BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

1. Berdasarkan dokumen-dokumen SKPL, DPPL, dan PDHUPL yang telah dibuat serta hasil uji coba yang telah dilakukan maka dapat disimpulkan bahwa penulis telah berhasil mengembangkan dan mengimplementasi kan suatu aplikasi IM berbasis Jabber pada ponsel dengan menggunakan Java 2 Micro Edition.
2. Berdasarkan hasil pengujian biaya maka dapat disimpulkan biaya pengiriman pesan melalui aplikasi JAIM lebih rendah daripada biaya pengiriman pesan melalui SMS. Biaya pengiriman pesan melalui JAIM tergantung tarif GPRS dari masing-masing provider.
3. Dari rumusan masalah yang ada dapat disimpulkan aplikasi Instant Messaging Client pada ponsel sudah berhasil dibuat dan aplikasi JAIM ini dapat terhubung dengan piranti lain selain ponsel misalnya PC.

5.2. Saran

Beberapa hal yang di sarankan untuk pengembangan lebih lanjut aplikasi game maze ini adalah sebagai berikut :

1. Menambah fungsi registrasi yang memungkinkan user melakukan registrasi langsung dari ponsel dan tidak perlu meregistrasi melalui website server.

2. Membuat tampilan dari aplikasi ini menjadi lebih menarik seperti menambahkan warna pada form, dan memberikan beberapa animasi.
3. Menambah fungsi-fungsi yang lain seperti Jabber User Directory(JUD) yaitu fungsi untuk mencari suatu user pada suatu server, voice, dll.

DAFTAR PUSTAKA

Dodson, Chaterine, 2000, *Jabber Technical White Paper*, Jabber.com, Inc

Dreamtech Software Team, 2002, *Instant Messaging Systems Cracking The Code*, Whiley Publishing, Inc., New York.

Hartanto, Antonius Aditya, 2003, Java 2 Micro Edition Tingkat Lanjut, PT Elex Media Komputindo, Jakarta.

Kozakai, Maiko , 2004, *Jabber/J2ME based Instant Messaging Client for Presence Service*, University of Applied Sciences, Sttutgart.

Shigeoka, Iain, 2002, *Instant Messaging in Java Jabber Protocols*, Manning Publications Co., Greenwich.

Suyoto, 2005, Membuat Sendiri Aplikasi Ponsel, Gava Media, Yogyakarta

Yenis, Rita, 2004, Aspek Keamanan Sistem *Instant Messaging* pada Protokol Jabber, Institut Teknologi Bandung, Bandung.

SKPL

SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK

Pembangunan Aplikasi Instant Messaging Client Berbasis Jabber pada Ponsel (JAIM)

Dipersiapkan oleh:

I MADE HEDI JULIARTA (3155/TF)

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Atma Jaya Yogyakarta

Jalan Babarsari 44, Yogyakarta 55281

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	
G	

INDEX TGL	-	A	B	C	D	E	F	G
Ditulis oleh								
Diperiksa oleh								
Disetujui oleh								

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi

Daftar Isi

Daftar Gambar

Daftar Tabel

1 Pendahuluan

1.1 Tujuan

Dokumen ini akan berisi penjelasan pemakaian dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) bertujuan untuk mendokumentasikan kebutuhan perangkat lunak yang akan dikembangkan. Dokumen ini digunakan oleh pengembang perangkat lunak sebagai acuan pengembangan perangkat lunak selanjutnya.

1.2 Lingkup Masalah

JAIM (Aplikasi *Instant Messsaging* Berbasis Jabber pada Ponsel) adalah perangkat lunak pada telepon selular (ponsel) yang digunakan untuk mengirim dan menerima pesan, *presence*, dan *chatting* antara dua buah ponsel atau antara ponsel dengan piranti yang lain seperti Laptop, PDA, dan PC. Aplikasi ini dijalankan pada ponsel yang mendukung Java dan GPRS, dan pesan yang dikirim hanya berupa teks.

1.3 Definisi, Akronim dan Singkatan

1. SKPL adalah Spesifikasi Kebutuhan Perangkat Lunak, atau dalam bahasa Inggrisnya sering juga disebut sebagai SRS (*Software Requirements Spesification*), merupakan spesifikasi dari perangkat lunak yang akan dikembangkan
2. SKPL-JAIM.K-xxxx adalah kode yang merepresentasikan kebutuhan (*requirement*) pada JAIM, dengan JAIM merupakan kode perangkat lunak, JAIM.K adalah kode fase dan xxxx adalah digit/nomor kebutuhan (*requirement*).
3. SKPL-JAIM.UC-xxxx adalah kode yang merepresentasikan Use Case Diagram pada JAIM, dengan JAIM merupakan kode perangkat lunak, JAIM.UC adalah Use Case dan xxxx adalah digit/nomor urutan Use Case.

1.4 Referensi

Referensi yang digunakan pada perangkat lunak ini adalah :

1. GL01, *Template* Spesifikasi Kebutuhan Perangkat Lunak, Jurusan Teknik Informatika-ITB

1.5 Deskripsi umum (Overview)

Dokumen SKPL ini dibagi menjadi tiga bagian utama. Bagian utama berisi penjelasan tentang dokumen SKPL yang mencakup tujuan pembuatan dokumen ini, lingkup masalah yang diselesaikan oleh perangkat lunak yang dikembangkan, definisi, referensi dan deskripsi umum.

Bagian kedua berisi penjelasan secara umum mengenai perangkat lunak JAIM yang akan dikembangkan, meliputi perspektif produk, fungsi produk, karakteristik pengguna, batasan dalam pengembangan perangkat lunak.

Bagian ketiga berisi uraian kebutuhan perangkat lunak secara lebih rinci.

2 Deskripsi Keseluruhan

2.1 Perspektif produk

Aplikasi *Instant Messaging Client* berbasis Jabber pada Ponsel (JAIM) adalah perangkat lunak yang digunakan untuk berkomunikasi antara dua ponsel atau antara ponsel dengan piranti lainnya seperti Laptop, PDA dan PC yang terhubung internet. Perangkat lunak ini dikembangkan dengan menggunakan bahasa pemrograman J2ME(*Java 2 Micro Edition*). Pengguna berinteraksi dengan ponsel atau emulator yang mendukung J2ME. Masukan dari perangkat lunak ini adalah berupa teks yang diketikkan melalui *keyboard* pada emulator atau tombol pada ponsel dan pilihan menu, yang dipilih oleh pengguna menggunakan tombol-tombol pada ponsel atau emulator. Perangkat lunak ini dapat dijalankan pada ponsel yang mendukung J2ME. Sedangkan emulator dapat dijalankan di sistem operasi Windows XP.

2.2 Fungsi Produk

Perangkat lunak ini berfungsi untuk melakukan komunikasi seperti mengirim atau menerima pesan, *presence*, dan *chatting* antara satu ponsel dengan ponsel, PDA, Laptop, PC yang lain. Fungsi-fungsi yang terdapat dalam perangkat lunak ini antara lain :

1. *Login* [SKPL-JAIM.**K-0001**].

   Digunakan untuk dapat terhubung ke *server* Jabber dan dapat menggunakan sistem.. Tampilan halaman *login* berisi *textbox* untuk diisi dengan *Jabber Indentity* (JID) dengan format [*node@domain/resource*] contohnya hedi@localhost/mobile dan *textbox password*. Jika berhasil maka akan masuk kehalaman roster dan status presence adalah *online*.

2. Kirim *Presence* [SKPL-JAIM.**K-0002**]

   Digunakan untuk mengganti *presence*. *Presence* berfungsi antara lain untuk mengetahui status user apakah *online* atau *offline*, untuk melakukan *subscribe* kepada teman kontak, dan agar teman kontak mengetahui status *user*. Tampilan halaman ini berisi pilihan-pilihan *presence* antara lain *online*, *offline*, *away*, dan *dnd*.

3. Kirim Pesan [SKPL-JAIM.**K-0003**]

   Digunakan untuk mengirim pesan kepada suatu account pada server yang sama atau server yang berbeda. Pesan dibatasi sebanyak 2048 karakter.

4. Terima Pesan [SKPL-JAIM.**K-0004**]

   Digunakan untuk membaca pesan yang masuk

5. *Chat* [SKPL-JAIM.**K-0005**]

   Digunakan untuk melakukan melakukan obrolan dengan teman kontak. *Chat* dapat dilakukan dengan satu orang maupun dengan lebih dari satu orang (*multichat*). User memilih teman yang akan diajak mengobrol, kemudian tampil halaman yang berisi *textbox* untuk diisi dengan teks dan teks balasan dari teman kontak tampil dibawahnya.

6. Tambah Kontak [SKPL-JAIM.**K-0006**]

Digunakan untuk menambah teman kontak pada list roster. Tampilan halaman ini berisi berupa form yang berisi *textbox* JID, dan email kemudian oleh user diisi dengan jid dan email dari teman kontak yang baru.

7. Edit Kontak **[SKPL-JAIM.K-0007]**

   Digunakan untuk melakukan *update* teman kontak seperti mengganti JID atau mengganti alamat e-mail. Setelah memilih teman yang akan diupdate kemudian tampil data JID dan email dari teman kontak tersebut. User kemudian menghapus data dan mengganti dengan yang baru.

8. Hapus Kontak **[SKPL-JAIM.K-0008]**

   Digunakan untuk menghapus teman kontak dari list *roster*. Prosesnya adalah *user* memilih teman yang akan dihapus kemudian pada menu pilih hapus.

9. Membaca Bantuan **[SKPL-JAIM.K-0009]**

   Digunakan sebagai panduan penggunaan aplikasi. Tampilan halaman berisi teks bantuan penggunaan aplikasi.

10. Melihat Info Pembuat **[SKPL-JAIM.K-0010]**

    Digunakan untuk mengetahui profil pembuat aplikasi. Tampilan halaman ini berisi profil pembuat.

2.3 Karakteristik Pengguna

Pengguna perangkat lunak ini adalah masyarakat yang memiliki ponsel berbasis Java dan mendukung MIDP 2.0 serta bisa mengoperasikannya.

2.4 Batasan-batasan

Batasan masalah dalam mengembangkan Aplikasi *Instant Messaging* berbasis Jabber pada Ponsel ini adalah sebagai berikut :

1. Diimplementasikan sebatas pada piranti ponsel yang mendukung Java dan GPRS.
2. Pesan dibatasi hanya berupa teks.
3. Implementasi perangkat lunak pada ponsel *client* menggunakan J2ME.

4. Analisa yang digunakan tidak menjelaskan secara detail mengenai *traffic* dan aspek keamanan.
5. Pengujian program dilakukan menggunakan emulator yang ada pada J2ME.

3 Deskripsi Rinci Kebutuhan

3.1 Kebutuhan antarmuka eksternal

Kebutuhan antarmuka eksternal pada JAIM meliputi kebutuhan antarmuka pemakai, kebutuhan antarmuka perangkat keras, kebutuhan antarmuka perangkat lunak.

3.1.1 Antarmuka pemakai

Pemakai berinteraksi dengan perangkat lunak JAIM dengan antarmuka layar ponsel atau emulator. Masukan dari perangkat lunak ini adalah pilihan menu yang ditekan melalui tombol pada ponsel atau dengan meng-*click* gambar tombol ponsel pada emulator dan teks yang diketikan user melalui tombol pada ponsel atau *keyboard*. Keluaran dari perangkat lunak JAIM berupa teks.

3.1.2 Antarmuka perangkat keras

Antarmuka perangkat keras JAIM meliputi:

a. Prosesor Intel Pentium
b. RAM 256 MB
c. Kapasitas sisa harddisk 500 MB
d. Keyboard
e. Mouse
f. Card Reader/Writer dan kabel USB
g. Ponsel berbasis Java (Nokia 6600)

3.1.3 Antarmuka perangkat lunak

Antarmuka perangkat lunak JAIM dengan menggunakan :

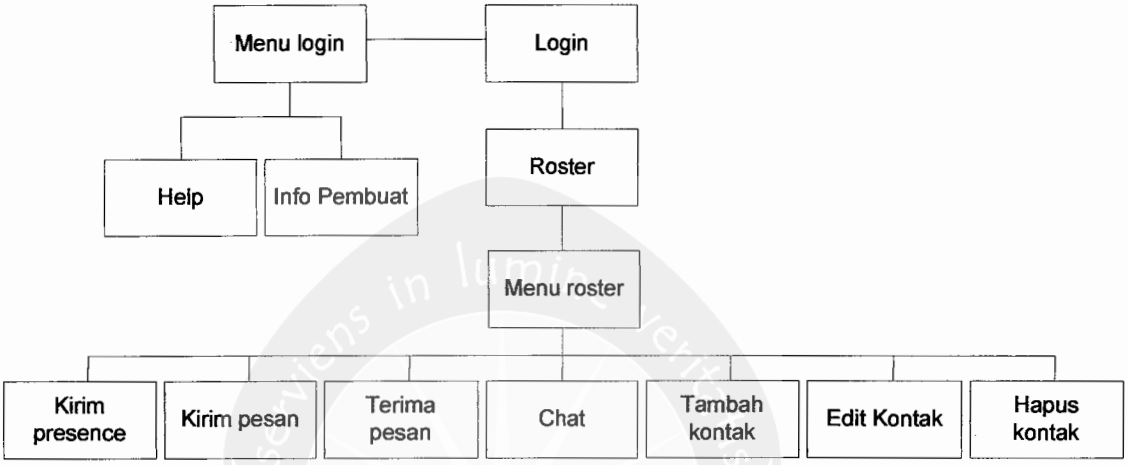
a. Sistem Operasi : Windows XP.
b. *Java 2 Software Development Kit* (J2SDK) versi 1.4.1

c. Antarmuka : J2ME *Wireless Tool Kit* (J2MEWTK) versi 2.1_01

d. Server: JabberD 1.4.2_2

3.2. Arsitektur Antarmuka Aplikasi

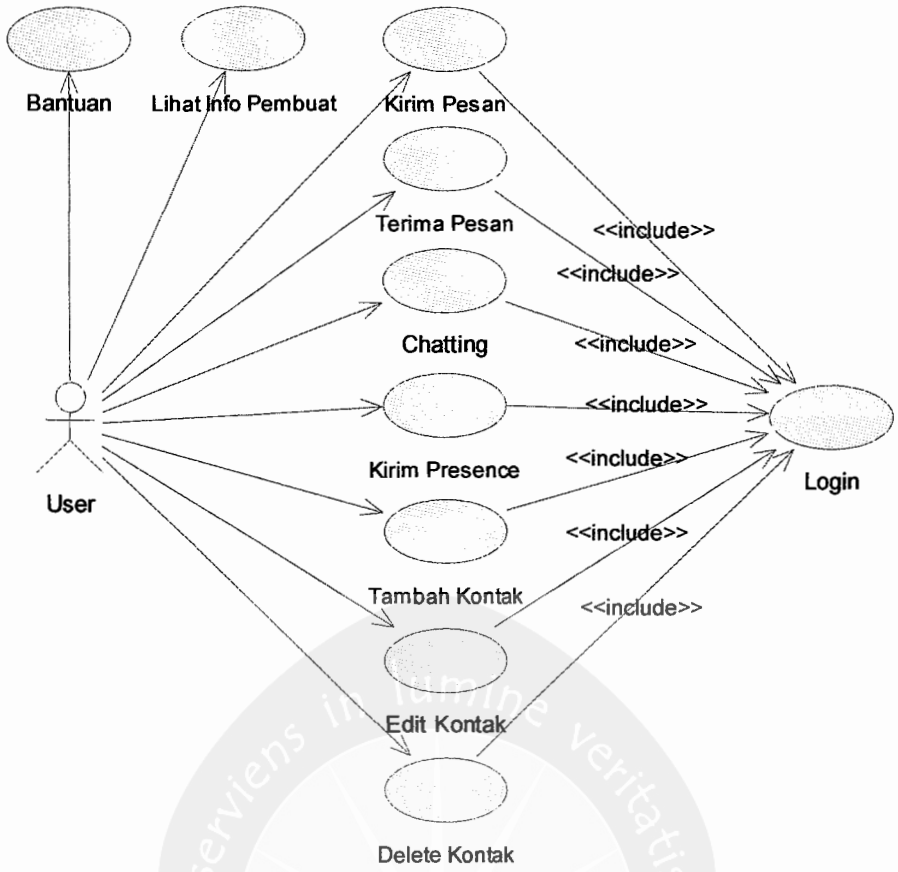
Arsitektur antarmuka aplikasi merupakan gambaran secara umum antarmuka aplikasi yang akan dibuat. Arsitektur antarmuka aplikasi JAIM seperti yang ditunjukkan pada gambar 1 di bawah ini.



Gambar 3.1 Arsitektur Antarmuka Aplikasi

3.3. Kebutuhan Fungsionalitas

Kebutuhan fungsionalitas adalah gambaran aksi-aksi apa sajakah yang dapat dilakukan oleh *user* secara umum dalam aplikasi. Kebutuhan fungsionalitas JAIM digambarkan dengan *use case diagram* seperti yang ditunjukkan pada gambar 3 di bawah ini :



Gambar 3.2 *Use Case Diagram* JAIM

Penjelasan *use case diagram* :

a. *User*

User adalah pengguna yang akan mengakses sistem pakar. *User* bertindak sebagai aktor.

b. *Login*

Aksi yang dilakukan agar dapat terhubung ke *server* Jabber dan dapat menggunakan sistem. User mengetikkan JID dan *password*-nya pada form kemudian menekan tombol OK. Jika berhasil maka *user* masuk ke sistem tapi jika gagal maka *user* tetap berada pada halaman *login*.

c. Kirim *Presence*

Aksi yang digunakan untuk mengganti *presence*. *Presence* berfungsi antara lain untuk mengetahui keadaan user apakah *online* atau *offline*, untuk melakukan *subscribe*

kepada teman kontak, dan agar teman kontak mengetahui keadaan *user*.

d. Kirim Pesan

   Aksi yang digunakan untuk mengirim pesan ke suatu account.

e. Terima Pesan

   Aksi yang digunakan untuk membaca pesan yang masuk.

f. *Chat*

   Aksi yang digunakan untuk melakukan obrolan dengan teman kontak. *Chat* dapat dilakukan dengan satu orang maupun dengan lebih dari satu orang (*multichat*).

g. Tambah Kontak

   Pilihan ini menampilkan *form* yang berisi *textbox* JID dan *password* yang diisi user untuk menambah teman kontak baru. Setelah *form* diisi kemudian menekan tombol OK pada ponsel atau emulator maka pada halaman *roster* nama teman kontak yang baru terlihat.

h. Edit Kontak

   Aksi yang digunakan untuk melakukan meng-*edit* profil teman kontak seperti mengganti JID atau mengganti alamat e-mail.

i. Hapus Kontak

   Aksi yang digunakan untuk menghapus teman kontak dari list kontak (*roster*.

j. Membaca Bantuan

   Aksi yang digunakan bila *user* kesulitan dalam menggunakan sistem dan memerlukan bantuan untuk menggunakan aplikasi.

k. Melihat Info Pembuat

   Aksi yang dilakukan user untuk dapat mengetahui profil pembuat aplikasi.

3.3.1. Fungsi Login

3.3.1.1. Skenario

Use Case ID	SKPL-JAIM.UC-01
Use Case Name	Login
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan agar user dapat mengakses server Jabber. User mengisi JID dan password.
Preconditions	-
Basic Path	1. JAIM menampilkan form login. 2. Aktor mengisi form dengan JID dan password-nya dan menekan tombol OK. 3. JAIM mengirim data login ke server. 4. Server memeriksa data login. 5. JAIM memberikan akses ke user dan menampilkan halaman roster.
Alternative Path	-
Postconditions	User terhubung ke server dan sistem menampilkan halaman kontak list.
Exception Path	Data login tidak terdapat dalam basis data server: 1. Server mengirim pesan error. 2. JAIM mengirim stanza untuk melakukan registrasi dengan data login yang dimasukkan oleh user.
Extends	-
Includes	-

3.3.2. Fungsi Kirim Presence

3.3.2.1. Skenario

Use Case ID	SKPL-JAIM.UC-02
Use Case Name	Kirim Presence
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user untuk mengirim presence.
Preconditions	Use case : login sudah dilakukan.
Basic Path	1. JAIM menampilkan halaman change presence. 2. User memilih presence apakah online, offline, atau away. 3. JAIM mengirim data presence ke server. 4. Server mengirim status presence user ke semua teman kontak user yang sedang online.
Alternative Path	-
Postconditions	JAIM menampilkan status presence yang baru pada halaman roster.
Exception Path	-
Extends	-
Includes	-

3.3.3. Fungsi Kirim Pesan

3.3.3.1. Skenario

Use Case ID	SKPL-JAIM.UC-03
Use Case Name	Kirim pesan
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user untuk

	mengirim pesan ke suatu account.
Preconditions	Use case : login sudah dilakukan.
Basic Path	1. JAIM menampilkan halaman kirim pesan. 2. User menulis teks pesan. 3. JAIM mengirim pesan ke server.
Alternative Path	
Postconditions	JAIM mengirim pesan ke server.
Exception Path	-
Extends	-
Includes	-

3.3.4. Fungsi Terima Pesan

3.3.4.1. Skenario

Use Case ID	SKPL-JAIM.UC-04
Use Case Name	Terima pesan
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user untuk membaca pesan yang masuk.
Preconditions	Use case : login sudah dilakukan.
Basic Path	1. JAIM menampilkan peringatan ada pesan masuk. 2. User menekan tombol terima pesan. 3. JAIM menampilkan form terima pesan yang berisi teks pesan masuk.
Alternative Path	
Postconditions	JAIM menampilkan pesan yang masuk pada form terima pesan.
Exception Path	-
Extends	-
Includes	-

3.3.5. Fungsi Chat

3.3.5.1. Skenario

Use Case ID	SKPL-JAIM.UC-05
Use Case Name	Chat
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user untuk melakukan obrolan dengan seseorang atau lebih dari satu orang.
Preconditions	Use case : login sudah dilakukan.
Basic Path	1. User memilih teman kontak yang akan diajak mengobrol pada list kontak. 2. JAIM menampilkan halaman chat. 3. User mengetikkan teks obrolan. 4. JAIM menampilkan teks balasan dari teman kontak.
Alternative Path	-
Postconditions	JAIM menampilkan teks balasan dari teman kontak yang diajak mengobrol.
Exception Path	-
Extends	-
Includes	-

3.3.6. Fungsi Tambah Kontak

3.3.6.1. Skenario

Use Case ID	SKPL-JAIM.UC-06
Use Case Name	Tambah Kontak
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user untuk menambah teman kontak.
Preconditions	Use case : login sudah dilakukan.

Basic Path	1. JAIM menampilkan halaman tambah teman kontak. 2. User mengisi form. 3. JAIM menampilkan nama teman kontak yang baru pada kontak list.
Alternative Path	1. JAIM menerima presence dari server bahwa ada permintaan subscribe dari seseorang untuk menjadi teman kontak user. 2. User menerima teman kontak tersebut. 3. JAIM menampilkan teman kontak yang baru pada kontak list.
Postconditions	JAIM menampilkan teman kontak yang baru pada kontak list.
Exception Path	-
Extends	-
Includes	-

3.3.7. Fungsi Edit Kontak

3.3.7.1. Skenario

Use Case ID	SKPL-JAIM.UC-07
Use Case Name	Edit Kontak
Use Case Type	Primary
Actors	User
Preconditions	Use case : login sudah dilakukan.
Description	Use Case ini digunakan user untuk melakukan edit data teman kontak.
Basic Path	1. User memilih teman kontak yang di-edit datanya. 2. JAIM menampilkan data teman kontak tersebut. 3. User mengganti data yang lama dengan data yang baru.

	4. Data teman kontak ter-update.
Alternative Path	-
Postconditions	Data teman kontak ter-update.
Exception Path	-
Extends	-
Includes	-

3.3.8. Fungsi Hapus Kontak

3.3.8.1. Skenario

Use Case ID	SKPL-JAIM.UC-08
Use Case Name	Hapus Kontak
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user untuk menghapus teman kontak dari list kontak.
Preconditions	Use case : login sudah dilakukan.
Basic Path	1. User memilih teman kontak yang akan dihapus. 2. JAIM menghapus teman kontak dari list kontak.
Alternative Path	-
Postconditions	JAIM menghapus teman kontak dari list kontak.
Exception Path	-
Extends	-
Includes	-

3.3.9. Fungsi Membaca Help

3.3.9.1. Skenario

Use Case ID	SKPL-JAIM.UC-09

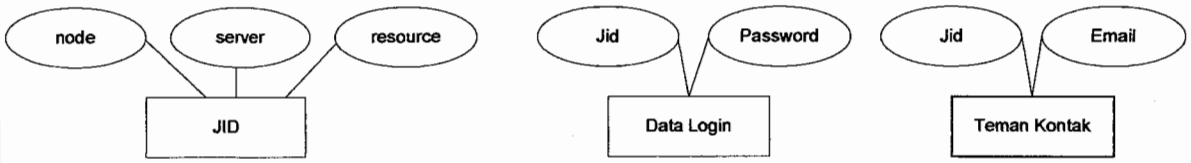
Use Case Name	Membaca Help
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user bila menemui kesulitan dalam menggunakan aplikasi.
Preconditions	-
Basic Path	1. User memilih help pada menu. 2. JAIM menampilkan teks help.
Alternative Path	-
Postconditions	JAIM menampilkan teks bantuan.
Exception Path	-
Extends	-
Includes	-

3.3.10. Fungsi Melihat Info Pembuat

3.3.10.1. Skenario

Use Case ID	SKPL-JAIM.UC-10
Use Case Name	Melihat Info Pembuat
Use Case Type	Primary
Actors	User
Description	Use Case ini digunakan user untuk mengetahui profil pembuat aplikasi.
Preconditions	-
Basic Path	1. User memilih lihat info pembuat pada menu. 2. JAIM menampilkan halaman info pembuat.
Alternative Path	-
Postconditions	JAIM menampilkan halaman info pembuat.
Exception Path	-
Extends	-
Includes	-

3.4. Persistent Data

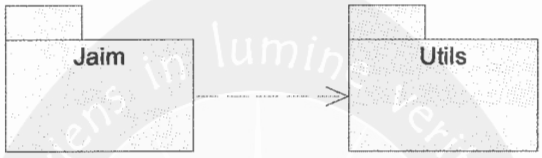


Gambar 3.3 *Persistent Data.*

4. Realisasi Use Case

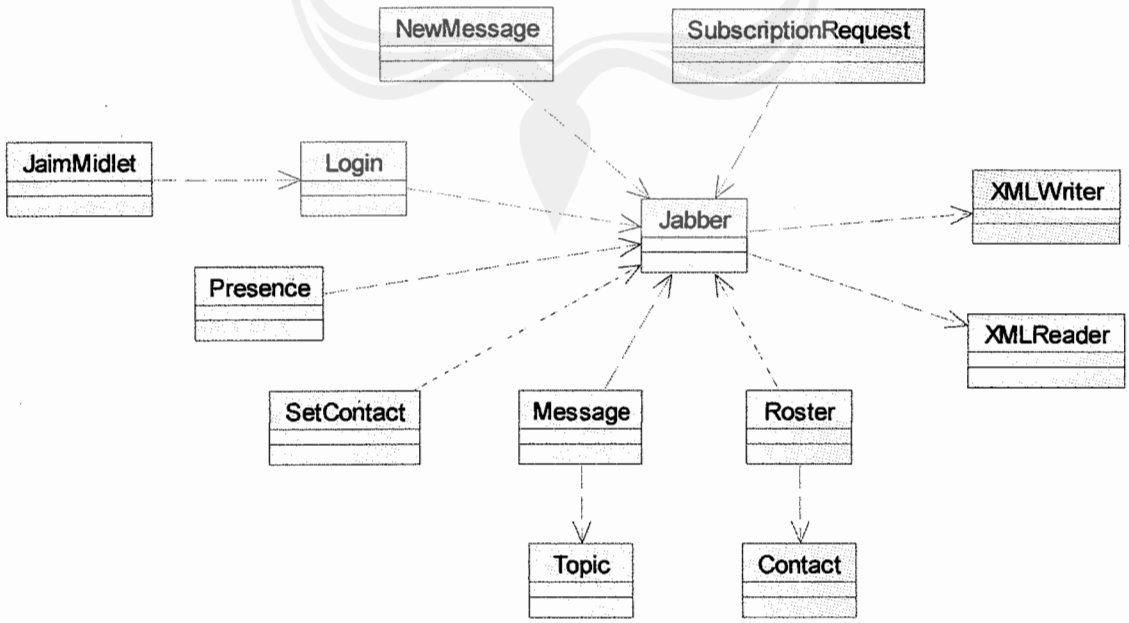
4.1 Static Structure Diagram

4.1.1 Analysis Class Diagram : Package Dependencies



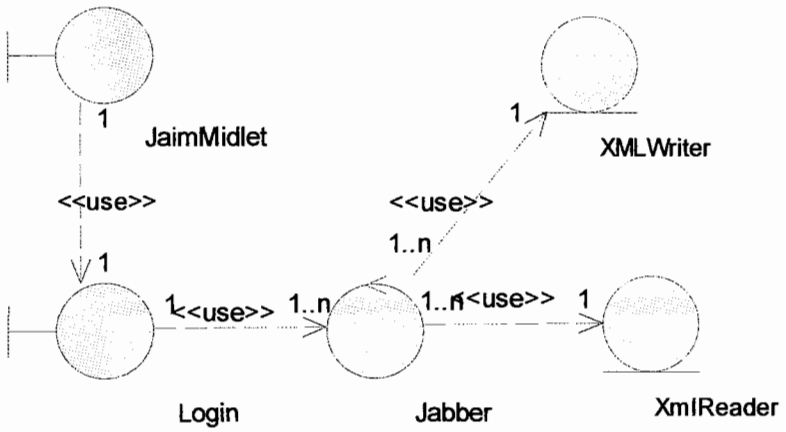
Gambar 4.1 *Analysis Class : Package Dependencies*

4.1.2 Analysis Class Diagram



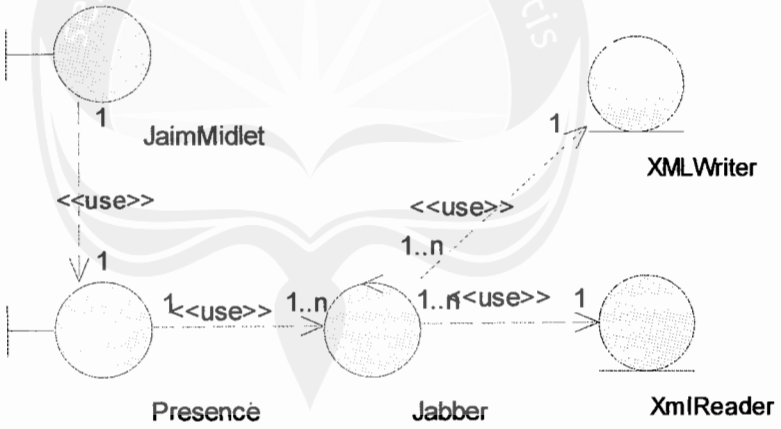
Gambar 4.2 *Analysis Class*

4.1.3 Analysis Class Diagram : Package Login



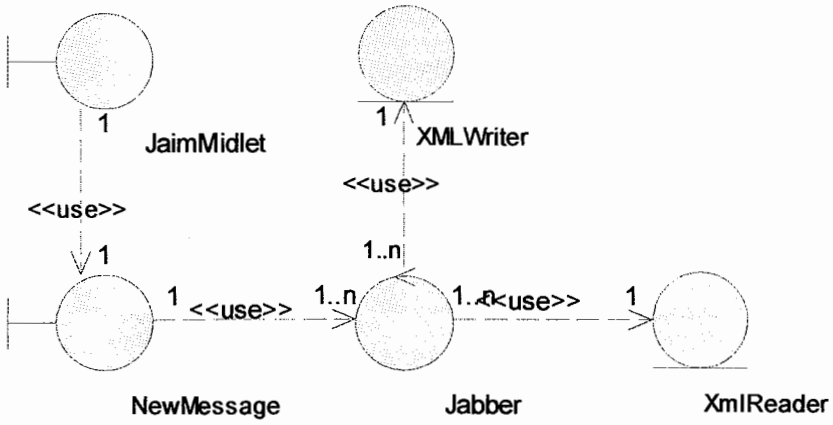
Gambar 4.3 *Analysis Class : Package Login*

4.1.4 Analysis Class Diagram : Package Kirim Presence



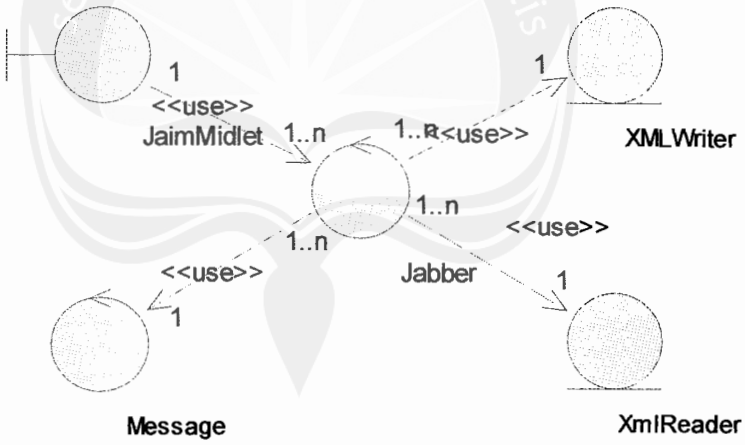
Gambar 4.4 *Analysis Class : Package* Kirim *Presence*

4.1.5 Analysis Class Diagram : Package Kirim Pesan



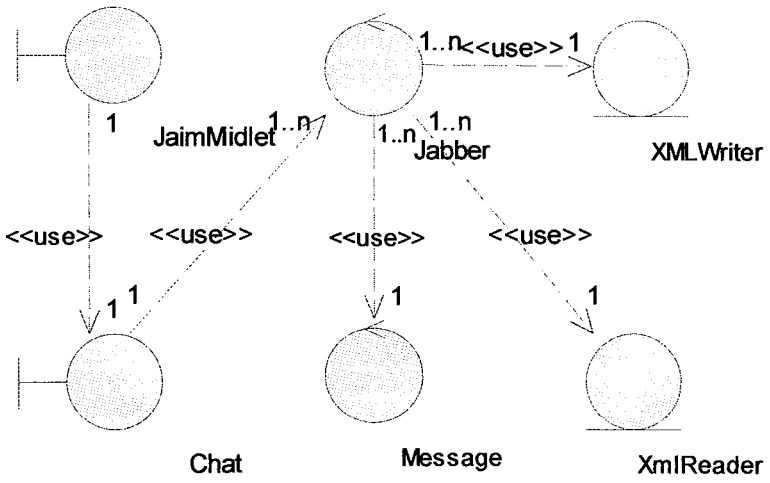
Gambar 4.5 *Analysis Class : Package* Kirim Pesan

4.1.6 Analysis Class Diagram : Package Terima Pesan



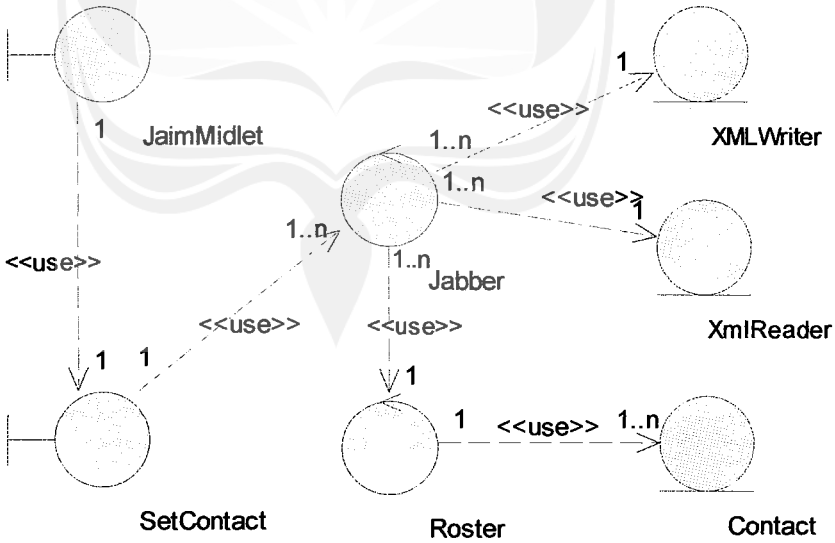
Gambar 4.6 *Analysis Class : Package* Terima Pesan

4.1.7 Analysis Class Diagram : Package Chat



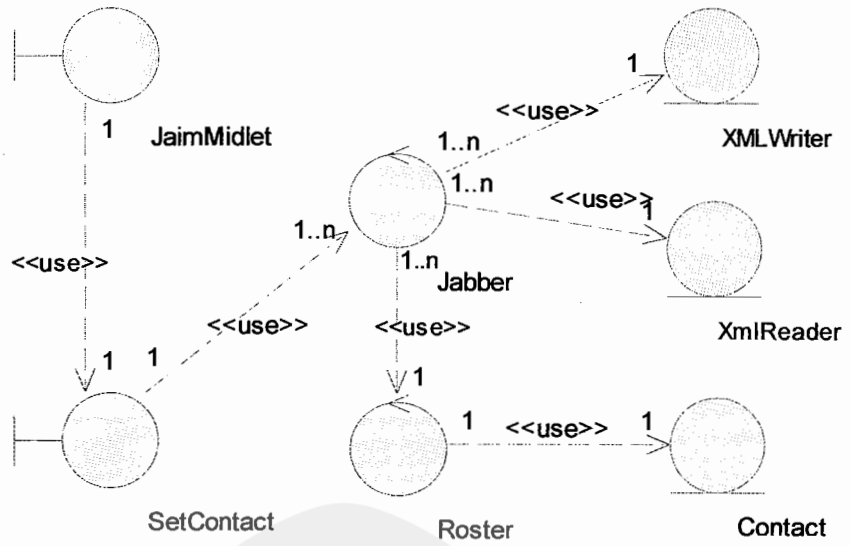
Gambar 4.7 *Analysis Class : Package Chat*

4.1.8 Analysis Class Diagram : Package Tambah Kontak



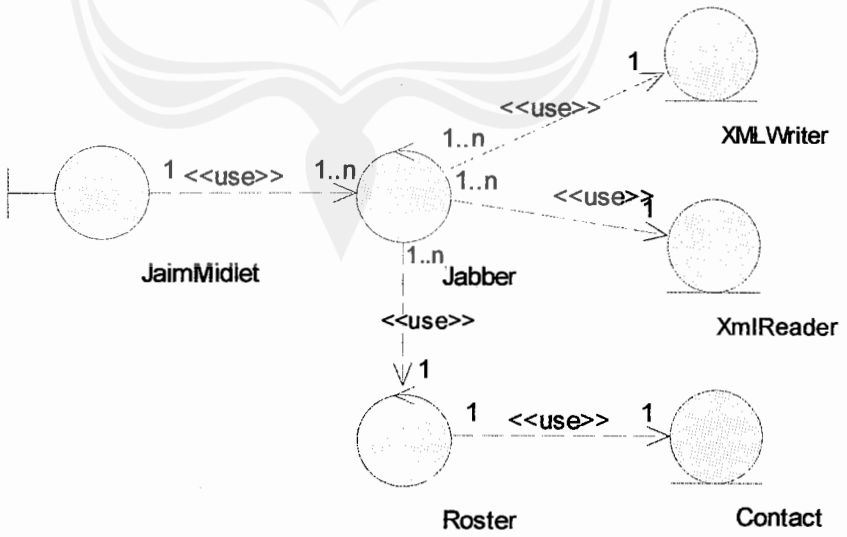
Gambar 4.8 *Analysis Class : Package* Tambah Kontak

4.1.9 Analysis Class Diagram : Package Edit Kontak



Gambar 4.9 *Analysis Class : Package* Edit Kontak

4.1.10 Analysis Class Diagram : Package Hapus Kontak



Gambar 4.10 *Analysis Class : Package* Hapus Kontak

4.1.11 Analysis Class Diagram : Package Melihat Help

Gambar 4.11 *Analysis Class : Package* Melihat Help

4.1.12 Analysis Class Diagram : Package Melihat Info Pembuat

Gambar 4.12 *Analysis Class : Package* Melihat Info Pembuat

DPPL

DESKRIPSI PERANCANGAN PERANGKAT LUNAK

Pembangunan Aplikasi Instant Messaging Client Berbasis Jabber pada Ponsel (JAIM)

Dipersiapkan oleh:

I MADE HEDI JULIARTA 03155 / TF

Jurusan Teknik Informatika

Universitas Atma Jaya Yogyakarta

Jalan Babarsari 44, Yogyakarta 55281

	Program Studi Teknik Informatika	Nomor Dokumen	Halaman
		DPPL- JAIM	1/50
		Revisi	

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	
G	

INDEX	-	A	B	C	D	E	F	G
TGL								
Ditulis oleh								
Diperiksa oleh								
Disetujui oleh								

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi

Daftar Isi

Daftar Gambar

Daftar Tabel

1 Pendahuluan

1.1 Tujuan

Dokumen Deskripsi Perancangan Perangkat Lunak (DPPL) bertujuan untuk mendokumentasikan perancangan dari perangkat lunak yang akan dikembangkan. Dokumen ini digunakan oleh pengembang perangkat lunak sebagai acuan pada tahap implementasi perangkat lunak selanjutnya.

1.2 Lingkup Dokumen

JAIM (Aplikasi *Instant Messsaging* Berbasis Jabber pada Ponsel) adalah perangkat lunak pada telepon selular (ponsel) yang digunakan untuk mengirim dan menerima pesan, *presence*, dan *chatting* antara dua buah ponsel atau antara ponsel dengan piranti yang lain seperti *Laptop*, PDA, dan PC. Aplikasi ini dijalankan pada ponsel yang mendukung Java dan GPRS, dan pesan yang dikirim hanya berupa teks.

1.3 Istilah dan Singkatan

Daftar definisi akronim dan singkatan :

Keyword/Phrase	Definisi
DPPL	Deskripsi Perancangan Perangkat Lunak disebut juga Software Design Description (SDD) merupakan deskripsi dari perancangan produk/perangkat lunak yang akan dikembangkan.
JAIM	Aplikasi *Instant Messaging* berbasis Jabber pada Ponsel

1.4 Referensi

Referensi yang digunakan pada perangkat lunak ini adalah :

1. GL02, Template Deskripsi Perancancangan Perangkat Lunak, Jurusan Teknik Informatika-ITB

2. Hartanto, Antonius Aditya, Pemprograman Mobile Java dengan MIDP 2.0, 2003.
3. Suyoto, Membuat Sendiri Aplikasi Ponsel, 2005.

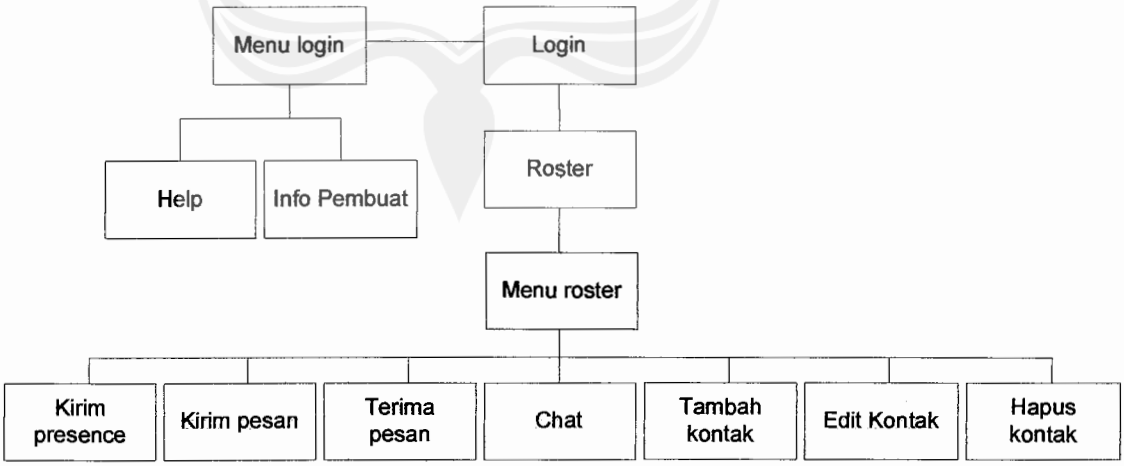
1.5 Deskripsi Umum Dokumen

Dokumen ini terdiri dari empat bab. Bab pertama adalah **Pendahuluan**, yang berisi deskripsi dokumen. Bab kedua adalah **Deskripsi Perancangan Arsitektural**, yang berisi deskripsi arsitektur sistem. Bab ketiga adalah **Deskripsi Perancangan Persistent Data**, yang berisi deskripsi data-data yang akan disimpan pada persistent storage. Bab keempat adalah **Deskripsi Perancangan Antarmuka**, yang berisi deskripsi rancangan GUI yang digunakan sistem untuk berinteraksi dengan *user*.

2. Deskripsi Arsitektural

2.1. Deployment Diagram

Deployment Diagram ini dibuat untuk menunjukkan *node* dengan sistem, hubungan di antara mereka, dan proses yang akan dijalankan di masing-masing *node*.



Gambar 2.1 *Deployment Diagram* JAIM

1. *Login*

   Merupakan fungsi yang digunakan oleh *user* untuk dapat menggunakan aplikasi dan juga untuk melakukan auntentifikasi ke *server*.

2. Kirim *presence*

   Merupakan fungsi untuk mengirim *presence* ke teman kontak.

3. Kirim pesan

   Merupakan fungsi untuk mengirim pesan ke kontak yang ada di roster maupun kontak yang tidak berada di roster.

4. Terima pesan

   Merupakan fungsi untuk membaca pesan yang masuk

5. *Chat*

   Merupakan fungsi untuk melakukan obrolan (*chatting*) dengan kontak yang ada di roster.

6. Tambah kontak

   Merupakan fungsi untuk menambah teman kontak pada roster.

7. Edit kontak

   Merupakan fungsi untuk mengedit profil dari kontak yang dipilih.

8. Hapus kontak

   Merupakan fungsi untuk menghapus kontak dari roster.

9. Membaca bantuan.

   Merupakan fungsi untuk membaca teks bantu cara penggunaan aplikasi JAIM.

10. Melihat info pembuat.

    Merupakan fungsi untuk membaca profil pembuat aplikasi.

2.2 Design Class

2.2.1 Pengantar

Nama *class* yang digunakan dalam *design class* adalah nama *class* yang *valid*, termasuk nama *package*-nya. Untuk *class-class* yang berasal dari framework Java juga digunakan nama *class* dengan *package* lengkap. Tipe primitive seperti integer atau double hanya akan dituliskan tanpa *package* dan diawali dengan huruf non-kapital. Untuk penjelasan tipe data yang utuh dapat dilihat pada bagian deskripsi *class*, sedangkan gambar *design class* tidak akan menggunakan nama *package* yang lengkap.

Stereotype yang digunakan dalam *design class* adalah :

- << *boundary* >>

  Boundary class merupakan *class* yang berfungsi untuk menghubungkan sistem dengan *user* di luar sistem.

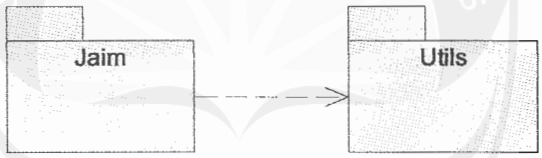
- << *control* >>

  Control class adalah suatu *class* yang objek-nya melakukan interaksi antar sekelompok objek lain. *Control class* biasanya memiliki karakteristik yang spesifik untuk satu *use case*, dan objek *class* ini biasanya hanya aktif pada realisasi *use case*.

- << *entity* >>

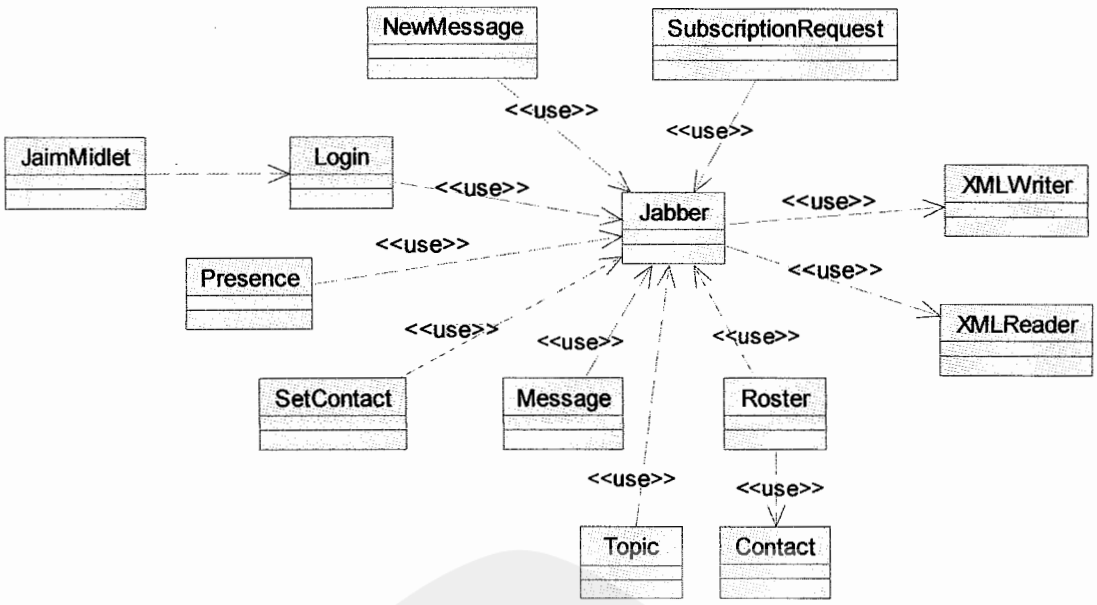
  Entity class adalah *class* yang bersifat pasif, dalam arti *class* tersebut tidak memulai interaksi dengan *class* lain. *Entity class* ini biasanya merepresentasikan suatu objek yang disimpan dalam *persistent storage*.

2.2.2 Package Dependencies



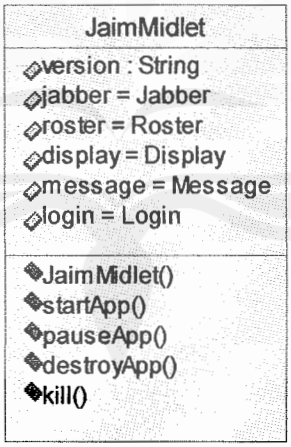
Gambar 2.2 *Package Dependencies* JAIM

2.2.3 Class Diagram



Gambar 2.3 *Class Diagram* JAIM

2.2.3.1 Class JaimMidlet



Gambar 2.4 *Class* JaimMidlet

Deskripsi

Kelas JaimMIDlet pada gambar 2.4 merupakan *class main program*. JaimMidlet merupakan kelas yang mengendalikan seluruh aplikasi.

Atribut

- version : String

Merupakan variabel untuk mendeklarasikan versi aplikasi.

- jabber : Jabber

  Variabel untuk mendeklarasikan kelas Jabber.
- roster : Roster

  Variabel untuk mendeklarasikan kelas Roster.
- display : Display

  Variabel untuk mendeklarasikan kelas Display.
- message : Message

  Variabel untuk mendeklarasikan kelas Message.
- login : Login

  Variabel untuk mendeklarasikan kelas Login.

Method

- jaimMIDlet()

  Merupakan konstruktor kelas jaimMidlet.
- startApp()

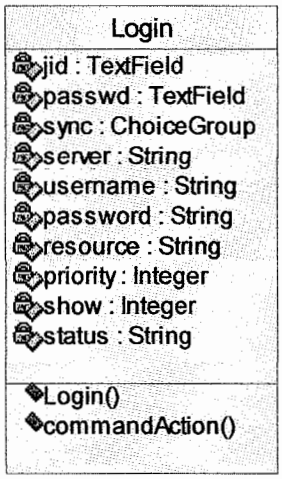
  Merupakan method standar yang dipanggil ketika MIDlet akan dijalankan.
- pauseApp()

  Merupakan method yang digunakan untuk menghentikan MIDlet sementara.
- destroyApp()

  Merupakan method yang dipanggil untuk mengakhiri MIDlet apabila terjadi kesalahan dan mengakhiri MIDlet yang sedang aktif.
- kill()

  Merupakan method yang memanggil fungsi destroyApp() dan fungsi notifikasi destroy.

2.2.3.2 Class Login



Gambar 2.5 *Class* Login

Deskripsi

Kelas Login digunakan untuk menyimpan data login di memori dan memanggil kelas Jabber untuk melakukan autentifikasi ke server.

Atribut

- jid : TextField
  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan Jid.
- passwd : TextField
  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan password
- sync : Choicegroup
  Variabel untuk memilih mengembalikan roster yang disimpan sebelumnya.
- server : String
  Variabel untuk mendeklarasi nama *server*.
- username : String
  Variabel untuk mendeklarasi nama *user*.
- password : String
  Variabel untuk mendeklarasi *password*
- resource : String

Variabel untuk mendeklarasi *resource* yang digunakan.

- priority : Integer

  Variabel untuk mendeklarasi prioritas dari *resource* yang digunakan

- show : Integer

  Variabel untuk mendeklarasi *presence* yang ditampilkan.

- status : String

  Variabel untuk mendeklarasi nama *presence* yang ditampilkan oleh variabel show.

Method

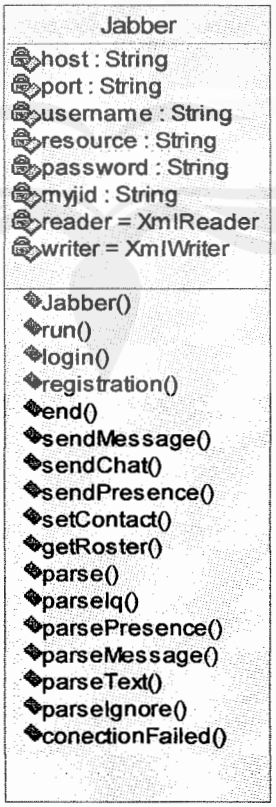
- Login()

  Konstruktor kelas Login.

- commandAction()

  Method untuk aksi-aksi tombol.

2.2.3.3 Class Jabber



Gambar 2.6 *Class* Jabber

Deskripsi

Kelas Jabber bertugas antara lain untuk melakukan koneksi ke *server*, autentifikasi, registrasi, mengirim *stanza* dan *parsing* data dari *server*.

Atribut

- host : String
  Variabel untuk mendeklarasikan nama *host*.
- port : String
  Variabel untuk mendeklarasikan nomor *port*.
- username : String
  Variabel untuk mendeklarasikan nama *user*.
- resource : String
  Variabel untuk mendeklarasikan nama *resource*.
- password : String
  Variabel untuk mendeklarasikan password.
- myjid : String
  Variabel untuk mendeklarasikan jid yang merupakan gabungan dari *username*, *host* dan *resource*.
- reader : XmlReader
  Variabel untuk mendeklarasikan kelas XmlReader.
- writer : XmlWriter
  Variabel untuk mendeklarsikan kelas XmlWriter.

Method

- Jabber()
  Konstruktor kelas Jabber.
- run()
  Method untuk menjalankan koneksi ke *server* bila kelas Jabber dipanggil.
- login()
  Method untuk melakukan autentifikasi ke *server*.
- registration()
  Method untuk melakukan registrasi ke *server*.

- end()

  Method untuk mengakhiri koneksi ke *server*.

- sendMessage()

  Method untuk mengirim teks pesan.

- sendChat()

  Method untuk mengirim teks *chat*.

- sendPresence()

  Method untuk mengirim *presence*.

- setContact()

  Method untuk menambah kontak baru.

- getRoster()

  Method untuk mendapatkan data roster dari *server*.

- parse()

  Method yang memanggil fungsi *parsing* yang lain.

- parseIq()

  Method untuk melakukan *parsing* data iq.

- parsePresence()

  Method untuk melakukan *parsing* data *presence*.

- parseMessage()

  Method untuk melakukan *parsing message*.

- parseText()

  Method untuk melakukan *parsing text*.

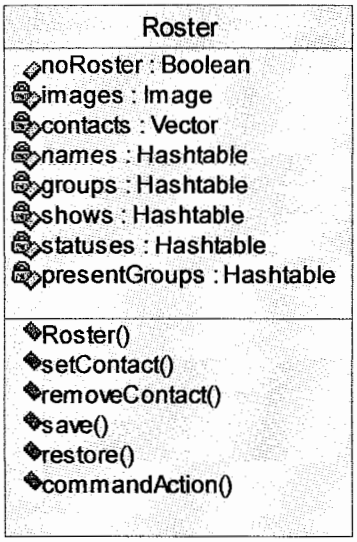
- parseIgnore()

  Method untuk melakukan *parsing* data yang tidak diketahui formatnya.

- connectionFailed()

  Method untuk memberitahukan koneksi ke *server* gagal.

2.2.3.4 Class Roster



Gambar 2.7 *Class* Roster

Deskripsi

Kelas Roster bertugas mengelola data roster.

Atribut

- noRoster : Boolean

  Variabel untuk mendeklarasikan apakah ada data roster di memori atau tidak.

- images : Image

  Variabel untuk mendeklarasikan gambar yang ditampilkan.

- contacts : Vector

  Variabel untuk mendeklarasikan sebuah *array* nama kontak.

- names : Hashtable

  Variabel untuk mendeklarasikan sebuah *hastable* untuk menyimpan nama kontak.

- groups : Hashtable

  Variabel untuk mendeklarasikan sebuah *hastable* untuk menyimpan nama group.

- shows : Hashtable

  Variabel untuk mendeklarasikan sebuah *hastable* untuk menyimpan nama presence.

- statuses : Hashtable

Variabel untuk mendeklarasikan sebuah *hastable* untuk menyimpan nama status *presence*.

- presentGroups : Hashtable

  Variabel untuk mendeklarasikan sebuah *hastable* untuk menyimpan nama kontak didalam *group*.

Method

- Roster()

  Konstruktor kelas Roster.

- setContact()

  Method untuk menambah kontak di roster.

- removeContact()

  Method untuk menghapus kontak di roster.

- registration()

  Method untuk melakukan registrasi ke *server*.

- save()

  Method untuk menyimpan data roster di memori.

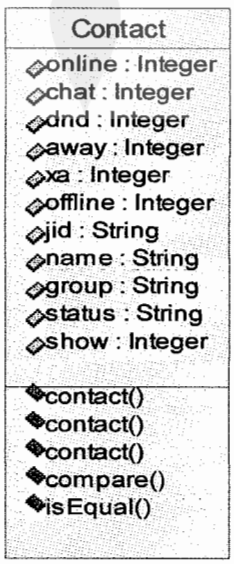
- restore()

  Method untuk memanggil data roster di memori.

- commandAction()

  Method untuk aksi-aksi tombol

2.2.3.5 Class Contact



Gambar 2.8 *Class* Contact

Deskripsi

Kelas untuk menginisialisasi data kontak.

Atribut

- online : Integer

  Variabel untuk mendeklarasikan status *presence online*.

- chat : Integer

  Variabel untuk mendeklarasikan status *presence chat*.

- dnd : Integer

  Variabel untuk mendeklarasikan status *presence dnd*.

- away : Integer

  Variabel untuk mendeklarasikan status *presence away*.

- xa : Integer

  Variabel untuk mendeklarasikan status *presence xa*.

- offline : Integer

  Variabel untuk mendeklarasikan status *presence offline*.

- jid : String

  Variabel untuk mendeklarasikan jid kontak.

- name : String

  Variabel untuk mendeklarasikan nama kontak.

- group : String

  Variabel untuk mendeklarasikan *group* kontak.

- status : String

  Variabel untuk mendeklarasikan status kontak.

- show : Integer

  Variabel untuk mendeklarasikan *presence* yang ditampilkan.

Method

- Contact()

  Konstruktor kelas Contact untuk edit kontak.

- Contact()

  Konstruktor kelas Contact untuk *presence*.

- Contact()

  Konstruktor kelas Contact untuk *group*.

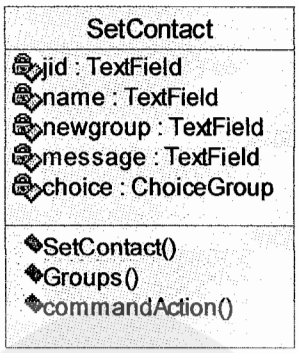
- compare()

  Method untuk membandingkan data kontak.

- isEqual()

  Method untuk mengembalikan data kontak yang sesuai.

2.2.3.6 Class SetContact



Gambar 2.9 *Class* SetContact

Deskripsi

Kelas untuk menambah kontak baru dan mengedit kontak.

Atribut

- jid : TextField

  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan Jid.

- name : TextField

  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan nama.

- jid : TextField

  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan nama *group* baru.

- jid : TextField

  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan teks pesan.

- choice : Choicegroup

  Variabel untuk mendeklarasi sebuah *choicegroup* untuk menerima inputan nama *group* yang dipilih.

Method

- SetContact()

  Konstruktor kelas SetContact.

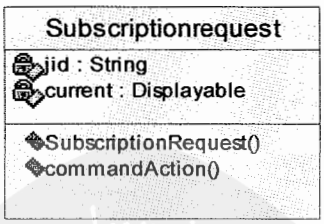
- Groups()

  Method untuk mengelola *group*.

- commandAction()

  Method untuk aksi-aksi tombol.

2.2.3.7 Class Subscriptionrequest



Gambar 2.10 *Class* Subscriptionrequest

Deskripsi

Kelas Subscriptionrequest untuk menerima atau menolak permintaan dari suatu kontak untuk menjadi teman kontak dari user.

Atribut

- jid : String

  Variabel untuk mendeklarasikan jid dari kontak yang meminta.

- current : Displayable

  Variabel untuk mendeklarasikan *form* saat ini.

Method

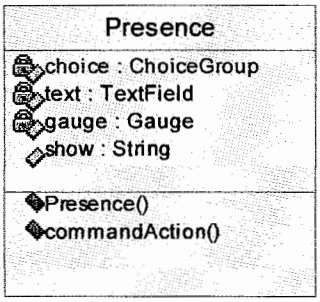
- SubscriptionRequest()

  Konstruktor kelas SubscriptionRequest.

- commandAction()

  Method untuk aksi-aksi tombol.

2.2.3.8 Class Presence



Gambar 2.11 *Class* Presence

Deskripsi

Kelas Presence bertugas untuk mengelola *presence*.

Atribut

- choice : ChoiceGroup

  Variabel untuk mendeklarasikan pilihan *presence*.

- text : TextField

  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan teks status.

- gauge : Gauge

  Variabel untuk mendeklarasikan *gaugè* untuk prioritas dari *resource*.

- show : String

  Variabel untuk mendeklarasikan *presence* yang dipilih.

Method

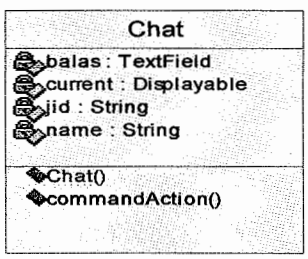
- Presence()

  Konstruktor kelas Presence.

- commandAction()

  Method untuk aksi-aksi tombol.

2.2.3.9 Class Chat



Gambar 2.12 *Class* Chat

Deskripsi

Kelas bertugas untuk mengirim data dan pesan ke *server* bila *user* melakukan *chatting*.

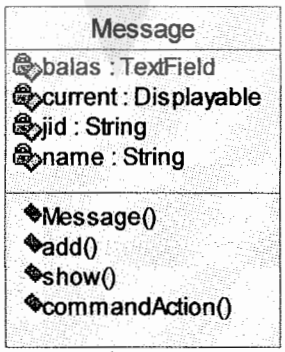
Atribut

- balas : TextField
  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan teks.
- current : Displayable
  Variabel untuk mendeklarasikan *form* saat ini.
- jid : String
  Variabel untuk mendeklarasikan jid dari kontak yang diajak melakukan obrolan.
- name : String
  Variabel untuk mendeklarasikan nama dari kontak yang diajak melakukan obrolan.

Method

- Chat()
  Konstruktor kelas Chat.
- commandAction()
  Method untuk aksi-aksi tombol.

2.2.3.10 Class Message



Gambar 2.13 *Class* Message

Deskripsi

Kelas Message bertugas untuk menerima semua pesan dan data pengirim dari *server*.

Atribut

- balas : TextField

  Variabel untuk mendeklarasi sebuah *textfield* untuk menerima inputan teks.

- current : Displayable

  Variabel untuk mendeklarasikan *form* saat ini.

- jid : String

  Variabel untuk mendeklarasikan jid dari kontak yang diajak melakukan obrolan.

- name : String

  Variabel untuk mendeklarasikan nama dari kontak yang diajak melakukan obrolan.

Method

- Message()

  Konstruktor kelas Message.

- add()

  Method untuk memasukkan pesan yang masuk *form*.

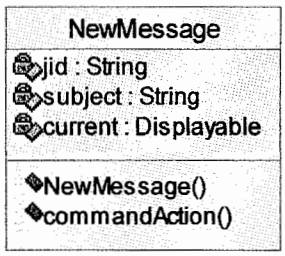
- show()

  Method untuk menampilkan *form message*.

- commandAction()

  Method untuk aksi-aksi tombol.

2.2.3.11 Class NewMessage



Gambar 2.14 *Class* NewMessage

Deskripsi

Kelas NewMessage bertugas menerima inputan teks pesan dan mengirimkan pesan ke *server*.

Atribut

- jid : String

  Variabel untuk mendeklarasikan jid dari kontak yang dikirimi pesan.

- subject : String

  Variabel untuk mendeklarasikan teks subject.

- current : Displayable

  Variabel untuk mendeklarasikan *form* saat ini.

Method

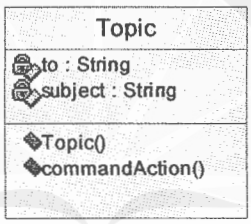
- NewMessage()

  Konstruktor kelas NewMessage.

- commandAction()

  Method untuk aksi-aksi tombol.

2.2.3.12 Class Topic



Gambar 2.15 *Class* Topic

Deskripsi

Kelas Topic bertugas untuk menerima inputan data kontak yang dikirimi pesan dan teks subject.

Atribut

- to : String

  Variabel untuk mendeklarasikan jid yang dikirimi pesan.

- subject : String

  Variabel untuk mendeklarasikan *subject*

Method

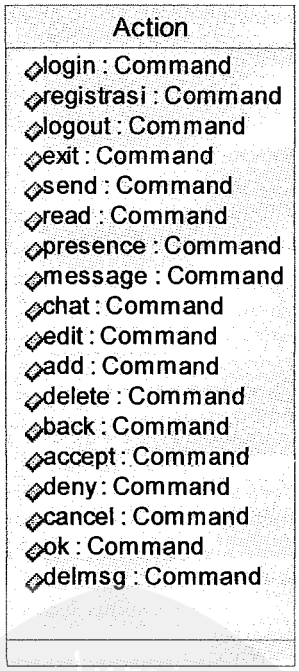
- Topic()

  Konstruktor kelas Topic.

- commandAction()

  Method untuk aksi-aksi tombol.

2.2.3.13 Class Action



Gambar 2.16 *Class* Action

Deskripsi

Kelas Action adalah kelas yang berisi kumpulan aksi-aksi tombol dari seluruh *form*.

Atribut

- login : Command

  Variabel untuk mendeklarasikan tombol login.
- registrasi : Command

  Variabel untuk mendeklarasikan tombol registrasi.
- logout : Command

  Variabel untuk mendeklarasikan tombol logout.
- exit : Command

  Variabel untuk mendeklarasikan tombol exit.
- send : Command

  Variabel untuk mendeklarasikan tombol send.
- read : Command

  Variabel untuk mendeklarasikan tombol read.
- presence : Command

  Variabel untuk mendeklarasikan tombol presence.

- message : Command

  Variabel untuk mendeklarasikan tombol message.
- chat : Command

  Variabel untuk mendeklarasikan tombol chat.
- edit : Command

  Variabel untuk mendeklarasikan tombol edit.
- add : Command

  Variabel untuk mendeklarasikan tombol add.
- delete : Command

  Variabel untuk mendeklarasikan tombol delete.
- back : Command

  Variabel untuk mendeklarasikan tombol back.
- accept : Command

  Variabel untuk mendeklarasikan tombol accept.
- deny : Command

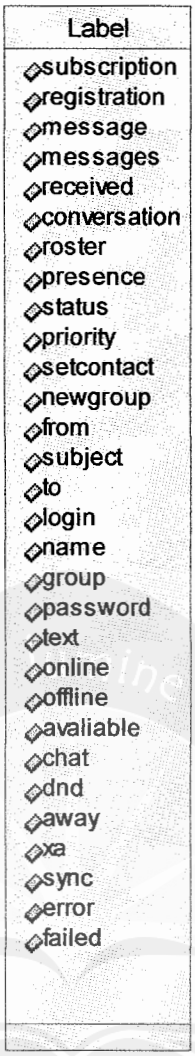
  Variabel untuk mendeklarasikan tombol deny.
- cancel : Command

  Variabel untuk mendeklarasikan tombol cancel.
- ok : Command

  Variabel untuk mendeklarasikan tombol ok.
- delmsg : Command

  Variabel untuk mendeklarasikan tombol delmsg.

2.2.3.14 Class Label



Gambar 2.17 *Class* Label

Deskripsi

Kelas Label berisi kumpulan label-label dari seluruh *form*.

Atribut

- subscription

  Variabel untuk mendeklarasikan label subscription.
- registration

  Variabel untuk mendeklarasikan label registration.
- message

  Variabel untuk mendeklarasikan label message.
- messages

Variabel untuk mendeklarasikan label messages.

- received

  Variabel untuk mendeklarasikan label received.

- conversation

  Variabel untuk mendeklarasikan label conversation.

- roster

  Variabel untuk mendeklarasikan label roster.

- presence

  Variabel untuk mendeklarasikan label presence.

- status

  Variabel untuk mendeklarasikan label status.

- priority

  Variabel untuk mendeklarasikan label priority.

- setcontact

  Variabel untuk mendeklarasikan label setcontact.

- newgroup

  Variabel untuk mendeklarasikan label newgroup.

- from

  Variabel untuk mendeklarasikan label from.

- subject

  Variabel untuk mendeklarasikan label subject.

- to

  Variabel untuk mendeklarasikan label to.

- login

  Variabel untuk mendeklarasikan label login.

- name

  Variabel untuk mendeklarasikan label name.

- group

  Variabel untuk mendeklarasikan label group.

- password

  Variabel untuk mendeklarasikan label password.

- text

Variabel untuk mendeklarasikan label text.

- online

  Variabel untuk mendeklarasikan label online.

- offline

  Variabel untuk mendeklarasikan label offline.

- available

  Variabel untuk mendeklarasikan label available.

- chat

  Variabel untuk mendeklarasikan label chat.

- dnd

  Variabel untuk mendeklarasikan label dnd.

- away

  Variabel untuk mendeklarasikan label away.

- xa

  Variabel untuk mendeklarasikan label xa.

- sync

  Variabel untuk mendeklarasikan label sync.

- error

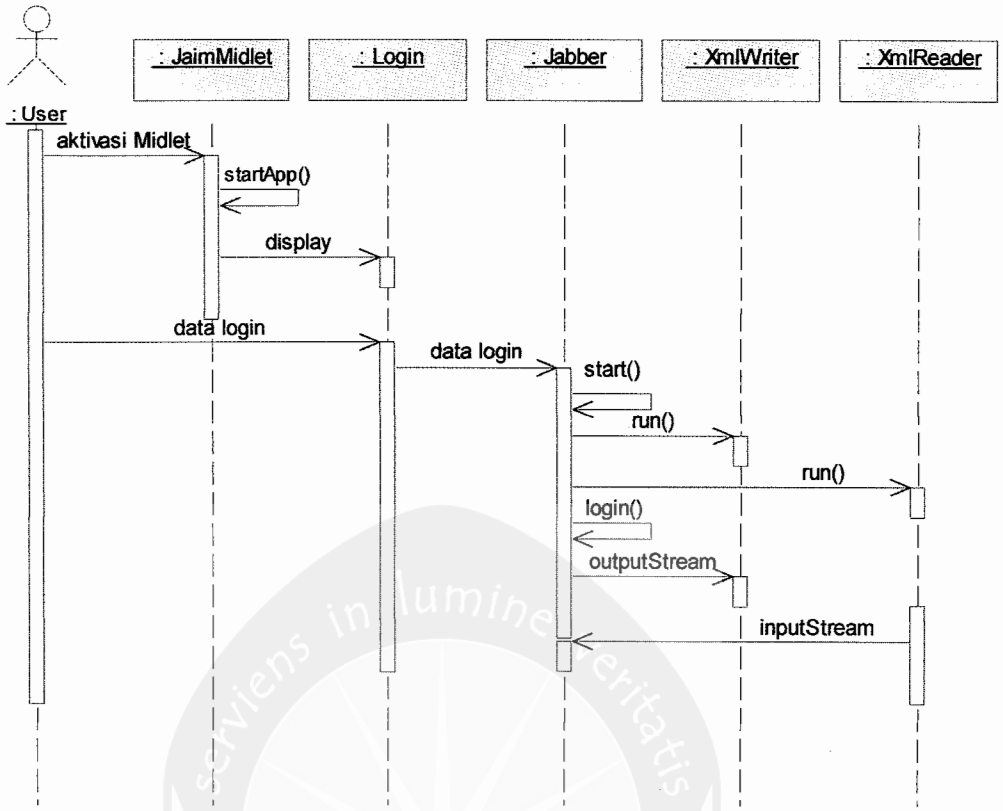
  Variabel untuk mendeklarasikan label error.

- failed

  Variabel untuk mendeklarasikan label failed.

2.3 Realisasi Use Case

2.3.1 Use Case:Login



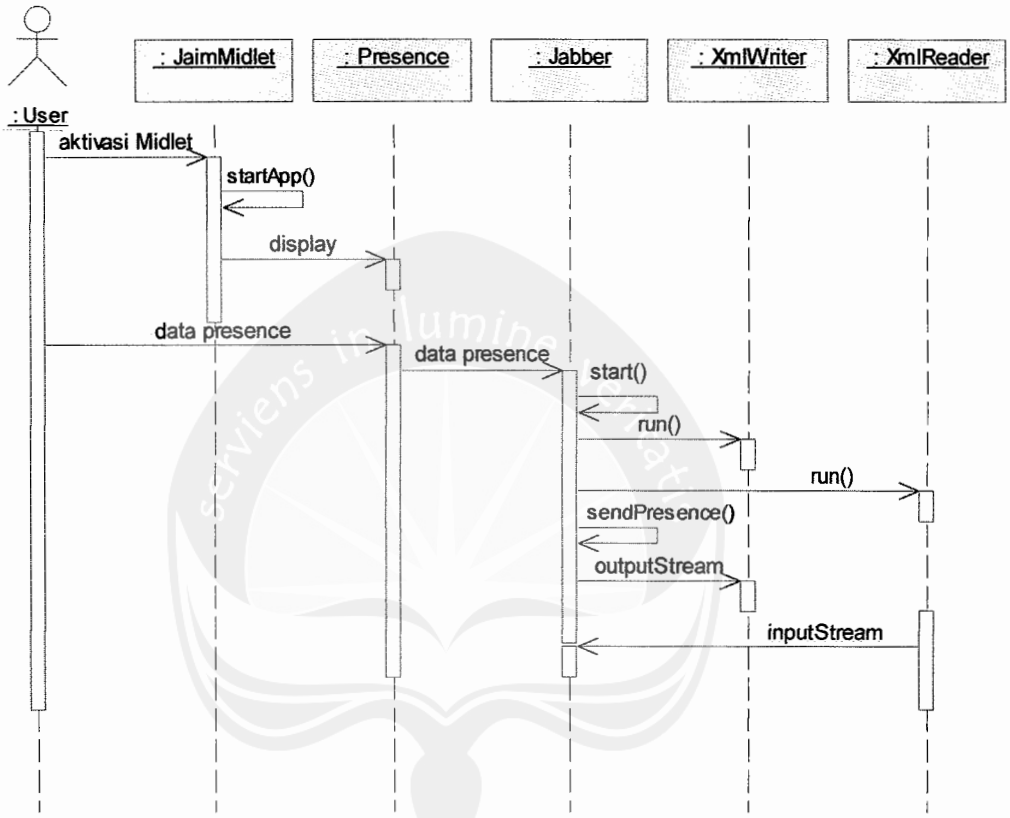
Gambar 2.18 Desain *sequence diagram use case Login*

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* Login.
3. *User* memasukkan data login yang terdiri dari Jabber ID (jid) dan *password* untuk melakukan aktivasi ke *server* pada *form login*.
4. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.
5. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
6. Kelas Jabber menjalankan fungsi login() untuk melakukan aktivasi ke *server*.

7. Data login ditulis kedalam bentuk format xml (*stanza*) oleh kelas XmlWriter dan kemudian dikirim ke *server*.
8. *Stanza* aktivasi diterima oleh kelas XmlReader dari *server* dan kemudian oleh kelas Jabber *stanza* Xml tersebut dibaca.

2.3.2 Use Case : Kirim Presence



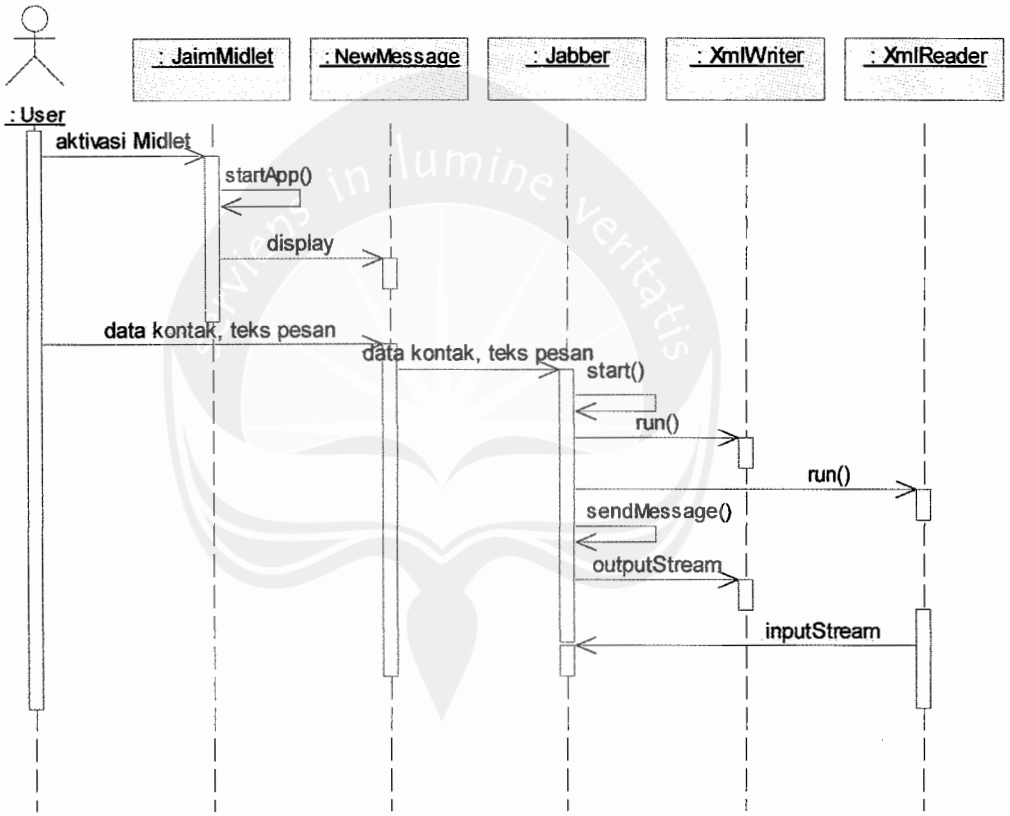
Gambar 2.19 Desain *sequence diagram use case* Kirim presence

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* Presence.
3. *User* memasukkan data *presence* yaitu berupa pilihan *online, offline, away, dnd* dan *xa*.
4. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.

5. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
6. Kelas Jabber menjalankan fungsi sendPresence() untuk mengirim *presence* ke *server*.
7. Data *presence* ditulis kedalam bentuk *format* xml (*stanza*) oleh kelas XmlWriter dan kemudian dikirim ke *server*.
8. *Stanza* balasan diterima oleh kelas XmlReader dari *server* dan kemudian oleh kelas Jabber *stanza* Xml tersebut dibaca.

2.3.3 Use Case : Kirim Pesan



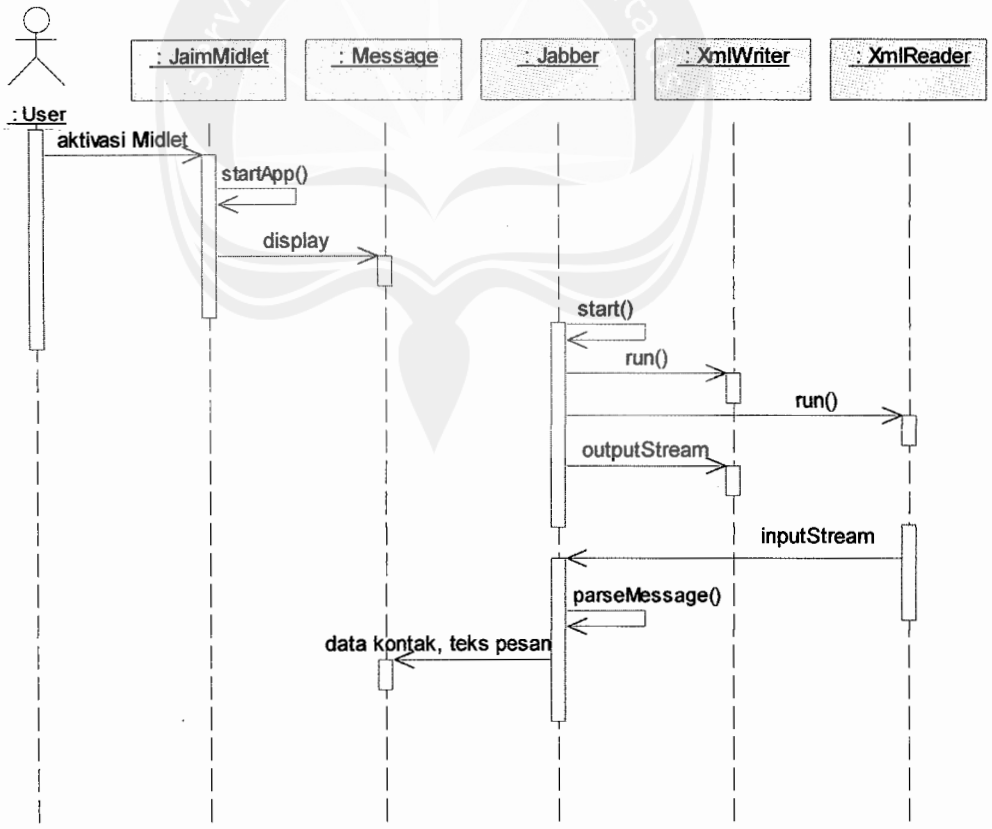
Gambar 2.20 Desain *sequence diagram use case* Kirim pesan

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* NewMessage.

3. *User* memasukkan data kontak yang akan dikirimi pesan dan teks pesannya.
4. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.
5. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
6. Kelas Jabber menjalankan fungsi sendMessage() untuk mengirim pesan ke *server*.
7. Data kontak dan pesan ditulis kedalam bentuk format xml (*stanza*) oleh kelas XmlWriter dan kemudian dikirim ke *server*.
8. *Stanza* balasan diterima oleh kelas XmlReader dari *server* dan kemudian oleh kelas Jabber *stanza* Xml tersebut dibaca.

2.3.4 Use Case : Terima Pesan

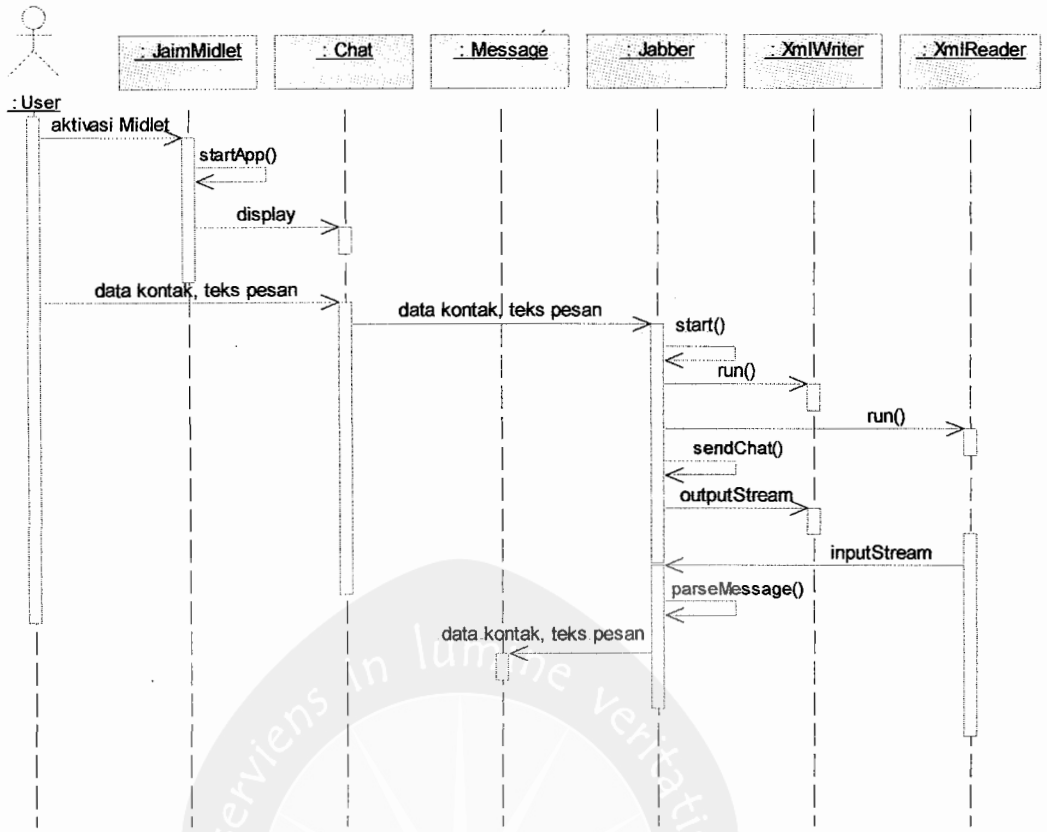


Gambar 2.21 Desain *sequence diagram use case* Terima pesan

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* Message.
3. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.
4. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
5. Kelas Jabber menjalankan fungsi sendMessage() untuk mengirim pesan ke *server*.
6. *Stanza* pesan masuk diterima oleh kelas XmlReader dari *server*.
7. *Stanza* Xml tersebut kemudian dibaca dengan fungsi parseMessage() pada kelas Jabber.
8. Pesan dan data pengirim kemudian ditampilkan pada *form* Message.

2.3.5 Use Case : Chat



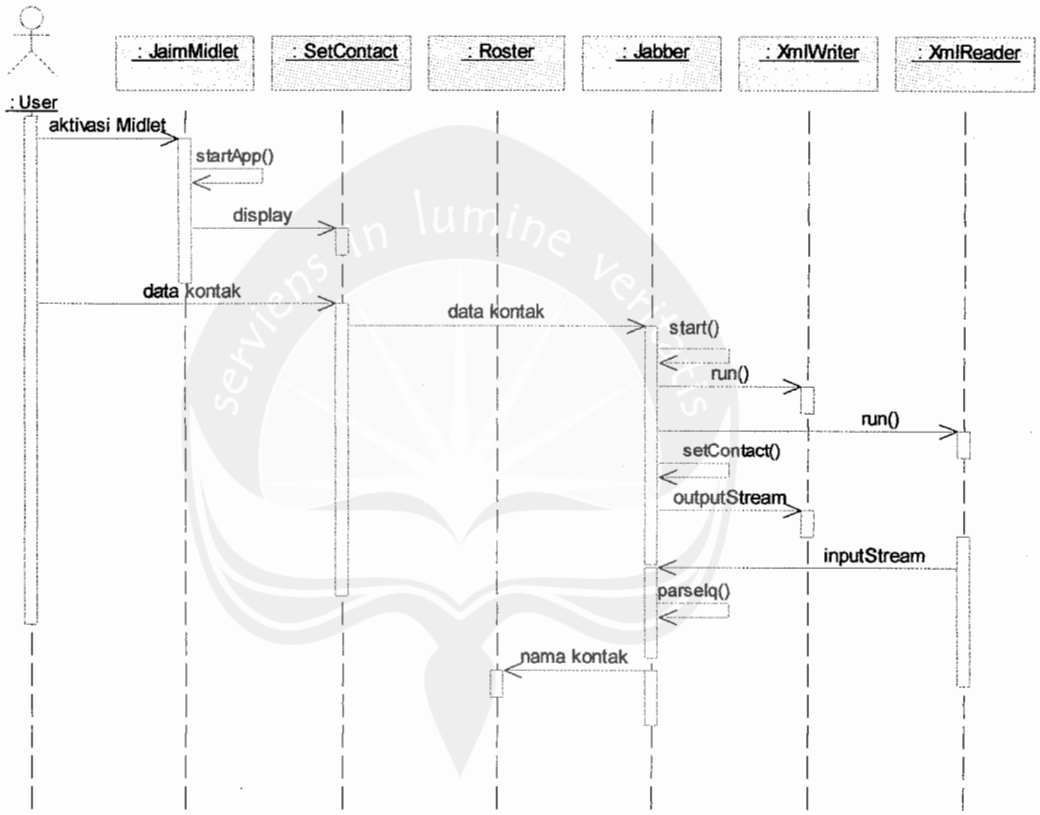
Gambar 2.22 Desain *sequence diagram use case* Chat

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* Chat.
3. *User* memasukkan data kontak yang akan diajak *chatting* dan teks *chat*-nya.
4. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.
5. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
6. Kelas Jabber menjalankan fungsi sendChat() untuk mengirim pesan *chat* ke *server*.

7. Data kontak dan pesan *chat* ditulis kedalam bentuk format xml (*stanza*) oleh kelas XmlWriter dan kemudian dikirim ke *server*.
8. *Stanza* balasan *chat* diterima oleh kelas XmlReader dari *server* dan kemudian oleh kelas Jabber *stanza* Xml tersebut dibaca dengan fungsi parseMessage().
9. Pesan balasan ditampilkan pada *form* Message.

2.3.6 Use Case : Tambah Kontak



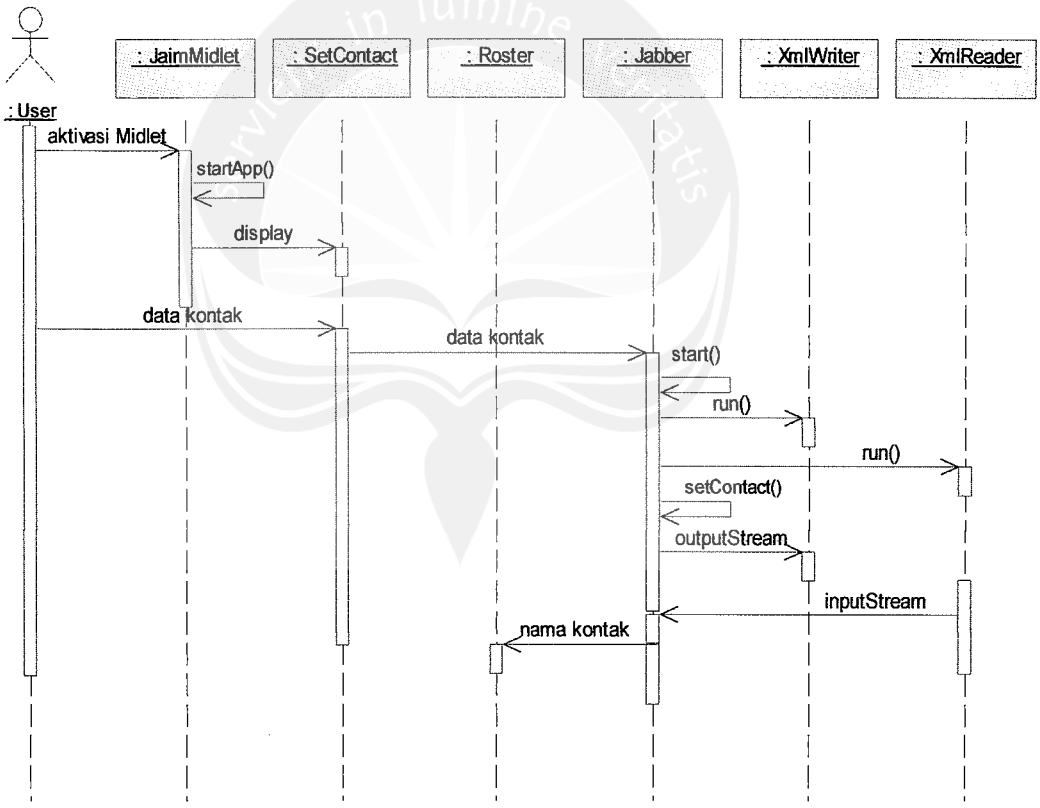
Gambar 2.23 Desain *sequence diagram use case* Tambah kontak

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* SetContact.
3. *User* memasukkan data kontak yang ingin ditambahkan.

4. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.
5. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
6. Kelas Jabber menjalankan fungsi setContact() untuk mengirim data kontak ke *server*.
7. Data kontak ditulis kedalam bentuk format xml (*stanza*) oleh kelas XmlWriter dan kemudian dikirim ke *server*.
8. *Stanza* balasan diterima oleh kelas XmlReader dari *server* dan kemudian oleh kelas Jabber *stanza* Xml tersebut dibaca dengan fungsi parseIq().
9. Kontak baru ditampilkan pada Roster.

2.3.7 Use Case : Edit kontak



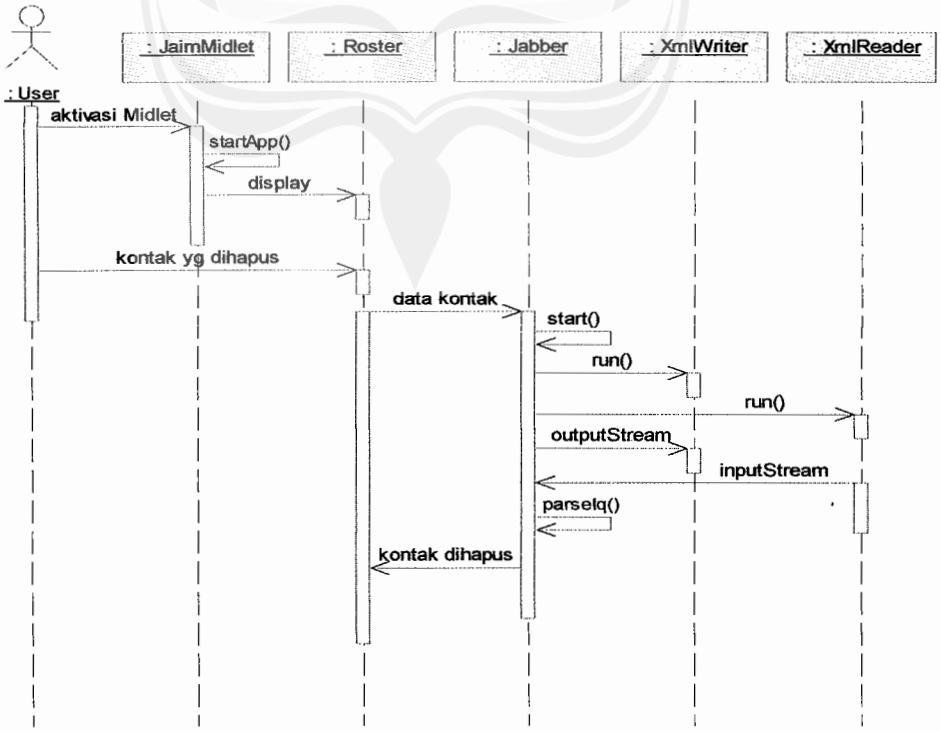
Gambar 2.24 Desain *sequence diagram use case* Edit kontak

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().

2. Midlet memanggil fungsi display untuk menampilkan Roster.
3. *User* memilih kontak yang akan diedit
4. *User* memasukkan data kontak yang baru pada form SetContact.
5. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.
6. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
7. Kelas Jabber menjalankan fungsi setContact() untuk mengirim data kontak ke *server*.
8. Data kontak ditulis kedalam bentuk format xml (*stanza*) oleh kelas XmlWriter dan kemudian dikirim ke *server*.
9. *Stanza* balasan diterima oleh kelas XmlReader dari *server* dan kemudian oleh kelas Jabber *stanza* Xml tersebut dibaca dengan fungsi parseIq().
10. Kontak baru ditampilkan pada Roster.

2.3.8 Use Case : Hapus kontak

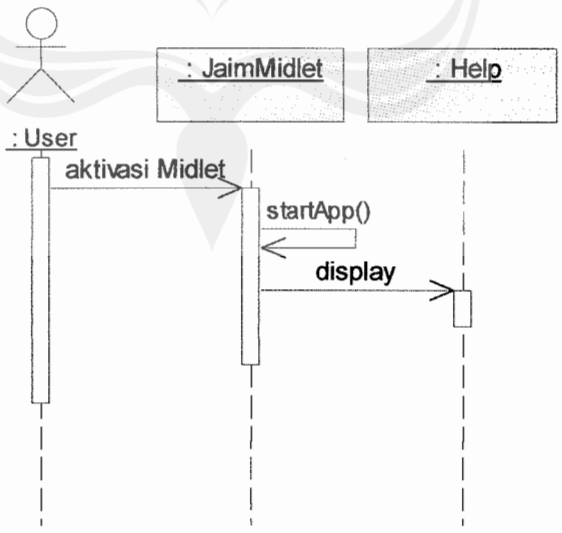


Gambar 2.25 Desain *sequence diagram use case* Hapus kontak

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* Roster.
3. *User* memilih kontak yang ingin akan dihapus.
4. Kelas Jabber menjalankan fungsi start() untuk koneksi ke *server*.
5. Kelas Jabber memanggil kelas XmlWriter dan XmlReader.
6. Kelas Jabber menjalankan fungsi setContact() untuk mengirim data kontak yang akan dihapus ke *server*.
7. Data kontak ditulis kedalam bentuk format xml (*stanza*) oleh kelas XmlWriter dan kemudian dikirim ke *server*.
8. *Stanza* balasan diterima oleh kelas XmlReader dari *server* dan kemudian oleh kelas Jabber *stanza* Xml tersebut dibaca dengan fungsi parseIq().
9. Kontak yang dihapus tidak tampil di Roster.

2.3.9 Use Case : Membaca bantuan

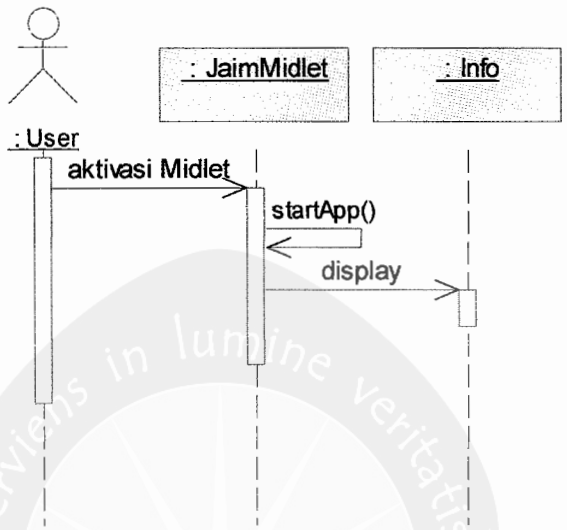


Gambar 2.26 Desain *sequence diagram use case* Membaca bantuan

Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* Help.

2.3.10 Use Case : Melihat info pembuat



Gambar 2.27 Desain *sequence diagram use case* Melihat info pembuat

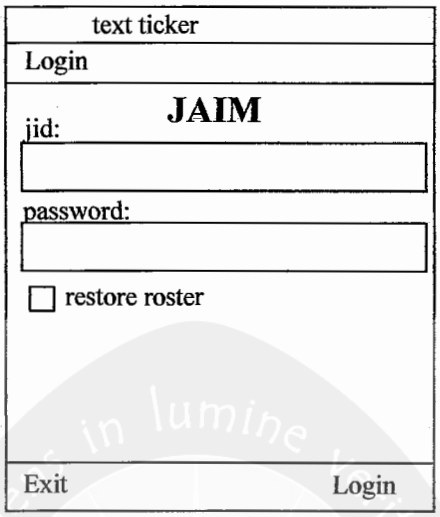
Flow of events

1. *User* menjalankan aplikasi dan midlet menjalankan metode startApp().
2. Midlet memanggil fungsi display untuk menampilkan *form* Info.

3. Deskripsi Antarmuka

Pada perancangan antarmuka ini dibagi menjadi beberapa *form*, yang tiap - tiap *form* terdiri dari berbagai obyek yang berfungsi untuk menampilkan informasi tertentu. Penjelasan dari tiap-tiap *form* tersebut adalah sebagai berikut :

3.1 Use Case : Login



Gambar 3.1 Rancangan antarmuka *use case* Login

Deskripsi

- Rancangan antarmuka pada gambar 3.1 diimplementasikan pada *class* Login, antarmuka ini digunakan untuk menampilkan *form* Login untuk melakukan aktivasi ke *server*.

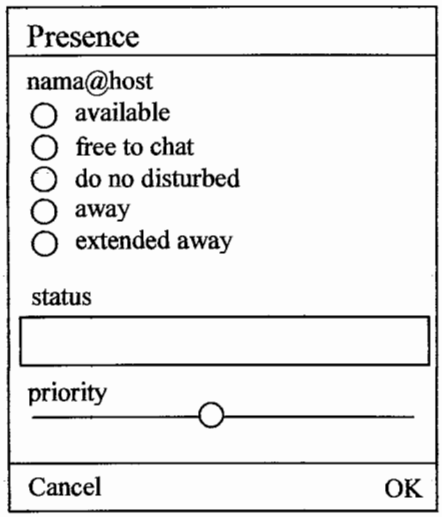
Event

- *Login*

  Urutan aksi yang terjadi :

  1. *User* menekan *icon* JAIM.
  2. JAIM menampilkan *form* Login.
  3. *User* mengisi *form* Login dengan jid, *password* dan memilih apakah mengembalikan roster yang tersimpan dimemori atau tidak.
  4. *User* menekan tombol login untuk aktivasi ke server dan masuk ke *form* Roster atau menekan tombol exit untuk keluar dari JAIM.

3.2 Use Case : Kirim Presence



Gambar 3.2 Rancangan antarmuka *use case* Kirim presence

Deskripsi

- Rancangan antarmuka pada gambar 3.2 diimplementasikan pada *class* Presence, antarmuka ini digunakan untuk menampilkan *form* Presence untuk mengganti *presence*.

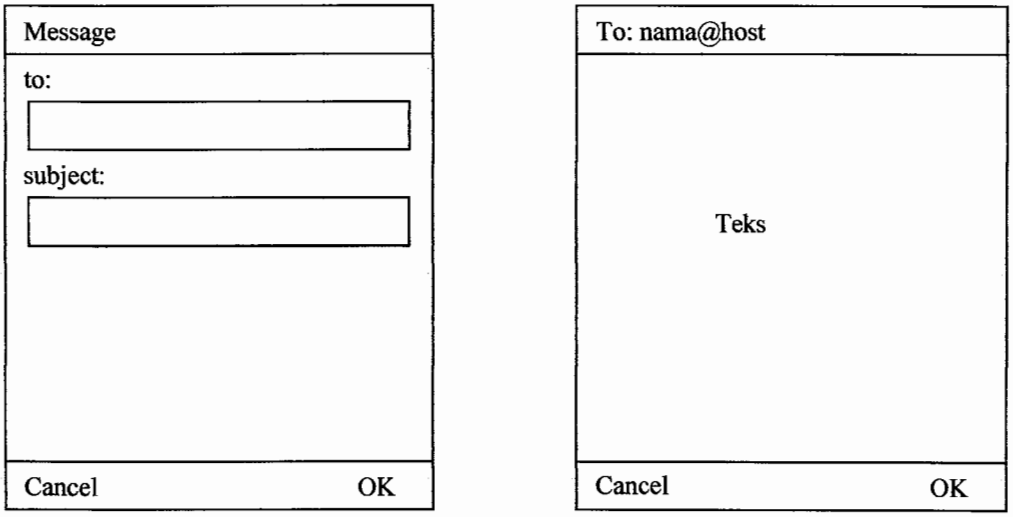
Event

- Kirim *presence*

  Urutan aksi yang terjadi :

  1. *User* memilih menu presence.
  2. JAIM menampilkan *form* Presence.
  3. *User* memilih pilihan *presence*, mengisi teks status dan memilih prioritas *resource*.
  4. *User* menekan tombol OK untuk mengirim *presence* ke *server* dan kembali ke *form* Roster atau menekan tombol cancel untuk membatalkan mengganti *presence*.

3.3 Use Case : Kirim pesan



Gambar 3.3 Rancangan antarmuka *use case* Kirim pesan

Deskripsi

- Rancangan antarmuka pada gambar 3.3 diimplementasikan pada *class* NewMessage dan Topic, antarmuka ini digunakan untuk menampilkan *form* Topic dan *form* Message.

Event

- Kirim pesan
  - Skenario1

  Urutan aksi yang terjadi :

  1. *User* memilih kontak dari roster kemudian memilih menu send.
  2. JAIM menampilkan *form* Message.
  3. *User* mengisi *form* dengan teks pesan.
  4. *User* menekan tombol OK untuk mengirim pesan ke *server* dan kembali ke *form* roster atau menekan tombol cancel untuk membatalkan mengirim pesan.

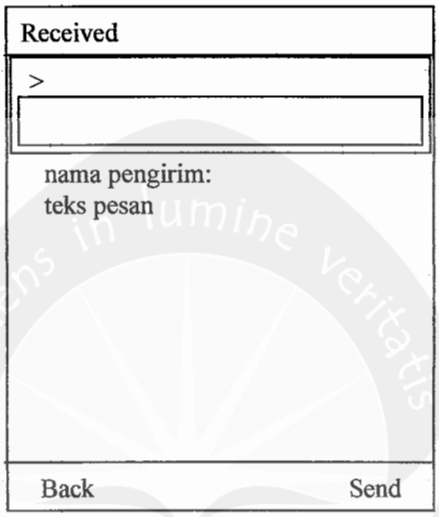
  - Skenario2

  Urutan aksi yang terjadi :

  1. *User* memilih menu message.
  2. JAIM menampilkan *form* topic.

3. *User* mengisi *form* Topic dengan kontak yang dikirimi pesan dan *subject* pesan
4. *User* menekan tombol OK dan masuk ke *form* Message atau menekan tombol cancel untuk membatalkan.
5. *User* mengisi *form* Message dengan teks pesan.
6. *User* menekan tombol OK untuk mengirim pesan ke *server* dan kembali ke *form* Roster atau menekan tombol cancel untuk membatalkan mengirim pesan.

3.4 Use Case : Terima pesan



Gambar 3.4 Rancangan antarmuka *use case* Terima pesan

Deskripsi

- Rancangan antarmuka pada gambar 3.4 diimplementasikan pada *class* Message, antarmuka ini digunakan untuk menampilkan *form* Received untuk membaca pesan.

Event

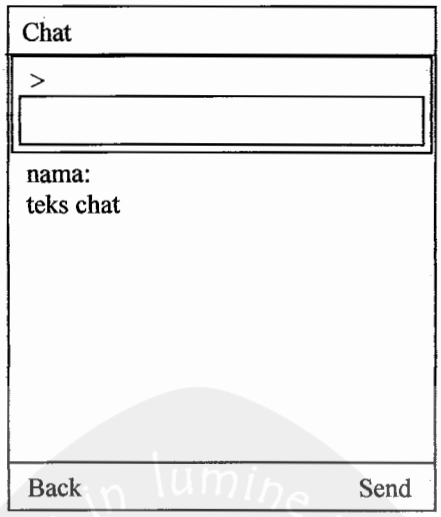
- Terima pesan

  Urutan aksi yang terjadi :

  1. *User* memilih menu read.
  2. JAIM menampilkan *form* Received.
  3. *User* membaca pesan kemudian mengisi teks balasan.

4. *User* menekan tombol OK untuk mengirim pesan ke *server* dan kembali ke *form* roster atau menekan tombol cancel untuk membatalkan.

3.5 Use Case : Chat



Gambar 3.5 Rancangan antarmuka *use case* Chat

Deskripsi

- Rancangan antarmuka pada gambar 3.5 diimplementasikan pada *class* Chat, antarmuka ini digunakan untuk menampilkan *form* Chat untuk melakukan obrolan.

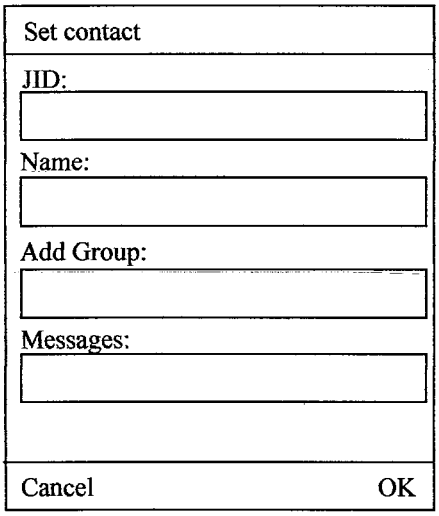
Event

- Chat

  Urutan aksi yang terjadi :

  1. *User* memilih kontak yang diajak *chatting* kemudian memilih menu chat.
  2. JAIM menampilkan *form* chat.
  3. *User* mengisi *textbox* dengan teks *chat*.
  4. *User* menekan tombol send untuk mengirim teks *chat* ke *server* atau menekan tombol back untuk kembali ke *form* roster.
  5. *User* menerima teks balasan pada *form* Message kemudian *user* kembali mengisi *textbox* dengan teks balasan kemudian menekan tombol send.

3.6 Use Case : Tambah kontak



Gambar 3.6 Rancangan antarmuka *use case* Tambah kontak

Deskripsi

- Rancangan antarmuka pada gambar 3.6 diimplementasikan pada *class* SetContact, antarmuka ini digunakan untuk menampilkan *form* Set Contact untuk menambah kontak pada roster.

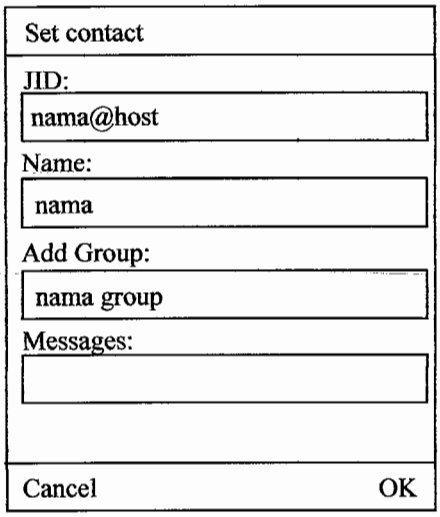
Event

- Tambah kontak

  Urutan aksi yang terjadi :

  1. *User* memilih menu Add.
  2. JAIM menampilkan *form* Set Contact.
  3. *User* mengisi *form* dengan jid, nama dan *group* kontak baru.
  4. *User* menekan tombol OK untuk mengirim data kontak ke *server* dan kembali ke *form* Roster atau menekan tombol cancel untuk membatalkan.

3.7 Use Case : Edit kontak



Gambar 3.7 Rancangan antarmuka *use case* Edit Kontak

Deskripsi

- Rancangan antarmuka pada gambar 3.7 diimplementasikan pada *class* SetContact, antarmuka ini digunakan untuk menampilkan *form* Set Contact untuk mengedit kontak yang ada di roster.

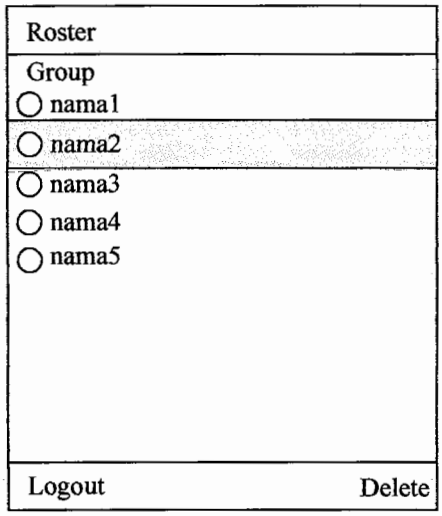
Event

- Edit kontak

  Urutan aksi yang terjadi :

  1. *User* memilih kontak yang akan diedit.
  2. *User* memilih menu Edit.
  3. JAIM menampilkan *form* Set Contact.
  4. *User* mengisi *form* dengan data jid, nama dan *group* kontak baru.
  5. *User* menekan tombol OK untuk mengirim data kontak ke *server* dan kembali ke *form* Roster atau menekan tombol cancel untuk membatalkan.

3.8 Use Case : Hapus kontak



Gambar 3.8 Rancangan antarmuka *use case* Hapus kontak

Deskripsi

- Rancangan antarmuka pada gambar 3.8 diimplementasikan pada *class* SetContact, untuk menghapus kontak yang ada di roster.

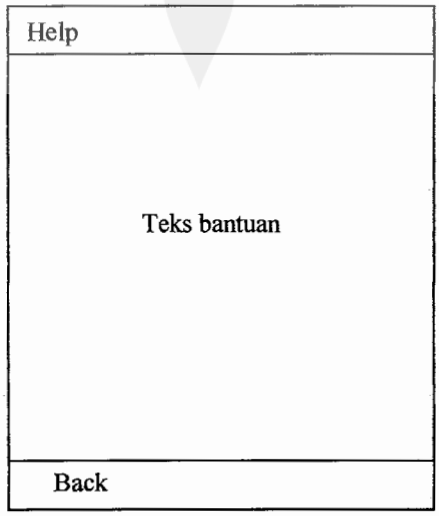
Event

- Hapus kontak

  Urutan aksi yang terjadi :

  1. *User* memilih kontak yang akan dihapus.
  2. *User* memilih menu Delete.
  3. Kontak terhapus dari roster.

3.9 Use Case : Membaca bantuan



Gambar 3.9 Rancangan antarmuka *use case* Membaca bantuan

Deskripsi

- Rancangan antarmuka pada gambar 3.9 diimplementasikan pada *class* Bantuan, antarmuka ini digunakan untuk menampilkan *form* Help.

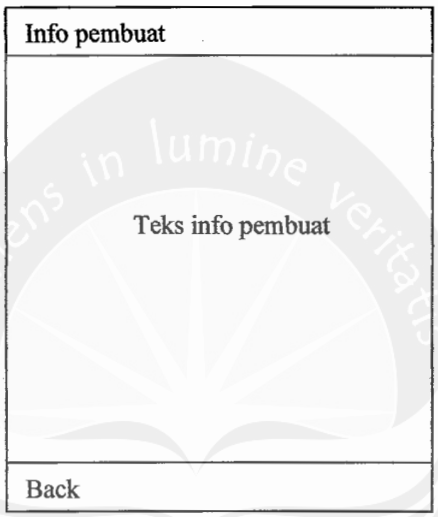
Event

- Membaca bantuan

  Urutan aksi yang terjadi :

  1. *User* memilih menu Help.
  2. JAIM menampilkan *form* Help.

3.10 Use Case : Melihat info pembuat



Gambar 3.10 Rancangan antarmuka *use case* Melihat info pembuat

Deskripsi

- Rancangan antarmuka pada gambar 3.10 diimplementasikan pada *class* Info, antarmuka ini digunakan untuk menampilkan *form* Info pembuat.

Event

- Melihat info pembuat

  Urutan aksi yang terjadi :

  1. *User* memilih menu Info.
  2. JAIM menampilkan *form* Info pembuat.

PDHUPL

PERENCANAAN, DESKRIPSI, DAN HASIL

UJI PERANGKAT LUNAK

Pembangunan Aplikasi Instant Messaging Client

Berbasis Jabber pada Ponsel

(JAIM)

Dipersiapkan oleh:

I Made Hedi Juliarta 03155/ TF

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Atma Jaya Yogyakarta

Jl. Babarsari 44, Jogjakarta 50281

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	
G	

INDEX TGL	-	A	B	C	D	E	F	G
Ditulis oleh								
Diperiksa oleh								
Disetujui oleh								

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi

Daftar Isi

Daftar Tabel

1 Pendahuluan

1.1 Tujuan Pembuatan Dokumen

Dokumen PDHUPL ini dibuat untuk menyediakan perencanaan, deskripsi, dan hasil pengujian perangkat lunak *Instant Messaging* JAIM. Dokumen ini ditujukan untuk pembuat perangkat lunak, dan orang lain yang tertarik untuk mengembangkan perangkat lunak ini lebih lanjut.

1.2 Deskripsi Umum Sistem

JAIM adalah aplikasi *Instant Messaging* pada ponsel yang berbasis Jabber sebagai sarana untuk mengirim pesan, *presence*, dan *chat* bagi pengguna ponsel yang mendukung MIDP 2.0. Sistem ini secara garis besar terdiri dari 10 komponen besar, yaitu:

a. Login untuk proses login ke sistem dan proses autentifikasi ke *server*.
b. Kirim Presence untuk mengirim *presence*.
c. Kirim Pesan untuk mengirim pesan.
d. Terima Pesan untuk menerima pesan yang masuk.
e. Chat untuk melakukan *chatting*.
f. Tambah Kontak untuk menambah kontak baru ke *roster*.
g. Edit Kontak untuk mengedit profil kontak.
h. Hapus Kontak untuk menghapus kontak dari *roster*.
i. Membaca Bantuan untuk menampilkan teks penggunaan aplikasi.
j. Melihat Info Pembuat untuk menampilkan profil pembuat aplikasi.

1.3 Definisi dan Singkatan

PDHUPL : Perencanaan, Deskripsi dan Hasil Uji Perangkat Lunak, merupakan dokumen untuk menguji perangkat lunak yang dikembangkan.

1.4 Dokumen Referensi

- GL01, SKPL, Jurusan Teknik Informatika -ITB
- Hartanto, Antonius Aditya, Pemprograman Mobile Java dengan MIDP 2.0, 2003.
- Suyoto, Membuat Sendiri Aplikasi Ponsel, 2005.

2 Lingkungan Pengujian Perangkat Lunak

2.1 Perangkat Lunak Pengujian

Perangkat lunak Pengujian berupa:

a. Sistem Operasi : Windows XP.
b. *Java 2 Software Development Kit* (J2SDK) versi 1.4.2_04
c. Antarmuka : J2ME *Wireless Tool Kit* (J2MEWTK) versi 2.2
d. *Server*: JabberD 1.4.2_2
e. Aplikasi IM untuk PC *Exodus ver. 6.0*

2.2 Perangkat Keras Pengujian untuk Simulasi

a. Prosesor Intel Pentium atau AMD Athlon
b. RAM 256 MB
c. Kapasitas sisa harddisk 500 MB
d. Keyboard
e. Mouse

2.3 Perangkat Keras Pengujian untuk Intalalasi

a. *Card Reader*

b. Ponsel berbasis Java (Nokia 6600)

2.4 Material Pengujian

Material tambahan untuk pengujian ini yaitu:

1. User Manual JAIM.

2.5 Sumber Daya Manusia

Sumber daya pengujian ini berupa:

1. Pembuat Perangkat Lunak.

2.6 Prosedur Umum Pengujian

2.6.1 Pengenalan dan Latihan

Pengenalan dan Pelatihan Aplikasi JAIM ini tidak memerlukan waktu yang lama karena aplikasi ini sangat mudah untuk dipahami dan digunakan.

2.6.2 Persiapan Awal

2.6.2.1 Persiapan Perangkat Keras

Perangkat Keras beserta spesifikasinya berupa:

- Prosesor Intel Pentium atau AMD Athlon
- RAM 256 MB
- Kapasitas sisa harddisk 500 MB
- Keyboard
- Mouse
- *Card Reader*
- Ponsel berbasis Java (Nokia 6600)
- Keyboard dan mouse standar.

2.6.2.2 Persiapan Perangkat Lunak

1. Install aplikasi *Java 2 Software Development Kit* (J2SDK) versi 1.4.2_04 pada komputer.
2. Install aplikasi : J2ME *Wireless Tool Kit* (J2MEWTK) versi 2.2 pada komputer.
3. Install aplikasi *Server*: JabberD 1.4.2_2.
4. Install aplikasi IM untuk PC *Exodus ver. 6.0*.

2.6.3 Pelaksanaan

Pelaksanan pengujian akan dilakukan untuk masing-masing use case. Untuk deskripisi use case dapat mengacu ke SKPL JAIM.

2.6.4 Pelaporan Hasil

Hasil pengujian akan diserahkan Dosen Pembimbing.

3 Identifikasi dan Rencana Pengujian

Tabel 1. Identifikasi Pengujian

Kelas Uji	Butir Uji	Identifikasi		Tingkat Pengujian	Jenis Pengujian	Jadwal
		SKPL	PDHUPL			
Pengujian Use Case Login	Login	UC-JAIM-01	AU_01_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Use Case Kirim Presence	Kirim Presence	UC-JAIM-02	AU_02_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Kirim Presence	Kirim Presence	UC-JAIM-03	AU_03_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Use case	Terima Pesan	UC-JAIM-04	AU_04_01	Pengujian Unit	Black Box	Mei 2007

Terima Pesan						
Pengujian Use Case Chat	Chat	UC-JAIM-05	AU_05_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Use Case Tambah Kontak	Tambah Kontak	UC-JAIM-06	AU_06_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Use Case Edit Kontak	Edit Kontak	UC-JAIM-07	AU_07_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Use Case Hapus Kontak	Hapus Kontak	UC-JAIM-08	AU_08_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Use Case Membaca Bantuan	Membaca Bantuan	UC-JAIM-09	AU_09_01	Pengujian Unit	Black Box	Mei 2007
Pengujian Use Case Melihat Info Pembuat	Melihat Info Pembuat	UC-JAIM-10	AU_10_01	Pengujian Unit	Black Box	Mei 2007

4 Deskripsi dan Hasil Uji

4.1 Identifikasi Kelas Pengujian Login oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Login dengan aktor Penguji sebagai penggunanya.

4.1.1 Identifikasi Butir Pengujian Login – AU_01_01

Butir pengujian ini melakukan pengujian terhadap antarmuka Login dengan input berupa penekanan tombol Keyboard.

4.2 Identifikasi Kelas Pengujian Kirim Presence oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Kirim Presence dengan aktor Penguji sebagai penggunanya.

4.2.1 Identifikasi Butir Pengujian Kirim Presence-AU_02_01

Butir pengujian ini melakukan pengujian terhadap antarmuka Presence dengan input berupa penekanan tombol Keyboard.

4.3 Identifikasi Kelas Pengujian Kirim Pesan oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Kirim Pesan dengan aktor Penguji sebagai penggunanya.

4.3.1 Identifikasi Butir Pengujian Kirim Pesan - AU_03_01

Butir pengujian ini melakukan pengujian terhadap antarmuka NewMessage dan Topic dengan input berupa penekanan tombol Keyboard.

4.4 Identifikasi Kelas Pengujian Terima Pesan oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Terima Pesan dengan aktor Penguji sebagai penggunanya.

4.4.1 Identifikasi Butir Pengujian Terima Pesan - AU_04_01

Butir pengujian ini melakukan pengujian terhadap antarmuka Received dengan input berupa penekanan tombol Keyboard.

4.5 Identifikasi Kelas Pengujian Chat oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Chat dengan aktor Penguji sebagai penggunanya

4.5.1 Identifikasi Butir Pengujian Chat - AU_05_01

Butir pengujian ini melakukan pengujian terhadap antarmuka Chat dengan input berupa penekanan tombol Keyboard.

4.6 Identifikasi Kelas Pengujian Tambah Kontak oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Tambah Kontak dengan aktor Penguji sebagai penggunanya

4.6.1 Identifikasi Butir Pengujian Tambah Kontak – AU_06_01

Butir pengujian ini melakukan pengujian terhadap antarmuka SetContact dengan input berupa penekanan tombol Keyboard.

4.7 Identifikasi Kelas Pengujian Edit Kontak oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Edit Kontak dengan aktor Penguji sebagai penggunanya

4.7.1 Identifikasi Butir Pengujian Edit Kontak – AU_07_01

Butir pengujian ini melakukan pengujian terhadap antarmuka SetContact dengan input berupa penekanan tombol Keyboard.

4.8 Identifikasi Kelas Pengujian Hapus Kontak oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Hapus Kontak dengan aktor Penguji sebagai penggunanya

4.8.1 Identifikasi Butir Pengujian Hapus Kontak - AU_08_01

Butir pengujian ini melakukan pengujian terhadap antarmuka Roster dengan input berupa penekanan tombol Keyboard.

4.9 Identifikasi Kelas Pengujian Membaca Bantuan oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Membaca Bantuan dengan aktor Penguji sebagai penggunanya

4.9.1 Identifikasi Butir Pengujian Membaca Bantuan - AU_09_01

Butir pengujian ini melakukan pengujian terhadap antarmuka Help dengan input berupa penekanan tombol Keyboard.

4.10 Identifikasi Kelas Pengujian Melihat Info Pembuat oleh User

Kelas pengujian ini meliputi pengujian-pengujian yang melibatkan fungsi antarmuka use case Melihat Info Pembuat dengan aktor Penguji sebagai penggunanya

4.10.1 Identifikasi Butir Pengujian Melihat Info Pembuat - AU_10_01

Butir pengujian ini melakukan pengujian terhadap antarmuka Info dengan input berupa penekanan tombol Keyboard.

Tabel 2. Deskripsi dan Hasil Pengujian

Identifikasi	Deskripsi	Prosedur Pengujian	Masukan	Keluaran yg diharapkan	Kriteria Evaluasi Hasil	Hasil yang Didapat	Kesimpulan
AU_01_01	Pengujian Login	- Jalankan Aplikasi	- Jid dan password - Tekan tombol Login.	Antarmuka Roster.	Antarmuka ditampilkan	Antarmuka ditampilkan	Handal
AU_01_02	Pengujian Kirim Presence	- Jalankan Aplikasi - Pilih Menu Presence. - Pilih status Presence.	- Tekan tombol OK untuk mengirim presence.	Presence berubah sesuai pilihan.	Presence ditampilkan	Presence ditampilkan	Handal
AU_01_03	Pengujian Kirim Pesan	- Jalankan Aplikasi - Pilih Menu Send.	- Jid yang dituju. - Teks pesan. - Tekan tombol Send.	Pesan tampil dilayar penerima.	Pesan tampil dilayar penerima.	Pesan tampil dilayar penerima.	Handal
AU_01_04	Pengujian Terima Pesan	- Jalankan Aplikasi - Pilih Menu Received	Tekan tombol OK.	Antarmuka Received	Pesan masuk ditampilkan.	Pesan masuk ditampilkan.	Handal
AU_01_05	Pengujian Chat	- Jalankan Aplikasi - Pilih Menu Chat	Tekan tombol OK.	Antarmuka Chat	Pesan terkirim dan pesan masuk ditampilkan.	Pesan terkirim dan pesan masuk ditampilkan.	Handal
AU_01_06	Pengujian Tambah Kontak	- Jalankan Aplikasi - Pilih menu Add	Tekan tombol OK	Antarmuka SetContact	Kontak baru tampil di roster.	Kontak baru tampil di roster.	Handal

AU_01_07	Pengujian Edit Kontak	- Jalankan Aplikasi - Pilih menu Edit	Tekan tombol OK	Antarmuka SetContact	Kontak ter-update.	Kontak ter-update.	Handal
AU_01_08	Pengujian Hapus Kontak	- Jalankan Aplikasi - Pilih menu Delete	Tekan tombol OK	Antarmuka Roster	Kontak ter-hapus.	Kontak ter-hapus.	Handal
AU_01_09	Pengujian Membaca Bantuan	- Jalankan Aplikasi - Pilih menu Help	Tekan tombol OK	Antarmuka Help	Antarmuka ditampilkan	Antarmuka ditampilkan	Handal
AU_01_10	Pengujian Melihat Info Pembuat	- Jalankan Aplikasi - Pilih menu Info	Tekan tombol OK	Antarmuka Info	Antarmuka ditampilkan	Antarmuka ditampilkan	Handal