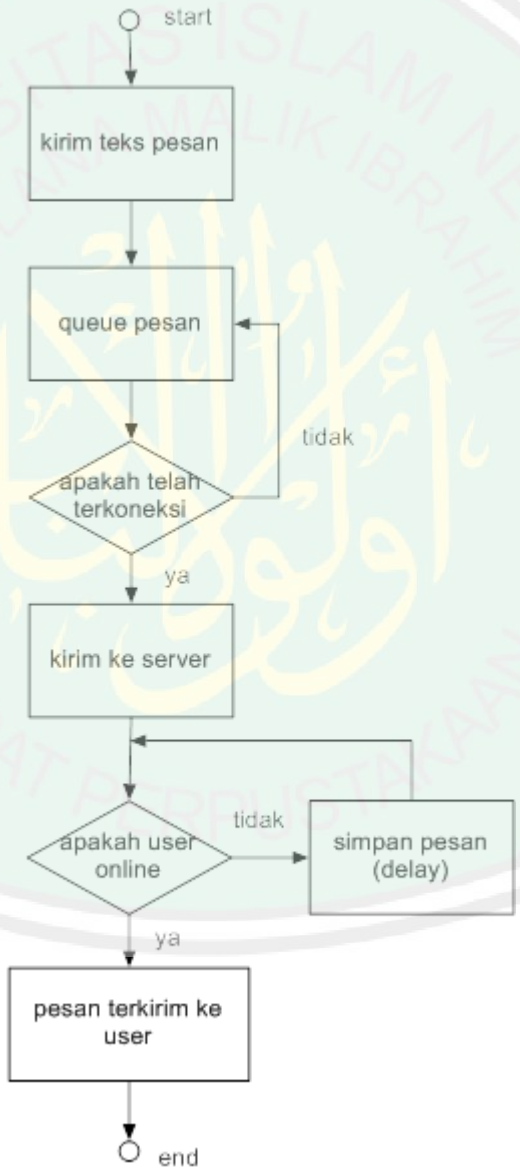
3.3.2.3 Flowchart Diagram



Gambar 3.19 Flowchart join sebuah room

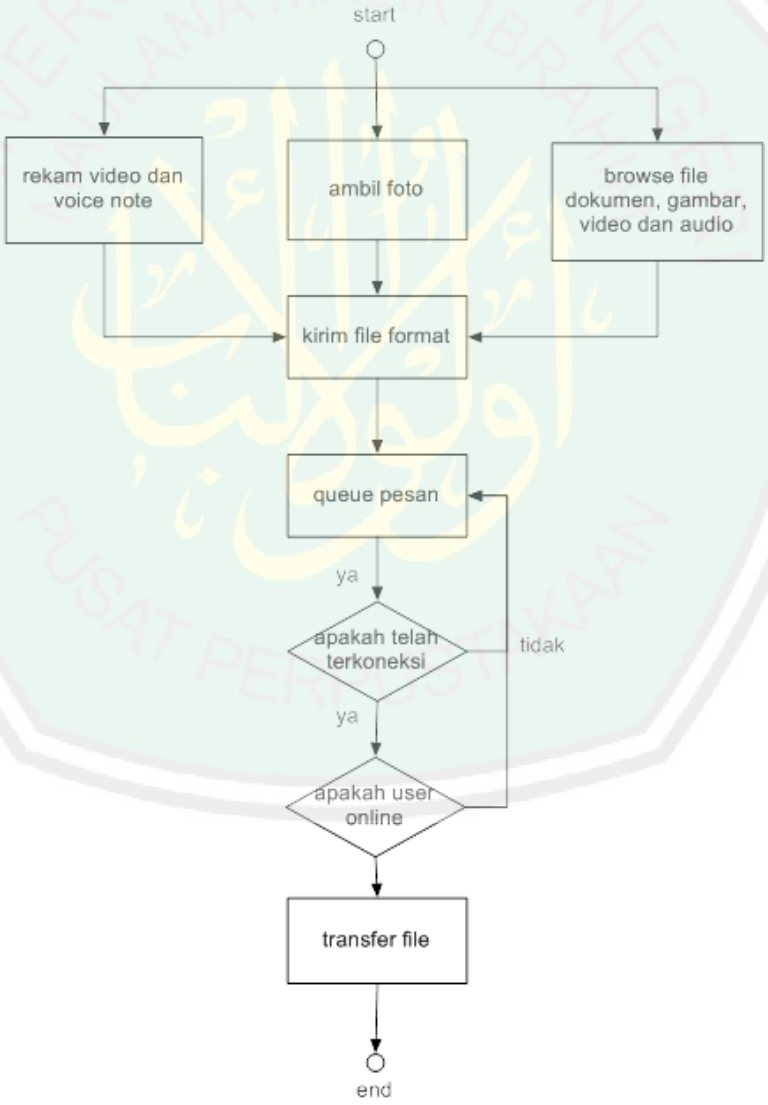
Flowchart ini menunjukkan aliran operasi *join* pada sebuah room chat. Pada setiap room chat memiliki opsi yang dapat dikonfigurasi secara berbeda oleh admin room bersangkutan. Setiap user harus mengikuti aliran ini agar dapat bergabung pada sebuah room. Pada *flowchart* diatas menunjukkan ada dua aliran terpisah untuk bergabung, yaitu user melakukan *discovery* semua room, kemudian setuju untuk bergabung pada salah satu room dan user menerima sebuah paket *invite* dari member room untuk bergabung dalam room bersangkutan. Pada aliran pertama, privat room user harus melakukan permintaan

request dulu kepada admin room bersangkutan. Dan untuk room yang terproteksi password, user harus memasukkan password yang benar agar dapat bergabung. Sedangkan pada aliran kedua, user dapat memilih apakah dia menerima *invite* untuk bergabung ataukah tidak.



Gambar 3.20 Flowchart kirim pesan teks

Mengirim sebuah pesan teks kepada user kontak dapat dijelaskan pada flowchart ini. Semua pesan akan terlebih dulu masuk kedalam antrian *queue* yang ada pada lokal aplikasi dan kemudian akan terkirim ke server. Server akan menyimpannya jika user tujuan tidak dalam keadaan *available (delay)*, dan ketika user tujuan sudah dalam keadaan available server kemudian akan meneruskan pesan delay tersebut kepada user tujuan.

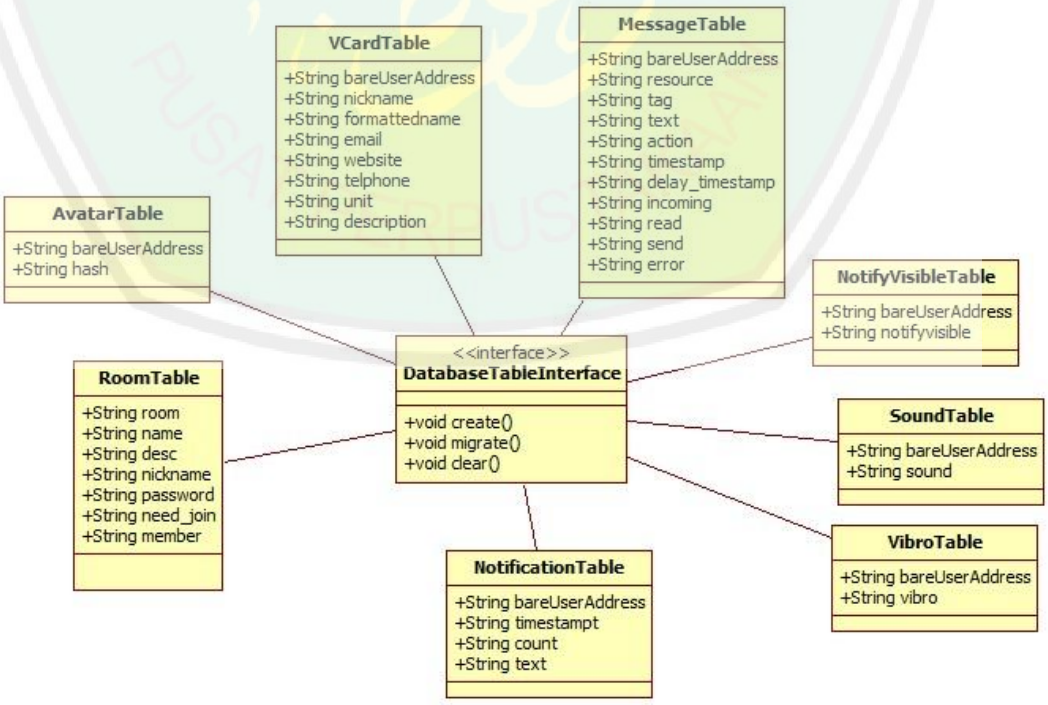


Gambar 3.21 Flowchart kirim file

Flowchart ini adalah alur proses untuk mengirim sebuah file. File disini merupakan data biner dengan berbagai macam format, seperti format video, audio dan dokumen. Sama seperti mengirim teks, bahwa file message akan terlebih dulu masuk antrian pada lokal aplikasi, tapi perbedaannya disini pesan tidak akan terkirim ke tujuan jika baik user pengirim maupun user tujuan tidak dalam keadaan *online*. Pada proses ini tidak ada mekanisme untuk *delay* pesan pada server. Mekanisme proses ini sesuai dengan spesifikasi protokol pada XEP-0065 yaitu mengenai penggunaan standar file transfer pada jaringan jabber.

3.3.2.4 Class diagram

Class Diagram menunjukan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.



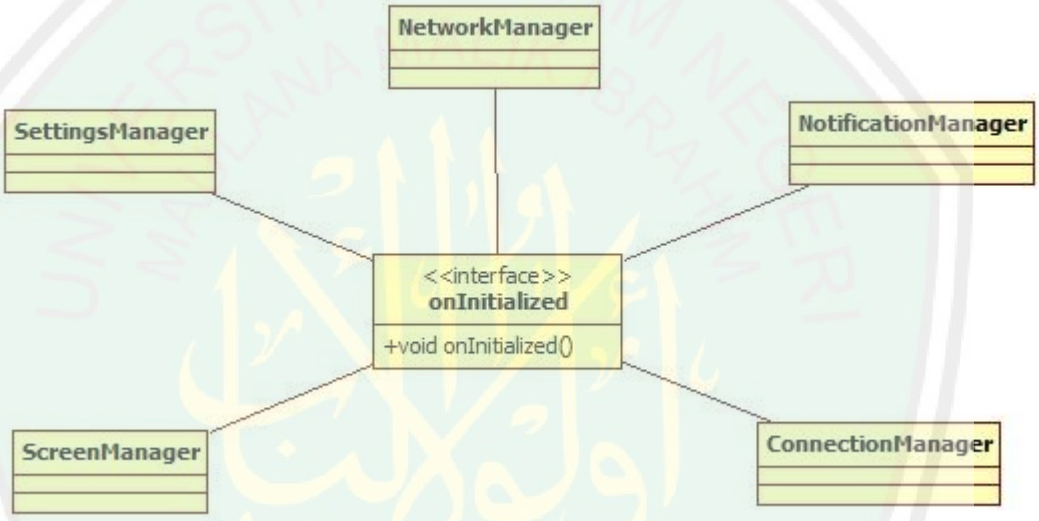
Gambar 3.22 Class diagram schema lokal database

Class diagram berikut ini dapat memberikan gambaran perancangan database lokal pada aplikasi yang akan dibangun. Interface DatabaseTableInterface merupakan merupakan kelas abstraksi untuk representasi sebuah struktur tabel pada database. Tiga method yang harus terimplementasi pada kelas implementasi, yaitu *create*, *migrate* dan *clear*. Method *create* dipanggil untuk membangun struktur data pada tabel, *migrate* untuk operasi *migration* database pada android versi tertentu dan method clear untuk operasi penghapusan tabel pada database. Berikut adalah daftar nama tabel beserta field entitas yang ada pada masing-masing kelas implementasi tersebut.

Nama	**Deskripsi**	***Fields***
vcards	Penyimpanan lokal untuk informasi vCard pada masing-masing kontak	user, nick_name, formatted_name, email, website, telphone, unit, description
avatars	Penyimpanan lokal untuk avatar kontak	user, hash
messages	Tabel simpan untuk setiap pesan chat	user, tag, resource, text, action, timestamp, delay_timestamp, incoming, read, sent, error
notifications	Tabel simpan setiap notif masuk	user, text, timestamp, count
rooms	Tabel simpan untuk setiap room yang telah join	room, nickname, password, need_join, room_name, room_desc, room_member
chat_notify_visible	Tabel simpan untuk pengaturan notif pada masing-masing chat	user, value

sound	Tabel simpan untuk pengaturan sound pada masing-masing chat	user, value
vibro	Tabel simpan untuk pengaturan vibro masing-masing chat	user, value

Tabel 3.5. Local storage android

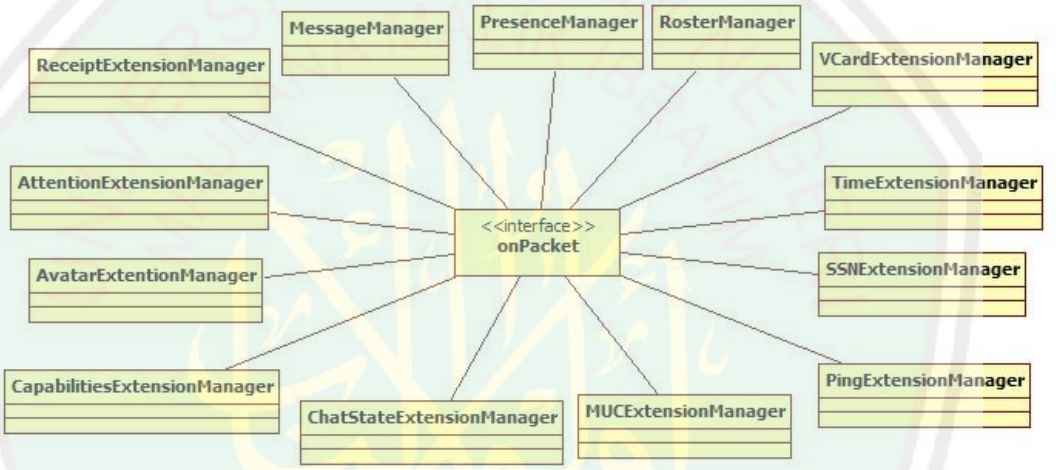


Gambar 3.23 Class diagram kelas onInitialized

Class diagram diatas menunjukkan beberapa kelas implementasi dari interface onInitialized. Kelas implementasi onInitialized akan dipanggil setelah data-data yang tersimpan pada lokal *storage* selesai terload. Beberapa kelas implementasi dan penjelasannya adalah sebagai berikut.

1. NetworkManager: merupakan sebuah source class yang akan bertanggung jawab terhadap urusan network API OS.
2. NotificationManager: merupakan sebuah source class yang akan mengurusi semua bentuk notifikasi pada aplikasi.

3. ConnectionManager: merupakan sebuah source class yang akan bertanggung jawab terhadap mengadakan sebuah koneksi aplikasi server.
4. ScreenManager: source class untuk callback action dari screen broadcast android.
5. SettingsManager: source class untuk setting aplikasi.

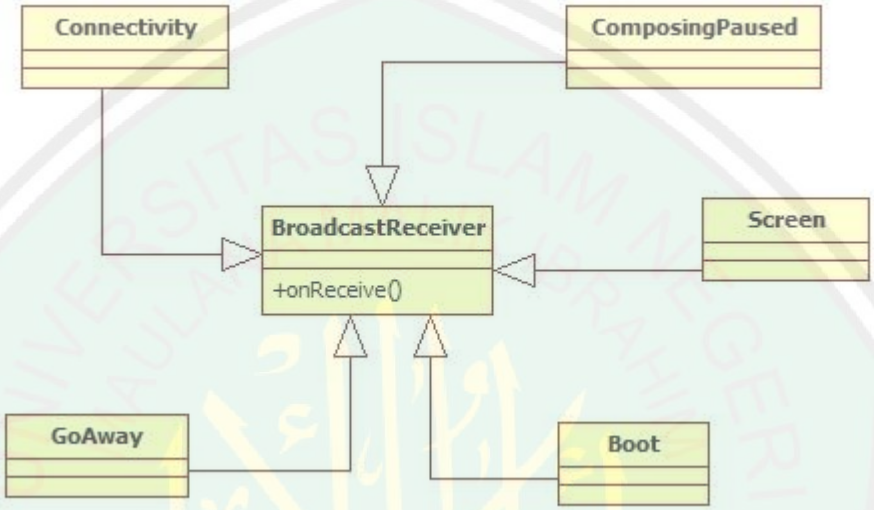


Gambar 3.24 Class diagram onPacket

Interface onPacket adalah kandidat kelas implementasi yang memiliki tanggung jawab terhadap setiap paket jaringan jabber yang masuk pada aplikasi *client* uin-messenger. Seperti paket presence ditangani oleh kelas PresenceManager dan paket message ditangani oleh kelas MessageManager. Pada rancangan sebuah *class* aplikasi yang dibangun disini, kelas manager secara umum bertanggung jawab terhadap implementasi spesifikasi XEP yang direpresentasikannya, seperti MUCExtensionManager sebagai implementasi XEP-0045 dan ChatStateExtensionManager dengan XEP-0085. Berikut adalah masing-masing penjelasan dari tiap kelas implementasi diatas.

1. MessageManager: merupakan *source class* yang akan bertanggung jawab terhadap chat pesan baik pesan dari user kontak maupun MUC.
2. PresenceManager: merupakan *source class* yang akan mengurusi presence kehadiran user kontak.
3. RosterManager: merupakan *source class* yang akan bertanggung jawab daftar user kontak yang pengguna miliki.
4. vCardExtensionManager: merupakan *source class* sebagai implementasi sebuah spesifikasi ekstensi vCard kontak.
5. TimeExtensionManager: merupakan *source class* sebagai implementasi spesifikasi ekstensi XEP-0202.
6. SSNExtensionManager: merupakan *source class* sebagai implementasi spesifikasi ekstensi XEP-0155.
7. PingExtensionManager: merupakan *source class* sebagai implementasi spesifikasi *ping request*.
8. MUCExtensionManager: merupakan *source class* sebagai implementasi spesifikasi ekstensi XEP-0045.
9. ChatStateExtensionManager: merupakan *source class* sebagai implementasi spesifikasi ekstensi XEP-0085.
10. CapabilitiesExtensionManager: merupakan *source class* sebagai implementasi informasi *entity capabilities*.
11. AvatarExtensionManager: merupakan *source class* dengan tanggung jawab untuk *manage* foto avatar pada aplikasi.
12. AttentionExtensionManager: merupakan *source class* sebagai implementasi spesifikasi ekstensi XEP-0224.

13. ReceiptExtensionMananger: merupakan *source class* sebagai implementasi spesifikasi ekstensi XEP-0184.

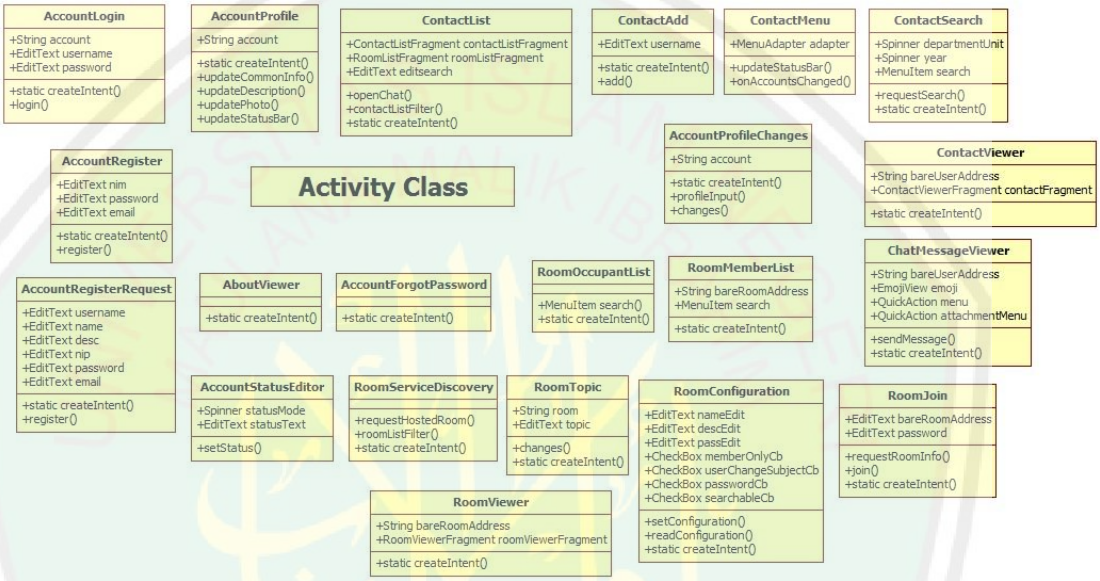


Gambar 3.25 Class diagram broadcast android

Class diagram ini menunjukkan respon aplikasi pada beberapa *event* yang ada pada *device* pengguna. Beberapa *event* tersebut diantaranya adalah booting OS device, perubahan status network device, perubahan *screen device* (screen ON/OFF) dan broadcast event dengan kondisi tertentu yang ditentukan oleh aplikasi (*go away status* dan *paused composing chat*). Berikut adalah penjelasan dari masing-masing respon aplikasi tersebut.

1. Connectivity: merupakan sebuah receiver untuk beberapa network event. Network event yang dimaksud seperti network available, suspend atau unavailable.
2. GoAway: merupakan sebuah receiver untuk membuat jadwal perubahan status menjadi away.
3. Boot: merupakan receiver untuk booting OS android.

4. Screen: merupakan sebuah receiver untuk event screen OFF dan ON.
5. ComposingPaused: merupakan sebuah receiver untuk membuat pengaturan berhenti (pause) saat user typing (saat chat session dimulai).



Gambar 3.26 Class diagram activity

Diagram kelas diatas menunjukkan semua activity *class* android yang ada pada aplikasi. Terdapat 23 *class* activity yang terdaftar pada manifest.xml aplikasi. Berikut adalah penjelasan masing-masing activity tersebut.

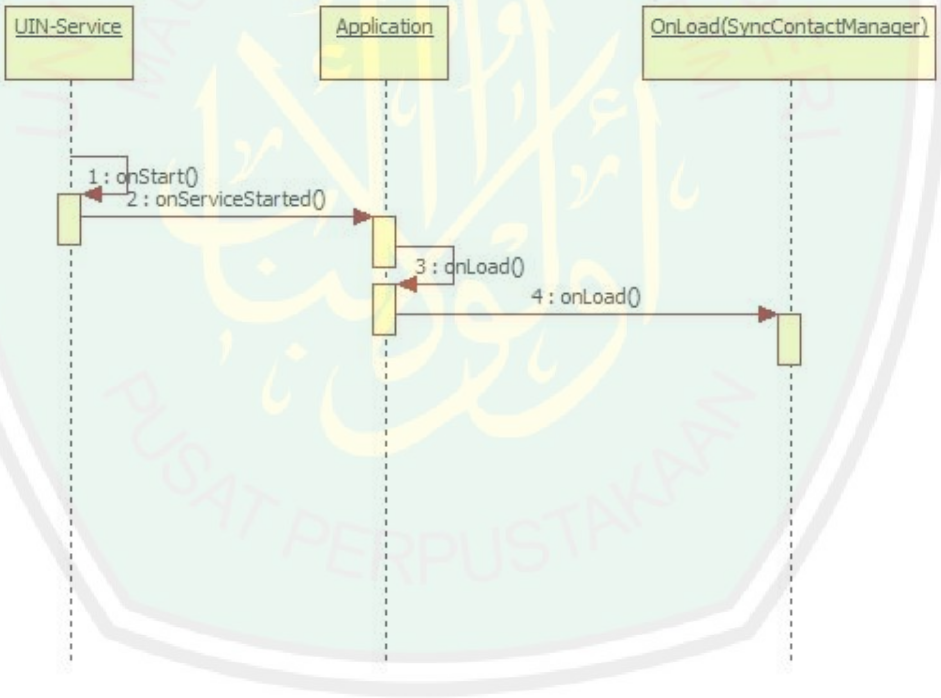
1. AccountLogin: Activity untuk *form* login pengguna
2. AccountRegister: Activity untuk *form* registrasi mahasiswa
3. AccountRegisterRequest: Activity untuk *form* registrasi dosen dan karyawan
4. AccountForgotPassword: Activity untuk *form* prosedur jika pengguna lupa password yang dia miliki
5. AccountProfile: Activity untuk display informasi akun pengguna

6. AccountProfileChanges: Activity untuk form perubahan informasi akun pengguna
7. AccountStatusEditor: Activity untuk form pengaturan status teks dan status mode pengguna
8. ContactList: Activity untuk display daftar kontak aplikasi dan juga room
9. ContactMenu: Berisi daftar menu aplikasi
10. ContactSearch: Activity untuk form pencarian kontak
11. ContactAdd: Activity untuk operasi menambah kontak (subscribe contact)
12. ContactViewer: Activity untuk display informasi vCard kontak
13. RoomJoin: Activity untuk operasi *join* room
14. RoomTopic: Activity untuk display dan perubahan topik sebuah room
15. RoomViewer: Activity untuk display informasi sebuah room
16. RoomServiceDiscovery: Activity untuk operasi pencarian room yang telah terdaftar pada sistem
17. RoomMemberList: Activity untuk *display* dan *form* yang berkaitan dengan daftar member sebuah room seperti *grant* admin sebuah room
18. RoomOccupantList: Activity untuk *display* daftar partisipan sebuah room
19. RoomConfiguration: Activity untuk *form* perubahan konfigurasi sebuah room oleh admin
20. RoomRegistrationMember: Activity untuk operasi permintaan menjadi member sebuah room
21. RosterList: Activity untuk *display* dan *form* untuk operasi yang berkaitan dengan daftar semua roster kontak seperti operasi *invite* kontak masuk room
22. ChatMessageViewer: Activity untuk display dan operasi chat message kontak dan room

23. AboutViewer: Activity untuk display informasi *about* aplikasi

3.3.2.4 Sequence Diagram

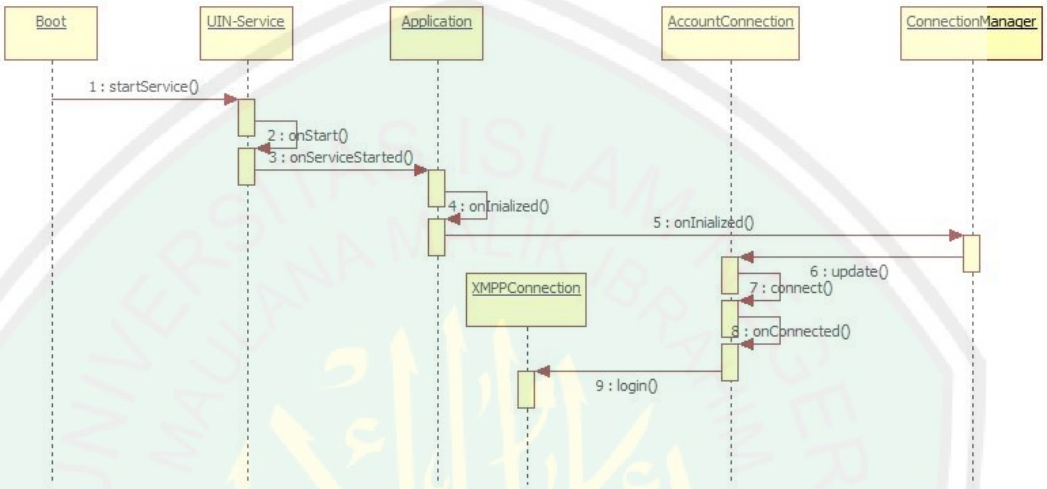
Sequence diagram digunakan untuk memodelkan aliran jalannya proses yang ditunjukkan dengan interaksi antar objek atau klas dan disusun berdasarkan urutan waktu. *Sequence diagram* disusun dengan mengambil acuan pada *use case* dan kelas yang telah diturunkan untuk membentuk fungsionalitas yang digambarkan pada *use case* tersebut.



Gambar 3.27 Sequence diagram onLoad

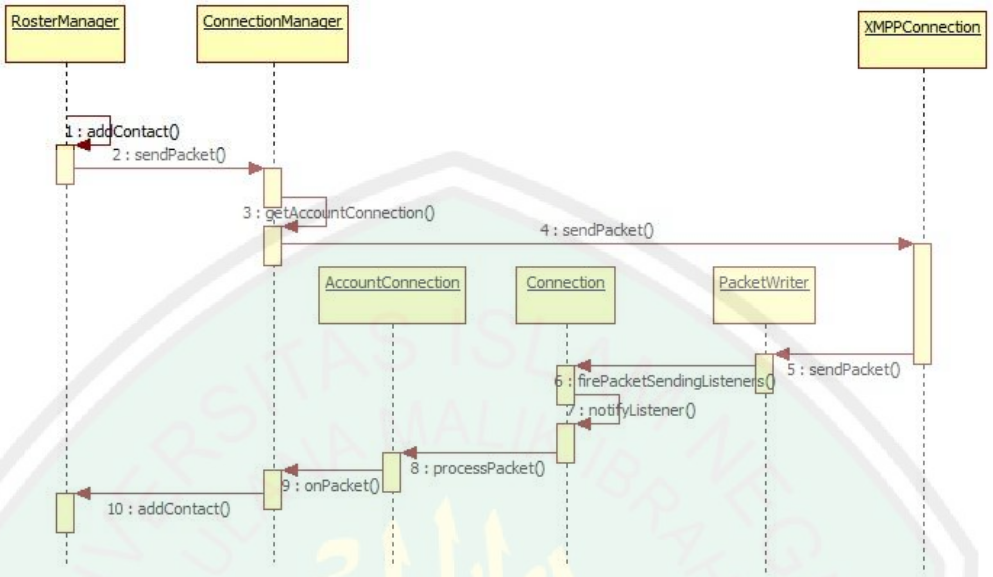
Sequence diagram diatas dapat menunjukkan sebuah *flow* bagaimana semua kontak yang telah tersimpan pada aplikasi akan terload. Sesuai dengan analisa kebutuhan sistem bahwa kontak aplikasi masuk juga pada kontak device android. Sebuah kelas yang bertanggung jawan untuk itu adalah kelas SyncContactManager. Load semua kontak pada

aplikasi akan dipanggil pada method onLoad() kelas Application tepat setelah service aplikasi dimulai onServiceStarted().



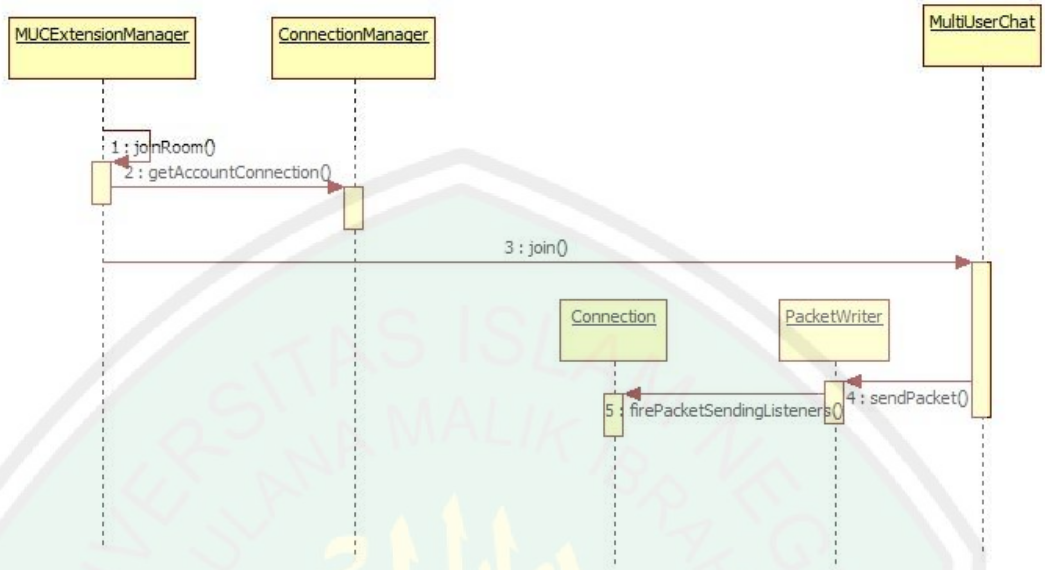
Gambar 3.28 Sequence diagram otentikasi user

Sequence diagram ini dapat menjelaskan aliran proses dan letak pemanggilan dari methode otentikasi pada aplikasi. Dimulai dengan proses booting OS, kemudian akan menjalankan sebuah service aplikasi UIN-Service. Terlihat pada diagram bahwa pemanggilan method login() dilakukan melalui onInitialized() tepat setelah service dimulai. Kelas ConnectionManager akan melakukan sebuah koneksi ke server terlebih dahulu connect() sebelum melakukan proses otentikasi pada onConnected().



Gambar 3.29 Sequence diagram subscribe user

Sequence diagram ini memaparkan beberapa kelas aplikasi yang nantinya terlibat dalam *subscribe user contact*. Terlihat kelas manager yang bertanggung jawab untuk *subcsribe* contact adalah kelas RosterManager. Pembuatan kontak baru melalui sebuah fungsi yaitu createContact(). Fungsi ini akan menyimpan kontak baru pada lokal database dan juga mengirim *subscribing* paket melalui sendPacket(). Paket yang dikirim merupakan paket permintaan untuk melakukan *subscribe* kepada user contact tujuan. Permintaan paket *subscribe* kemudian akan masuk melalui method onPacket() kelas RosterManager pada user *contact* tujuan. Kontak tujuan melakukan konfirmasi permintaan dan menyimpan kontak yang diterimannya pada database lokal serta juga disini mengajukan permintaan *subscribe* sebaliknya pada kontak asal.



Gambar 3.30 Sequence diagram multi user chat

MUCExtensionManager merupakan kelas aplikasi sebagai implementasi spesifikasi XEP-0045. Pada sequence diagram diatas menunjukkan aliran method untuk operasi join sebuah room. Semua operasi aplikasi terdapat pada kelas manager MUCExtensionManager. Untuk operasi join, kelas manager akan memanggil method join() pada kelas MultiUserChat.

3.4 Uji Kelayakan Aplikasi

Uji kelayakan aplikasi dilakukan untuk menguji apakah sistem pada aplikasi sudah berjalan sesuai dengan fungsinya. Dalam uji kelayakan aplikasi ini terdapat pertanyaan dalam bentuk kuisioner yang disesuaikan untuk menguji sistem pada aplikasi.

3.4.1 Uji Kelayakan Aplikasi Berdasarkan Kuisioner

Kuesioner dilakukan untuk menghitung hasil survei kelayakan aplikasi. Survei ditujukan kepada pengguna dengan mengisi pernyataan sesuai dengan pengamatan pada

aplikasi. Pengukuran kuesioner ini menggunakan skala Likert, dimana digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang. Dengan skala Likert, maka variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrument yang dapat berupa pernyataan atau pertanyaan (Sugiyono, 2010) Berikut pernyataan pada kuesioner:

a. Ketertarikan dengan Aplikasi
b. Tampilan Aplikasi
c. Kesesuaian desain warna
d. Kelengkapan fitur
e. Kemudahan penggunaan fitur
f. Penggunaan bahasa
g. Manfaat Aplikasi
h. Pengembangan Aplikasi
i. Publikasi Aplikasi pada masyarakat
j. Tanggapan pengguna secara keseluruhan

Adapun nilai dari pernyataan diasumsikan dengan huruf, maka keterangan sebagai berikut:

1. SB = Sangat Baik, diberi skor 5
2. B = Baik, diberi skor 4
3. C = Cukup, diberi skor 3
4. K = Buruk, diberi skor 2
5. SK = Sangat Buruk, diberi skor 1

Setelah hasil dari kuesioner dianalisis, maka akan dicari tingkat kelayakan produk yang ditentukan dari rata-rata prosentase jawaban responden, berikut tingkatan kelayakan produk:

1. Sangat layak = 81% - 100%
2. Layak = 61% - 80%
3. Biasa = 41% - 60%
4. Tidak layak = 21% - 40%
5. Sangat tidak layak = 1% - 20%

Berdasarkan tingkatan tersebut, maka akan diperoleh hasil yang nanti akan digunakan dalam menetapkan tingkat kelayakan produk dari hasil kuesioner.

Uji kelayakan aplikasi di atas diharapkan dapat membantu menentukan apakah aplikasi ini sudah berjalan dengan benar sesuai fungsinya atau tidak sebelum digunakan untuk kebutuhan sehari-hari. Hasil dari uji kelayakan aplikasi di atas akan dijelaskan secara lengkap pada bab berikutnya.

BAB IV

IMPLEMENTASI HASIL DAN PEMBAHASAN

Implementasi merupakan proses pembangunan komponen-komponen pokok sebuah sistem berdasarkan desain yang sudah dibuat. Implementasi sistem juga merupakan sebuah proses pembuatan dan penerapan sistem secara utuh baik dari sisi perangkat keras maupun perangkat lunaknya. Pada bab berikut akan dipaparkan implementasi ruang lingkup sistem baik dari perangkat keras maupun perangkat lunak yang dibutuhkan aplikasi, kemudian pemaparan implementasi antar muka aplikasi dan terakhir juga pembahasan sistem aplikasi instant messenger.

4.1 Ruang Lingkup Perangkat Keras

Dalam proses pengembangan aplikasi permainan ini menggunakan komputer dengan spesifikasi perangkat keras sebagai berikut.

1. CPU Processor Intel® Core™ i5-520M 2.40GHz
2. Memory Storage 500 GB
3. RAM memory 4 GB
4. Graphic Card VGA ATI Mobility Radeon HD 5650
5. Memory VGA 2733 MB

Dalam proses instalasi dan pengujian, perangkat yang digunakan adalah smartphone android dengan spesifikasi perangkat keras sebagai berikut.

1. Memory storage 158 MB
2. RAM memory 278 MB
3. CPU Processor 800 MHz ARM 11 GPU Adreno 200

4.2 Ruang Lingkup Perangkat Lunak

Dalam proses pengembangan aplikasi messenger ini menggunakan komputer dengan spesifikasi perangkat lunak sebagai berikut.

1. Sistem Operasi Linux Mint 10
2. Bahasa pemrograman Java dan XML
3. IDE (Integrated Development Environment) Eclipse Hellios
4. Control system Mercurial
5. Android Development Tools (ADT)
6. Gimp Image Editor
7. Pencil Prototype
8. StarUML
9. ejabberd server v13
10. Erlang VM
11. Pidgin Internet Messenger
12. Spark

Dalam proses instalasi dan pengujian perangkat yang digunakan adalah smartphone android dengan spesifikasi perangkat lunak sebagai berikut.

1. Android OS 2.3 (Gingerbread)

4.3 Implementasi Antarmuka Sistem

Antarmuka pokok aplikasi instant messanger adalah halaman daftar kontak, halaman daftar room dan halaman chat baik itu chat user maupun chat room. Ketiga halaman ini yang akan sering diakses oleh pengguna. Implementasi antarmuka pada aplikasi ini berfokus pada kemudahan akses pengguna untuk ketiga halaman pokok tersebut.

4.3.1 Landing page

Halaman landing page berikut ini akan muncul ketika aplikasi pertama kali dijalankan (*launching application*).

Gambar 4.1. Launching application

4.3.2 Login, registrasi akun dan halaman forgot password

Pengguna memasukkan username dan password pada halaman login. Tapi sebelum itu, pengguna harus registrasi dan aktivasi akun yang dia miliki pada halaman registrasi akun. Registrasi terbagi menjadi dua, yaitu registrasi untuk mahasiswa dan registrasi untuk dosen dan karyawan.

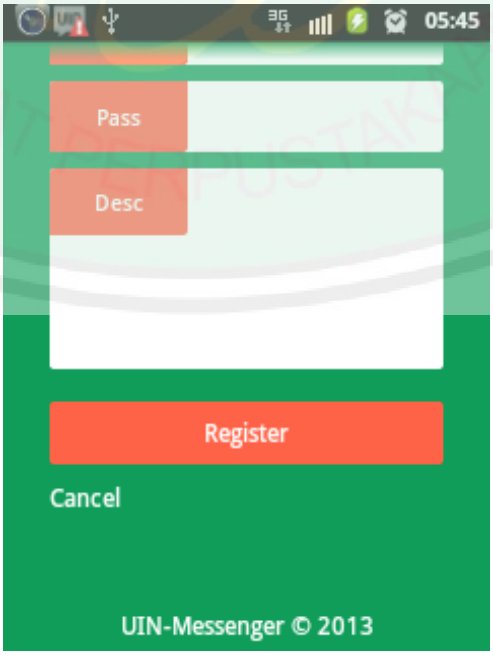
Gambar 4.2. Halaman login

Gambar 4.3. Halaman register

Halaman login (Gambar 4.2) muncul hanya sekali yaitu setelah aplikasi selesai terinstal. Pada halaman ini *field* NIM diisi dengan NIM yang dimiliki mahasiswa dan diisi dengan username bagi dosen maupun staf. Gambar 4.3 adalah halaman registrasi sebuah akun mahasiswa. *Field* masukan yang muncul

pada halaman ini adalah NIM mahasiswa, email dan password. Pengguna memberikan email pada masukkan untuk pemberitahuan sebagai hasil aktivasi.

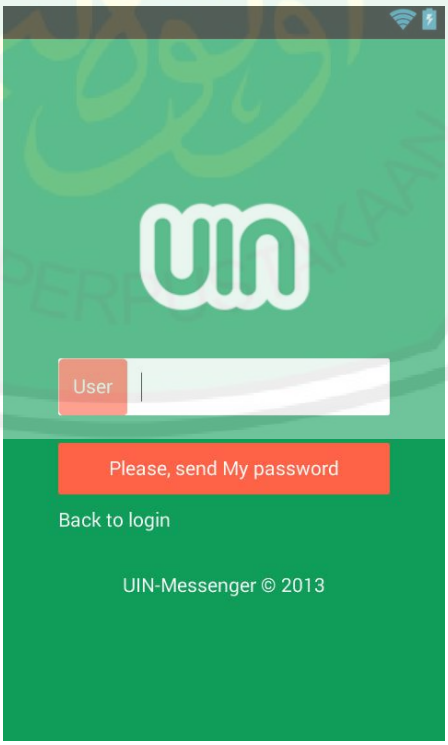
Gambar 4.4. Halaman register dosen 1



Gambar 4.5. Halaman register dosen 2

Gambar 4.4 dan gambar 4.5 merupakan tampilan halaman registrasi untuk dosen dan staf karyawan. Pada bagian ini user dapat memilih sendiri username yang akan user pakai. Beberapa *field* masukan lain yang harus diisi oleh user yaitu nama, NIP, email, password dan deskripsi singkat mengenai tugas, posisi atau pekerjaan di kampus. Terdapat dua button yaitu *register* untuk submit proses registrasi dan *cancel* untuk membatalkan registrasi dan kembali ke halaman login.

Jika pengguna aplikasi lupa akan password yang dia miliki, pengguna dapat mengakses halaman *forgot password* untuk mekanisme pergantian password. Sistem akan mengirim sebuah link yang akan mengarah ke halaman pergantian password via mail.



Gambar 4.6. Halaman lupa password

4.3.3 Contact list

Halaman *contact* berikut adalah halaman utama aplikasi. Halaman ini berisi daftar semua kontak yang user miliki (*roster*). Informasi-informasi penting pada kontak terlihat disini, seperti foto, nama lengkap, isi percakapan terakhir, status user, serta status mode kontak.

Gambar 4.7. Halaman kontak

Search icon pada bar aplikasi akan memfilter kontak dengan karakter yang pengguna berikan. Beberapa status mode ikon yang muncul disini memiliki keterangan berikut ini:

●	Available	User online dan aktif (mengakses aplikasi)

	Away	User online atau sudah sejak 15 menit lalu tidak mengakses aplikasi
	Do not disturb	User memilih status ini agar user sementara waktu jangan diganggu
	Offline	User sedang tidak dalam terkoneksi
	Unsubscribe	User belum menyetujui subscribe yang pengguna lakukan
	Shared roster	Kontak user yang otomatis dimiliki semua user

Tabel 4.1 Mode ikon status

4.3.4 Room chat

Halaman ini berisi daftar semua room yang pengguna miliki. Setiap item room disini, berisi informasi seperti nama room, topik yang sedang didiskusikan dan status user terhadap room.

Gambar 4.8. Room chat

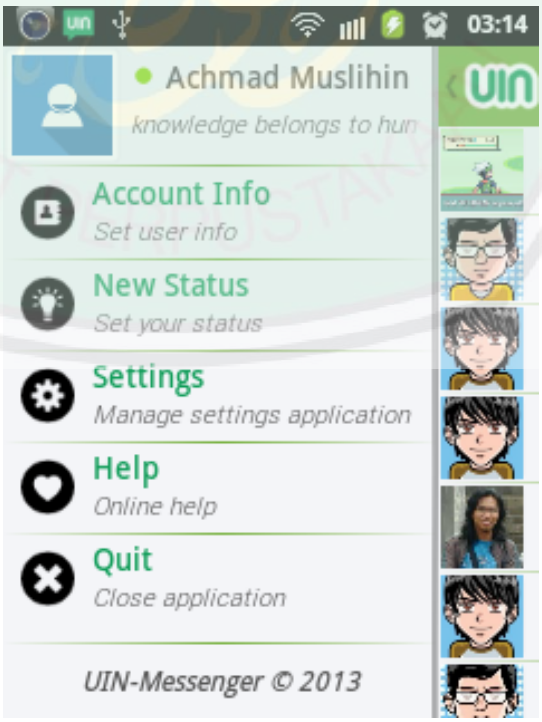
Search icon pada bar aplikasi dapat memfilter room berdasarkan karakter yang pengguna masukkan. Beberapa *icon* status pengguna terhadap room diterangkan pada tabel berikut ini:

	Administrator	Pengguna sebagai administrator pada room ini
	Member/Partisipator	Pengguna sebagai partisipator pada room ini
	Offline	Pengguna sedang tidak dalam keadaan terkoneksi

Tabel 4.2 Status pengguna room

4.3.5 Main menu

Halaman ini berisi menu-menu utama aplikasi. Pada bagian atas halaman terdapat informasi pengguna yaitu foto, nama lengkap, status mode dan status teks yang telah dia masukkan.



Gambar 4.9. Menu utama

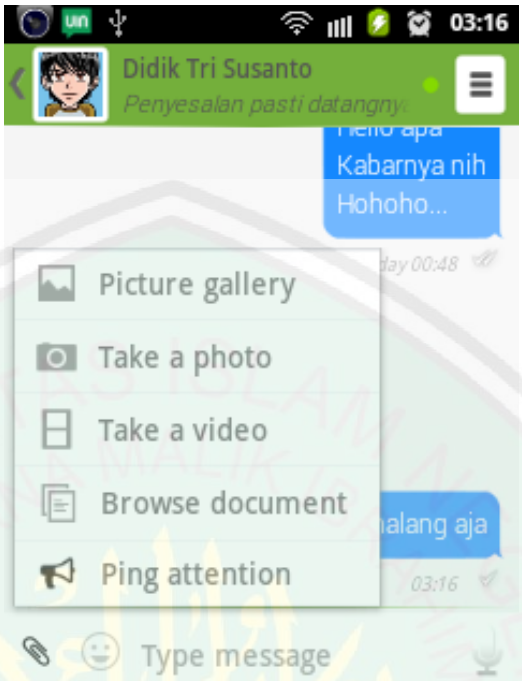
Menu utama aplikasi terdiri dari (1) menu *Account Info* yang akan mengantarkan pengguna menuju informasi profil yang pengguna miliki, (2) *New Status* untuk mengubah status teks dan status mode pengguna, (3) *Settings* berisi semua pengaturan aplikasi instant messanger, (4) *Help* yang akan mengantarkan pengguna ke situs dokumentasi aplikasi, dan (5) *Quit* untuk menutup aplikasi (bukan *logout*).

4.3.6 Halaman chat kontak

Halaman berikut adalah halaman percakapan dengan kontak pengguna. Pada bagian atas, terdapat informasi terhadap user partner. Beberapa menu percakapan terletak di sudut kanan atas. Bagian bawah halaman, berisi menu attachment, emoji dan masukan teks pesan. Isi percakapan ditampilkan dalam list dengan format atau layout disesuaikan dengan isi atau konten pesan.

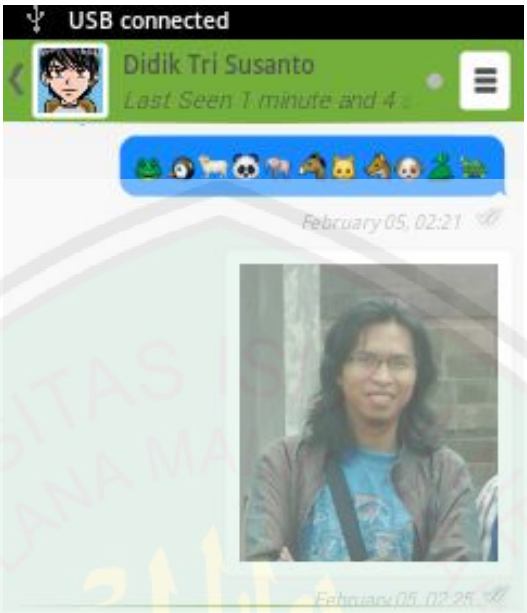
Gambar 4.10. Halaman chat 1

Menu samping kanan atas (bar aplikasi) terdiri dari tiga menu item yaitu menu *View Contact*, *Edit Alias Contact* dan *Close Chat*.

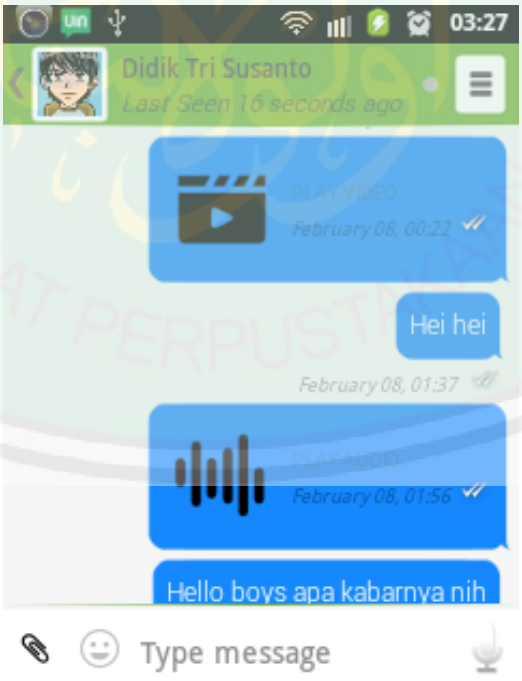


Gambar 4.11. Menu attachment

Attachment menu pada samping kiri bawah terdiri dari 5 menu item attachment. Kelima menu item ini adalah *attachment picture gallery* (FR-1.6), *attachment take a photo* (FR-1.2), *attachment take a video* (FR-1.4), *attachment browse document* (FR-1.5), dan terakhir *ping attention* (NF-1). Icon samping kanan bawah disini memiliki dua action yang berbeda, pertama icon bergambar mikrofon untuk merekam *voice note* (FR-1.3) dan kedua *icon* bergambar panah untuk mengirim pesan teks yang telah dimasukkan (FR-1).



Gambar 4.12. Halaman chat 2



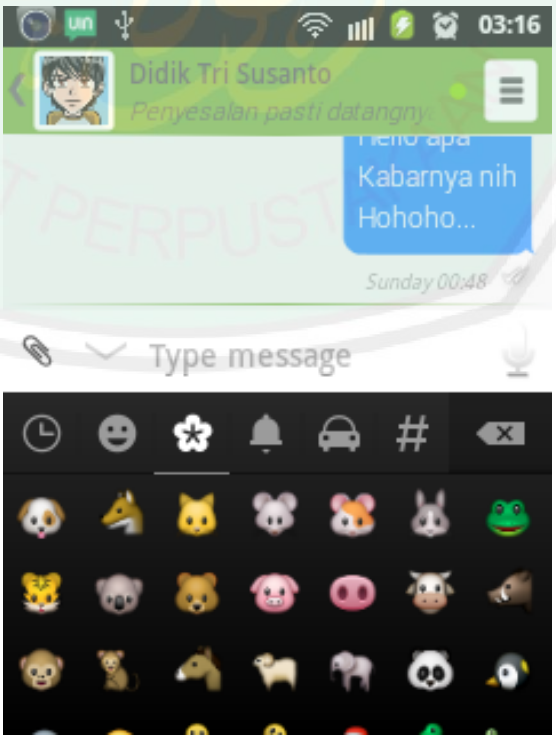
Gambar 4.13. Halaman chat 3

Pada gambar dan gambar merupakan tampilan layout pesan dengan konten image, video dan voice note ditampilkan.

	Delivered	Pesan terkirim ke user tujuan
	Sent	Pesan tersimpan pada server karena user tujuan offline
	Save	Pesan tersimpan pada lokal aplikasi karena pengguna offline
	Error	Terjadi kegagalan pada pengiriman pesan

Tabel 4.3. Status ikon pesan

Ikon *smile* akan berubah menjadi sebuah ikon panah bawah ketika ikon ini ditap. Hal ini akan membuka panel yang berisi daftar emoji yang disediakan aplikasi dan ketika ikon panah ditap, aplikasi akan menutup panel emoji ini.



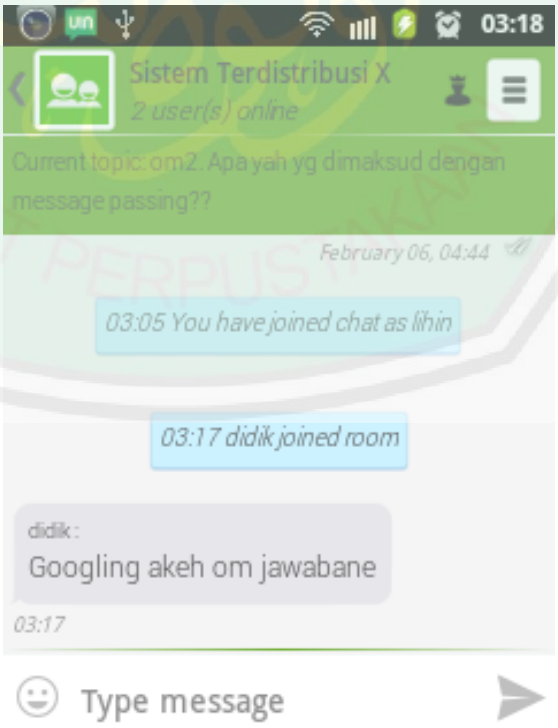
Gambar 4.14. Panel emoji

	Mengirim pesan teks
	Membuka emoji panel
	Membuka menu *attachment*
	Merekan *voice note*

Tabel 4.4. Ikon button halaman chat

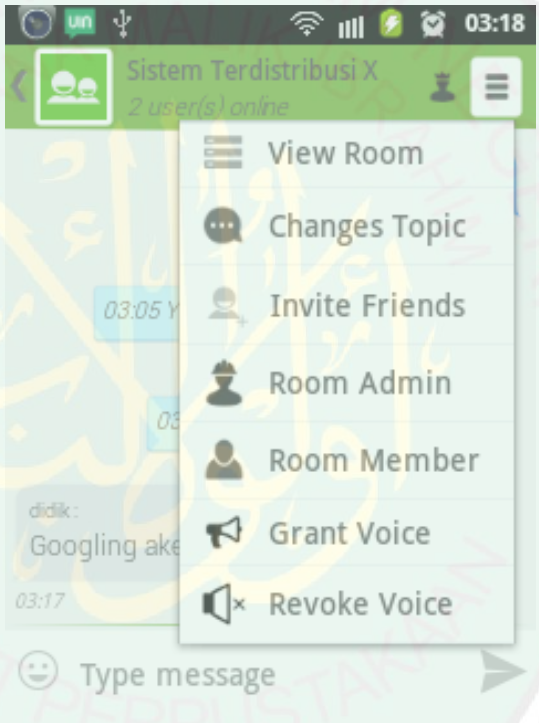
4.3.7 Halaman chat room

Halaman berikut adalah halaman percakapan room. Informasi room chat terletak di bagian atas halaman. Kemudian menu-menu room chat di sudut kanan atas. Bagian bawah terdapat button emoji dan masukkan teks pesan.



Gambar 4.15. Halaman chat room

Menu samping kanan atas pada *bar* aplikasi merupakan menu yang dimiliki pengguna terhadap room. Menu yang muncul tergantung pada status pengguna, apakah dia seorang administrator atau member. Administrator room akan memiliki beberapa menu item seperti yang tampil pada gambar, sedangkan untuk member menu yang tampil tampak pada gambar.

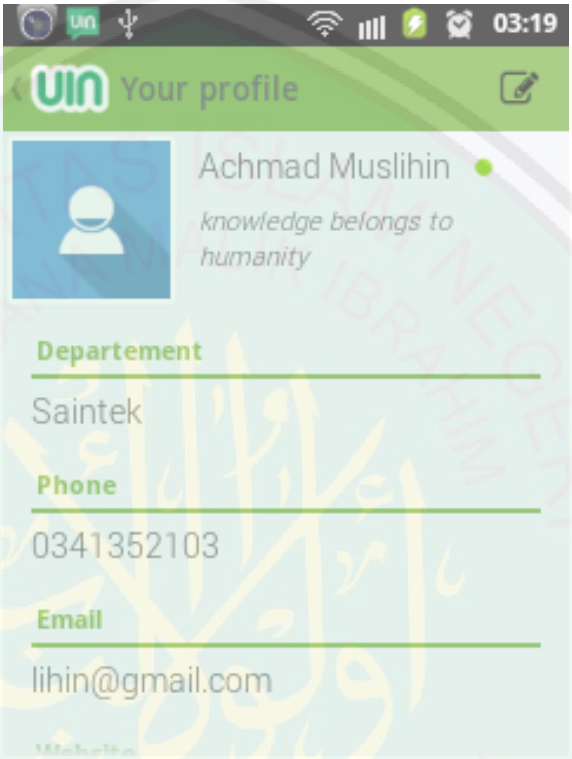


Gambar 4.16. Menu item chat room

4.3.8 Profil dan form perubahan profil akun

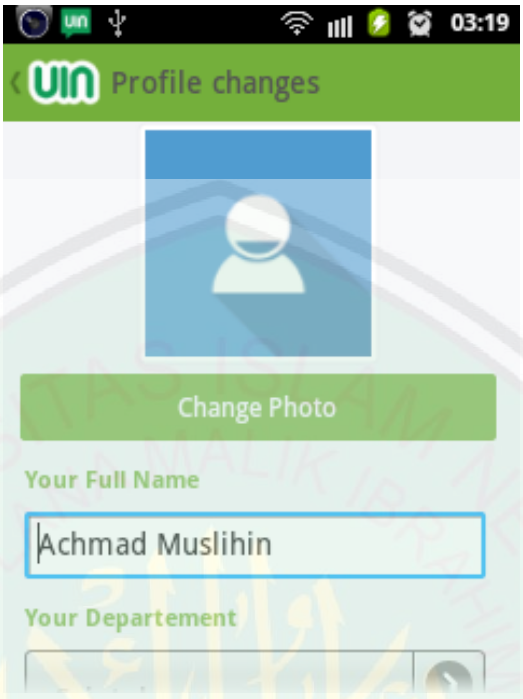
Profil pengguna memuat beberapa informasi penting tentang pengguna bersangkutan. Informasi ini diinputkan oleh pengguna pada form perubahan profil. Informasi yang dapat dimasukkan adalah nama lengkap, foto, jurusan, telpon, email, website, dan deskripsi secara singkat mengenai user. Informasi ini

bersifat opsional artinya pengguna boleh saja tidak menginputkan salah satu dari informasi ini.

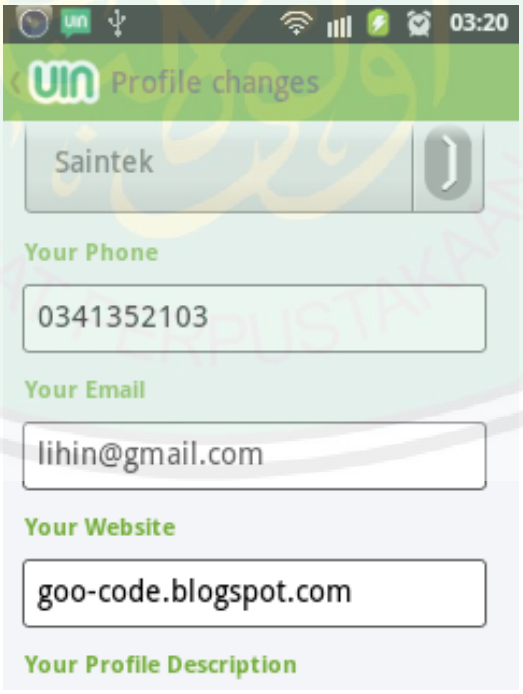


Gambar 4.17. Profil pengguna

Pada samping kanan atas pada halaman profil terdapat sebuah ikon yang akan mengantarkan pengguna pada halaman edit profil. Tampilan edit profil seperti yang terlihat pada gambar dan gambar. Semua informasi yang ada pada halaman profil dapat diubah pengguna melalui halaman ini.



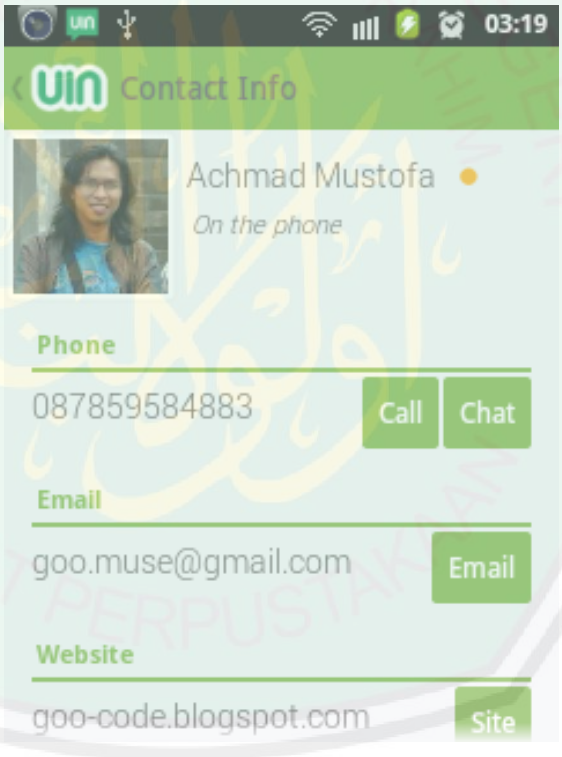
Gambar 4.18. Halaman edit profil 1



Gambar 4.19. Halaman edit profil 2

4.3.9 Profil kontak user

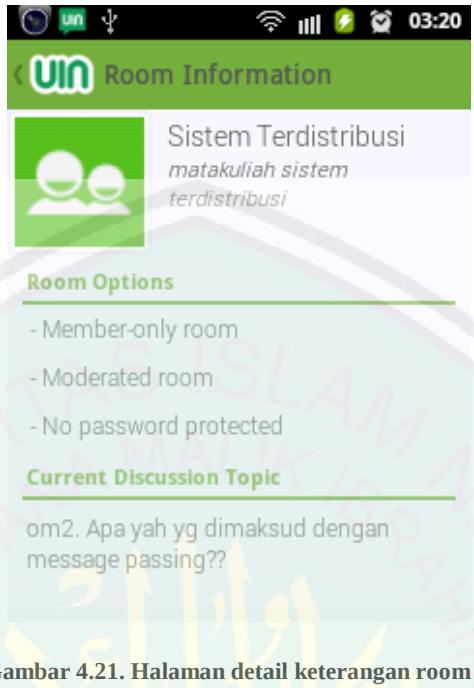
Pengguna dapat melihat informasi user kontak yang dia miliki pada halaman ini. Terdapat tombol *Call* disamping kanan nomer telpon user untuk memanggil user via telpon. Selain itu terdapat tombol *Email* untuk mengirim pesan ke user via mail dan juga tombol *Site* untuk membuka *browser* yang akan langsung menuju website atau blog user kontak.



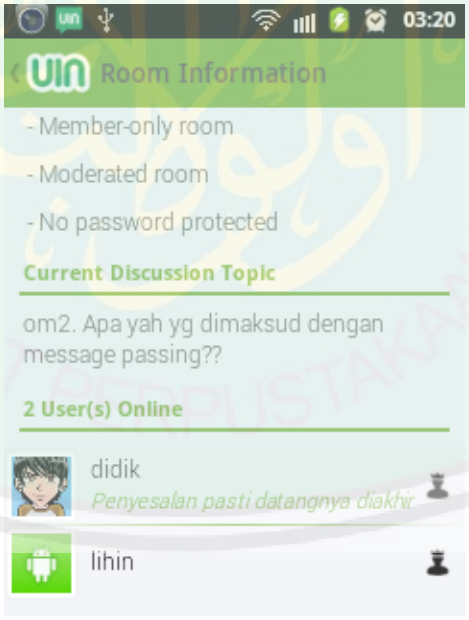
Gambar 4.20. Halaman informasi user

4.3.10 Profil room chat

Pengguna melihat beberapa informasi room chat melalui halaman ini. Informasi itu seperti deskripsi singkat mengenai room, topik yang sedang didiskusikan, room opsi dan siapa saja yang sedang online sekarang.



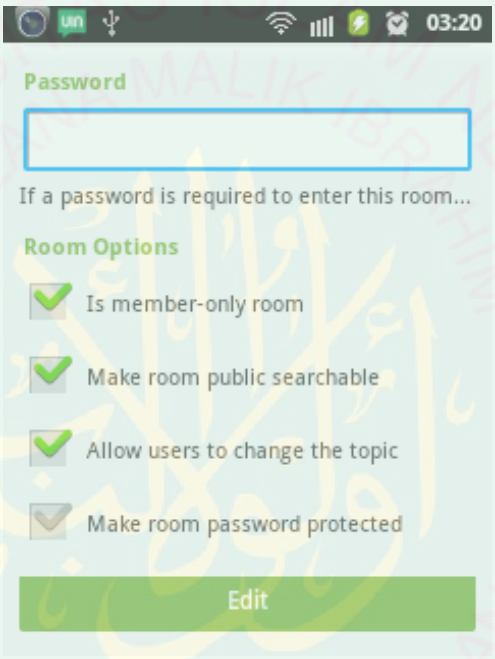
Gambar 4.21. Halaman detail keterangan room 1



Gambar 4.22. Halaman detail keterangan room 2

4.3.11 Form perubahan profil room chat

Halaman ini hanya dapat diakses oleh administrator room *chat*. Informasi seperti nama, deskripsi dan beberapa room opsi dapat diubah melalui halaman ini oleh administrator.



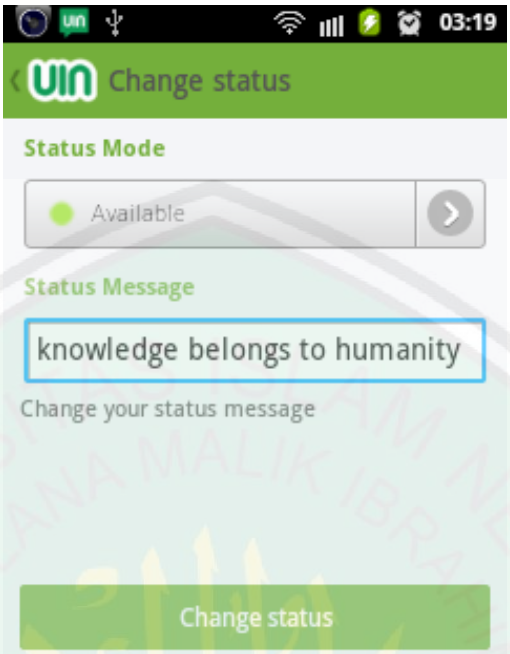
Gambar 4.23. Halaman edit detail room 1

Gambar 4.24. Halaman edit detail 2

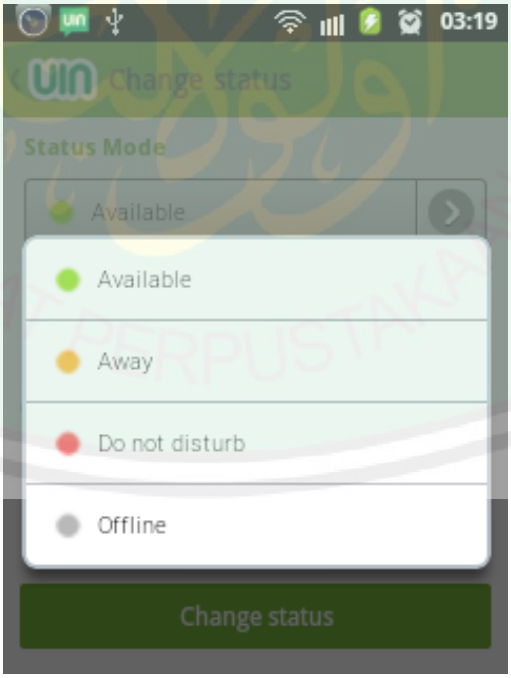
Administrator dapat mengubah beberapa informasi room seperti nama, deskripsi room dan memberikan password sebagai proteksi calon member untuk bergabung. Untuk memberikan password room, maka administrator juga harus menyetujui *"protected password"* pada pengaturan room *(room options)*. Pengaturan room yang lain seperti apakah room termasuk room private *(is member-only room*), room muncul pada halaman discovery *(searchable)*, apakah semua partisipator dapat mengubah topik room.

4.3.12 Form perubahan status pengguna

Pengguna aplikasi dapat mengubah status yang dimilikinya melalui form yang ada pada halaman ini. Terdapat dua kontrol masukkan, yaitu status teks dan status mode dalam bentuk opsi radio.



Gambar 4.25. Halaman edit status pengguna

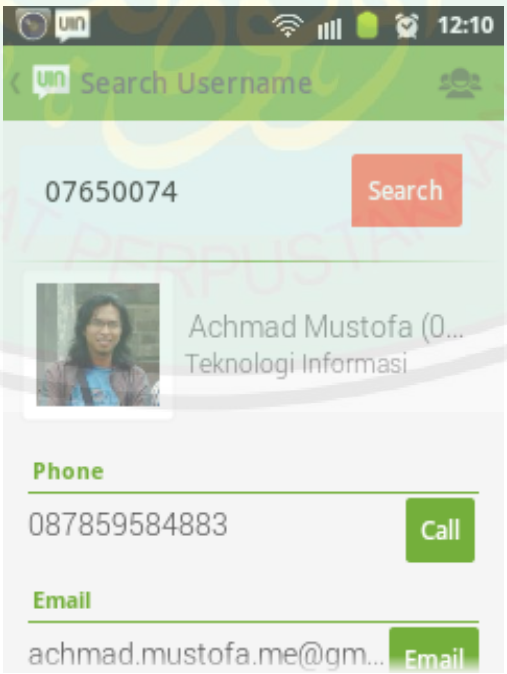


Gambar 4.26. Status mode pengguna

Opsi yang tersedia pada status mode adalah status *available*, *away*, *status do not distrub*, dan status *offline*. Beberapa opsi status mode ini ditampikan dalam bentuk popup seperti ditunjukkan pada gambar.

4.3.13 Halaman pencarian user

Pengguna dapat mencari user lain melalui halaman ini. Terdapat dua versi pencarian, yaitu pencarian user berdasarkan username(NIM) dan juga pencarian kolektif berdasarkan beberapa opsi yang ada. Pada pencarian user secara kolektif, *filter* opsi yang disediakan hanya pada opsi jurusan dan tahun angkatan user. Hasil yang dikeluarkan, terdapat sebuah *label* pada samping kanan untuk pemberitahuan bahwa pengguna telah *subscribe* user.



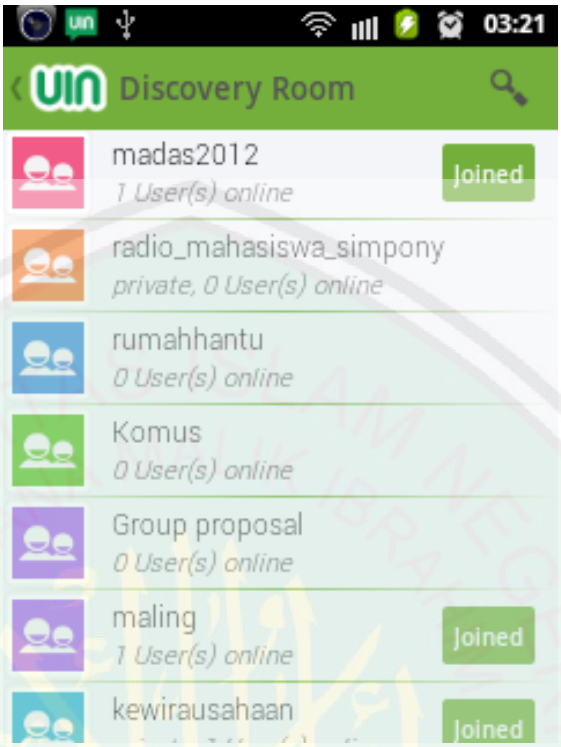
Gambar 4.27 Halaman pencarian user 1

Gambar 4.28 Halaman pencarian user 2

Pada setiap user item terdapat informasi foto user, nama lengkap dan alamat email yang user miliki. Pengguna melakukan filter pencarian terhadap user hanya pada kontrol jurusan yang user miliki. Konten user berasal dari server sehingga pengguna harus dalam keadaan online untuk mengakses halaman ini. Ikon search pada samping kanan atas, membantu pengguna untuk filter karakter terhadap hasil yang muncul.

4.3.14 Discovery room chat

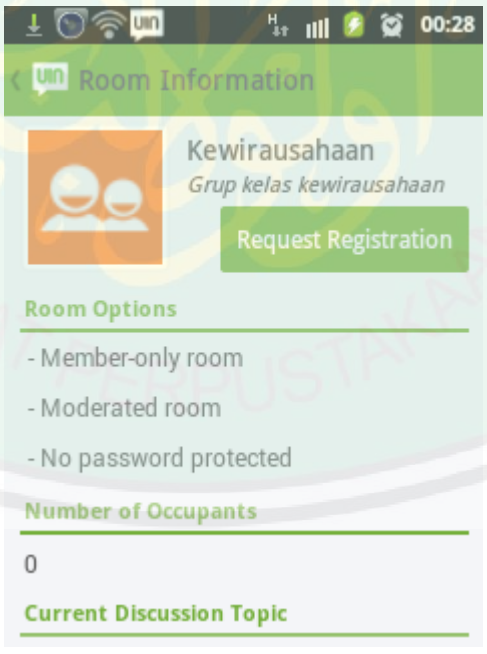
Room chat yang sudah teregistrasi pada server akan tampak pada halaman *discovery room*, tapi hanya room chat yang memiliki opsi *searchable* saja yang akan tampak. Pengguna dapat mencari dan melihat informasi mengenai room chat sebelum pengguna setuju untuk ikut bergabung didalamnya.



Gambar 4.29 Halaman discovery room

Pada setiap item *discrovery room* terdapat *label* disamping kanan untuk memberitahukan pengguna bahwa pengguna telah mengikuti room. Sedangkan untuk dapat mengikuti atau join sebuah room, disediakan sebuah button join pada halaman detail room. Baik konten pada *room discovery* maupun detail room diambil langsung dari ejabberd server sehingga pengguna harus dalam keadaan *online* untuk mengakses kedua halaman ini.

Gambar 4.30 Halaman detail discovery room 1



Gambar 4.31 Halaman detail discovery room 2

4.3.15 Room Topik

Halaman berikut adalah form untuk mengubah konten topik chat room. Topik room akan selalu terlihat di bagian atas halaman chat dari room. Semua member room dapat mengakses form halaman ini, semua member dapat mengganti topik room bersangkutan.

Gambar 4.32 Room topik 1

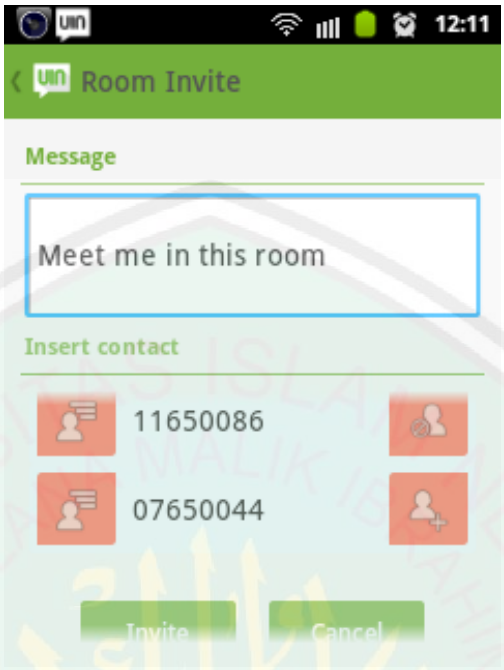
Gambar 4.33 Halaman edit topik room

Topik akan terlihat diposisinya saat pengguna masuk ke halaman chat *room*, dan selama kurang lebih 20 detik kemudian, topik akan hilang disembunyikan pada halaman chat. Ketika terdapat user lain mengubah topik maka kemudian topik akan kembali muncul lagi dan disembunyikan lagi pada 20 detik selanjutnya. Detail topik kemudian juga dapat dilihat dari halaman berikut.

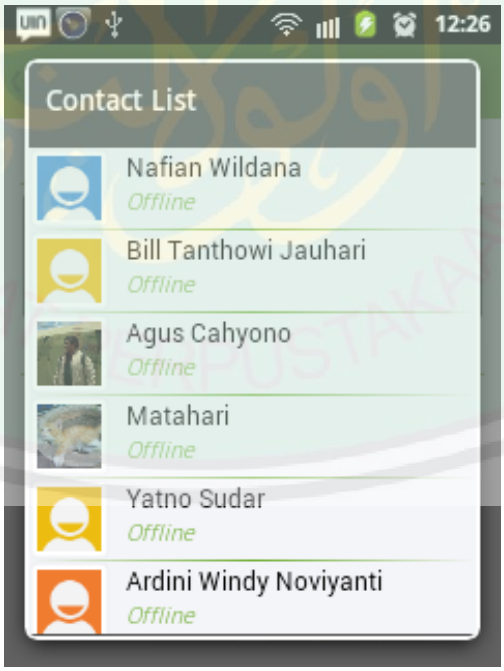
Gambar 4.34 Halaman topik room

4.3.16 *Invite* user

Pengguna dapat mengundang user lain untuk bergabung room chat pada halaman ini. Terdapat dua versi halaman, yaitu pertama halaman *invite* dengan *control* masukan *text* username dan yang kedua halaman *invite* dengan *control* daftar *contact* user yang pengguna miliki. Pada halaman pertama, pengguna bebas memasukkan username yang walaupun tidak terdapat pada daftar kontak. Pada setiap masukan *text* juga dapat menampilkan daftar kontak dan kemudian memilih siapa yang akan dia invite dari daftar tersebut. Pembuatan versi ini dimaksudkan untuk memudahkan pengguna untuk *invite* teman-temannya.



Gambar 4.35 Halaman invite user 1



Gambar 4.36 Halaman invite user 2

Daftar user item yang muncul pada halaman versi kedua ini adalah semua user kontak pengguna. Disini pengguna dapat memilih beberapa user sekaligus

dengan mencentang *checkbox* dari setiap item. Sebelum mengirimkannya, pengguna dapat memasukkan pesan teks kepada user *invite*.

Gambar 4.37 Halaman invite friend

Terdapat informasi nama dan deskripsi singkat room pada bagian atas halaman. Search icon pada bar aplikasi akan memfilter daftar kontak yang muncul sesuai dengan karakter yang diketikkan. Setelah semua beres, pengguna aplikasi dapat men-*submit invite* user dengan menekan *check icon* yang juga pada bagian bar aplikasi. Notifikasi kemudian akan muncul pada masing-masing user *invite* seperti yang akan tampak pada halaman notifikasi *invite* diatas.

4.3.17 *Grant* dan *kick* member room

Halaman ini hanya dapat diakses oleh administrator room. Administrator dapat memasukkan user untuk bergabung dan juga mengeluarkan member room melalui halaman ini.

Gambar 4.38 Halaman kick member

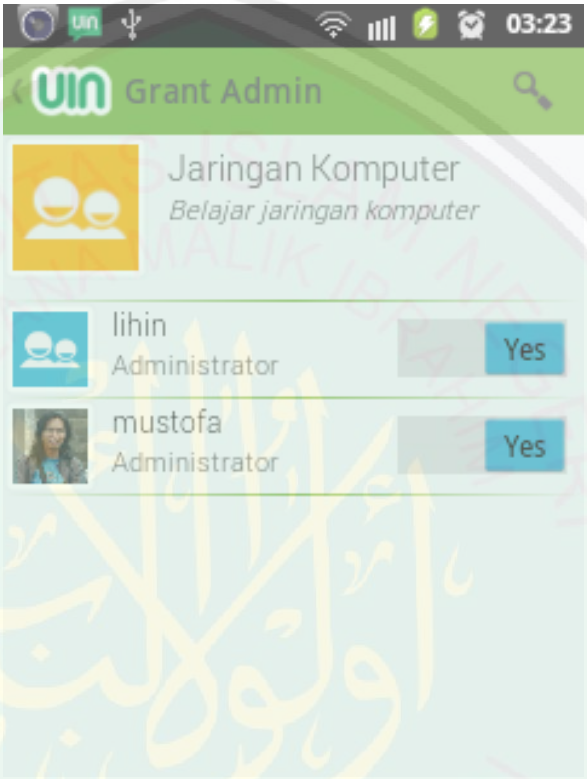
Gambar 4.39 Halaman membership room

Gambar 4.39 merupakan tampilan *membership* room. Daftar user yang muncul merupakan semua user kontak yang pengguna (administrator) miliki. Samping kanan setiap user terdapat checkbox berguna seleksi pengguna untuk menjadi member room. Sedangkan gambar adalah tampilan untuk mengeluarkan member dari sebuah room. Daftar item user yang muncul disini adalah semua member yang telah bergabung pada room.

4.3.18 *Grant* dan *revoke* administrator

Halaman ini hanya dapat diakses oleh administrator room. Administrator room dapat memberikan ijin member sebagai administrator pada room tersebut melalui halaman ini. User yang muncul merupakan semua member yang sudah

tergabung pada room. Pada samping kanan member terdapat tombol *On/Off* untuk memberikan ijin atau mencabut ijin member untuk menjadi administrator.

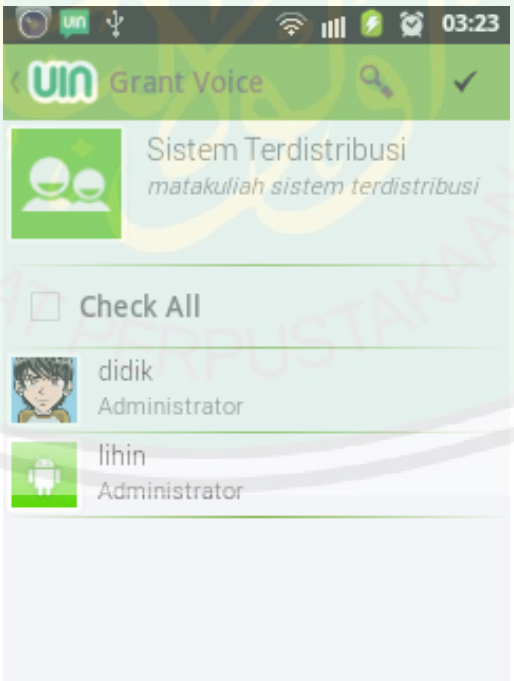


Gambar 4.40 Halaman grant admin room

4.3.19 *Grant* dan *revoke* voice room

Halaman ini hanya dapat diakses oleh administrator room. Administrator dapat mencabut hak berbicara (mengirimkan pesan baru) pada room chat melalui halaman ini. User item yang muncul pada halaman ini adalah semua member yang sudah tergabung pada room. Terdapat sebuah checkbox disamping kanan user item untuk menyeleksi user.

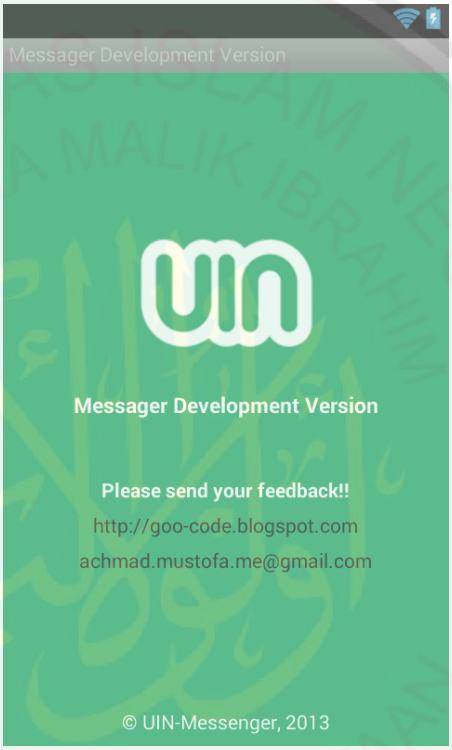
Gambar 4.41 Halaman revoke voice



Gambar 4.42 Halaman grant voice

4.3.20 *About*

Berisi informasi singkat mengenai aplikasi dan himbauan untuk mengirim feedback pada alamat email yang tertera.



Gambar 4.43 Halaman about

4.3.21 *Persistent Notification*

Notifikasi ini akan selalu muncul pada bar device android. Hal ini dapat memberitahukan pengguna status aplikasi, seperti apakah aplikasi sedang berjalan, aplikasi terkoneksi pada server dan aplikasi sedang tidak terkoneksi server. Secara teknis notifikasi ini merupakan foreground service aplikasi messenger. Ada beberapa icon yang mewakili status aplikasi, diantaranya adalah:

	Aplikasi dalam keadaan terkoneksi (Online)
	Aplikasi tidak dalam keadaan terkoneksi (Offline)
	Terdapat user yang sedang melakukan *request subscribe* pada pengguna
	Pemberitahuan terdapat pesan baru yang masuk
	Aplikasi gagal terautentikasi

Tabel 4.5 Status ikon notifikasi service

4.4 Pembahasan Sistem

Pada bab ini akan dipaparkan implementasi sistem berdasarkan rancangan program. Implementasi sistem terdiri dari dua bagian, implementasi server ejabberd dan impelementasi aplikasi instant messaging. Berikut adalah paparan implementasi dari perangkat lunak yang telah di bangun.

4.4.1 Instalasi dan Konfigurasi Server

Pada implementasinya, server yang digunakan adalah ejabberd. Skalabilitas, reliabilitas dan modular server yang membuat ejabberd sangat cocok untuk aplikasi instant messaging. Pada pembahasan bagian ini, akan dijelaskan instalasi ejabberd dan juga instalasi serta konfigurasi beberapa modul XEP yang dibutuhkan sistem aplikasi.

4.4.1.1 Instalasi ejabberd server

Sebelum melakukan instalasi ejabberd, dibutuhkan beberapa paket yang harus terinstall terlebih dahulu. Tanpa program ini, ejabberd tidak akan berjalan dengan baik atau tidak akan berjalan sama sekali. Dalam implementasi sistem instant messaging ini, ejabberd diinstal menggunakan Sistem Operasi Linux Mint 10. Paket yang dibutuhkan antara lain :

1. GNU Make
2. GCC
3. libexpat 1.95
4. Erlang/OTP R16B
5. Web server apache
6. MySQL dan PHP 5.5.1

Setelah semua *dependencies* yang dibutuhkan terinstall, maka kemudian ejabberd dapat diinstall melalui beberapa *shell command* berikut ini:

```
tar xzvf ejabberd-R16B.tar.gz
cd ejabberd-R16B
./configure
make
make install
```

Shell command instalasi ejabberd

4.4.1.2 Instalasi dan konfigurasi modul ejabberd

Standar XEP kebutuhan sistem instant messenger terdapat pada bab perancangan sistem. Dari semua XEP standar yang dibutuhkan oleh aplikasi, ejabberd menyediakan beberapa modul yang dapat dipakai. Di antaranya adalah:

mod_disco	Spesifikasi untuk memperoleh informasi feature setiap entitas jaringan jabber
mod_caps	Spesifikasi untuk memperoleh informasi capabilities pada setiap entitas
mod_last	Spesifikasi untuk mengetahui kapan waktu entitas terakhir terkoneksi
mod_muc	Spesifikasi protokol untuk group chat
mod_offline	Mekanisme untuk pesan offline termasuk dalam modul ini adalah XEP-203
mod_proxy65	Spesifikasi untuk mengadakan koneksi SOCKS5
mod_roster	Mekanisme untuk record subscribe antar entitas pada core protokol jabber
mod_shared_roster	Mekanisme untuk mengorganisasi roster kontak
mod_vcard	Spesifikasi untuk penerapan vCard user/entitas

Pengaturan beberapa modul ejabberd dapat dilakukan melalui sebuah file konfigurasi yang terletak pada */etc/ejabberd/ejabberd.yml*. Berikut adalah potongan file konfigurasi ejabberd untuk beberapa modul yang telah terinstal pada sistem aplikasi.

```
mod_disco: {}
mod_caps: {}
mod_last: {}
    ##  db_type: odbc
mod_muc:
    host: "conference.@HOST@"
    ##  db_type: odbc
    access: muc
    access_create: muc_create
    access_persistent: muc_create
    access_admin: muc_admin
    max_room_name: 50
    max_room_description: 100
    default_room_options:
        persistent: true
        allow_change_subj: true
        allow_query_users: true
        allow_user_invites: true
        allow_visitor_nickchange: false
        allow_visitor_status: true
        anonymous: true
        members_only: true
        members_by_default: true
        moderated: true
        password_protected: false
        public: false
        public_list: true
        max_users: 100



mod_muc_log:
    access: muc
    cssfile: false
    dirtype: subdirs
    outdir: "/var/www/muclogs"
    timezone: local
mod_offline:
    access_max_user_messages: max_user_offline_messages
    ##  db_type: odbc
mod_proxy65:
    ip: "10.10.0.1"
    host: "proxy.@HOST@"
    name: "File Transfer Proxy"
    port: 7777
mod_roster: {}
    ##  db_type: odbc
mod_shared_roster: {}
mod_vcard: {}
```

Pada beberapa modul ejabberd masih memakai pengaturan *default*. Pada *mod_muc*, terdapat beberapa pengaturan yang disesuaikan dengan kebutuhan

sistem aplikasi. Sesuai dengan spesifikasnya, modul ini membuka service yang berada pada *host* domain *conference.uin-messenger.com*. `access_create` merupakan pengaturan agar *create room* hanya bisa dilakukan oleh user tertentu yang telah didefinisikan sebelumnya (FR), sedangkan `default_room_options` pengaturan default pada room yang baru saja dibuat (belum dikonfigurasi oleh administrator room). Pengaturan pada mod_proxy65 adalah pemakaian sebuah proxy server yang terletak pada alamat IP 10.10.0.1 dengan nama host *proxy.uin-messenger.com* dan port yang terbuka pada 7777.

4.4.2 Koneksi dan Otentikasi Aplikasi

Terdapat tiga proses penting pada aplikasi uin-messenger sebelum pengguna dapat saling bertukar pesan. **(1)** Proses open stream atau koneksi ke server uin-messenger, **(2)** proses otentikasi pengguna menggunakan protokol SASL dengan metode PLAIN, dan **(3)** proses setting resource aplikasi client dalam jaringan jabber uin-messenger. Ketiga proses ini harus dilakukan karena memang bagian dari core protokol jabber RFC 3620. Ketiga proses ini akan menjadi pembahasan sistem berikut ini,

(1) Koneksi aplikasi client uin-messenger ditandai dengan pengiriman paket stream berikut kepada server uin-messenger (*open stream*).

```
<stream:stream
      to="uin-messenger.com"
      xmlns="jabber:client"
      xmlns:stream="http://etherx.jabber.org/streams"
      version="1.0">
```

Code aplikasi *client* uin-messenger untuk *builder* paket XML stream ini ditunjukkan pada snippet berikut,

```
void openStream() throws IOException {
      StringBuilder stream = new StringBuilder();
      stream.append("<stream:stream");
      stream.append(" to=\"")
            .append(connection.getServiceName()).append("\"");
      stream.append(" xmlns=\"jabber:client\"");
      stream.append(" xmlns:stream=\"http://etherx.jabber.org/streams\"");
      stream.append(" version=\"1.0\">");
      writer.write(stream.toString());
      writer.flush();
}
```

(2) Server mengetahui dan mengirimkan *response* dengan beberapa mekanisme SASL yang didukung oleh server *uin-messenger* sendiri. Berikut merupakan contoh paket *stream response* yang dikirimkan pada aplikasi *client*.

```
<stream:features>
<mechanisms xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
      <mechanism>PLAIN</mechanism>
</mechanisms>
<c
      xmlns='http://jabber.org/protocol/caps'
      hash='sha-1'
      node='http://www.process-one.net/en/ejabberd/'
      ver='bdaZP72YVSx00gwWoF2nf0UNYHQ='/>
<register xmlns='http://jabber.org/features/iq-register'/>
</stream:features>
```

Pada contoh paket *response* diatas tampak bahwa server uin-messenger memiliki hanya satu mekanisme otentikasi yaitu *PLAIN*. ejabberd secara *default* memang hanya menyediakan mekanisme satu ini untuk proses yang memakai otentikasi eksternal ejabberd. Mekanisme *PLAIN* sangat sederhana dimana token atau password user cukup dienkode dengan *Base64*. Aplikasi *client* uin-messenger

kemudian juga memilih mana mekanisme yang sama yang juga dia dukung. Bagaimana pun juga aplikasi *client* uin-messenger harus juga memiliki dukungan terhadap mekanisme *PLAIN*, karena memang hanya satu pilihan yang tersedia. Disini smack library yang digunakan pada aplikasi, menyediakan dukungan terhadap beberapa mekanisme SASL seperti yang tampak pada snippet code smack berikut ini.

```
// Register SASL mechanisms supported by Smack
registerSASLMechanism("EXTERNAL", SASLExternalMechanism.class);
registerSASLMechanism("GSSAPI", SASLGSSAPIMechanism.class);
registerSASLMechanism("DIGEST-MD5", SASLDigestMD5Mechanism.class);
registerSASLMechanism("CRAM-MD5", SASLCramMD5Mechanism.class);
registerSASLMechanism("PLAIN", SASLPlainMechanism.class);
registerSASLMechanism("ANONYMOUS", SASLAnonymous.class);
```

Code yang dilakukan aplikasi *client* untuk memilih mekanisme yang sesuai ditunjukkan pada *snippet* berikut,

```
// Locate the SASLMechanism to use
String selectedMechanism = null;
for (String mechanism : mechanismsPreferences) {
      if (implementedMechanisms.containsKey(mechanism) &&
                  serverMechanisms.contains(mechanism)) {
            selectedMechanism = mechanism;
            break;
      }

}
```

Selesai menemukan mekanisme untuk otentikasi, aplikasi *client* kemudian mengirimkan paket *auth* dengan atribut *mechanism="PLAIN"*, sesuai dengan mekanisme yang keduanya dukung.

```
<auth
      mechanism="PLAIN"
      xmlns="urn:ietf:params:xml:ns:xmpp-sasl">
            bGloaW4AbGloaW4AbGloaW4=
</auth>
```

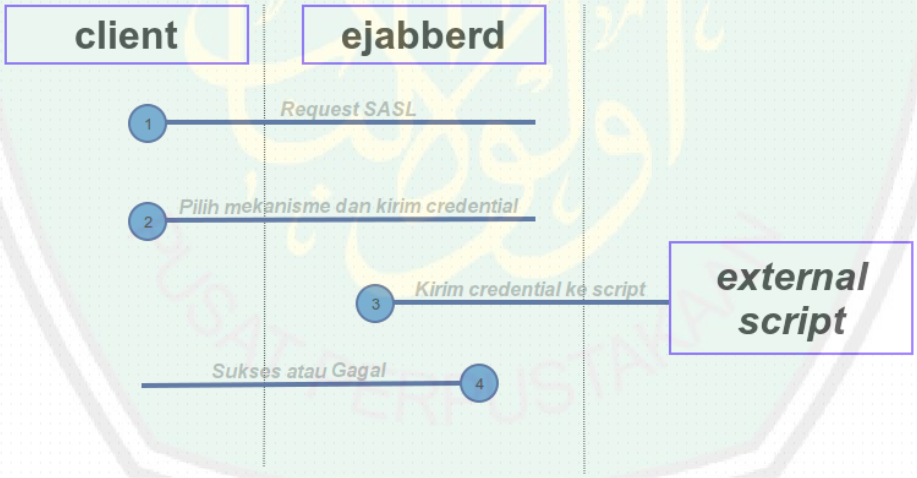
Paket auth ini ditunjukkan pada code SASLPlainMechanism pada bagian bawah ini. Tampak bahwa keamanan password pengguna hanya dilakukan dengan enkoding Base64.

```
// SASLPlainMechanism.java
// Send the authentication to the server
String authenticationText = null;
try {
      if(sc.hasInitialResponse()) {
            byte[] response = sc.evaluateChallenge(new byte[0]);
            authenticationText=Base64.encodeBytes(
                        response,Base64.DONT_BREAK_LINES);
    }
} catch (SaslException e) {
      throw new XMPPException("SASL authentication failed", e);
}
// Send the authentication to the server
getSASLAuthentication().send(
      new AuthMechanism(getName(), authenticationText));
```

Class AuthMechanism merupakan builder untuk membentuk paket auth. Class ini akan mengembalikan XML string seperti yang ditunjukkan pada code snippet berikut ini

```
// AuthMechanism.java
StringBuilder stanza = new StringBuilder();
stanza.append("<auth mechanism=\"").append(name);
stanza.append("\" xmlns=\"urn:ietf:params:xml:ns:xmpp-sasl\">");
if (authenticationText != null &&
            authenticationText.trim().length() > 0) {
      stanza.append(authenticationText);
}
stanza.append("</auth>");
return stanza.toString();
```

Sesuai dengan analisa kebutuhan sistem (FR-2) bahwa user aplikasi berada pada sebuah sistem eksternal dari sistem *uin-messenger*. Tujuannya disini bahwa sistem *uin-messenger* yang dibangun memiliki kemampuan untuk integrasi dengan sistem lain terutama pada bagian manajemen user. Proses otentikasi dan aktivasi harus bisa mengakses aplikasi manajemen user yang dibangun diluar aplikasi uin-messager tersebut. Aplikasi manajemen user yang dibuat disini memiliki antar muka *dashboard* halaman web untuk memudahkan admin manambah, menghapus atau menonaktifkan user. Solusi yang terjadi pada *backend* sistem uin-messenger ini dapat ditunjukkan pada bagan berikut ini,



Gambar 4.44 Skema SASL otentikasi user

Eksternal *script* pada bagan bukanlah bagian dari *core* protokol jabber. Fitur ini merupakan implementasi dan setup pada software ejabberd. Setup ejabberd untuk melakukan otentikasi eksternal ini adalah dengan menulis beberapa baris konfigurasi berikut pada file ejabberd.yml.

```
auth_method: external
extauth_program: "/www/UinMessenger/UserManajemen/ejabberd/login.php"
extauth_cache: 600
extauth_instances: 3
```

Penggunaan *external* otentikasi ditunjukkan pada bagian pengaturan *auth_method: external*, sedangkan *script* program yang bertanggung jawab untuk otentikasi user diserahkan pada file *script* login.php seperti yang tampak pada pengaturan *extauth_program*. Dan berikut adalah potongan *script* login.php yang digunakan untuk *check* akun *username* dan *password* pada aplikasi manajemen user. Penyimpanan data akun user dilakukan pada database RDBMS MySQL dengan nama schema *"UinMessenger"* dan ditaruh pada tabel *"civitas"*.

```
#!/usr/bin/php
<?php
...
$data=explode(":", $this->data);
$this->jabber_user = $data[1];
$this->jabber_pass = $data[3];
$return=checkpass();
@fwrite($this->stdout, $return);

function checkpass() {
    $query = "SELECT 1 FROM civitas WHERE username='".$this->jabber_user."'
AND password='".$this->jabber_pass."' LIMIT 1";
    return @mysql_num_rows(mysql_query($query)) == 1;
}
```

Response yang dilakukan server, untuk proses otentikasi user ini adalah mengirimkan paket *success* atau *failure* kepada aplikasi *client* uin-messenger.

```
<success xmlns='urn:ietf:params:xml:ns:xmpp-sasl'/>

<failure xmlns='urn:ietf:params:xml:ns:xmpp-sasl'>
  <invalid-authzid/>
</failure>
</stream:stream>
```

(3) Setelah otentikasi SASL berhasil dilakukan, proses selanjutnya adalah *setting* sebuah *resource* aplikasi *client*. Dengan *resource* yang telah disetting ini, alamat JID pengguna aplikasi menjadi *mustofa@uin-messenger/android* <*node@domain/resource*>. Untuk pengaturan ini, *client* harus mengirimkan sebuah paket *iq-set* dengan kualifikasi data namespace *urn:ietf:params:xml:ns:xmpp-bind*. Pada aplikasi ini, setiap *resource* pengguna telah ditentukannya sendiri dengan nilai *"android"*. Sehingga paket untuk set *resource* menjadi seperti berikut.

```
<iq type='set' id='bind_2'>
 <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
   <resource>android</resource>
 </bind>
</iq>
```

Server uin-messenger kemudian melaporkan dengan mengirimkan paket *iq-result* bahwa set *resource* telah berhasil dilakukan.

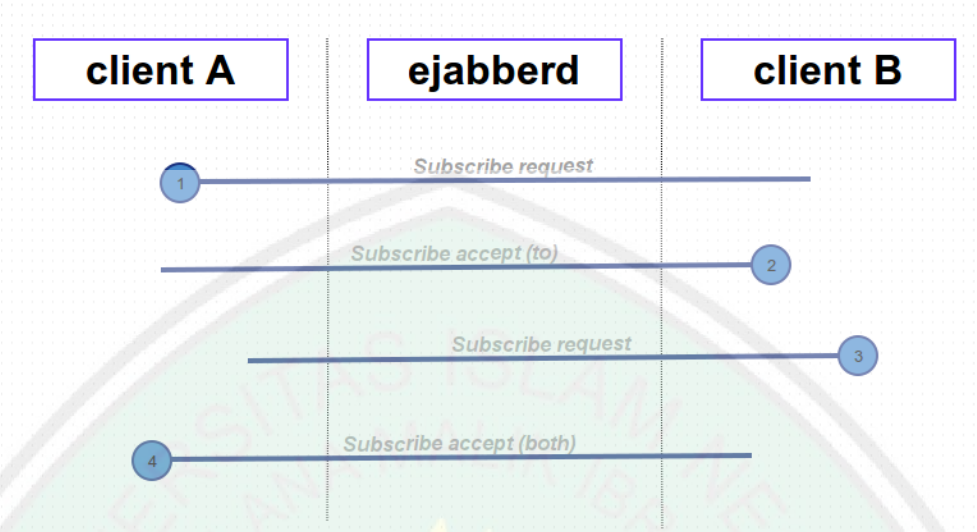
```
<iq type='result' id='bind_2'>
 <bind xmlns='urn:ietf:params:xml:ns:xmpp-bind'>
   <jid>mustofa@uin-messenger.com/android</jid>
 </bind>
</iq>
```

Code aplikasi uin-messenger untuk pengaturan *resource binding* ini dapat ditunjukkan pada *snippet* berikut.

```
Bind bindResource = new Bind();
bindResource.setResource(resource);

PacketCollector collector = connection.createPacketCollector(new
PacketIDFilter(bindResource.getPacketID()));
// Send the packet
connection.sendPacket(bindResource);
// Wait up to a certain number of seconds for a response from the server.
Bind response = (Bind)
collector.nextResult(SmackConfiguration.getPacketReplyTimeout());
collector.cancel();
if (response == null) {
      throw new XMPPException("No response from the server.");
}
// If the server replied with an error, throw an exception.
else if (response.getType() == IQ.Type.ERROR) {
      throw new XMPPException(response.getError());
}
String userJID = response.getJid();
return userJID;
```

4.4.3 Add Contact

Secara teknis fitur *add contact* disebut juga sebagai *subscribing presence*. Istilah ini merupakan mekanisme jabber yang memungkinkan kontrol atas bagaimana informasi *presence* sebuah entitas dapat diketahui oleh entitas lain. Secara *default*, setiap entitas tidak mengetahui informasi *presence* entitas lain. Proses fitur *add contact* pada aplikasi akan tampak seperti bagan berikut ini.



Gambar 4.45 Skema subscribe kontak

Pada implementasi aplikasi, fitur *add contact* akan selalu memiliki orientasi dua arah. Client B harus menyetujui (*accept*) permintaan *subcribe client* A tapi kemudian permintaan B untuk *subscribe* ke A secara otomatis selalu *accepted* oleh sistem aplikasi. Kedua *client* akan selalu *subcribe* satu sama lain, sehingga masing-masing akan selalu mendapatkan informasi status mode dan status teks. Setiap *client* memiliki daftar semua user yang dia *subscribe*. Daftar ini kemudian dinamakan *roster* yang disimpan pada database server ejabberd. Setiap terjadi perubahan *roster* di server, akan selalu tersinkron pada daftar roster yang ada pada lokal aplikasi. Contoh berikut adalah skenario untuk menunjukkan bagaimana *subscribe presence* bekerja, disini pengguna *lihin@uin-messenger.com* melakukan *subscribe* kepada user *saleh@uin-messenger.com*.

```
<iq id="3vh21-13" type="set">
<query xmlns="jabber:iq:roster" >
<item
      jid="saleh@uin-messenger.com"
      name="Ali Saleh">
</item>
</query>
</iq>
```

```
RosterPacket packet = new RosterPacket();
packet.setType(IQ.Type.SET);

RosterPacket.Item item = new RosterPacket.Item(bareAddress, name);
// Add group name to new contact
for (String group : groups)
      if (group.trim().length() > 0)
            item.addGroupName(group);
packet.addRosterItem(item);

// Send request subscribing
ConnectionManager.getInstance().sendPacket(account, packet);
```

Pertama kali pengguna akan mencatat user baru bernama *saleh@uin-messenger.com* pada roster yang dia miliki di server *uin-messenger*. Respon yang dikembalikan server adalah paket dengan status *subscription=”none”*. Status ini menginformasikan bahwa baik *requester* dan *receiver* tidak dalam keadaan *subscribe* satu sama lain.

```
<iq
      from='lihin@uin-messenger.com'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      id='push2065561227' type='set'>
<query xmlns='jabber:iq:roster'>
<item
      subscription='none'
      name='Ali Saleh'
      jid='saleh@uin-messenger.com'/>
</query>
</iq>

<iq
      from='lihin@uin-messenger.com'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      id='3vh21-13'
      type='result'/>
```

Selain mengembalikan paket *iq* dengan status *subscription*, server juga mengirimkan paket *iq* dengan tipe *iq-result* untuk menginformasikan bahwa roster tercatat pada server. Setelah proses ini, kemudian *lihin@uin-messenger.com* mengirimkan sebuah paket *presence* dengan tipe *subscribe* yang ditujukan pada user *saleh@uin-messenger.com*. Dengan mengirimkan paket ini, pengguna menunggu jawaban user, apakah *requester subscribe* diterima ataukan tidak.

```
<presence
      id="3vh21-14"
      to="saleh@uin-messenger.com"
      type="subscribe">
<c
      xmlns='http://jabber.org/protocol/caps'
      hash='sha-1'
      node='http://www.igniterealtime.org/projects/smack/'
      ver='6df9ZPMsG17RYRV0AF6ML1sPAt8='/>
</presence>
```

```
Presence packet = new Presence(Presence.Type.subscribe);
packet.setTo(bareAddress);
ConnectionManager.getInstance().sendPacket(account, packet);
```

Bersamaan dengan pengiriman paket *presence* diatas, server kemudian mengupdate roster *lihin@uin-messenger.com* dengan *subscription=”none”* dan *ask=”subscribe”*, dan kemudian melaporkannya kepadanya.

```
<iq
      from='lihin@uin-messenger.com'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      id='push691520285'
      type='set'>
<query xmlns='jabber:iq:roster'>
<item
      subscription='none'
      ask='subscribe'
      name='Ali Saleh'
      jid='saleh@uin-messenger.com'/>
</query>
</iq>
```

saleh@uin-messenger kemudian mengirimkan paket presence dengan *type=”subscribed”* kepada *lihin@uin-messenger* yang akan menginformasikan bahwa dia mengijinkan *(accept)* permintaan *subscribe*.

```
<presence
      from='saleh@uin-messenger.com'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      type='subscribed'/>
```

```
Presence packet = new Presence(Presence.Type.subscribed);
packet.setTo(bareAddress);
ConnectionManager.getInstance().sendPacket(account, packet);
```

Bersamaan dengan user mengijinkan permintaan *subscribe*, server kemudian mengupdate roster menjadi *subscription=”to”* dan mengirimkan paket *iq-set* ini kepada *lihin@uin-messenger.com*.

```
<iq
      from='lihin@uin-messenger.com'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      id='push616075080'
      type='set'>
<query xmlns='jabber:iq:roster'>
<item
      subscription='to'
      name='Ali Saleh'
      jid='saleh@uin-messenger.com'/>
</query>
</iq>
```

Dari paket terakhir yang terkirim, pengguna kemudian juga mendapatkan informasi presence berupa beberapa informasi terkait dengan user *saleh@uin-messenger.com*.

```
<presence
      from='saleh@uin-messenger.com/37167413031392919731914411'
      to='lihin@uin-messenger.com/androidjIuf5eDw'>
<status>Available</status>
<priority>1</priority>
<c
      xmlns='http://jabber.org/protocol/caps'
      node='http://pidgin.im/'
      hash='sha-1'
      ver='AcN1/PEN8nq7AHD+9jpxMV4U6YM='
      ext='voice-v1 camera-v1 video-v1'/>
<x xmlns='vcard-temp:x:update'>
      <photo>db45e218150110d6f7052225a1527f4e05aaadf4</photo>
</x>
</presence>
```

Tentu saja sampai disini bahwa subscribe user hanya satu arah yaitu dari *lihin@uin-messenger.com* kepada *saleh@uin-messenger.com*. Aplikasi kemudian tanpa kontrol user *saleh@uin-messenger.com* mengirimkan paket presence dengan *type=”subscribe”* kepada *lihin@uin-messenger.com*.

```
<presence
      from='saleh@uin-messenger.com'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      type='subscribe'/>
```

Permintaan *subscribe* ini akan selalu disetujui oleh aplikasi tanpa kontrol *lihin@uin-messenger.com*. Dan kemudian user akan mendapatkan paket *presence* balasan dengan *type="subscribed"* yang memberitahukan bahwa permintaan *subscribe* telah disetujui.

```
<presence
      id="3vh21-18"
      to="saleh@uin-messenger.com"
      type="subscribed">
<c
      xmlns='http://jabber.org/protocol/caps'
      hash='sha-1'
      node='http://www.igniterealtime.org/projects/smack/'
      ver='6df9ZPMsG17RYRV0AF6ML1sPAt8='/>
</presence>
```

Proses *add contact* kemudian diakhiri server dengan mengupdate roster yang dimiliki masing-masing user menjadi *subscription="both"*. Atribut ini mengartikan bahwa pemilik roster dan user item saling melakukan *subscribe* satu-sama lain.

```
<iq
      from='lihin@uin-messenger.com'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      id='push922793971'
      type='set'>
<query xmlns='jabber:iq:roster'>
<item
      subscription='both'
      name='Ali Saleh'
      jid='saleh@uin-messenger.com'/>
</query>
</iq>
```

4.4.4 Kirim teks message

Pengiriman pesan baru dari *sender* kepada *receiver* dilakukan dengan mengirimkan paket *message* dengan tipe *chat*. Pada setiap paket *message* tipe *chat pada* aplikasi uin-messenger diikutsertakan juga tiga spesifikasi XEP standar yaitu *receipts message* (XEP-0184), *chat state notification* (XEP-0085) dan *delayed delivery* (XEP-0203). Implementasi kedua spesifikasi ini didasari pada kebutuhan non-fungsional aplikasi yaitu informasi *user typing* (lihat NF-8) dan status pesan *delivered* (lihat NF-9).

```
<message
      id="rkJ27-89"
      to="mustofa@uin-messenger.com"
      type="chat">
<body>Hello mus</body>
<thread>3797BCNrr1D1</thread>
<active xmlns="http://jabber.org/protocol/chatstates" />
<request xmlns="urn:xmpp:receipts" />
</message>
```

Pesan baru dikirim ke mustofa@uin-messenger.com

Tag <*active/*> pada paket diatas memulai *session* percakapan sesuai spesifikasi XEP-0085. Pada saat *session* percakapan dimulai, pengguna akan

mendapatkan beberapa status user seperti *typing, offline, gone, pause dan inactive* tergantung dari beberapa aktivitas yang dilakukan user. Berikut adalah paket typing yang user mustofa@uin-messenger.com kirimkan saat dia sedang proses *typing* ketika session masih dalam keadaan *active*.

```
<message
      from = 'mustofa@uin-messenger.com/android4mNDzl1Q '
      to='lihin@uin-messenger.com/androidjIuf5eDw '
      type='chat'>
      <composing
            xmlns='http://jabber.org/protocol/chatstates'/>
</message>
```

chat state composing

Sedangkan tag *<request/>* pada paket message menginformasikan pada receiver *mustofa@uin-messenger.com* untuk mengirimkan paket *"ack"* yang menandakan bahwa pesan sudah diterimanya (XEP-0184). Berikut adalah paket yang dikirimkan user *mustofa@uin-messenger.com* kepada sender message *lihin@uin-messenger.com* bahwa pesan dengan thread pesan 3797BCNrr1D1 telah diterimanya.

```
<message
      from='mustofa@uin-messenger.com/android4mNDzl1Q'
      to='lihin@uin-messenger.com/androidjIuf5eDw'
      id='rkJ27-89'>
<thread>3797BCNrr1D1</thread>
<received xmlns='urn:xmpp:receipts' id='rkJ27-105'/>
</message>
```

Info ke sender bahwa pesan telah diterima

Code pada aplikasi *uin-messenger* yang menunjukkan pengiriman paket *message* ini adalah sebagai berikut. Tampak pada code snippet berikut, message baru diupdate oleh ketiga standar extension sebelum dikirimkan ketujuan. Ketiga extensions ini terimplementasi pada tiga class ChatStateExtensionManager (XEP-0085), ReceiptExtensionManager (XEP-0184), dan DelayInformation (XEP-0203).

```
Message message = createMessagePacket(text);
ChatStateExtensionManager.getInstance()
            .updateOutgoingMessage(this, message);
ReceiptExtensionManager.getInstance()
            .updateOutgoingMessage(this, message, messageItem);
message.addExtension(new DelayInformation(messageItem.getTimestamp()));

try {
      ConnectionManager.getInstance().sendPacket(account,message);
} catch (NetworkException e) {
      // Message not sent, break!!
      break;

}
```

4.4.5 Kirim file media

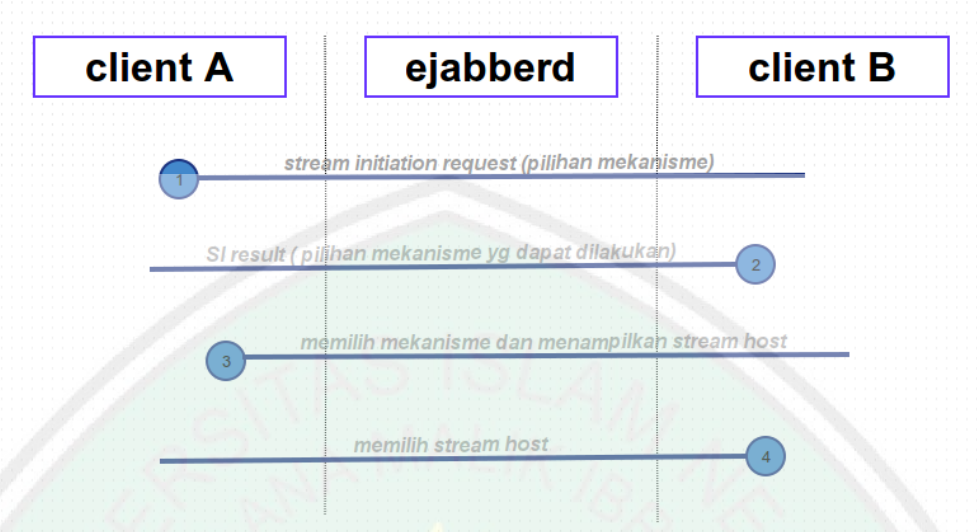
Terdapat dua metode yang ditawarkan protokol jabber untuk bertukar file, yaitu *in-band* dan *out-band*. Metode pertama, *in-band*, yaitu mengubah file *binary* menjadi *text string* dengan cara enkode *Base64*. String enkode kemudian dipecah-pecah menjadi beberapa bagian dan kemudian per-bagian diikutkan dalam paket stream protokol jabber. Metode ini tidak direkomendasi karena tidak efektif untuk mengirim file dengan ukuran besar seperti video dan dokumen. Metode kedua, *out-band*, yaitu memanfaatkan sebuah proxy server untuk bertukar file. Cara ini efektif dan dapat mengirim file dengan ukuran yang relatif besar. Pada dasarnya

juga proses ini merupakan proses negosiasi untuk mengadakan koneksi stream antara dua entitas client guna mengirim sebuah file.

Dalam implementasi pada sistem aplikasi ini, kedua metode akan digunakan, *in-band* dan *out-band*. Penentuan salah satu metode yang digunakan dalam satu transaksi (tranfer file) disini kemudian disebut sebagai proses negosiasi. Setelah proses negosiasi terjadi baru kemudian proses transfer file bisa dilakukan melalui stream yang telah disetujui antar kedua client. Pada aplikasi uin-messenger, proses transfer file ini dapat ditunjukkan dengan melihat *code snippet* berikut ini,

```
// Negotiate the stream
StreamNegotiator streamNegotiator = negotiator
                        .negotiateOutgoingTransfer(
                                getPeer(), streamID, fileName,
                                fileSize, description,
                                RESPONSE_TIMEOUT);

if (streamNegotiator == null) {
      setStatus(Status.error);
      setError(Error.no_response);
      return null;
}

...

outputStream = streamNegotiator.createOutgoingStream(streamID,
            initiator, getPeer());

// Send file process
```

Pada proses negosiasi stream sistem yang dibuat ini, menggunakan spesifikasi standar dari *Stream Initiation XEP-0095*. Secara garis besar proses dari spesifikasi *stream initiation* ini dapat dilihat pada bagan berikut ini.



Gambar 4.46. Proses negosiasi dengan stream initiation

Contoh skenario yang dibuat adalah, pengguna aplikasi *lihin@uin-messenger.com* ingin mengirimkan sebuah file gambar (jpeg) kepada user *didik@uin-messenger.com*. Paket pertama yang dikirimkan pengguna aplikasi adalah seperti berikut ini:

```
<iq
      id="1eF0f-12"
      to="didik@uin-messager.com"
      from="lihin@uin-messager.com/androidDJulp480"
      type="set">
<si
      xmlns="http://jabber.org/protocol/si"
      id="jsi_2818282238378498576"
      mime-type="image/jpeg"
      profile="http://jabber.org/protocol/si/profile/file-transfer">
<file
      xmlns="http://jabber.org/protocol/si/profile/file-transfer"
      name="IMG_20131218_211036_151713282.jpg"
      size="44508" >
<desc>Description</desc>
</file>
<feature xmlns="http://jabber.org/protocol/feature-neg">
      <x type="form" xmlns="jabber:x:data">
      <field var="stream-method" type="list-single">
      <option>
            <value>http://jabber.org/protocol/bytestreams</value>
      </option>
      <option>
            <value>http://jabber.org/protocol/ibb</value>
      </option>
      </field>
      </x>
</feature>
</si>
</iq>
```

Memulai SI untuk transfer gambar

Tiga elemen tag penting dari paket diatas adalah *<si/>* *<file/>* dan *<feature/>*. Tag *si*, kepanjangan dari *stream initiation*, digunakan untuk transaksi transfer file (atribut profile). Informasi file yang akan ditransfer diletakkan pada tag file. Sedangkan tag *feature* merupakan sebuah form berisi dua mekanisme stream yang dapat digunakan oleh sender (*lihin@uin-messenger.com*). Pada dasarnya, receiver *didik@uin-messenger.com* diminta untuk menjawab dari dua mekanisme pilihan, meknisme mana yang dapat juga digunakan oleh dia untuk transfer sebuah file. Dua opsi mekanisme itu adalah *http://jabber.org/protocol/bytestreams (out-band)* dan mekanisme

http://jabber.org/protocol/ibb (in-band). Pemilihan mekanisme ini dapat ditunjukkan pada *code snippet* berikut ini,

```
if (!isByteStream && !isIBB) {
      XMPPError error = new XMPPError(XMPPError.Condition.bad_request,
                  "No acceptable transfer mechanism");
      throw new XMPPException(error.getMessage(), error);
}

if (isByteStream && isIBB) {
      return new FaultTolerantNegotiator(connection,
                  byteStreamTransferManager, inbandTransferManager);
}
else if (isByteStream) {
      return byteStreamTransferManager;
}
else {
      return inbandTransferManager;
}
```

Spesifikasi *stream initiation* mendefinisikan bahwa *receiver* dapat saja menolak *si-request* yang dilakukan sender, tapi dalam implementasi aplikasi ini, semua *si-request* akan diterima secara otomatis oleh aplikasi. Paket berikut merupakan *submit form* yang dikirimkan *receiver* ke *sender*.

```
<iq
      id="1eF0f-12"
      to="lihin@uin-messager.com/androidDJulp480"
      from="didik@uin-messager.com/351858161313888416857923l7"
      type="result">
<si xmlns="http://jabber.org/protocol/si">
<feature xmlns="http://jabber.org/protocol/feature-neg">
      <x type="submit" xmlns="jabber:x:data">
      <field var="stream-method">
      <value>http://jabber.org/protocol/bytestreams</value>
      <value>http://jabber.org/protocol/ibb</value>
      </field>
      </x>
</feature>
</si>
</iq>
```

Submit form mekanisme yang juga dapat digunakan receiver

Dari listing paket diatas, diketahui bahwa *receiver* juga dapat memanfaatkan kedua metode untuk bertukar file. Disini tentu saja *receiver* dapat mendukung semua mekanisme yang ditawarkan *sender* karena memang baik *sender* maupun *receiver* masih dalam satu aplikasi. *Sender* memilih satu mekanisme prioritas *(out-band)* dari submit paket *receiver* dan kemudian mengirimkannya beserta beberapa *stream host* yang memungkinkan untuk diadakannya koneksi *out-band*.

```
<iq
      from='lihin@uin-messager.com/androidDJulp48O'
      to='didik@uin-messager.com/351858161313888841685792317'
      type='set'
      id='1eF0f-12'>
<query
      xmlns='http://jabber.org/protocol/bytestreams'
      sid='purpledb92fd83'>
<streamhost
      jid='lihin@uin-messager.com/androidDJulp48O'
      host='10.10.0.2'
      port='54247'/>
<streamhost
      jid='lihin@uin-messager.com/androidDJulp48O'
      host='10.190.170.171'
      port='54247'/>
<streamhost
      jid='proxy.uin-messager.com'
      host='10.10.0.1'
      port='7777'/>
</query>
</iq>
```

Metode dan stream host sender

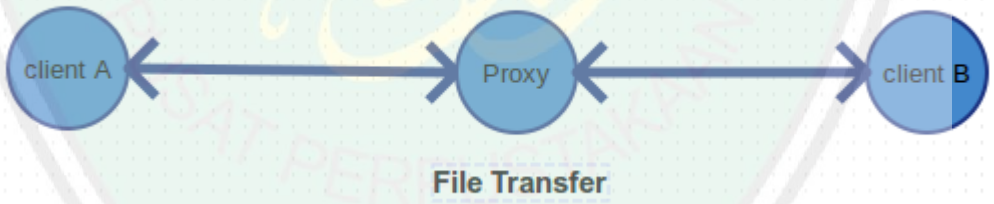
Terdapat tiga pilihan *stream host* yang dapat dilakukan *sender*. Dua diantaranya, receiver dapat melakukan *direct connection* pada alamat sender 10.10.0.2 atau 10.190.170.171 dengan service port 54247. Ketiga adalah dengan *mediated connection* pada proxy server yang memiliki JID *proxy.uin-messenger.com*. *Receiver* kemudian mengirimkan satu *stream host* yang

memungkinkannya diadakan koneksi ke sender. Receiver setuju untuk mengadakan koneksi untuk file transfer dengan menggunakan mediated connection melalui proxy server pada alamat proxy.uin-messenger.com.

```
<iq
      id="1eF0f-12"
      to="lihin@uin-messager.com/androidDJulp480"
      type="result">
<query xmlns="http://jabber.org/protocol/bytestreams">
      <streamhost-used jid="proxy.uin-messager.com" />
</query>
</iq>
```

Receiver setuju dengan stream host

Dengan pengiriman paket result ini, maka proses negosiasi berakhir. Proses kemudian akan diserahkan kepada *sender* dan *receiver* untuk mengadakan koneksi yang dimediasi oleh sebuah proxy server *proxy.uin-messenger.com*.



Gambar 4.47 Mediated Connection

Koneksi stream yang sudah disepakati dan terbentuk kemudian digunakan untuk mengirimkan file dari *sender* ke *receiver*. Proses ini kemudian dapat ditunjukkan pada baris *code* aplikasi berikut ini,

```
public synchronized void sendFile(
                  final File file,
                  final String description)
                  throws XMPPException {
      transferThread = new Thread(new Runnable() {
            @Override
            public void run() {
            try {
                  outputStream = negotiateStream(
                        file.getName(), file.length(), description);
            } catch (XMPPException e) {
                  return;
            }
            if (outputStream == null) {
                  return;
            }

            InputStream inputStream = null;
            try {
                  inputStream = new FileInputStream(file);
                  writeToStream(inputStream, outputStream);
            } catch (FileNotFoundException e) {
            } catch (XMPPException e) {
            } finally {
                  try {
                        if (inputStream != null) {
                              inputStream.close();
                        }
                        outputStream.flush();
                        outputStream.close();
                  } catch (IOException e) {
                        /* Do Nothing */
                  }
            }
            }
      }, "File Transfer " + streamID);
      transferThread.start();
}
```

4.4.6 Multi-User Chat

Groupchat pada sistem yang dibangun sepenuhnya menggunakan spesifikasi *Multi-user chat (MUC)* pada XEP-0045. Pada spesifikasinya, MUC

membuat sebuah komponen service sendiri yang pada sistem aplikasi ini terletak pada host *conference.uin-messenger.com*. Setiap room memiliki identitas unik sebagai *room JID* yang memiliki format penulisan *room@service*, contoh *web_master@conference.uin-messenger.com*. *web_master* adalah nama *room*. Dan setiap partisipator dalam *room* teridentifikasi sebagai *occupant JID* dengan format penulisan *room@service/nickname*, contoh *web_master@conference.uin-messenger.com/mustofa*. *mustofa* disini adalah room *nickname* dari user yang tergabung pada room.

Pada implementasinya setiap user aplikasi tidak memiliki akses untuk membuat *groupchat*, hanya beberapa user yang telah didefinisikan sebelumnya dapat membuatnya. Hal ini sesuai dengan kebutuhan aplikasi (lihat FR-3.2), yang dimaksudkan untuk kontrol semua room yang terdaftar pada sistem aplikasi. Untuk pengaturannya pada file konfigurasi *ejabberd.yml* dengan nama *access_create: muc_create*.

```
mod_muc:
    host: "conference.@HOST@"
    access: muc
    access_create: muc_create
    access_persistent: muc_create
    access_admin: muc_admin
    max_room_name: 50
    max_room_description: 100
    default_room_options:
        persistent: true
        allow_change_subj: true
        allow_query_users: true
        allow_user_invites: true
        allow_visitor_nickchange: false
        allow_visitor_status: true
        anonymous: true
        members_only: true
        members_by_default: true
        moderated: true
        password_protected: false
        public: false
        public_list: true
        max_users: 100
```

muc_create disini adalah sebuah referensi kepada daftar user yang memiliki akses untuk membuat sebuah room chat. Sama dengan *muc_admin* yang juga referensi kepada user sebagai administrator sistem aplikasi *multi user chat*. Administrator dapat mengakses service MUC aplikasi pada ejabberd server. *default_room_options* merupakan opsi default untuk konfigurasi sebuah room. Selesai dibuat oleh seorang *muc_create*, sebuah room pertama kali akan memiliki beberapa opsi konfigurasi default. Tapi nantinya setiap room memiliki opsi konfigurasi yang boleh berbeda, tergantung dari konfigurasi oleh administrator room tersebut.

Sesuai spesifikasi XEP-0045, pengguna dapat mengirimkan message baru yang ditujukan ke room chat dengan mengirimkannya ke alamat room semisal

web_master@conference.uin-messenger.com. Paket message yang akan dikeluarkan aplikasi adalah seperti ini:

```
public void sendMessage(String text) throws XMPPException {
      Message message = new Message(room, Message.Type.groupchat);
      message.setBody(text);
      connection.sendPacket(message);
}
```

```
<message
    from='lihin@uin-messenger.com' id='hysf1v37'
    to='web_master@conference.uin-messenger.com'
    type='groupchat'>
<body></body>
</message>
```

Kemudian setiap partisipan yang sedang tergabung dalam *room* akan mendapatkan sebuah *message* baru seperti pada paket berikut:

```
<message
      from='web_master@conference.uin-messenger.com/lihin'
      to='mustofa@uin-messenger.com/androiddysgBVz0'
      id='3JarI-45'
      type='groupchat'>
<body>:-)</body>
<thread>T8yt2lIkRW3V</thread>
</message>
```

Menerima pesan room

Paket message yang pengguna kirimkan untuk meng-*invite* beberapa kontak user ke sebuah room chat adalah seperti berikut ini:

```
public void invite(Message message, String user, String reason) {
      // TODO listen for 404 error code
      // when inviter supplies a non-existent JID
      message.setTo(room);
      // Create the MUCUser packet
      // that will include the invitation
      MUCUser mucUser = new MUCUser();
      MUCUser.Invite invite = new MUCUser.Invite();
      invite.setTo(user);
      invite.setReason(reason);
      mucUser.setInvite(invite);
      // Add the MUCUser packet that includes the invitation
      // to the message
      message.addExtension(mucUser);

      connection.sendPacket(message);
}
```

```
<message
    from ='mustofa@uin-messenger.com/androiddysgBVz0'
    id='nzd143v8'
    to='web2013@conference.uin-messenger.com'>
<x xmlns ='http://jabber.org/protocol/muc#user'>
    <invite to='kiki@uin.messenger.com'>
    <reason></reason>
    </invite>
</x>
</message>
```

User *invite* kemudian mendapatkan paket message ajakan bergabung kepada sebuah room seperti pada contoh berikut ini, *kiki@uin-messenger* mendapatkan invite untuk gabung kesebuah room chat *web_master@conference.uin-messenger.com* oleh *mustofa@uin-messenger.com*.

```
<message
      from='web2013@conference.uin-messenger.com'
      to='kiki@uin-messenger.com'
      type='normal'>
<x xmlns='http://jabber.org/protocol/muc#user'>
      <invite from='mustofa@uin-messenger.com/androiddysgBVz0'>
      <reason></reason>
      </invite>
</x>
<x
      xmlns='jabber:x:conference'
      jid='web2013@conference.uin-messenger.com'>
</x>
<body>
      Achmad mustofa invites you to the room Web Master</body>
</message>
```

Room invites

4.5 Pengujian Sistem

Pada bab ini akan dibahas mengenai skenario pengujian yang dilakukan terhadap aplikasi yang meliputi aspek prototip aplikasi dan performa aplikasi. Selanjutnya hasil pengujian akan dianalisis dari beberapa segi, untuk mengetahui kesesuaian aplikasi terhadap persyaratan standar yang telah ada sehingga mencapai tujuan yang telah ditentukan. Hasil analisis ini pada akhirnya akan digunakan untuk dijadikan bahan hipotesis guna penelitian dan pengembangan komunikasi aplikasi lebih lanjut.

4.5.1 Pengujian Prototipe Aplikasi

Salah satu teknik yang digunakan pada pengujian prototipe aplikasi Instant Messenger adalah Black-Box testing. Black-Box testing adalah suatu teknik pengujian yang mengamati proses masukan dan keluaran dari sistem perangkat lunak tanpa memperhatikan apa yang terjadi di dalam sistem. Salah satu cara yang

terdapat pada Black-Box testing adalah dengan membuat tabel-tabel yang di dalamnya berisi skenario, output yang diharapkan dan validasi. Hal tersebut dilakukan untuk menguji kesesuaian antara desain dengan implementasi.

4.5.1.1 Test case pengujian otentikasi dan registrasi akun

Test case	Keluaran aplikasi	Keluaran yang seharusnya
Pengguna memasukkan username dan password yang belum terdaftar	Muncul peringatan popup bahwa user belum terdaftar	Terdapat sebuah peringatan bahwa user belum terdaftar
Pengguna memasukkan username dan password yang belum aktivasi	Muncul peringatan popup bahwa akun belum teraktivasi	Terdapat sebuah peringatan akun belum diaktifkan
Pengguna memasukkan username dan password yang valid	Masuk halaman kontak dan display semua user kontak	Pengguna masuk ke halaman daftar kontak
Pengguna tidak memasukkan/invalid NIM atau email atau password pada registrasi mahasiswa	Muncul peringatan popup untuk mengisi form dengan benar	Terdapat peringatan untuk mengisi form dengan valid
Pengguna tidak memasukkan/invalid username, NIP, email, password dan deskripsi pada registrasi dosen	Muncul peringatan popup untuk mengisi form dengan benar	Terdapat peringatan untuk memasukkan form dengan valid
Pengguna memasukkan valid data pada registrasi mahasiswa (submit)	Muncul teks informasi untuk melakukan cek email untuk aktivasi	Display untuk cek email untuk aktivasi
Pengguna memasukkan valid data pada registrasi dosen dan staf (submit)	Muncul teks informasi untuk menunggu akun teraktivasi oleh admin	Display untuk menunggu email bahwa akun telah teraktivasi

Tabel 4.6. Test case otentikasi user login

4.5.1.2 Test case profil pengguna

Test case	Keluaran aplikasi	Keluaran yang seharusnya
Pengguna mengedit semua field vCard yang disediakan aplikasi	Kembali ke profil dan merubah semua field sesuai masukan pengguna	Tidak terdapat loading proses, berjalan pada background aplikasi

Pengguna mengedit foto vCard	Thumbnail foto avatar berubah sesuai pilihan pengguna	Thumbnail foto avatar berubah sesuai pilihan pengguna
Field required vCard (nama dan telpon) kosong	Muncul peringatan popup untuk mengisi field dengan benar	Terdapat warning popup untuk field required.

Tabel 4.7 Test case profil pengguna

4.5.1.3 Test case pengujian kontak user

Test case	Keluaran aplikasi	Keluaran yang seharusnya
Pengguna melakukan *Add Contact* kepada user yang sedang offline	Muncul user pada daftar kontak dengan status unsubscribe	User terdapat pada daftar kontak pengguna dengan status unsubscribe
Pengguna melakukan *Add Contact* kepada user yang sedang online	Muncul user pada daftar kontak dengan status unsubscribe	User terdapat pada daftar kontak pengguna dengan status unsubscribe
User menyetujui konfirmasi yang diberikan	Status user pada daftar kontak menjadi available	Presence kontak user berubah sesuai dengan status user
User tidak menyetujui konfirmasi yang diberikan	Status user tidak berubah tetap dengan unsubscribe	Pengguna tidak perlu ada notifikasi bahwa user tidak menyetujui permintaan
Satu atau lebih user kontak mengubah pesan teks status	Teks status masing-masing user kontak berubah	Teks status pada daftar kontak subscriber juga otomatis berubah
Satu atau lebih user kontak mengubah status mode	Status presence masing-masing user kontak berubah	Status mode pada daftar kontak subscriber juga otomatis berubah
User random melakukan *Add Contact* kepada pengguna	Muncul notifikasi permintaan subscribe	Muncul notifikasi bahwa terdapat permintaan subscribe pada pengguna
Pengguna melakukan pencarian user dengan memasukkan username	Keluaran informasi user dengan username yang cocok	Berisi satu user yang cocok dengan semua informasi yang dia miliki
Pengguna melakukan pencarian user dengan memilih jurusan	Keluar daftar user yang sesuai dengan jurusan yang dicari	Berbentuk daftar user dengan jurusan yang sama

Tabel 4.8 Test case kontak user

4.5.1.4 Test case pengujian mengirim dan menerima konten pesan

Test case	Keluaran aplikasi	Keluaran yang seharusnya
Pengguna mengirim konten pesan ketika dia offline	Muncul pesan pada chat dengan status queue	Terdapat status pesan bahwa pesan tersimpan dalam antrian lokal aplikasi
Pengguna mengirim konten pesan pada user yang sedang offline	Muncul pesan pada chat dengan status delay pada server	Terdapat status pesan delay di server aplikasi
Pengguna mengirim konten pada user online	Muncul pesan pada chat dengan status pesan delivered	Terdapat status pesan bahwa pesan sudah terkirim ke user
Pesan delay yang pengguna kirim baru saja diterima oleh user	Pesan muncul/diterima dengan keterangan waktu kapan dikirim sender	Status pesan berubah dari pesan offline ke pesan terkirim
Pesan queue pengguna, terkirim pada server atau user	Status pesan queue berubah menjadi status delay ke server	Status pesan berubah dari offline pesan ke delay pesan di server
Pengguna menerima konten pesan image	Muncul notifikasi dan konten gambar/thumbnail muncul kemudian pada chat	Notifikasi pengguna bahwa terdapat kiriman foto atau gambar oleh user
Pengguna menerima konten pesan voice	Muncul notifikasi dan voice editor muncul pada chat	Notifikasi pengguna terdapat kiriman pesan suara oleh user
Pengguna menerima konten pesan video	Terdapat notifikasi dan video editor muncul pada chat	Notifikasi pengguna terdapat kiriman pesan video oleh user
Pengguna menerima konten pesan dokumen	Terdapat notifikasi dan dokumen editor muncul pada chat	Notifikasi pengguna terdapat kiriman dokumen oleh user

Tabel 4.9 Test case kirim dan terima pesan

4.5.1.5 Test case pengujian room chat

Test case	Keluaran aplikasi	Keluaran yang seharusnya
Pengguna masuk ke halaman discovery room	Muncul semua room teregistrasi dengan opsi searchable	Keluaran berbentuk daftar room yang sudah teregistrasi
Pengguna menekan button join pada room publik	Teks informasi telah bergabung dan muncul room pada daftar room pengguna	Teks informasi bahwa pengguna telah bergabung
Pengguna melakukan request untuk bergabung pada room privat	Error, server not implemented.	Teks informasi untuk menunggu konfirmasi permintaan oleh admin
Pengguna melakukan *invite* room pada lebih dari satu user kontak	Muncul teks informasi memberitahukan telah invite masing-masing user	Terdapat teks informasi bahwa invite telah dilakukan pada user
Pengguna mendapatkan *invite* room dari salah satu user	Muncul notifikasi dan pilihan opsi untuk menerima atau menolak	Mendapat notifikasi dan konfirmasi pengguna
Pengguna mengirim konten pesan pada room chat	Notifikasi atau counter pesan baru pada room	Notifikasi atau counter pesan baru pada room
Pengguna mengubah topik pada room chat	Muncul teks informasi bahwa topik room telah diubah	Muncul teks informasi bahwa topik room telah diubah
Salah satu member room mengubah topik pada room	Teks topik berubah ketika pengguna masuk ke halaman chat	Topik room chat berubah pada semua user yang telah bergabung pada room

Tabel 4.10 Test case multi user chat

4.5.2 Pengujian Kelayakan Aplikasi

Hasil perolehan perhitungan dari 40 responden, yang diambil dari berbagai pihak diantaranya mahasiswa pengguna device android dan mahasiswa jurusan teknik informatika dengan device dan operating sistem yang berbeda. Berdasarkan

hasil rekapitulasi kuesioner pada tabel 4.11, maka diperoleh hasil untuk masing-masing pertanyaan.

No	Pernyataan	Jumlah Penilaian Responden				
		SB	S	C	K	SK
1	Ketertarikan dengan Aplikasi	26	14	0	0	0
2	Tampilan Aplikasi	16	14	3	0	0
3	Kesesuaian desain warna	15	23	2	0	0
4	Kelengkapan fitur	14	23	3	0	0
5	Kemudahan penggunaan fitur	21	17	2	0	0
6	Penggunaan bahasa	20	19	1	0	0
7	Manfaat Aplikasi	21	19	0	0	0
8	Pengembangan Aplikasi	21	18	1	0	0
9	Tanggapan pengguna secara keseluruhan	26	14	0	0	0

Tabel 4.11 Tabel rekapitulasi hasil kuisioner

Data interval tersebut, analisis dengan menghitung rata-rata jawaban berdasarkan skoring setiap jawaban dari responden dan menghitung prosentase total hasil jawaban dengan membagi total jawaban responden dengan total skor ideal (dengan nilai 200, jika semua jawaban mendapat nilai SB). Berdasarkan skor yang telah ditetapkan dapat di hitung, sebagai berikut:

Keterangan:

- **t1**: Jumlah mendapatkan nilai
- **t2**: Skor dari masing nilai
- **T**: t1*t2

1. Ketertarikkan dengan Aplikasi

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
26	5	**130**	14	4	**56**	0	3	**0**	0	2	**0**	0	1	**0**

2. Tampilan Aplikasi

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
16	5	**80**	21	4	**84**	3	3	**9**	0	2	**0**	0	1	**0**

3. Kesesuaian desain warna

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
15	5	**75**	23	4	**92**	2	3	**6**	0	2	**0**	0	1	**0**

4. Kelengkapan fitur

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
14	5	**70**	23	4	**92**	3	3	**9**	0	2	**0**	0	1	**0**

5. Kemudahan penggunaan fitur

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
21	5	**105**	17	4	**68**	2	3	**6**	0	2	**0**	0	1	**0**

6. Penggunaan bahasa

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
20	5	**100**	19	4	**76**	1	3	**3**	0	2	**0**	0	1	**0**

7. Manfaat Aplikasi

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
21	5	**105**	19	4	**76**	0	3	**0**	0	2	**0**	0	1	**0**

8. Pengembangan Aplikasi

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
21	5	**105**	18	4	**72**	1	3	**3**	0	2	**0**	0	1	**0**

9. Tanggapan pengguna secara keseluruhan

SB			B			C			K			SK		
t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T	t1	t2	T
26	5	**130**	14	4	**56**	0	3	**0**	0	2	**0**	0	1	**0**

Data tersebut merupakan hasil dari perhitungan kuesioner dengan menggunakan skala Likert. Pada tabel berikut merupakan hasil perhitungan dengan skala Likert.

No	Pernyataan	Hasil Perhitungan					Prosentase
		SB	B	C	K	SK	
1	Ketertarikan dengan Aplikasi	130	56	0	0	0	93%
2	Tampilan Aplikasi	80	84	9	0	0	86.5%
3	Kesesuaian desain warna	75	92	6	0	0	86.5%
4	Kelengkapan fitur	70	92	9	0	0	86%
5	Kemudahan penggunaan fitur	105	68	6	0	0	89.5%
6	Penggunaan bahasa	100	76	3	0	0	89.5%
7	Manfaat Aplikasi	105	76	0	0	0	91%

8	Pengembangan Aplikasi	105	72	3	0	0	90%
9	Tanggapan pengguna secara keseluruhan	130	56	0	0	0	93%
Total	1610						
Rata-rata	89.5%						

Tabel 4.12 Hasil perhitungan skala Likers

Berdasarkan tabel 4.12, hasil dari analisis kuesioner uji kelayakan aplikasi dari jawaban 50 responden, **89.5%** menyatakan bahwa aplikasi pencarian messenger ini **sangat layak** berdasarkan tingkatan uji kelayakan, sehingga dapat digunakan dan dipublikasikan untuk civitas kampus.

4.6 Pemanfaatan aplikasi messenger dalam perspektif keislaman

Pemanfaatan aplikasi messenger ini sangat luas penggunaannya. Secara fungsional aplikasi ini dapat memudahkan komunikasi dan akses informasi terutama pada data-data akademik civitas kampus. Bagi seorang muslim harus membentengi diri dengan keimanan dan keislaman yang kuat. Tanpa iman yang kokoh kehidupan seorang muslim akan terombang-ambing dan bisa berujung pada kehancuran. Iman adalah pelita, yang menjadi penerang dan petunjuk pada jalan yang lurus. Berikut adalah beberapa poin manfaat-manfaat teknologi messenger dipandang dari diri seorang muslim.

1. Memperoleh kemudahan berkomunikasi

Kemampuan fisik manusia itu tidak sebanding dengan kebutuhan yang diinginkan. Tetapi manusia sebagai khalifah Allah diberikan kemampuan

akal-pikiran untuk memanfaatkannya menemukan cara-cara yang tepat dan efektif guna meraih kebutuhan hidup yang tidak mungkin dicapai melalui kemampuan fisik semata. Akal-pikiran manusia mampu mendayagunakan segala yang Allah ciptakan di bumi ini.

وَسَخَّرَ لَكُم مَّا فِي ٱلسَّمَٰوَٰتِ وَمَا فِي ٱلۡأَرۡضِ جَمِيعٗا مِّنۡهُۚ إِنَّ فِي ذَٰلِكَ لَأٓيَٰتٖ لِّقَوۡمٖ يَتَفَكَّرُونَ ١٣

Artinya : "Dan dia telah menundukkan untukmu apa yang di langit dan apa yang di bumi semuanya, (sebagai rahmat) daripada-Nya. Sesungguhnya pada yang demikian itu benar-benar terdapat tanda-tanda (kekuasaan Allah) bagi kaum yang berfikir." (QS. Al- Jaziyah : 13)

2. Mengenal dan mengagungkan Allah

Apabila manusia mampu menghayati akan makna sains dan teknologi yang dikembangkannya, bahwa sernua itu bukan semata-mata karena faktor diri pribadi manusia, tetapi ada faktor lain di luar dirinya, maka manusia akan memperoleh jalan untuk mengenal sesuatu yang lain di luar dirinya itu, yaitu Yang Maha Agung, Yang Maha Kuasa, dan Yang Maha Bijaksana, yaitu Allah SWT. Semakin luas dan dalam pengetahuan manusia akan rahasia alam ini, maka semakin dekat manusia untuk mengenal Pencipta alam ini, yaitu Allah, Sang Khalik. Ketika pertama manusia mengembangkan teknologi bangunan, manusia telah diberikan contoh langit yang tinggi, yang luas dan kokoh, yang tidak takut akan runtuh.

3. Meningkatkan kualitas pengabdian kepada Allah

Teknologi apabila dirancang dan dimanfaatkan secara benar dalam konteks tugas pengabdian manusia tersebut, maka teknologi diyakini akan mampu meningkatkan kualitas pengabdiannya kepada Allah.

وَمَا خَلَقْتُ ٱلْجِنَّ وَٱلْإِنسَ إِلَّا لِيَعْبُدُونِ ﴿٥٦﴾

Artinya : "Dan Aku tidak menciptakan jin dan manusia melainkan supaya mereka mengabdi kepada-Ku."(QS. Al-Dzariyat : 56)

4. Memperoleh kesenangan dan kebahagiaan hidup

Kemudahan-kemudahan yang diperoleh manusia melalui pemanfaatan teknologi membuat manusia dapat memperoleh kesenangan dan kebahagiaan hidup serta tetap dalam koridor kesenangan dan kebahagiaan yang halal, yang diridhai Allah. Allah tidak menghendaki manusia hidup susah, tetapi sebaliknya Allah menghendaki manusia hidup senang, hidup bahagia. Untuk memperoleh kesenangan dan kebahagiaan hidup yang disediakan oleh Allah itu, manusia diberikan sarana kebutuhan yang serba lengkap di bumi, sebagaimana Allah nyatakan:

هُوَ ٱلَّذِى خَلَقَ لَكُم مَّا فِى ٱلْأَرْضِ جَمِيعًا ثُمَّ ٱسْتَوَىٰٓ إِلَى
ٱلسَّمَآءِ فَسَوَّىٰهُنَّ سَبْعَ سَمَٰوَٰتٍ ۚ وَهُوَ بِكُلِّ شَىْءٍ عَلِيمٌ ﴿٢٩﴾

Artinya: “Dia-lah Alah yang menjadikan segala yang ada di bumi untuk kamu sekalian dan Dia berkehendak (menciptakan) langit, lalu dijadikan-Nya tujuh langit. Dan Dia Maha Mengetahui segala sesuatu” (QS. Al-Baqarah : 29)

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil pengamatan selama perancangan, implementasi, dan proses pengujian, diambil kesimpulan bahwa pemanfaatan aplikasi untuk memenuhi sebuah media komunikasi bersama kampus UIN Malang, memiliki skor sangat bagus pada tingkat kelayakan yaitu sebesar 89.5% untuk terimplementasikan pada kondisi yang sebenarnya. Pemanfaatan protokol jabber untuk membuat sebuah media komunikasi messenger ini merupakan pilihan teknologi yang tepat guna. Beberapa tahun terakhir banyak bermunculan spesifikasi standar extension protokol oleh XSF yang berjalan diatas *core* protokol jabber, terlebih *extension* standar untuk pembuatan aplikasi messenger. Media komunikasi messenger ini pun banyak memanfaatkan spesifikasi standar tersebut. Dari hasil implementasinya, semua XEP standar yang dipakai berjalan dengan baik sesuai dengan dokumentasi manual masing-masing spesifikasi yang dikeluarkan oleh XSF. Serta juga pembuatan aplikasi pada platform android terbukti dapat meng-*cover* semua *requirement* aplikasi (seperti yang dipaparkan pada Bab III) baik yang bersifat fungsional maupun yang non-fungsional dengan baik.

5.2 Saran

1. Implementasi transfer file yang ditunjukkan pada spesifikasi standar protokol XEP-0095 mengenai negosiasi stream konten, memungkinkan koneksi *peer-to-peer* untuk transport data file. Persyaratannya, kedua entitas baik sender dan receiver harus sama-sama dalam keadaan *available*. Tentu ini merupakan

sebuah permasalahan sendiri. Sebuah model yang dipaparkan oleh Benjemin Sollner pada jurnal yang dia tulis dengan judul *XMPP-based Media Sharing for Mobile Collaboration with Android Phones* berikut ini, akan sangat bagus untuk diimplementasikan pada pengembangan selanjutnya.

2. Penggunaan fitur room chat sebagai akomodasi ruang komunikasi forum setiap kelas matakuliah dirasa penulis masih kurang tepat pengimplementasiannya. Saran yang lebih bagus adalah dengan membuat sebuah chanel diskusi pada setiap kelas matakuliah yang ada kurikulum sekarang. Model protokol pun lebih bagus menggunakan model *publish-subscribe* (XEP-0060) daripada *Multi-User Chat* (XEP-0045).
3. Pembuatan sebuah robot kontak yang akan melayani beberapa permintaan mahasiswa untuk urusan administrasi juga menarik untuk diimplementasikan pada media komunikasi ini. Sebuah fitur yang mirip SMS premium dengan membaca kata atau frasa tertentu untuk mengambil data tertentu semisal nilai matakuliah yang ada pada server kampus.
4. Enkripsi paket pesan pada jaringan jabber adalah diluar implementasi yang dilakukan oleh rancang bangun aplikasi ini. Tetapi terdapat sebuah standar extension yang dapat dimanfaatkan untuk pengamanan encryption paket pada jaringan jabber. OTR (Off the Record) yaitu sebuah protokol standar kriptografi untuk pengamanan pesan pada *conversation chat*. Pemanfaatan standar extension ini bagus untuk pembuatan sebuah fitur *privat chat* yang memungkinkan pesan percakapan lebih aman karena terenkripsi.

Daftar Pustaka

Anabel Quan-Haase.2007. *Instant Messaging on Campus: Social and Academic Change*. London: The University of Western Ontario

Dave Smith, Matthew Miller, Justin Karneges, Peter Saint-Andre, 2011. *XEP-0065: SOCKS5 Bytestreams*. XMPP Standards Foundation (XSF)

Dave Smith, Peter Saint-Andre, 2009. *XEP-0085: Chat State Notifications*. XMPP Standards Foundation (XSF)

Dipen Hamal. 2012. *Near Real-Time Communication in the WWW, Extensible Messaging and Presence Protocol*. Tesis tidak diterbitkan. Helsinki Metropolia University

Jason Ostrander. 2012. *Android UI Fundamentals, Develop and Design*. Berkeley, CA: Peachpit Press

Joe Hildebrand, Peter Saint-Andre, 2011. *XEP-0184: Message Delivery Receipts*. XMPP Standards Foundation (XSF)

Leigh Griffin, Eamonn de Leastar, Dmitri Botvich. *Dynamic Shared Groups within XMPP: An Investigation of the XMPP Group Model*. Waterford Institute of Technology.

Martin A Coleman, 2013, *Integrating Communications and Merging Messaging via the eXtensible Messaging and Presence Protocol*. Caloundra, Queensland, Australia

Meilani Sri Sundari, 2010, *Analisis dan Implemantasi Aplikasi Client Instant Messenger Internal Menggunakan XMPP PT. Indosat Mega Media*. Universitas Mercu Buana.

Peter Saint-Andre, Kevin Smith, and Remko Tronçon, 2009, *XMPP: The Definitive Guide*. O'Reilly Media Inc.

Peter Saint-Andre, 2011, *RFC 6120*. Cisco. Internet Engineering Task Force (IETF)

Peter Saint-Andre, 2004, *RFC 3920*. Jabber Software Foundation (JSF). Internet Engineering Task Force (IETF)

Peter Saint-Andre, 2012, *XEP-0045: Multi-User Chat*. XMPP Standards Foundation (XSF)

Peter Saint-Andre, 2008, *XEP-0054: vcard-temp*. XMPP Standards Foundation (XSF)

Peter Saint-Andre, 2008, *XEP-0095: Stream Initiation*. XMPP Standards Foundation (XSF)

Peter Saint-Andre, Ed. 2004. *RFC 3920-Extensible Messaging and Presence Protocol (XMPP): Core*. Network Working Group.

Peter Saint-Andre, Jeremie Miller, Thomas Muldowney, 2008. *XEP-0012: Last Activity*. XMPP Standards Foundation (XSF)

Peter Saint-Andre, 2012. *XEP-0114: Jabber Component Protocol*. XMPP Standards Foundation (XSF)

Peter Saint-Andre, 2004. *XEP-0134: XMPP Design Guidelines*. XMPP Standards Foundation (XSF)

Peter Saint-Andre, 2009. *XEP-0203: Delayed Delivery*. XMPP Standards Foundation (XSF)

Pin Nie, 2006, *An open standard for instant messaging: eXtensible Messaging and Presence Protocol (XMPP)*. Helsinki University of Technology

Samir Chatterjee, Tarun Abhichandani, Haiqing Li, Bengisu Tulu. *Instant Messaging and Presence Technologies for College Campuses*. Claremont Graduate University.

Theresa Davey, Anastasia Envall, Mark Gernerd, Tiffanne Mahomes, Maria Monroe, Jenna Nowak, Mattew Patricoski, Jacob Weiler. *Instant Messaging: Functions of a New Communicative Tool*.

Thiago O. Fontes, Michelle O'Mahony, 2008, *In-depth interviewing by* *Instant Messaging*, University of Surrey.

7 things you should know about Instant Messaging, http://www.educause.edu/eli, diakses tanggal 1 Maret 2013.

Android Developer Official Site. 2011. *What is Android*. Diakses dari: http://developer.android.com/guide/basics/what-is-android.html. Tanggal Akses: 01-08-2011.

http://wiki.xmpp.org/, diakses tanggal 15 Juli 2014.

http://multazam-einstein.blogspot.com/2013/06/integrasi-ilmu-pengetahuan-dan.html, diakses tanggal 15 Juli 2014.

Lampiran

Lampiran 1. Kuesioner Uji Kelayakan Aplikasi

Bapak/Ibu/Saudara yang terhormat,

Demi kepentingan peningkatan dan uji kelayakan aplikasi UIN-messenger, saya mohon bantuan untuk memberikan informasi. Semua keterangan dan jawaban yang diperoleh semata-mata hanya untuk kepentingan penelitian dan dijamin kerahasiaannya. Atas bantuan Bapak/Ibu/Saudara saya mengucapkan terima kasih.

Keterangan:

SB = Sangat baik

B = Baik

C = Cukup/Biasa

K = Buruk/Kurang

SK = Sangat buruk/kurang

Nama : ……..

No	Pernyataan	Jawaban				
		SB	S	C	K	SK
1	Ketertarikan dengan Aplikasi					
2	Tampilan Aplikasi					
3	Kesesuaian desain warna					
4	Kelengkapan fitur					
5	Kemudahan penggunaan fitur					
6	Penggunaan bahasa					
7	Manfaat Aplikasi					
8	Pengembangan Aplikasi					
9	Tanggapan pengguna secara keseluruhan					
10	Saran dan lain-lain					

Lampiran 2. Hasil Kuesioner Uji Kelayakan Produk

Keterangan:

SB = Sangat Baik

B = Baik

C = Biasa/Cukup

K = Buruk/Kurang

SK = Sangat buruk/kurang

1-10 = Item pernyataan

No	Responden	Jawaban Kuesioner									
		1	2	3	4	5	6	7	8	9	10
1	Jona	SB	SB	SB	C	SB	SB	B	SB	SB	Android 4.3 Jelly Bean (API level 18)
2	Fauzadin	B	B	B	B	SB	B	SB	SB	B	Perangkat : Lenovo
3	Teguh	B	B	B	B	B	B	B	B	B	Apabila aplikasi ini dkembangkan lg n di perbaiki bug minnor yg ada. G akan kalah dengan aplikasi bbm.
4	Rismalil Ismi Afida	SB	B	B	SB	B	B	SB	B	SB	Samsung DUOS
5	Ahmad Rizkyka	SB	B	B	B	B	B	B	SB	SB	Samsung GT-S5300 Android 2.3.6 Gingerbread
6	Fiqqi	SB	B	B	B	B	B	SB	B	B	Sony Xperia L, JB 4.2.2
7	Prima Oktava	B	SB	B	SB	SB	B	B	SB	SB	Tested on cross andromeda a7* ICS
8	Moh. Miftakhur	SB	B	B	SB	SB	SB	B	SB	SB	dijalankan pada BlueStack App Player for Windows (beta-1)
9	Alfi Setyadi M.	B	B	B	SB	SB	B	SB	B	SB	
10	Agus Minanur	SB	C	C	B	B	B	SB	B	B	Axioo picophone4 GDX android 4. 2.1
11	Nurfan Dzihan	SB	B	B	B	B	SB	SB	C	B	xperia arc sony ericsson
12	Bill Tanthowi	SB	SB	SB	B	SB	SB	SB	SB	SB	android 2.3 ke atas
13	Fevi	SB	SB	SB	SB	SB	SB	SB	SB	SB	daebak (y)
14	Marinda Sari	SB	B	B	SB	SB	SB	SB	SB	SB	Note Android 4.1.2
15	Hento	SB	SB	SB	B	B	SB	B	B	B	Axioo picopad 5 GEW on ICS
16	Ahsannun Naseh	SB	B	B	SB	SB	B	SB	SB	SB	Android

											samsung galaxy mini,
17	Gery Aries Sukma	SB	B	B	B	C	C	SB	B	B	Evercoss a26c Android Jelly Bean 4.2
18	Bara	SB	SB	SB	SB	SB	SB	B	SB	SB	jelly bean 4.2.2
19	Arie Iskandar	SB	SB	SB	SB	SB	SB	B	SB	SB	segera di launching dah,, pake metode RAD aja,,
20	Liffindra Angga	SB	SB	SB	SB	SB	SB	SB	SB	SB	saya coba sudah bagus bang. dang maju wes bang
21	Agus Cahyono	SB	SB	SB	SB	SB	SB	SB	SB	SB	Segera di implementasikan dalam proses belajar mengajar di kampus, semisal digunakan untuk sarana peyebaran informasi perkuliahan, setiap mata kuliah membuat grup chat
22	Moh. Nuruddin	B	B	B	B	B	B	B	B	B	
23	Hamdi Musaad	B	B	B	B	C	SB	B	B	SB	
24	Rofiq Azhari	B	B	B	B	B	B	B	B	B	
25	Sofyan Setiawan	SB	C	C	C	SB	SB	SB	SB	SB	
26	M. Nuril Efendi	SB	SB	SB	SB	SB	SB	SB	SB	SB	
27	Rochman	SB	SB	SB	B	B	SB	B	SB	SB	
28	Anonim	B	B	B	B	B	B	B	B	SB	
29	Wildan Gunardi	SB	C	SB	C	SB	SB	SB	SB	SB	
30	Syaiful Arif	SB	SB	SB	B	SB	B	SB	B	SB	
31	Edi	B	B	B	B	B	B	B	B	B	
32	Rudi Setyawan	B	B	B	B	B	B	B	B	SB	semangat kk!
33	Luluk	SB	SB	B	B	SB	SB	SB	SB	SB	bagus bang mus...
34	Sugeng	B	B	B	B	B	B	B	B	B	
35	Dwi Kharisma	SB	SB	SB	SB	SB	SB	SB	SB	SB	
36	Perdana H.P	B	B	B	B	B	B	B	B	B	android young
37	Rina	SB	SB	SB	B	SB	SB	SB	SB	SB	semangat bang

											mus heheh
38	Vitri Nur Hayati	SB	SB	SB	SB	SB	SB	SB	SB	SB	
39	Robait Usman	B	B	B	B	B	B	B	B	B	
40	Eko	B	B	B	B	B	B	B	B	B	
Total											
SB		26	16	15	14	21	20	21	21	26	
B		14	21	23	23	17	19	19	18	14	
C		0	3	2	3	2	1	0	1	0	
K		0	0	0	0	0	0	0	0	0	
SK		0	0	0	0	0	0	0	0	0	