BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan pembahasan pada bab-bab sebelumnya maka dapat ditarik beberapa kesimpulan yaitu:

1. Metode *forward chaining* dapat membantu pengambilan keputusan dalam sistem pendukung keputusan.
2. Aplikasi Sistem Pendukung Keputusan Pemilihan Salon Kecantikan dengan Metode *Forward Chaining* Berbasis *Mobile* berhasil dibangun dengan *platform* Android dan menggunakan metode *forward chaining*.

6.2 Saran

Beberapa saran yang dapat diambil dari proses analisis sampai pada pembuatan tugas akhir ini adalah sebagai berikut:

1. Dapat membuat aplikasi ini berjalan di *platform* lain, seperti iOS, *windows phone*, atau di *mobile web*.
2. Diharapkam aplikasi ini dapat dilengkapi dengan *direction*.
3. *Website* yang sudah ada dibuat versi *mobile* dan sistem pendukung keputusannya dapat berjalan di *website* juga.

DAFTAR PUSTAKA

Abidin, Hasanuddin Z., 2007, *Advancement in GPS Technology and Application*, Geodesy Research Division.

Barry, Andrew, 2008, *Australian Inventor Develops Computer Program Forward-Chaining Inference System*, US Fed News Service.

Binder, Walter, Constantinescu, Ion, Boi, Faltings, 2007, *A Flexible Directory Query Language for the Efficient Processing of Service Composition Queries*, International Journal of Web Services Research, Vol. 4, No. 1, PP. 59-79.

Darden, Lindley, 2002, *Strategies for Discovering Mechanisms: Schema Instantiation, Modular Subassembly, Forward/Backward Chaining*, Philosophy of Science, Vol 69, No 3, PP 354-365.

Dvorski, Dalibor D., 2007, Installing, Configuring, and Developing with Xampp, Skill Canada.

Encheva, Sylvia, Kondratenko, Yuriy, Solesvik, Maryna Z., Tumin, Sharil, 2008, *Decision Support Systems in Logistics*, AIP Conference Proceedings, Vol. 1060, No. 1, PP. 254-256.

Fletcher, Thomas, 2003, *With Eclipse, Tools Find Common IDE, Electronic Engineering Times*, Vol 1287, PP 67,76.

Fulsang, Deborah, Tiyana, Grulovic, Montanera, Doris, 2007, *The Busy Woman's Guide to Beauty, Rogers Publishing Limited*, Vol. 80, No. 9, PP. 177-178,180,182,184,186,188,190,192,194,196,198,200,202,204,206,208.

Hamdani, 2010, Sistem Pakar Untuk Diagnosa Penyakit Mata Pada Manusia, Jurnal Informatika Mulawarman, Vol. 5, NO.2, PP. 13-21.

Harchaoui, Tarek M, Tarkani, Faouzi, Jackson, Chris, Armstrong, Philip, 2002, *Information Technology and Economic Growth in Canada and the U.S.*, Monthly Labor Review, Vol. 125, No. 10, PP. 3-12.

Hendrasurya, Wirawan, 2004, Perancangan dan Pembuatan Aplikasi Sistem Pakar tentang Pemilihan Obat Non Resep Dokter Berbasiskan Web, Universitas Kristen Petra.

Honggowibowo, Anton Setiawan, 2009, Sistem Pakar Diagnosa Penyakit Tanaman Padi Berbasis Web dengan *Forward* dan *Backward Chaining*, Telkomnika, Vol. 7, No. 3, PP 187-194.

Kadir, Abdul, 2008, Tuntunan Praktis Belajar *Database* menggunakan *MySQL*, Penerbit Andi, Yogyakarta.

Kushwaha, Amit, Kushwaha, Vineet, 2011, *Location Based Services using Android Mobile Operating System*, International Journal of Advances in Engineering & Technology.

Lambeek, Alex, 2009, *Mobile Technology: Driving Change and Opportunity in Developing Countries*, International Trade Forum, Vol 3,PP 26-27.

Lawson, Stephen, 2008, *Google I/O Event Spotlights Android*, Network World, Vol 25, No 22, PP 16.

Lee, Chang Won, 2006, *Development Of Web-Based Decision Support System For Business Process Reenginering in A Health-Care System*, Academy of Information and Management Sciences Journal, Vol 9, No 2, PP 33-34.

Li, B., Tan, Y. Khing, Dempster, A.G., 2011, *Using Two Global Positioning System Satellites to Improve Wireless Fidelity Positioning Accuracy in Urban Canyons*, IET Communications, Vol 5, No 2, PP 163-171.

Lydia, Maya Silvi, 2011, Perancangan Aplikasi Sistem Pendukung Keputusan Investasi Untuk Penentuan Potensi Batubara Pada Suatu Area Dengan Metode

Forward Chaining Berbasis Web, Universitas Sumatera Utara.

Nilawati, Eva Sativa, *Beautypreneurship*, 2010, Penerbit Andi, Yogyakarta.

Paz, Joel O, Batchelor, William D, Pedersen, Palle, 2004, *A Web-Based Soybean Management Decision Support System*, Agronomy Journal, Vol. 96, No. 6, PP. 1771-1779.

Pocatilu, Paul, 2010, *Developing Mobile Learning Applications for Android using Web Services*, Informatica Economica, Vol 14, No 3, PP 106-115.

Ponzo, J, Gruber, O, 2005, *Integrating Web Technologies in Eclipse*, IBM Systems Journal, Vol 44, No 2, PP 279-288.

Post, Gerald V, 1999, *Database Management Systems Designing and Building Applications*, McGraw-Hill, United States of America.

Russell, Stuart, Norvig, Peter, 2010, *Artificial Intelligence: A Modern Approach*, Pearson, New Jersey.

Saputra, Agus, 2011, Trik Kolaborasi *CodeIgniterI & jQuery*, Penerbit Lokomedia, Yogyakarta.

Straughan, Elizabeth Rachel, 2010, *The salon as Clinic: Problematising, Treating, and Caring for skin*,

Social & Cultural Geography, Vol. 11, No. 7, PP. 647-661.

Ting-Peng, Liang; Chen-Wei, Huang; Yeh, Yi-Hsuan; Lin, Binshan, 2007, *Adoption of Mobile Technology in Business: A Fit-Viability Model,Industrial Management + Data Systems*, Vol 107, No 8, PP 1154.

Turban, Efraim, 1992, *Expert Systems and Applied Artificial Intelligence*, Macmillan, New York.

Turban, Efraim, Aronson, Jay E., Liang, Ting-Peng, 2005, Penerbit Andi, Yogyakarta.

Tuttle, D Ray, 2010, *Android Advances: As Number of Apps Grows, Phones Become More Popular,* Journal Record.

Untoro, Wisnu Yudho, 2009, Penerapan Metode *Forward Chaining* pada Penjadwalan Mata Kuliah, Jurnal Matematika dan Komputer Indonesia, Vol.1, No.2, PP. 17-24.

Vardell, Emily, Moore, Mary, 2011, *Isabel, a Clinical Decision Support System*, Medical Reference Services Quarterly, Vol. 30, No. 2, PP. 158-166.

Walikota Yogyakarta, 2009, Izin Penyelenggaraan Salon Kecantikan, Peraturan Walikota Kota Yogyakarta Nomor 70 Tahun 2009, Yogyakarta.

Yuregir, Oya H, Oral, Mustafa, Kalan, Olcay, 2010, *A Decision Support System for Preventing Legionella Disease, Journal of Medical Systems*, Vol. 34, No.5, PP. 875.

LAMPIRAN

SKPL

SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK

getYourSalon

(Sistem Pendukung Keputusan Pemilihan Salon Kecantikan Berbasis *Mobile*)

Untuk :

Universitas Atma Jaya Yogyakarta

Dipersiapkan oleh:

Vera Hannyta / 080705565

Program Studi Teknik Informatika – Fakultas Teknologi Industri

Universitas Atma Jaya Yogyakarta

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	

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

2 Pendahuluan

1.1. Tujuan

Dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) ini merupakan dokumen spesifikasi kebutuhan perangkat lunak Sistem Pendukung Keputusan Pemilihan Salon Kecantikan Berbasis *Mobile* yang diberi nama *getYourSalon* untuk mendefinisikan kebutuhan perangkat lunak yang meliputi antarmuka eksternal (antarmuka antara sistem dengan sistem lain perangkat lunak dan perangkat keras, dan pengguna) dan atribut (*feature-feature* tambahan yang dimiliki sistem), serta mendefinisikan fungsi perangkat lunak. SKPL-*getYourSalon* ini juga mendefinisikan batasan perancangan perangkat lunak.

1.2. Lingkup Masalah

Perangkat Lunak *getYourSalon* dikembangkan dengan tujuan untuk :

1. Menangani proses pemilihan salon kecantikan dengan atribut pemilihan *budget*, jenis perawatan, suasana salon, dan obat yang digunakan.
2. Menangani pengelolaan Admin.
3. Menangani penampilan posisi salon terpilih dalam bentuk peta.

Aplikasi *getYourSalon* berjalan pada lingkungan *mobile* dengan platform *Android*.

1.3. Definisi, Akronim dan Singkatan

Daftar definisi akronim dan singkatan :

Keyword/Phrase	Definisi
Internet	Internet merupakan istilah umum yang dipakai untuk menunjuk *Network* global yang terdiri dari komputer dan layanan servis dengan sekitar 30 sampai 50 juta pemakai komputer dan puluhan layanan informasi termasuk e-mail, FTP, dan World Wide Web.
SKPL	Merupakan spesifikasi kebutuhan dari perangkat lunak yang akan dikembangkan.
SKPL-*getYourSalon*-XXX	Kode yang merepresentasikan kebutuhan pada *getYourSalon* dimana XXX merupakan nomor fungsi produk.
getYourSalon	Perangkat lunak untuk pendukung keputusan dalam pemilihan salon kecantikan.
DataBase	Kumpulan data yang terkait yang diorganisasikan dalam struktur tertentu dan dapat diakses dengan cepat.

1.4. Referensi

Referensi yang digunakan pada perangkat lunak tersebut adalah:

1. Hannyta Vera, Spesifikasi Kebutuhan Perangkat Lunak ASM, Universitas Atma Jaya Yogyakarta, 2011.
2. Hannyta Vera, Spesifikasi Kebutuhan Perangkat Lunak HyuBOS, CV Sumber Baru Agung, 2011.

3. Sapta Juli, *Spesifikasi Kebutuhan Perangkat Lunak SC3*, Universitas Atma Jaya Yogyakarta, 2006.

1.5. Deskripsi umum (Overview)

Secara umum dokumen SKPL ini terbagi atas 5 bagian utama. Bagian utama berisi penjelasan mengenai dokumen SKPL tersebut yang mencakup tujuan pembuatan SKPL, ruang lingkup masalah dalam pengembangan perangkat lunak tersebut, definisi, referensi dan deskripsi umum tentang dokumen SKPL ini.

Bagian kedua berisi penjelasan umum tentang perangkat lunak *getYourSalon* yang akan dikembangkan, mencakup perspektif produk yang akan dikembangkan, fungsi produk perangkat lunak, karakteristik pengguna, batasan dalam penggunaan perangkat lunak dan asumsi yang dipakai dalam pengembangan perangkat lunak *getYourSalon* tersebut.

Bagian ketiga mencakup kebutuhan khusus yang terdiri dari kebutuhan antarmuka eksternal, antarmuka pemakai, antarmuka perangkat keras, antarmuka perangkat lunak dan antarmuka komunikasi.

Bagian keempat berisi penjelasan secara lebih rinci tentang kebutuhan perangkat lunak *getYourSalon* yang akan dikembangkan.

Bagian terakhir atau kelima berisi *Entity Relationship Diagram* yang akan menggambarkan relasi yang dimiliki oleh setiap entitas yang ada di dalam perangkat lunak *getYourSalon*.

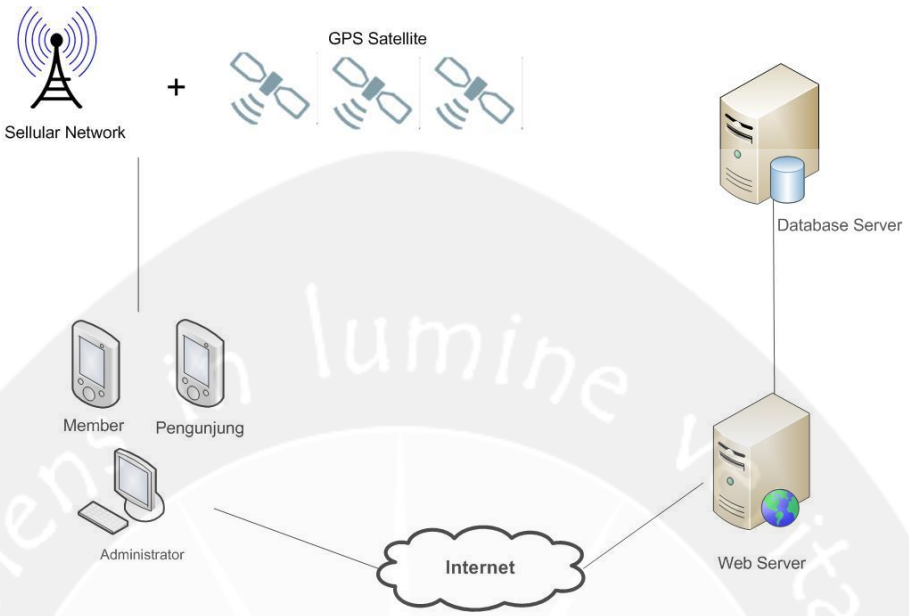
3 Deskripsi Kebutuhan

1.1. Perspektif produk

GetYourSalon merupakan perangkat lunak yang dikembangkan untuk membantu pengguna untuk mendapatkan salon yang sesuai dengan yang diinginkannya. Sistem ini dikembangkan untuk membantu proses pencarian salon sesuai dengan *budget* yang disediakan pengguna, pengguna juga dapat memberikan masukan tentang aplikasi ini, selain itu pengguna juga dapat mendaftarkan salon untuk bergabung dengan aplikasi ini. *getYourSalon* juga mampu menghasilkan peta dari posisi pengguna berada ke salon yang terdekat sesuai hasil pencarian salon yang dicari berdasar kriteria yang diinginkan pengguna.

Pengguna akan berinteraksi dengan sistem melalui antarmuka GUI (*Graphical User Interface*). Pada sistem ini, untuk berbasis *mobile*, arsitektur perangkat lunak yang digunakan berupa *client server*, di mana semua data disimpan di *server*.

Inputan data yang dimasukkan akan disimpan dalam *database server*, sehingga jika ada pencarian data, maka data yang diinginkan akan dicari ke *database server* yang selanjutnya dikirimkan ke client yang me*request* melalui *web server*.



Gambar 2.1 Arsitektur Perangkat Lunak *getYourSalon*

1.2. Fungsi Produk

Fungsi produk perangkat lunak *getYourSalon* adalah sebagai berikut :

1. Fungsi *Login* (**SKPL-*getYourSalon*-001**)

   Fungsi *Login* merupakan fungsi yang digunakan oleh administrator dan *member* untuk masuk ke sistem dan mendapatkan hak akses sesuai dengan *role* yang dimiliki.

2. Fungsi Pengelolaan Data Salon (**SKPL-*getYourSalon*-002**)

   Pengelolaan data salon merupakan fungsi yang digunakan oleh Administrator untuk mengelola data salon meliputi nama salon, alamat salon, telepon salon, perawatan yang ada di salon beserta harganya, dan info lain mengenai salon.

   Fungsi Pengelolaan Data Salon mencakup :

a. Fungsi *Entry* Data Salon (**SKPL-*getYourSalon*-002-01**).

   Fungsi *entry* data salon merupakan fungsi yang digunakan untuk menambahkan data salon yang baru.

b. Fungsi *Edit Data* Salon *(***SKPL-*getYourSalon*-002-02).**

   Fungsi *edit data* salon merupakan fungsi yang digunakan untuk mengubah data salon.

c. Fungsi *Read Data* Salon (**SKPL-*getYourSalon*-002-03**).

   Fungsi *read data* salon merupakan fungsi yang digunakan untuk menampilkan atau mencari data Salon.

d. Fungsi *Delete Data* Salon(**SKPL-*getYourSalon*-002-04**).

   Fungsi *delete data* salon merupakan fungsi yang digunakan untuk menghapus data salon.

3. Fungsi Pengelolaan Data Perawatan (**SKPL-*getYourSalon*-003**)

   Fungsi *pengelolaan* data perawatan merupakan fungsi yang digunakan oleh Administrator untuk mengelola data perawatan meliputi jenis perawatan.

   Fungsi Pengelolaan Data Perawatan mencakup :

   a. Fungsi *Entry* Data Perawatan (**SKPL-*getYourPerawatan*-003-01**).

      Fungsi *entry* data perawatan merupakan fungsi yang digunakan untuk menambahkan data perawatan yang baru.

b. Fungsi *Edit Data* Perawatan (**SKPL-*getYourPerawatan*-003-02).**

Fungsi *edit data* perawatan merupakan fungsi yang digunakan untuk mengubah data perawatan.

c. Fungsi *Read Data* Perawatan (**SKPL-*getYourPerawatan*-003-03**).

Fungsi *read data* perawatan merupakan fungsi yang digunakan untuk menampilkan atau mencari data perawatan.

d. Fungsi *Delete Data* Perawatan(**SKPL-*getYourPerawatan*-003-04**).

Fungsi *delete data* perawatan merupakan fungsi yang digunakan untuk menghapus data perawatan.

4. Fungsi *Pengelolaan* Data Detail Perawatan (**SKPL-*getYourSalon*-004**)

Fungsi *pengelolaan* data detail perawatan merupakan fungsi yang digunakan oleh Administrator untuk mengelola detail perawatan meliputi harga perawatan, jenis perawatan, gambar perawatan, dan keterangan tentang perawatan.

Fungsi Pengelolaan Data Perawatan mencakup :

a. Fungsi *Entry* Data Detail Perawatan (**SKPL-*getYourPerawatan*-003-01**).

Merupakan fungsi yang digunakan untuk menambahkan data detail perawatan yang baru.

b. Fungsi *Edit Data* Perawatan (**SKPL-*getYourPerawatan*-003-02).**

Fungsi *entry* data detail perawatan merupakan fungsi yang digunakan untuk mengubah detail perawatan.

c. Fungsi *Read Data* Perawatan (**SKPL-*getYourPerawatan*-003-03**).

   Fungsi *read data* perawatan merupakan fungsi yang digunakan untuk menampilkan atau mencari data detail perawatan.

d. Fungsi *Delete Data* Perawatan(**SKPL-*getYourPerawatan*-003-04**).

   Merupakan fungsi yang digunakan untuk menghapus data detail perawatan.

5. Fungsi Pencarian Salon Berdasarkan Spesifikasi Tertentu (**SKPL-*getYourSalon*-005**)

   Fungsi pencarian salon berdasarkan spesifikasi tertentu merupakan fungsi yang digunakan untuk mencari salon tertentu dengan *input user* berupa *budget*, jenis perawatan, suasana yang diinginkan, dan bahan yang digunakan.

6. Fungsi Daftar *Member*(**SKPL-*getYourSalon*-006**)

   Fungsi daftar *member* merupakan fungsi yang digunakan oleh pengunjung untuk mendaftar menjadi sebagai anggota, dengan menjadi anggota pengguna dapat memberikan komentar atau mendaftarkan salon miliknya ke aplikasi ini.

7. Fungsi *Manage Account*(**SKPL-*getYourSalon*-007**)

   Merupakan fungsi yang digunakan oleh *member* untuk mengubah akun yang dimilikinya.

8. Fungsi *Add Comment*(**SKPL-*getYourSalon*-08**)

   Fungsi *add comment* yang digunakan oleh *member* untuk memberi *comment*.

9. Fungsi *View Comment*(**SKPL-getYourSalon-09**)

   Fungsi *view comment* yang digunakan oleh *member*, pengunjung, atau admin untuk melihat *comment*.

10. Fungsi *Delete Comment* ***(SKPL-getYourSalon-010)***

    Fungsi *delete comment* yang digunakan oleh administrator untuk menghapus *comment*.

11. Fungsi *Manage Member* ***(SKPL-getYourSalon-011)***

    Fungsi *manage member* digunakan oleh administrator untuk mengelola *member*.

    a. Fungsi *Add Admin_child* (**SKPL-getYourSalon-011_1**)

       Fungsi *Add Admin_child* merupakan fungsi yang digunakan oleh administrator untuk member hak seseorang sebagai administrator.

    b. Fungsi *reset password* (**SKPL-getYourSalon-011_2**)

       Fungsi *reset password* merupakan fungsi yang digunakan oleh administrator mereset *password* member.

    c. Fungsi Delete Member (**SKPL-getYourSalon-011_3**)

       Fungsi *delete member* merupakan fungsi yang digunakan oleh administrator menghapus member.

12. Fungsi Pengelolaan Suasana (**SKPL-*getYourSalon*-012**)

    Fungsi *pengelolaan* suasana merupakan fungsi yang digunakan oleh Administrator (hanya 1 orang) untuk mengelola suasana meliputi deskripsi suasana.

Fungsi Pengelolaan suasana mencakup :

a. Fungsi *Entry* Suasana (**SKPL-*getYourPerawatan*-012-01**).

   Fungsi *entry* data suasana merupakan fungsi yang digunakan untuk menambahkan data suasana yang baru.

b. Fungsi *Edit* Suasana (**SKPL-*getYourPerawatan*-012-02**).

   Fungsi *edit* suasana merupakan fungsi yang digunakan untuk mengubah suasana.

c. Fungsi *Read* Suasana (**SKPL-*getYourPerawatan*-012-03**).

   Fungsi *read data* suasana merupakan fungsi yang digunakan untuk menampilkan atau mencari suasana.

d. Fungsi *Delete Suasana* (**SKPL-*getYourPerawatan*-012-04**).

   Fungsi *delete* suasana merupakan fungsi yang digunakan untuk menghapus data suasana.

13. Fungsi Pengelolaan Bahan (**SKPL-*getYourSalon*-010**)

    Fungsi *pengelolaan* bahan merupakan fungsi yang digunakan oleh Administrator (hanya 1 orang) untuk mengelola bahan meliputi nama bahan.

    Fungsi Pengelolaan bahan mencakup :

a. Fungsi *Entry* Bahan (**SKPL-*getYourPerawatan*-012-01**).

   Fungsi *entry* data bahan merupakan fungsi yang digunakan untuk menambahkan data bahan yang baru.

b. Fungsi *Edit* Bahan *(***SKPL-*getYourPerawatan*-012-02).**

Fungsi *edit* bahan merupakan fungsi yang digunakan untuk mengubah bahan.

c. Fungsi *Read* Bahan (**SKPL-*getYourPerawatan*-012-03**).

Fungsi *read data* bahan merupakan fungsi yang digunakan untuk menampilkan atau mencari bahan.

d. Fungsi *Delete Bahan*(**SKPL-*getYourPerawatan*-012-04**).

Fungsi *delete* bahan merupakan fungsi yang digunakan untuk menghapus data bahan.

1.3. Karakteristik Pengguna

Karakteristik dari pengguna perangkat lunak ASM adalah sebagai berikut :

1. Pengguna memahami penggunaan ponsel Android.
2. Pengguna memahami penggunaan internet dan GPS.

1.4. Batasan-batasan

Batasan-batasan dalam pengembangan perangkat lunak ASM tersebut adalah:

1. Aplikasi *getYourSalon* tidak menangani reservasi salon secara *online.*
2. Keterbatasan perangkat keras

   Dapat diketahui kemudian setelah sistem ini berjalan (sesuai dengan kebutuhan).

1.5. Asumsi dan Ketergantungan

Sistem ini dapat dijalankan pada perangkat yang menggunakan sistem operasi Android minimal versi 2.2 (Froyo) dan mempunyai modul GPS serta harus memenuhi ketersediaan *internet*.

4 Kebutuhan Khusus

1.1. Kebutuhan antarmuka eksternal

Kebutuhan antar muka eksternal pada perangkat lunak *getYourSalon* meliputi kebutuhan antarmuka pemakai, antarmuka perangkat keras, antarmuka perangkat lunak, antarmuka komunikasi.

4.1.1 Antarmuka pemakai

Pemakai berinteraksi langsung dengan sistem *getYourSalon* dengan antarmuka berbasis *mobile*. Antarmuka yang ditampilkan dalam bentuk *form-form* berbasis *layout*.

4.1.2 Antarmuka perangkat keras

Untuk antarmuka perangkat keras yang digunakan dalam perangkat lunak *getYourPerawatan* adalah:

3. Ponsel dengan sistem operasi minimum Android 2.2 (Froyo).
4. GPS.

4.1.2 Antarmuka perangkat lunak

Perangkat lunak yang dibutuhkan untuk membangun perangkat lunak *getYourSalon* adalah sebagai berikut :

1. Nama : MySQL
   Sumber : Sun Microsystem
   Sebagai *database management system* (DBMS).
2. Nama : Android (minimal 2.2)
   Sumber : Google
   Sebagai sistem operasi untuk perangkat *mobile*.
3. Nama : Google API v.8
   Sumber : Google
   Sebagai API yang digunakan untuk dapat menggunakan layanan Google, termasuk di dalamnya Google Map API.
4. Nama : Apache
   Sumber : Apache Software Foundation
   Sebagai *server*.
5. Nama : Google Chrome
   Sumber : Google.

   Sebagai web browser.

6. Nama : Code Igniter
   Sumber : EllisLab.

   Sebagai framework.

7. Nama : PHP
   Sumber : The PHP Group

   Sebagai bahasa pemrograman yang digunakan untuk membangun sistem *getYourSalon* dengan basis *web* dan sebagai penghubung antara aplikasi dengan *server*.

8. Nama : Eclipse Indigo
   Sumber : IBM VisualAge

   Sebagai IDE pembuatan aplikasi Android.

9. Nama : Java NetBeans 7.1
   Sumber : Sun Microsystems

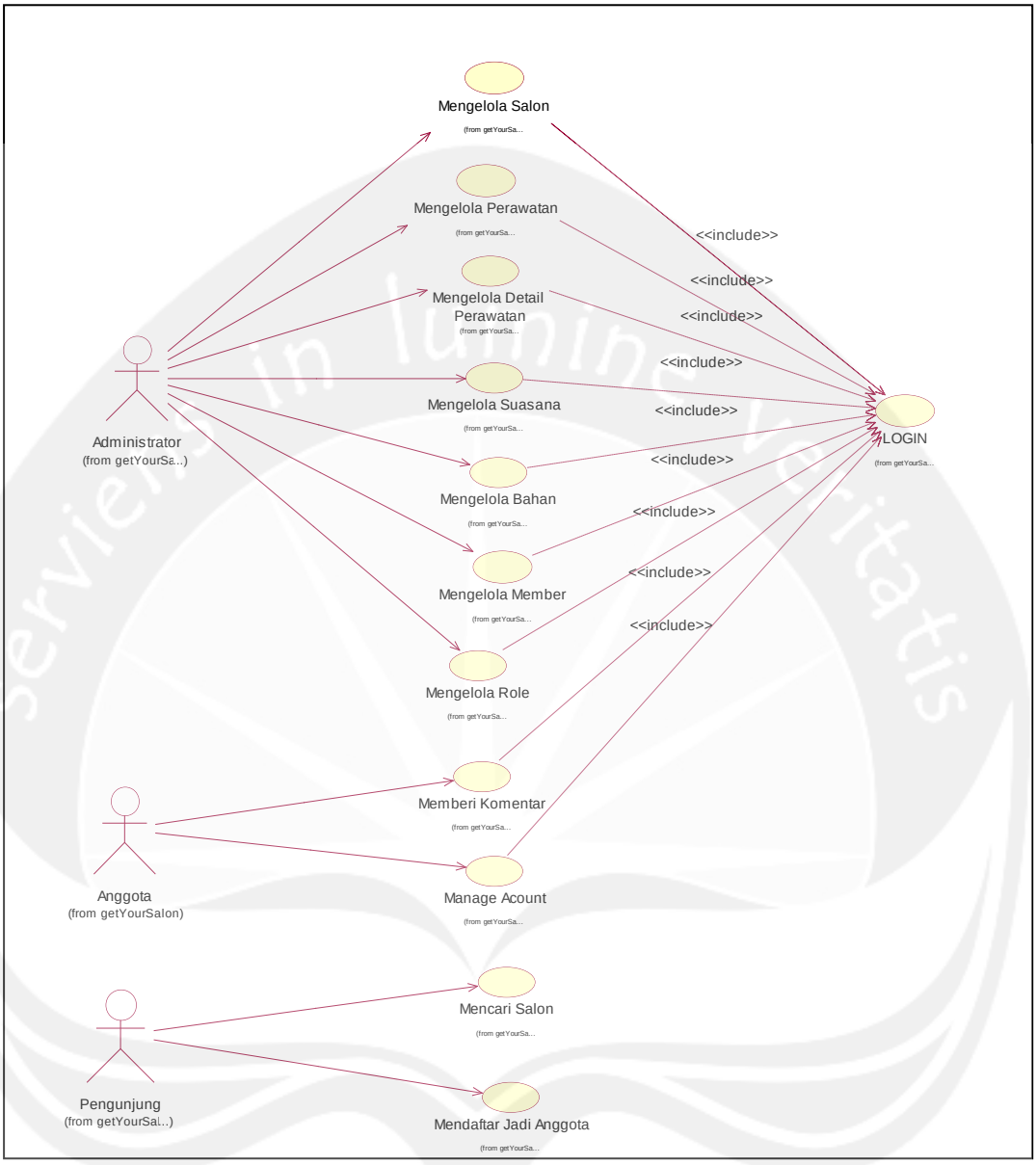
   Sebagai IDE pembuatan *web* dengan PHP.

4.1.3 Antarmuka Komunikasi

Antarmuka komunikasi *getYourPerawatan* menggunakan protokol HTTP.

1.2. Kebutuhan fungsionalitas Perangkat Lunak

3.2.1 Use Case Diagram



Gambar 3.1 Use Case Diagram

5 Spesifikasi Rinci Kebutuhan

1.1. Spesifikasi Kebutuhan Fungsionalitas

4.1.6.1.1 Use case Spesification : Login

9. Brief Description

Use Case ini digunakan oleh aktor untuk memperoleh akses ke sistem. Login didasarkan pada sebuah id unik dari user dan password yang berupa rangkaian karakter.

10. Primary Actor

1. *Member*

11. Supporting Actor

none

12. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk melakukan *login*
2. Sistem menampilkan antarmuka untuk *login*

   A-1 Aktor memilih untuk ganti *password*
3. Aktor memasukkan id dan *password*
4. Sistem memeriksa id dan *password* yang diinputkan aktor

   E-1 Password atau id *user* tidak sesuai
5. Sistem memberikan akses ke aktor
6. Use Case ini selesai

13. Alternative Flow

A-1 Aktor memilih untuk ganti password

1. Sistem menampilkan sub menu untuk ganti *password*
2. Aktor memasukan id, *password* lama, *password* baru dan konfirmasi p*assword* baru
3. Sistem memeriksa id, *password* lama, *password* baru dan konfirmasi *password* baru

E-2 *Password* atau id *user*, *password* baru dan konfirmasi *password* tidak sesuai

4. Sistem menyimpan data yang telah diedit ke database
5. Berlanjut ke Basic Flow langkah ke 6

14. Error Flow

E-1 *Password* atau nama *user* tidak sesuai

1. Sistem menampilkan peringatan bahwa id *user* atau *password* tidak sesuai
2. Kembali ke *Basic Flow* langkah ke 3

E-2 Password atau id *user*, password baru dan konfirmasi password tidak sesuai

3. Sistem menampilkan peringatan bahwa id *user* atau *password* tidak sesuai
4. Kembali ke Altenative Flow A-1 langkah ke 2

15. PreConditions

none

16. PostConditions

1. Aktor memasuki sistem dan dapat menggunakan fungsi-fungsi pada sistem
2. Aktor behasil mengganti password

4.1.6.1.2 Use case Spesification : Mengelola Daftar Perawatan

9. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data salon kecantikan. Aktor dapat melakukan *entry* data perawatan kecantikan, edit data

perawatan kecantikan, read data perawatan kecantikan, atau delete data perawatan kecantikan.

10. Primary Actor

    2. Administrator

11. Supporting Actor

    none

12. Basic Flow

    1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan data perawatan kecantikan
    2. Sistem memberikan pilihan untuk melakukan *entry* data perawatan kecantikan, *edit* data perawatan kecantikan, *read* data perawatan kecantikan, atau *delete* data perawatan kecantikan
    3. Aktor memilih untuk melakukan *entry* data perawatan kecantikan

        A-1 Aktor memilih untuk melakukan edit data perawatan kecantikan

        A-2 Aktor memilih untuk melakukan *read* perawatan kecantikan

        A-3 Aktor memilih untuk melakukan *delete* data perawatan kecantikan

    4. Aktor menginputkan data perawatan kecantikan
    5. Aktor meminta sistem untuk menyimpan data perawatan kecantikan yang telah diinputkan
    6. Sistem mengecek data perawatan kecantikan yang telah diinputkan

        E-1 Data perawatan kecantikan yang diinputkan aktor salah

7. Sistem menyimpan data perawatan kecantikan ke *database server*
8. Use Case selesai

13. Alternative Flow

A-1 Aktor memilih untuk melakukan edit data perawatan kecantikan

7. Sistem menampilkan data perawatan kecantikan
8. Aktor mengedit data perawatan kecantikan yang sudah ditampilkan
9. Aktor meminta sistem untuk menyimpan data perawatan kecantikan yang telah diedit
10. Sistem melakukan pengecekan terhadap data perawatan kecantikan yang telah diedit

E-2 Data perawatan kecantikan yang telah di*edit* salah

11. Sistem meyimpan data perawatan kecantikan yang telah diedit ke *database*
12. Berlanjut ke *Basic Flow* langkah ke 8

A-2 Aktor memilih untuk melakukan read data perawatan kecantikan

1. Sistem menampilkan pilihan kriteria pencarian/*display* data perawatan kecantikan
2. Aktor memilih pencarian/*display* seluruh data master barang

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

3. Sistem menampilkan seluruh data perawatan kecantikan yang tersimpan dalam *database*
4. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan delete data perawatan kecantikan

6. Sistem menampilkan data perawatan kecantikan
7. Aktor memilih data perawatan kecantikan yang akan dihapus
8. Aktor meminta sistem untuk menghapus data perawatan kecantikan yang dipilih
9. Sistem melakukan penghapusan data perawatan kecantikan dari *database*
10. Berlanjut ke *Basic Flow* langkah ke 8

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

3. Sistem menampilkan seluruh data perawatan kecantikan yang tersimpan dalam *database* berdasarkan kriteria yang dipilih aktor
4. Berlanjut ke Basic Flow langkah ke 8

14. Error Flow

E-1 Data perawatan kecantikan yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data perawatan kecantikan yang telah diedit salah

3. Sistem memberikan pesan peringatan bahwa data yang diedit salah
4. Kembali ke Alternative Flow A-1 langkah ke 2

15. PreConditions

1. Use Case *Login* telah dilakukan
2. Aktor telah memasuki sistem

16. PostConditions

Data perawatan kecantikan di database telah ter*update*

4.1.6.1.3 Use case Spesification : Mengelola Daftar Salon

1. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data salon. Aktor dapat melakukan *entry* data salon, edit data salon, read data salon, atau delete data salon.

3. Primary Actor
   1. Administrator
4. Supporting Actor

none

5. Basic Flow
   1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan data salon
   2. Sistem memberikan pilihan untuk melakukan *entry* data salon, *edit* data salon, *read* data salon, atau *delete* data salon
   3. Aktor memilih untuk melakukan *entry* data salon

      A-1 Aktor memilih untuk melakukan edit data salon

      A-2 Aktor memilih untuk melakukan *read* salon

A-3 Aktor memilih untuk melakukan *delete* data salon

4. Aktor menginputkan data salon
5. Aktor meminta sistem untuk menyimpan data salon yang telah diinputkan
6. Sistem mengecek data salon yang telah diinputkan

E-1 Data salon yang diinputkan aktor salah

7. Sistem menyimpan data salon ke *database server*
8. Use Case selesai

6. Alternative Flow

A-1 Aktor memilih untuk melakukan edit data salon

1. Sistem menampilkan data salon
2. Aktor mengedit data salon yang sudah ditampilkan
3. Aktor meminta sistem untuk menyimpan data salon yang telah diedit
4. Sistem melakukan pengecekan terhadap data salon yang telah diedit

E-2 Data salon yang telah di*edit* salah

5. Sistem meyimpan data salon yang telah diedit ke *database*
6. Berlanjut ke *Basic Flow* langkah ke 8

A-2 Aktor memilih untuk melakukan read data salon

a. Sistem menampilkan pilihan kriteria pencarian/*display* data salon
b. Aktor memilih pencarian/*display* seluruh data salon

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

c. Sistem menampilkan seluruh data salon yang tersimpan dalam *database*

d. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan delete data perawatan

1. Sistem menampilkan data salon
2. Aktor memilih data salon yang akan dihapus
3. Aktor meminta sistem untuk menghapus data salon yang dipilih
4. Sistem melakukan penghapusan data salon dari *database*
5. Berlanjut ke *Basic Flow* langkah ke 8

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

1. Sistem menampilkan seluruh data salon yang tersimpan dalam *database* berdasarkan kriteria yang dipilih aktor
2. Berlanjut ke Basic Flow langkah ke 8

7. Error Flow

E-1 Data perawatan yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data perawatan yang telah diedit salah

1. Sistem memberikan pesan peringatan bahwa data yang diedit salah
2. Kembali ke Alternative Flow A-1 langkah ke 2

8. PreConditions

    1. Use Case *Login* telah dilakukan
    2. Aktor telah memasuki sistem

9. PostConditions

    Data perawatan di database telah ter*update*

4.1.6.1.4 Use case Spesification : Mengelola Detail Perawatan

1. Brief Description

    Use Case ini digunakan oleh aktor untuk mengelola data daftar detail perawatan. Aktor dapat melakukan *entry* data detail perawatan, edit data detail perawatan perawatan, read data detail perawatan, atau delete data detail perawatan.

9. Primary Actor

    1. Administrator

10. Supporting Actor

    none

11. Basic Flow

    1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan data detail perawatan
    2. Sistem memberikan pilihan untuk melakukan *entry* data detail perawatan, *edit* data detail perawatan , *read* data detail perawatan , atau *delete* data detail perawatan
    3. Aktor memilih untuk melakukan *entry* data detail perawatan

        A-1 Aktor memilih untuk melakukan edit data detail perawatan

A-2 Aktor memilih untuk melakukan *read* detail perawatan

A-3 Aktor memilih untuk melakukan *delete* data detail perawatan

4. Aktor menginputkan data detail perawatan
5. Aktor meminta sistem untuk menyimpan data detail perawatan yang telah diinputkan
6. Sistem mengecek data detail perawatan yang telah diinputkan

E-1 Data detail perawatan yang diinputkan aktor salah

7. Sistem menyimpan data detail perawatan ke *database server*
8. Use Case selesai

12. Alternative Flow

A-1 Aktor memilih untuk melakukan edit data detail perawatan perawatan

7. Sistem menampilkan data detail perawatan perawatan
8. Aktor mengedit data detail perawatan perawatan yang sudah ditampilkan
9. Aktor meminta sistem untuk menyimpan data detail perawatan perawatan yang telah diedit
10. Sistem melakukan pengecekan terhadap data detail perawatan perawatan yang telah diedit

E-2 Data detail perawatan yang telah di*edit* salah

11. Sistem meyimpan data detail perawatan yang telah diedit ke *database*

12. Berlanjut ke *Basic Flow* langkah ke 8

A-2 Aktor memilih untuk melakukan read data detail perawatan

a. Sistem menampilkan pilihan kriteria pencarian/*display* data detail perawatan
b. Aktor memilih pencarian/*display* seluruh data master barang

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

c. Sistem menampilkan seluruh data detail perawatan yang tersimpan dalam *database*
d. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan *delete* data detail perawatan perawatan

6. Sistem menampilkan data detail perawatan perawatan
7. Aktor memilih data detail perawatan perawatan yang akan dihapus
8. Aktor meminta sistem untuk menghapus data detail perawatan perawatan yang dipilih
9. Sistem melakukan penghapusan data detail perawatan dari *database*
10. Berlanjut ke *Basic Flow* langkah ke 8

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

3. Sistem menampilkan seluruh data detail perawatan yang tersimpan dalam *database* berdasarkan kriteria yang dipilih aktor
4. Berlanjut ke Basic Flow langkah ke 8

13. Error Flow

E-1 Data detail perawatan yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data detail perawatan yang telah diedit salah

3. Sistem memberikan pesan peringatan bahwa data yang diedit salah
4. Kembali ke Alternative Flow A-1 langkah ke 2

14. PreConditions
    1. Use Case *Login* telah dilakukan
    2. Aktor telah memasuki sistem
15. PostConditions
    1. Data detail perawatan di database telah ter*update*

4.1.6.1.5 Use case Spesification : Mengelola Suasana

1. Brief Description

   Use Case ini digunakan oleh aktor untuk mengelola data suasana. Aktor dapat melakukan *entry* data suasana, edit data suasana, read data suasana, atau delete data suasana.

2. Primary Actor
   1. Administrator
3. Supporting Actor

   none
4. Basic Flow
   1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan suasana

2. Sistem memberikan pilihan untuk melakukan *entry* data suasana, *edit* data suasana, *read* data suasana, atau *delete* suasana
3. Aktor memilih untuk melakukan *entry* suasana

   A-1 Aktor memilih untuk melakukan edit suasana

   A-2 Aktor memilih untuk melakukan *read* suasana

   A-3 Aktor memilih untuk melakukan *delete* suasana
4. Aktor menginputkan suasana
5. Aktor meminta sistem untuk menyimpan data suasana yang telah diinputkan
6. Sistem mengecek data suasana yang telah diinputkan

   E-1 Data suasana yang diinputkan aktor salah
7. Sistem menyimpan data suasana ke *database server*
8. Use Case selesai

5. Alternative Flow

   A-1 Aktor memilih untuk melakukan edit data suasana

   1. Sistem menampilkan data suasana
   2. Aktor mengedit data suasana yang sudah ditampilkan
   3. Aktor meminta sistem untuk menyimpan data suasana yang telah diedit
   4. Sistem melakukan pengecekan terhadap suasana yang telah diedit

E-2 Data suasana yang telah di*edit* salah

5. Sistem meyimpan data suasana yang telah diedit ke *database*
6. Berlanjut ke *Basic Flow* langkah ke 8

A-2 Aktor memilih untuk melakukan read data suasana

a. Sistem menampilkan pilihan kriteria pencarian/*display* data suasana
b. Aktor memilih pencarian/*display* seluruh data suasana

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

c. Sistem menampilkan seluruh data suasana yang tersimpan dalam *database*
d. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan *delete* data detail perawatan perawatan

1. Sistem menampilkan data detail perawatan perawatan
2. Aktor memilih data detail perawatan perawatan yang akan dihapus
3. Aktor meminta sistem untuk menghapus data detail perawatan perawatan yang dipilih
4. Sistem melakukan penghapusan data detail perawatan dari *database*
5. Berlanjut ke *Basic Flow* langkah ke 8

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

1. Sistem menampilkan seluruh data detail perawatan yang tersimpan dalam *database* berdasarkan kriteria yang dipilih aktor
2. Berlanjut ke Basic Flow langkah ke 8

6. Error Flow

E-1 Data detail perawatan yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data detail perawatan yang telah diedit salah

5. Sistem memberikan pesan peringatan bahwa data yang diedit salah
6. Kembali ke Alternative Flow A-1 langkah ke 2

7. PreConditions

1. Use Case *Login* telah dilakukan
2. Aktor telah memasuki sistem

8. PostConditions

Data detail perawatan di database telah ter*update*

4.1.6.1.6 Use case Spesification : Mengelola Bahan

1. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data bahan. Aktor dapat melakukan *entry* data bahan, edit data bahan, read data bahan, atau delete data bahan.

2. Primary Actor

1. Administrator

3. Supporting Actor

None

4. Basic Flow

    1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan bahan
    2. Sistem memberikan pilihan untuk melakukan *entry* data bahan, *edit* data bahan, *read* data bahan, atau *delete* bahan
    3. Aktor memilih untuk melakukan *entry* bahan

        A-1 Aktor memilih untuk melakukan edit bahan

        A-2 Aktor memilih untuk melakukan *read* bahan

        A-3 Aktor memilih untuk melakukan *delete* bahan
    4. Aktor menginputkan bahan
    5. Aktor meminta sistem untuk menyimpan data bahan yang telah diinputkan
    6. Sistem mengecek data bahan yang telah diinputkan

        E-1 Data bahan yang diinputkan aktor salah
    7. Sistem menyimpan data bahan ke *database server*
    8. Use Case selesai

5. Alternative Flow

    A-1 Aktor memilih untuk melakukan edit data bahan

        1. Sistem menampilkan data bahan
        2. Aktor mengedit data bahan yang sudah ditampilkan
        3. Aktor meminta sistem untuk menyimpan data bahan yang telah diedit

4. Sistem melakukan pengecekan terhadap bahan yang telah diedit

E-2 Data bahan yang telah di*edit* salah

5. Sistem meyimpan data bahan yang telah diedit ke *database*
6. Berlanjut ke *Basic Flow* langkah ke 8

A-2 Aktor memilih untuk melakukan read data bahan

a. Sistem menampilkan pilihan kriteria pencarian/*display* data bahan
b. Aktor memilih pencarian/*display* seluruh data bahan

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

c. Sistem menampilkan seluruh data bahan yang tersimpan dalam *database*
d. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan *delete* data detail perawatan perawatan

1. Sistem menampilkan data detail perawatan perawatan
2. Aktor memilih data detail perawatan perawatan yang akan dihapus
3. Aktor meminta sistem untuk menghapus data detail perawatan perawatan yang dipilih
4. Sistem melakukan penghapusan data detail perawatan dari *database*
5. Berlanjut ke *Basic Flow* langkah ke 8

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

3. Sistem menampilkan seluruh data detail perawatan yang tersimpan dalam *database* berdasarkan kriteria yang dipilih aktor
4. Berlanjut ke Basic Flow langkah ke 8

6. Error Flow

E-1 Data detail perawatan yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data detail perawatan yang telah diedit salah

7. Sistem memberikan pesan peringatan bahwa data yang diedit salah
8. Kembali ke Alternative Flow A-1 langkah ke 2

7. PreConditions

1. Use Case *Login* telah dilakukan
2. Aktor telah memasuki sistem

8. PostConditions

Data detail perawatan di database telah ter*update*

4.1.6.1.7 Use case Spesification : Mengelola Member

1. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data member. Aktor dapat melakukan *entry* data member, edit data member, read data member, atau delete data member.

2. Primary Actor

1. Administrator

3. Supporting Actor

None

4. Basic Flow

    1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan member
    2. Sistem memberikan pilihan untuk melakukan *entry* data member, *edit* data member, *read* data member, atau *delete* member
    3. Aktor memilih untuk melakukan *entry* member

        A-1 Aktor memilih untuk melakukan edit member

        A-2 Aktor memilih untuk melakukan *read* member

        A-3 Aktor memilih untuk melakukan *delete* member
    4. Aktor menginputkan member
    5. Aktor meminta sistem untuk menyimpan data member yang telah diinputkan
    6. Sistem mengecek data member yang telah diinputkan

        E-1 Data member yang diinputkan aktor salah
    7. Sistem menyimpan data member ke *database server*
    8. Use Case selesai

5. Alternative Flow

    A-1 Aktor memilih untuk melakukan edit data member

        1. Sistem menampilkan data member
        2. Aktor mengedit data member yang sudah ditampilkan

3. Aktor meminta sistem untuk menyimpan data member yang telah diedit
4. Sistem melakukan pengecekan terhadap member yang telah diedit

E-2 Data member yang telah di*edit* salah

5. Sistem meyimpan data member yang telah diedit ke *database*
6. Berlanjut ke *Basic Flow* langkah ke 8

A-2 Aktor memilih untuk melakukan read data member

a. Sistem menampilkan pilihan kriteria pencarian/*display* data member
b. Aktor memilih pencarian/*display* seluruh data member

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

c. Sistem menampilkan seluruh data member yang tersimpan dalam *database*
d. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan *delete* data detail perawatan perawatan

1. Sistem menampilkan data member
2. Aktor memilih data member yang akan dihapus
3. Aktor meminta sistem untuk menghapus data member yang dipilih
4. Sistem melakukan penghapusan data member dari *database*
5. Berlanjut ke *Basic Flow* langkah ke 8

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

5. Sistem menampilkan seluruh member yang tersimpan dalam *database* berdasarkan kriteria yang dipilih aktor
6. Berlanjut ke Basic Flow langkah ke 8

6. Error Flow

E-1 Data member yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data member yang telah diedit salah

1. Sistem memberikan pesan peringatan bahwa data yang diedit salah
2. Kembali ke Alternative Flow A-1 langkah ke 2

7. PreConditions

1. Use Case *Login* telah dilakukan
2. Aktor telah memasuki sistem

8. PostConditions

Data member di database telah ter*update*

4.1.6.1.8 Use case Spesification : Mengelola Role

1. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data role. Aktor dapat melakukan *entry* data role, edit data role, read data role, atau delete data role.

2. Primary Actor

1. Administrator

3. Supporting Actor

None

4. Basic Flow

    1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan role
    2. Sistem memberikan pilihan untuk melakukan *entry* data role, *edit* data role, *read* data role, atau *delete* role
    3. Aktor memilih untuk melakukan *entry* role

        A-1 Aktor memilih untuk melakukan edit role

        A-2 Aktor memilih untuk melakukan *read* role

        A-3 Aktor memilih untuk melakukan *delete* role
    4. Aktor menginputkan role
    5. Aktor meminta sistem untuk menyimpan data role yang telah diinputkan
    6. Sistem mengecek data role yang telah diinputkan

        E-1 Data role yang diinputkan aktor salah
    7. Sistem menyimpan data role ke *database server*
    8. Use Case selesai

5. Alternative Flow

    A-1 Aktor memilih untuk melakukan edit data role

    1. Sistem menampilkan data role
    2. Aktor mengedit data role yang sudah ditampilkan
    3. Aktor meminta sistem untuk menyimpan data role yang telah diedit
    4. Sistem melakukan pengecekan terhadap role yang telah diedit

E-2 Data role yang telah di*edit* salah

5. Sistem meyimpan data role yang telah diedit ke *database*
6. Berlanjut ke *Basic Flow* langkah ke 8

A-2 Aktor memilih untuk melakukan read data role

1. Sistem menampilkan pilihan kriteria pencarian/*display* data role
2. Aktor memilih pencarian/*display* seluruh data role

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

1. Sistem menampilkan seluruh data role yang tersimpan dalam *database*
2. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan *delete* data detail perawatan perawatan

1. Sistem menampilkan data role
2. Aktor memilih data role yang akan dihapus
3. Aktor meminta sistem untuk menghapus data role yang dipilih
4. Sistem melakukan penghapusan data role dari *database*
5. Berlanjut ke *Basic Flow* langkah ke 8

A-4 Aktor memilih pencarian/*display* data berdasarkan kriteria tertentu

1. Sistem menampilkan seluruh role yang tersimpan dalam *database* berdasarkan kriteria yang dipilih aktor
2. Berlanjut ke Basic Flow langkah ke 8

6. Error Flow

E-1 Data role yang diinputkan aktor salah

1. Sistem roleikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data role yang telah diedit salah

1. Sistem roleikan pesan peringatan bahwa data yang diedit salah
2. Kembali ke Alternative Flow A-1 langkah ke 2

7. PreConditions
   1. Use Case *Login* telah dilakukan
   2. Aktor telah memasuki sistem
8. PostConditions

   Data role di database telah ter*update*

4.1.6.1.9 Use case Spesification : Mengelola *Account*

1. Brief Description

   Use Case ini digunakan oleh aktor untuk mengelola data *account*. Aktor dapat melakukan edit *account* atau delete *account*.
2. Primary Actor
   1. *Member*
3. Supporting Actor

   None
4. Basic Flow
   1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan *account*
   2. Sistem memberikan pilihan untuk melakukan edit *account* atau delete *account*
   3. Aktor memilih untuk melakukan edit *account*

A-1 Aktor memilih untuk melakukan *delete Account*

4. Sistem menampilkan data *account*
5. Aktor mengedit data *account* yang sudah ditampilkan
6. Aktor meminta sistem untuk menyimpan data *account* yang telah diedit
7. Sistem melakukan pengecekan terhadap role yang telah diedit

E-2 Data *account* yang telah di*edit* salah

8. Sistem menyimpan data *account* ke *database server*
9. Use Case selesai

5. Alternative Flow

A-1 Aktor memilih untuk melakukan *delete* data detail perawatan perawatan

1. Sistem menampilkan data *account*
2. Aktor memilih data *account* yang akan dihapus
3. Aktor meminta sistem untuk menghapus data *account* yang dipilih
4. Sistem melakukan penghapusan data *account* dari *database*
5. Berlanjut ke *Basic Flow* langkah ke 8

6. Error Flow

E-1 Data *account* yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

E-2 Data role yang telah diedit salah

3. Sistem memberikan pesan peringatan bahwa data yang diedit salah
4. Kembali ke Alternative Flow A-1 langkah ke 2

6. PreConditions
   1. Use Case *Login* telah dilakukan
   2. Aktor telah memasuki sistem
7. PostConditions
   Data role di database telah ter*update*

4.1.6.1.10 Use case Spesification : Memberi Komentar

1. Brief Description
   Use Case ini digunakan oleh aktor untuk member komentar.
2. Primary Actor
   1. Member, Administrator
3. Supporting Actor
   None
4. Basic Flow
   1. Use Case ini dimulai ketika aktor memilih untuk memberi komentar
   2. Sistem memberikan pilihan untuk memberi komentar
   3. Aktor memilih untuk memberi komentar
   4. Aktor meminta sistem untuk menyimpan komentar yang telah diedit
   5. Sistem melakukan pengecekan terhadap *account* yang telah diedit
      E-2 Komentar yang telah diinputkan salah

6. Sistem menyimpan data *account* ke *database server*
7. Use Case selesai

5. Alternative Flow

-

6. Error Flow

E-1 Komentar yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
2. Kembali ke Basic Flow langkah ke 4

6. PreConditions

1. Use Case *Login* telah dilakukan
2. Aktor telah memasuki sistem

7. PostConditions

Komentar di database telah ter*update*

4.1.6.1.11 Use case Spesification : Mendaftar jadi anggota

1. Brief Description

Use Case ini digunakan oleh aktor untuk mendaftar menjadi anggota.

2. Primary Actor

1. Guest

3. Supporting Actor

None

4. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk mendaftar menjadi anggota
2. Sistem memberikan pilihan untuk mendaftar menjadi anggota

3. Aktor memilih untuk mendaftar menjadi anggota
4. Aktor meminta sistem untuk menyimpan data anggota yang telah diinputkan
5. Sistem melakukan pengecekan terhadap data *account* yang telah diedit

   E-2 data *account* yang telah diinputkan salah
6. Sistem menyimpan data *account* ke *database server*
7. Use Case selesai

5. Alternative Flow

   -

7. Error Flow

   E-1 Data *Account* yang diinputkan aktor salah

   1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
   2. Kembali ke Basic Flow langkah ke 4

6. PreConditions

   1. Use Case *Login* telah dilakukan
   2. Aktor telah memasuki sistem

7. PostConditions

   Komentar di database telah ter*update*

4.1.6.1.12 Use case Spesification : Pencarian Salon Berdasarkan Spesifikasi Tertentu

1. Brief Description

   Use Case ini digunakan oleh aktor untuk mencari perawatan sesuai dengan *budget*, perawatan yang diinginkan, suasana yang diinginkan, dan bahan yang diinginkan.

2. Primary Actor

1. User

3. Supporting Actor

   none

4. Basic Flow

   1. Use Case ini dimulai ketika aktor memilih untuk melakukan pencarian perawatan.
   2. Sistem menampilkan antarmuka untuk melakukan pencarian perawatan kecantikan.
   3. Aktor menginputkan data-data yang diperlukan dalam pencarian, seperti *budget*, jenis perawatan yang diinginkan, suasana yang diinginkan, dan bahan yang diinginkan.
   4. Aktor meminta sistem untuk mencari data perawatan sesuai dengan data yang diinputkan.
   5. Sistem mengecek data pencarian perawatan kecantikan yang telah diinputkan

      E-1 Data yang diinputkan kurang lengkap atau salah

   6. Sistem menampilkan rekomendasi dari hasil pencarian perawatan kecantikan.
   7. Use Case selesai

5. Alternative Flow

   None

6. Error Flow

   E-1 Data yang diinputkan kurang lengkap atau salah

   1. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
   2. Kembali ke Basic Flow langkah ke 3

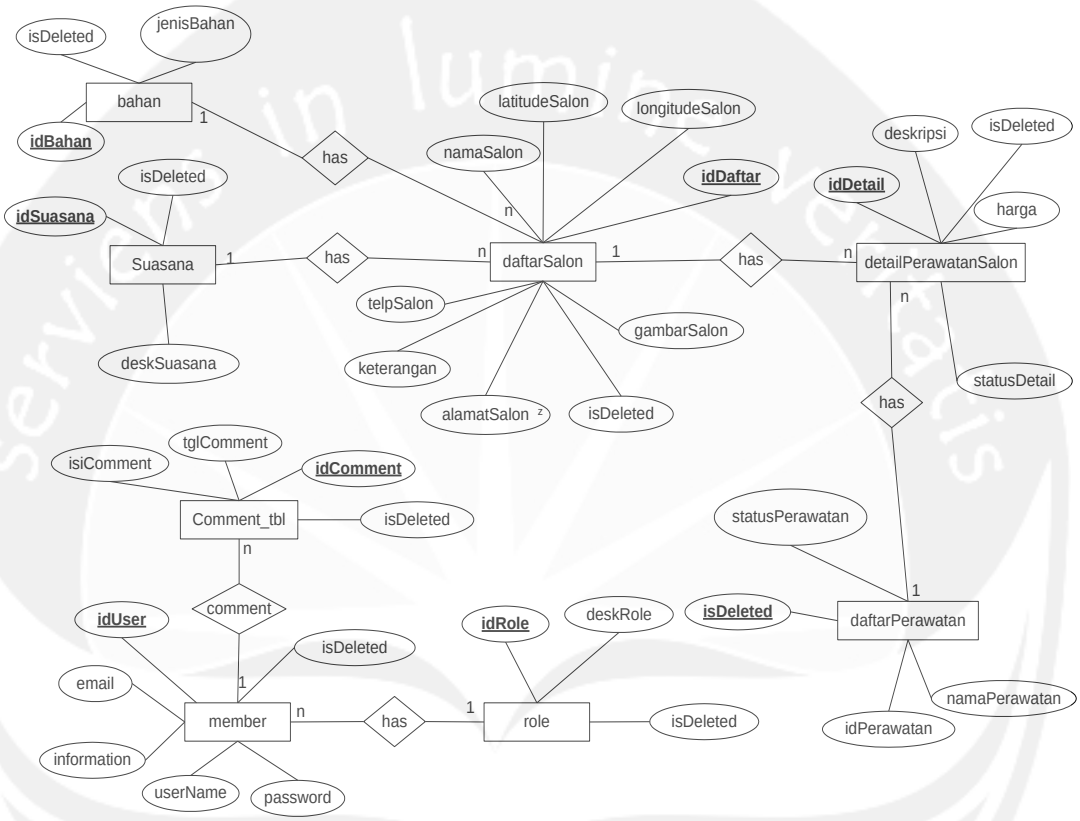
7. PreConditions

   1. Aktor telah memasuki sistem

8. PostConditions

Aktor memperoleh perawatan kecantikan hasil pencarian.

6 Entity Relationship Diagram (ERD)



Gambar 3 *Entity Relationship Diagram*

DPPL

DESKRIPSI PERANCANGAN PERANGKAT LUNAK

getYourSalon

(Sistem Pendukung Keputusan Pemilihan Salon Kecantikan Berbasis *Mobile*)

Untuk :

Universitas Atma Jaya Yogyakarta

Dipersiapkan oleh:

Vera Hannyta / 080705565

Program Studi Teknik Informatika – Fakultas Teknologi Industri

Universitas Atma Jaya Yogyakarta

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	

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

1 Pendahuluan

1.1. Tujuan

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

1.2. Ruang Lingkup

Perangkat Lunak *getYourSalon* dikembangkan dengan tujuan untuk :

4. Menangani proses pencarian salon kecantikan dengan atribut pencarian *budget*, jenis perawatan, suasana salon, dan bahan yang digunakan.
5. Menangani pengelolaan Admin.
6. Menangani penampilan posisi salon terpilih dalam bentuk peta.

1.3. Definisi dan Akronim

Daftar definisi akronim dan singkatan :

Keyword/Phrase	Definisi
Internet	Internet merupakan istilah umum yang dipakai untuk menunjuk *Network* global yang terdiri dari komputer dan layanan servis dengan sekitar 30 sampai 50 juta pemakai komputer dan puluhan layanan informasi termasuk e-mail, FTP, dan World Wide Web.
DPPL	Deskripsi Perancangan Perangkat Lunak disebut juga Software Design Description

	(SDD) merupakan deskripsi dari perancangan produk/perangkat lunak yang akan dikembangkan.
getYourSalon	Perangkat lunak untuk pendukung keputusan dalam pemilihan salon kecantikan.
DataBase	Kumpulan data yang terkait yang diorganisasikan dalam struktur tertentu dan dapat diakses dengan cepat.
Internet	Internet merupakan istilah umum yang dipakai untuk menunjuk *Network* global yang terdiri dari komputer dan layanan servis dengan sekitar 30 sampai 50 juta pemakai komputer dan puluhan layanan informasi termasuk e-mail, FTP, dan World Wide Web.

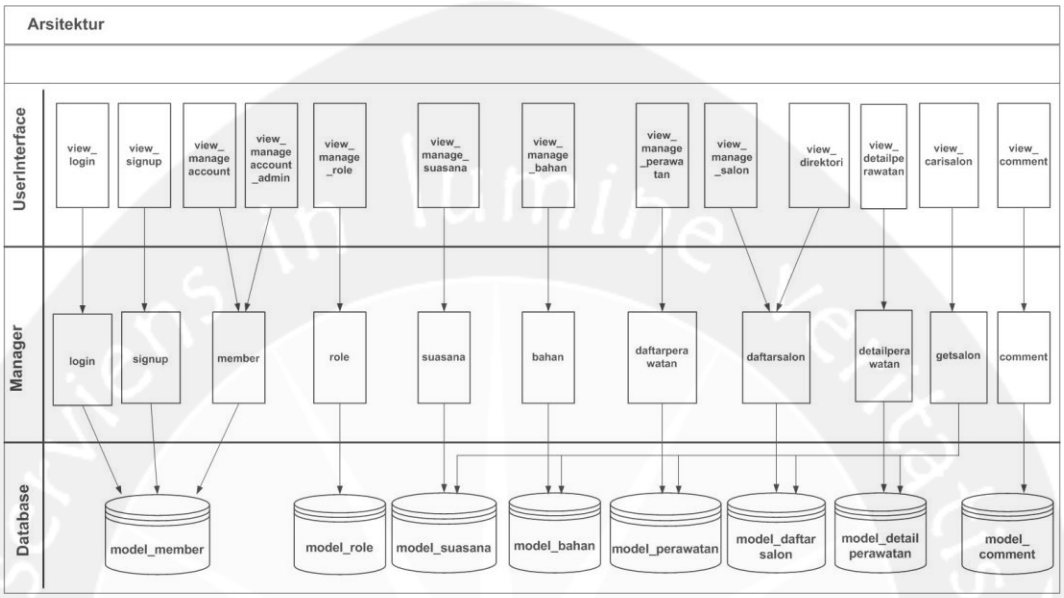
1.4. Referensi

Referensi yang digunakan pada perangkat lunak tersebut adalah:

1. Hannyta Vera, *Deskripsi Perancangan Perangkat Lunak ASM*, Universitas Atma Jaya Yogyakarta, 2011.
2. Hannyta Vera, *Deskripsi Perancangan Perangkat Lunak* HyuBOS, CV Sumber Baru Agung, 2011.
3. Hannyta Vera, Spesifikasi Kebutuhan Perangkat Lunak SPK_PeWaRu, Universitas Atma Jaya Yogyakarta, 2011.
4. Wibisono Aryo, *Deskripsi Perancangan Perangkat Lunak SIAMA*, Universitas Atma Jaya Yogyakarta, 2010.

2 Perancangan Sistem

1.1. Perancangan Arsitektur

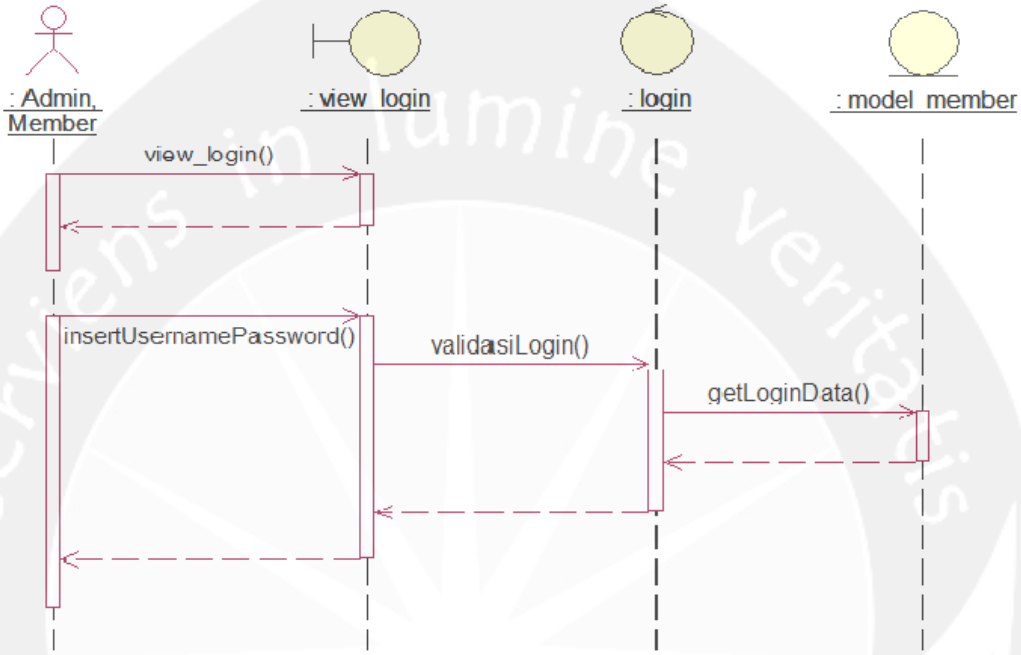


Gambar 2.1 Perancangan Arsitektur *getYourSalon*

1.2. Perancangan Rinci

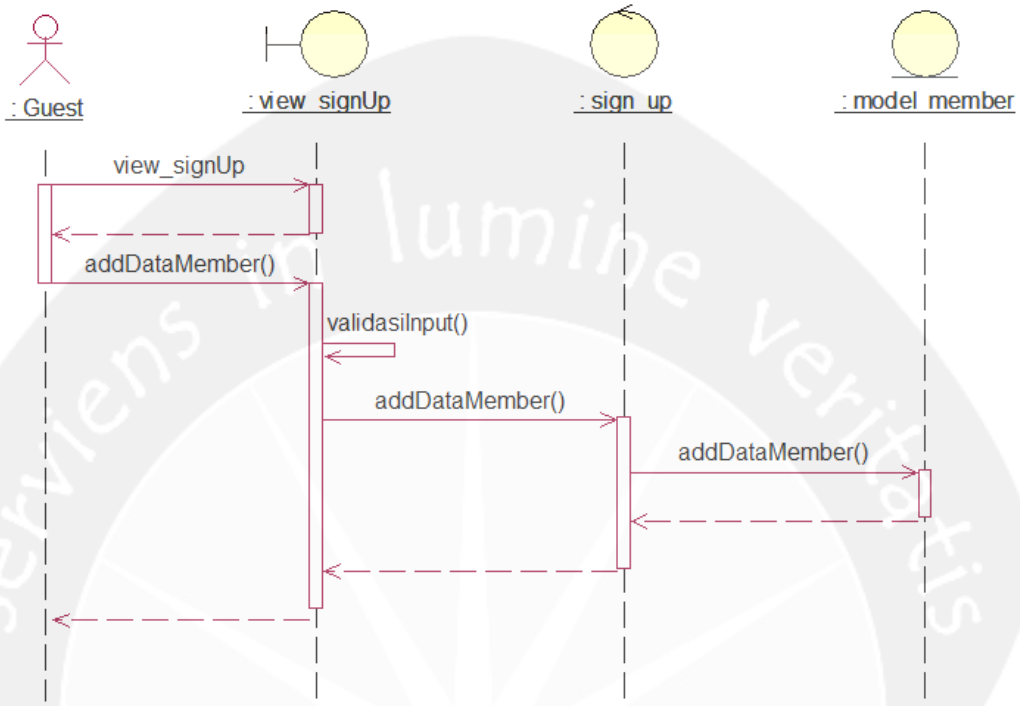
2.2.1 Sequence Diagram

2.2.1.5 Login



Gambar 2.2 Sequence Diagram : Login

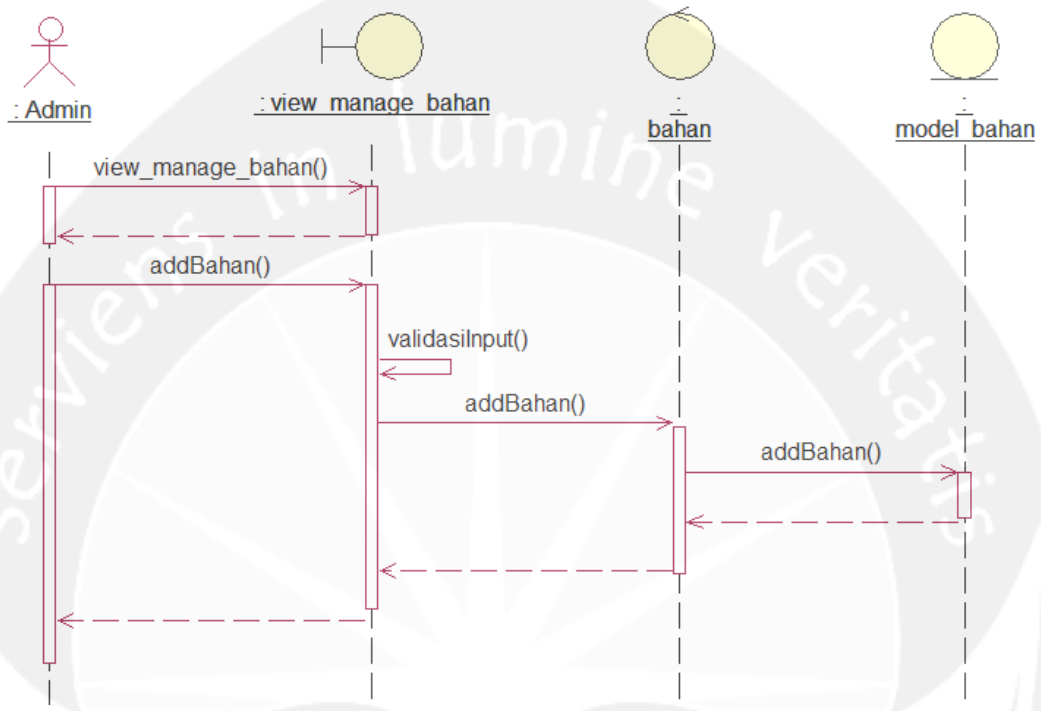
2.2.1.6 Daftar Add Member



Gambar 2.3 Sequence Diagram : Add Member

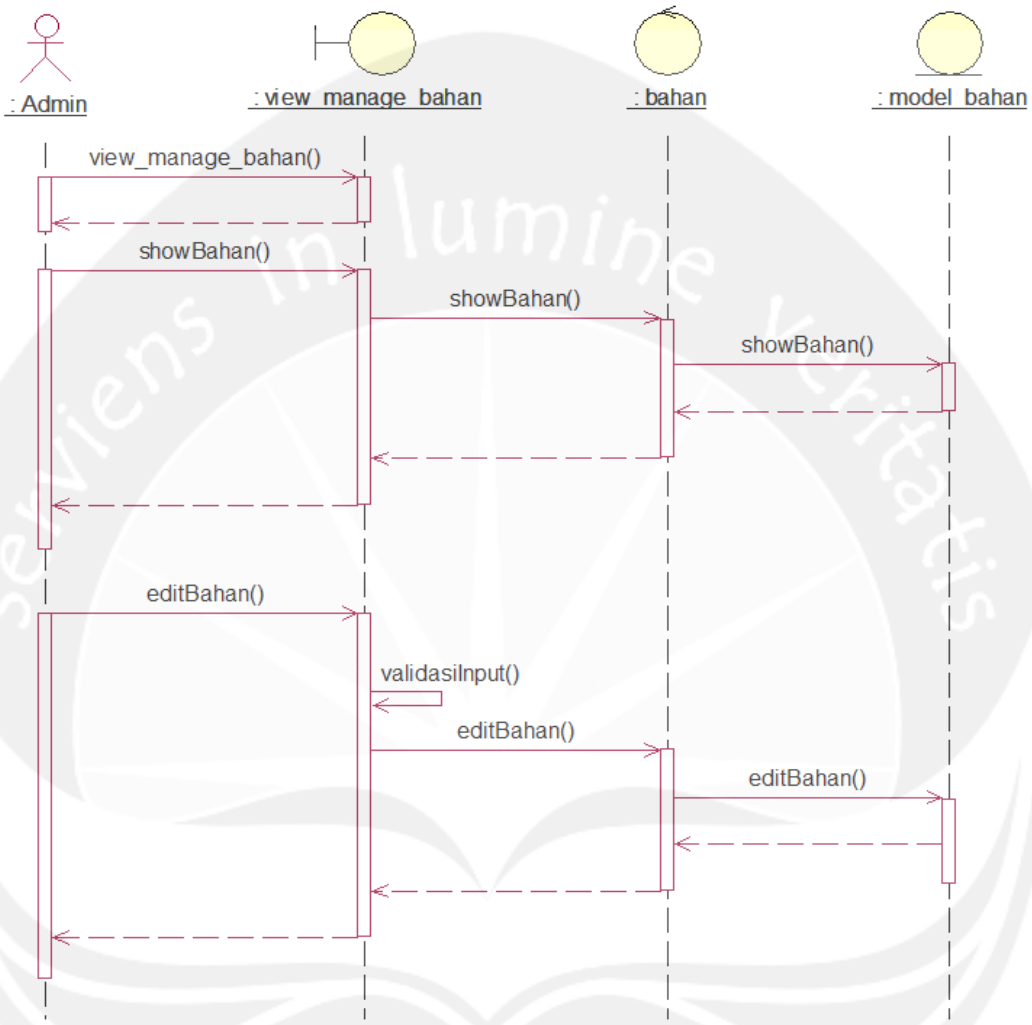
2.2.1.7 Pengelolaan Data Bahan

4.2.2.3.1 Add Data Bahan



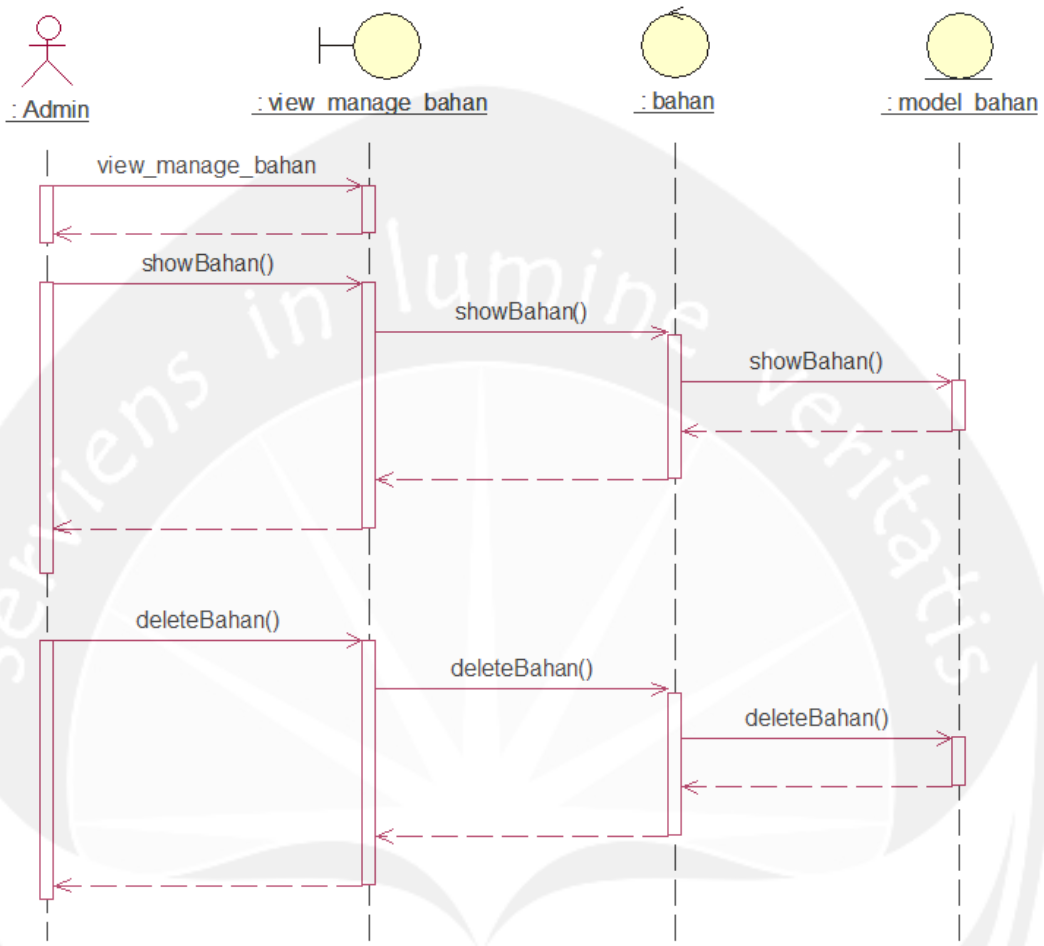
Gambar 2.4 Sequence Diagram : Add Data Bahan

4.2.2.3.2 Edit Data Bahan



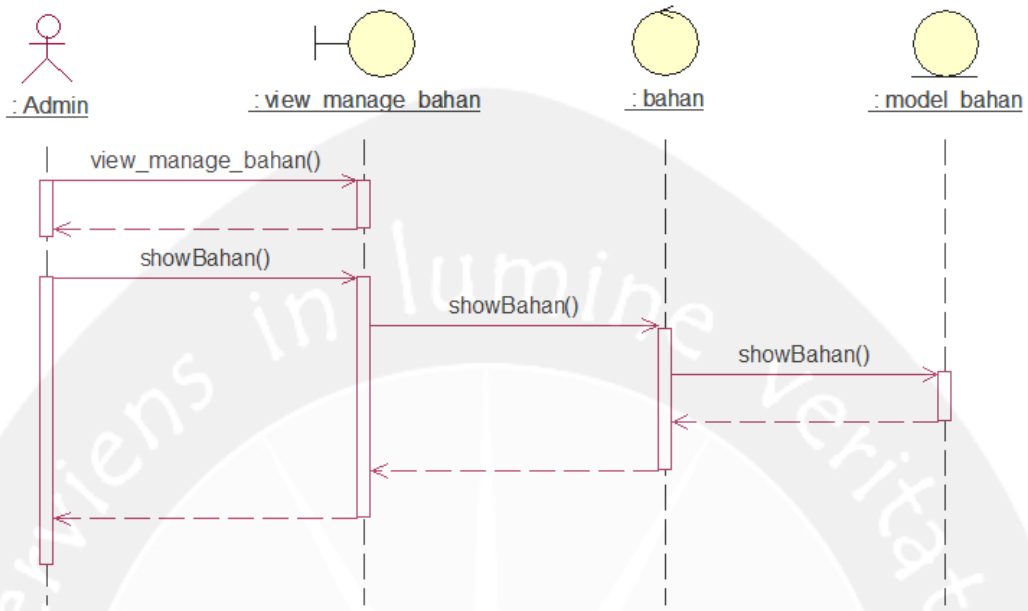
Gambar 2.5 Sequence Diagram : Edit Data Bahan

4.2.2.3.3 Delete Data Bahan



Gambar 2.6 Sequence Diagram : Delete Data Bahan

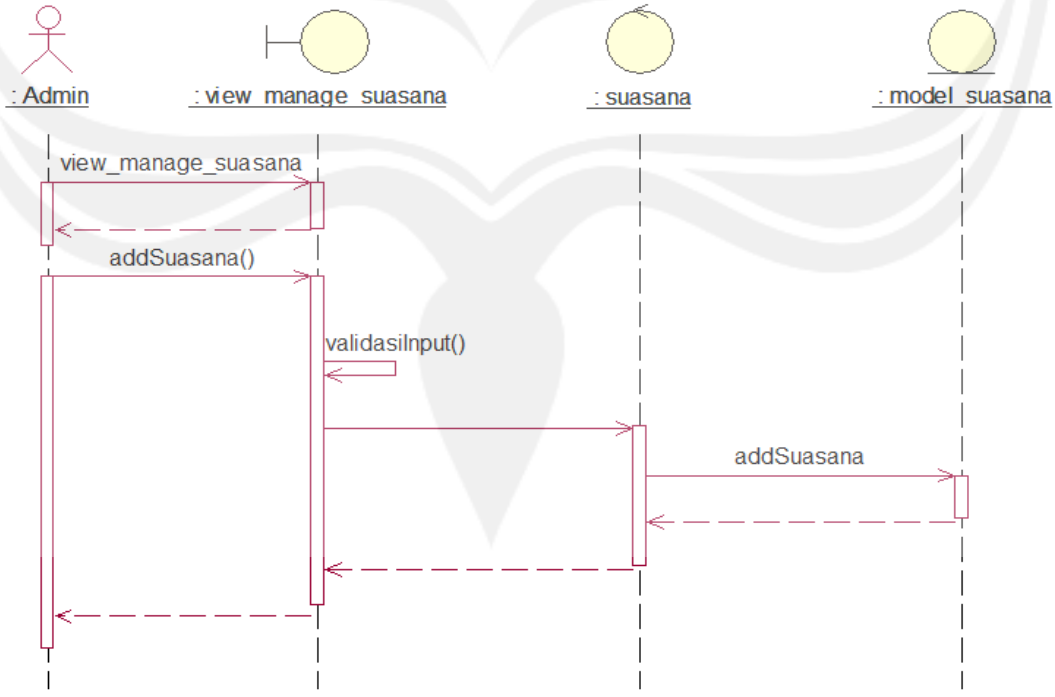
4.2.2.3.4 Show Data Bahan



Gambar 2.7 Sequence Diagram : Show Data Bahan

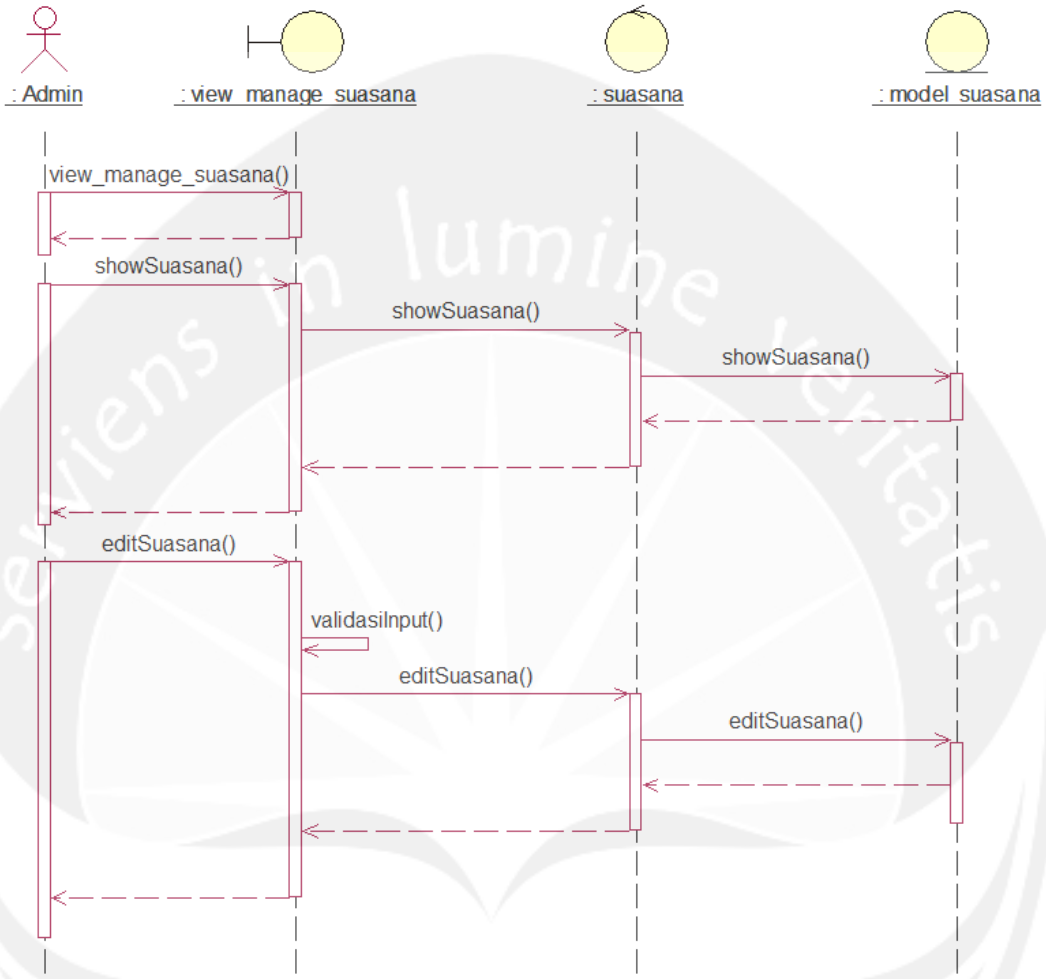
2.2.1.8 Pengelolaan Data Suasana

2.2.2.4.1 Add Data Suasana



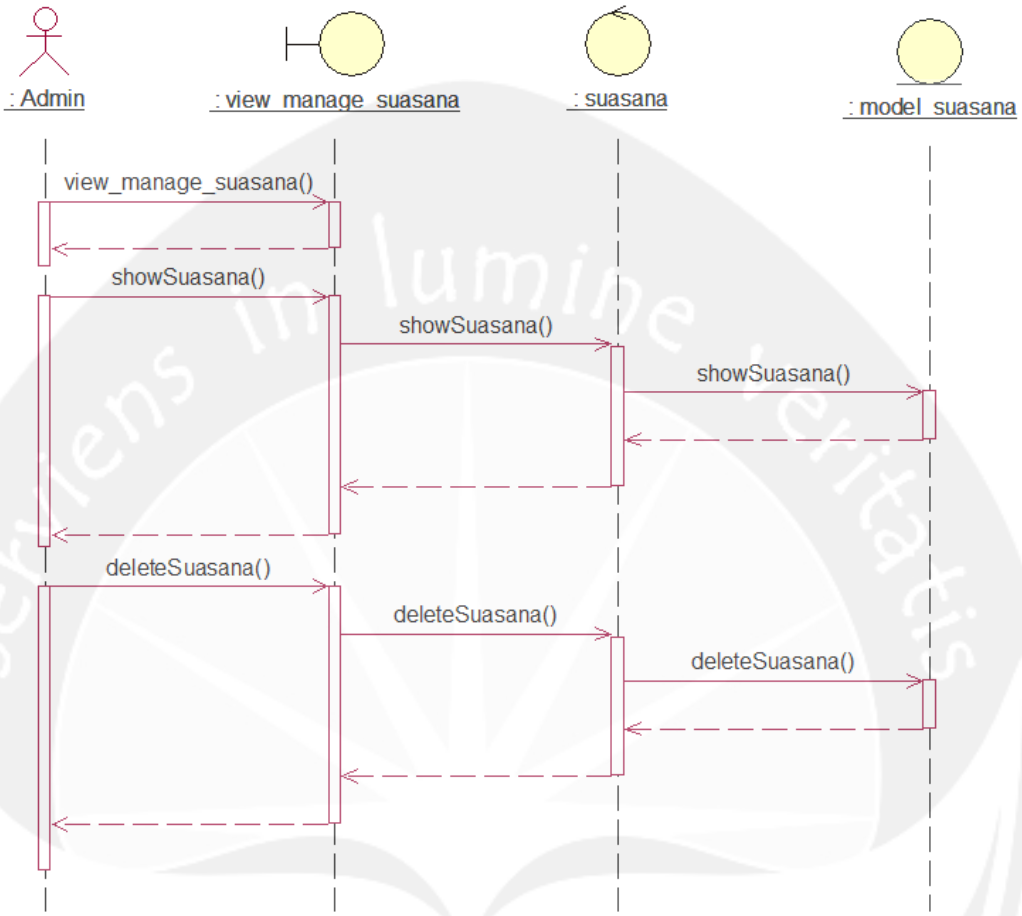
Gambar 2.8 Sequence Diagram : Add Data Suasana

2.2.2.4.2 Edit Data Suasana



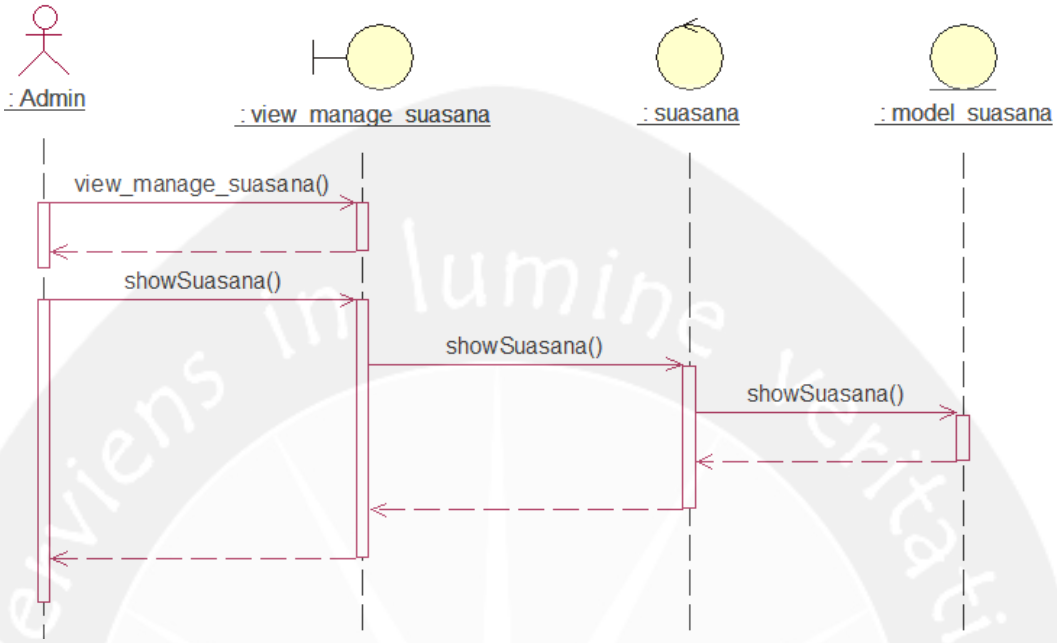
Gambar 2.9 Sequence Diagram : Edit Data Suasana

2.2.2.4.3 Delete Data Suasana



Gambar 2.10 Sequence Diagram : Delete Data Suasana

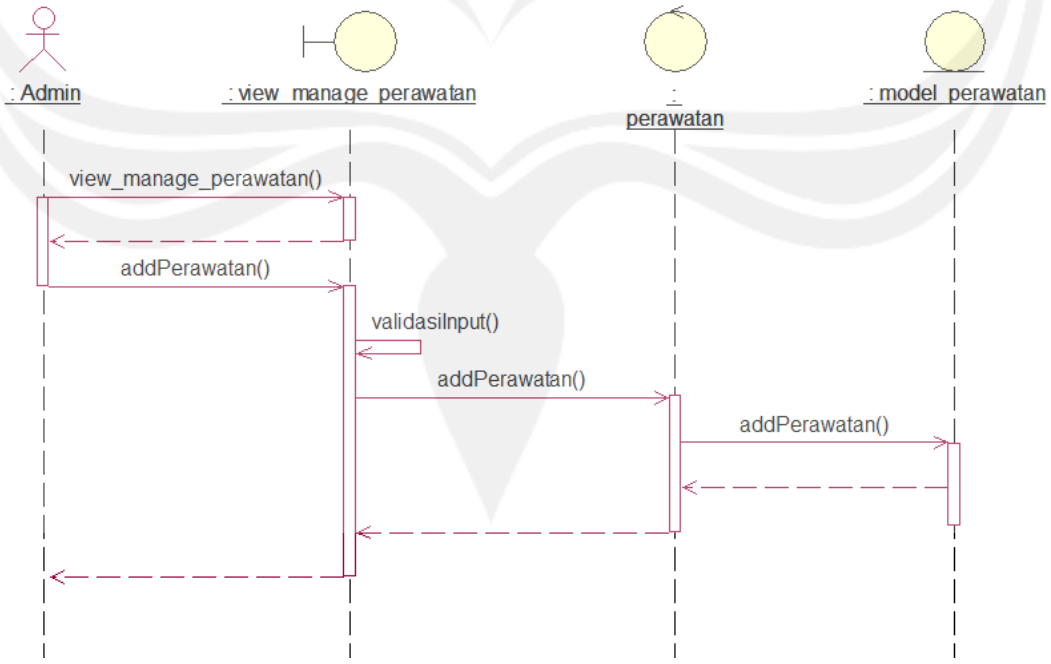
2.2.2.4.4 Show Data Suasana



Gambar 2.11 Sequence Diagram : Show Data Suasana

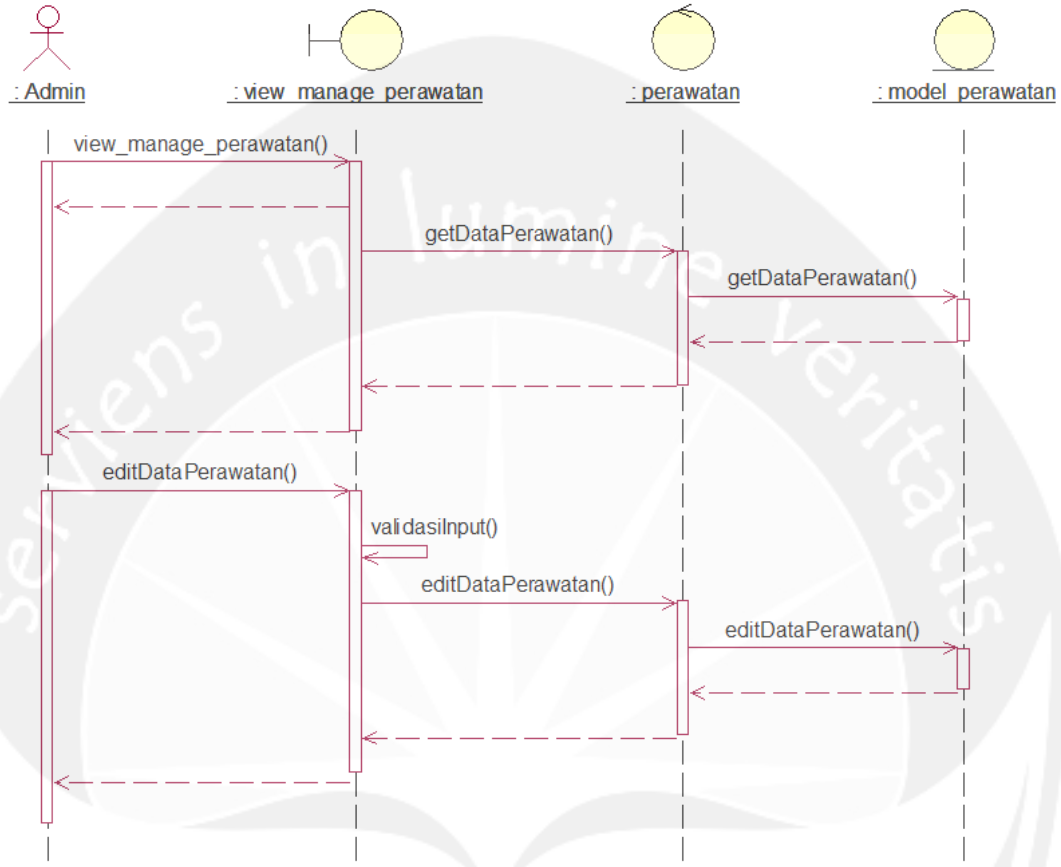
2.2.1.9 Pengelolaan Data Perawatan

2.2.2.5.1 Add Data Perawatan



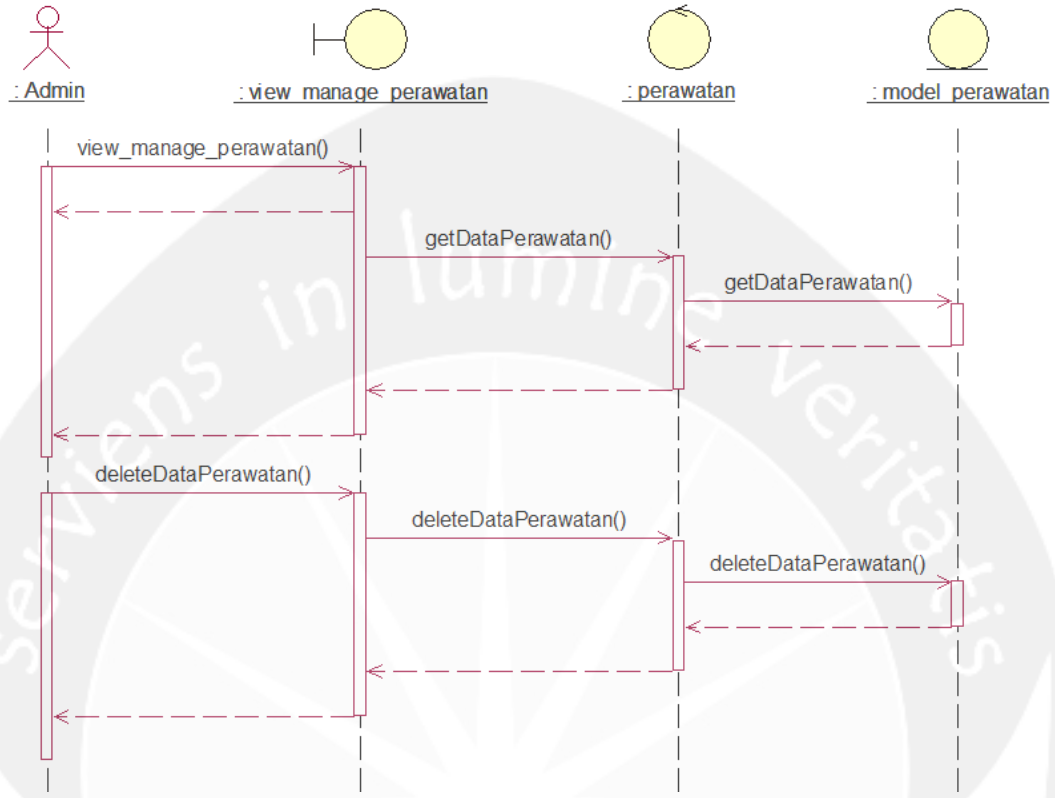
Gambar 2.12 Sequence Diagram : Add Data Perawatan

2.2.2.5.2 Edit Data Perawatan



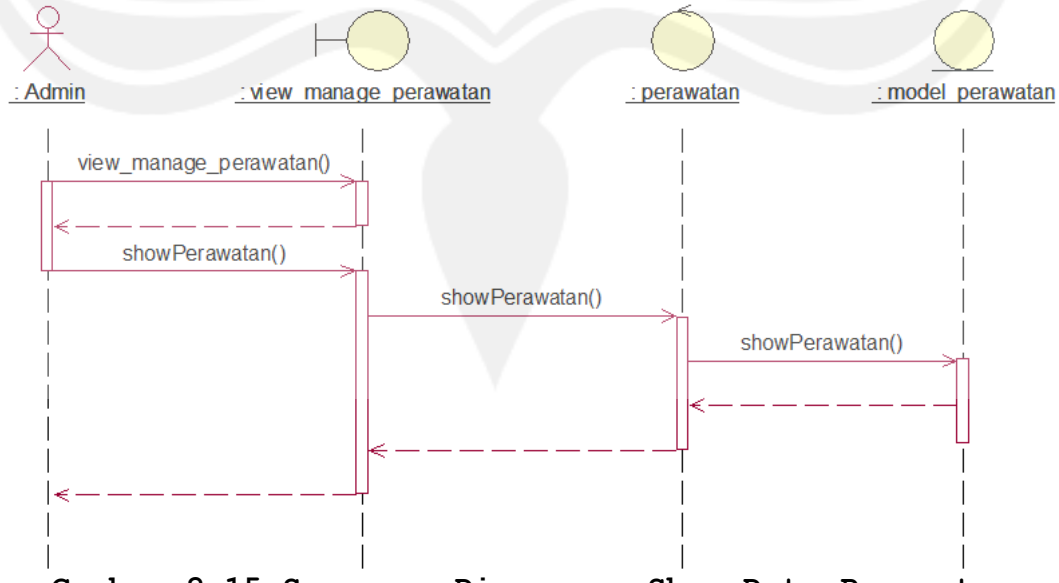
Gambar 2.13 Sequence Diagram : Edit Data Perawatan

2.2.2.5.3 Delete Data Perawatan



Gambar 2.14 Sequence Diagram : Delete Data Perawatan

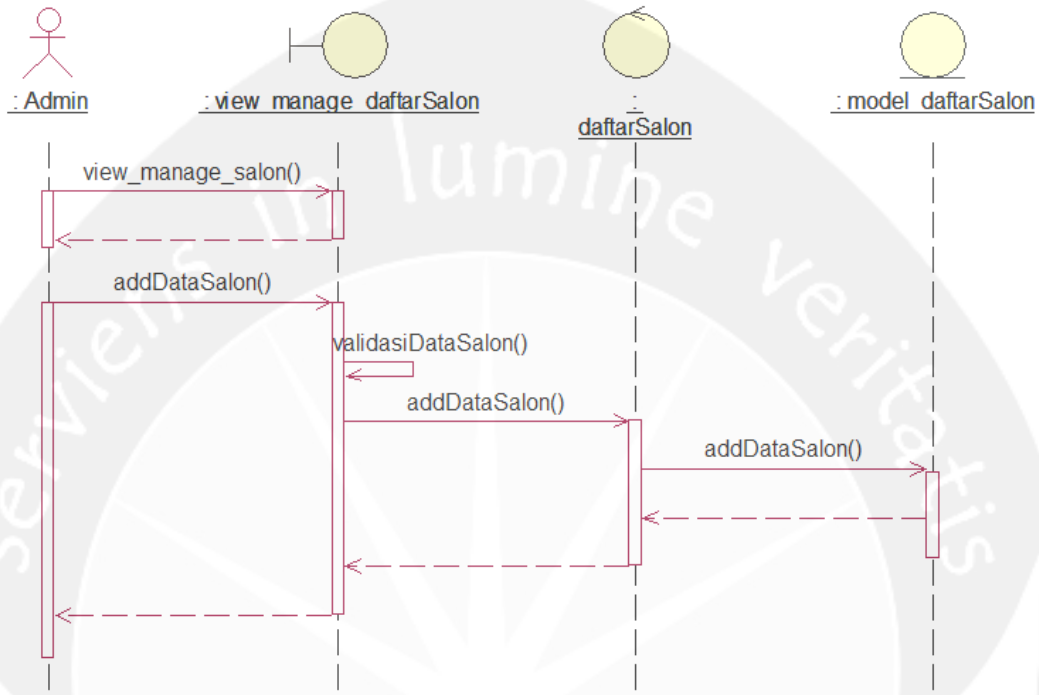
2.2.2.5.4 Show Data Perawatan



Gambar 2.15 Sequence Diagram : Show Data Perawatan

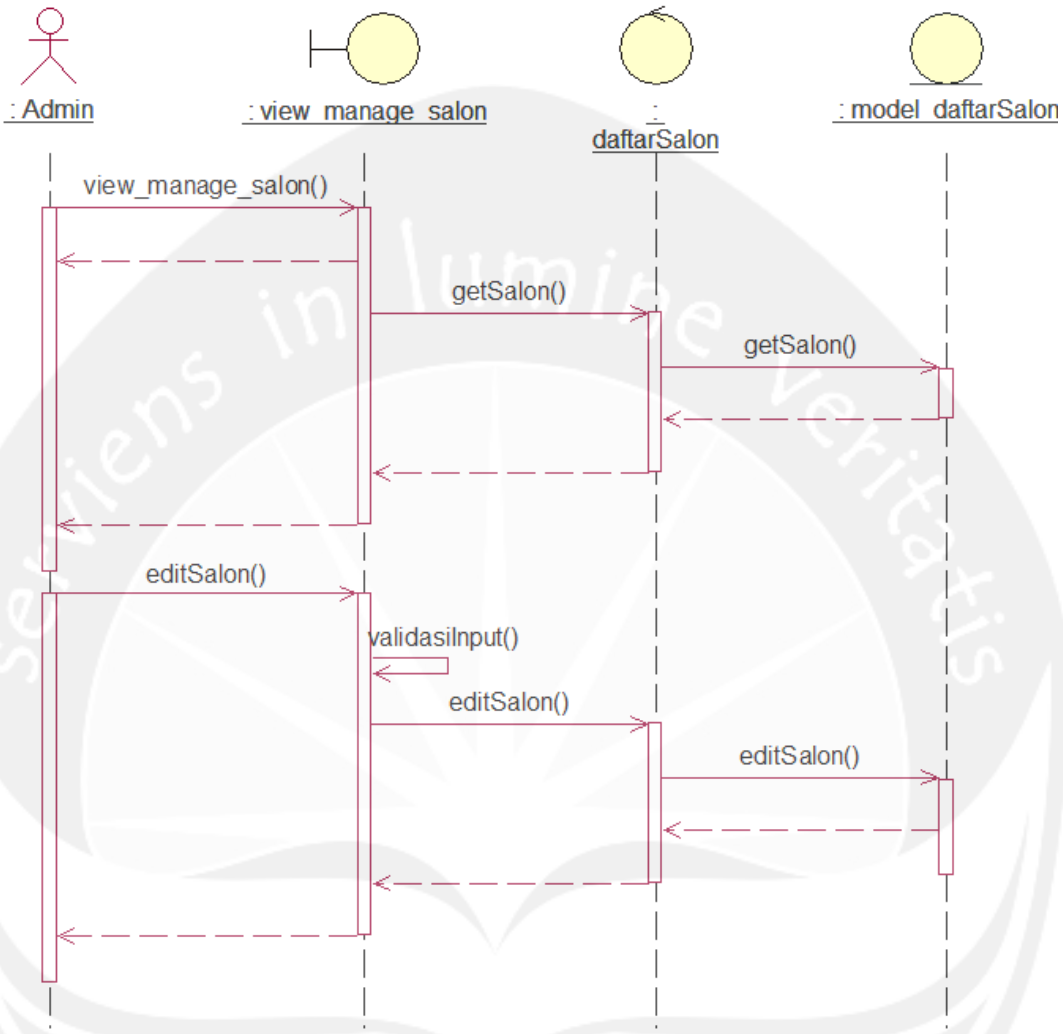
2.2.1.10 Pengelolaan Data Daftar Salon

4.2.2.2.5 Add Daftar Salon



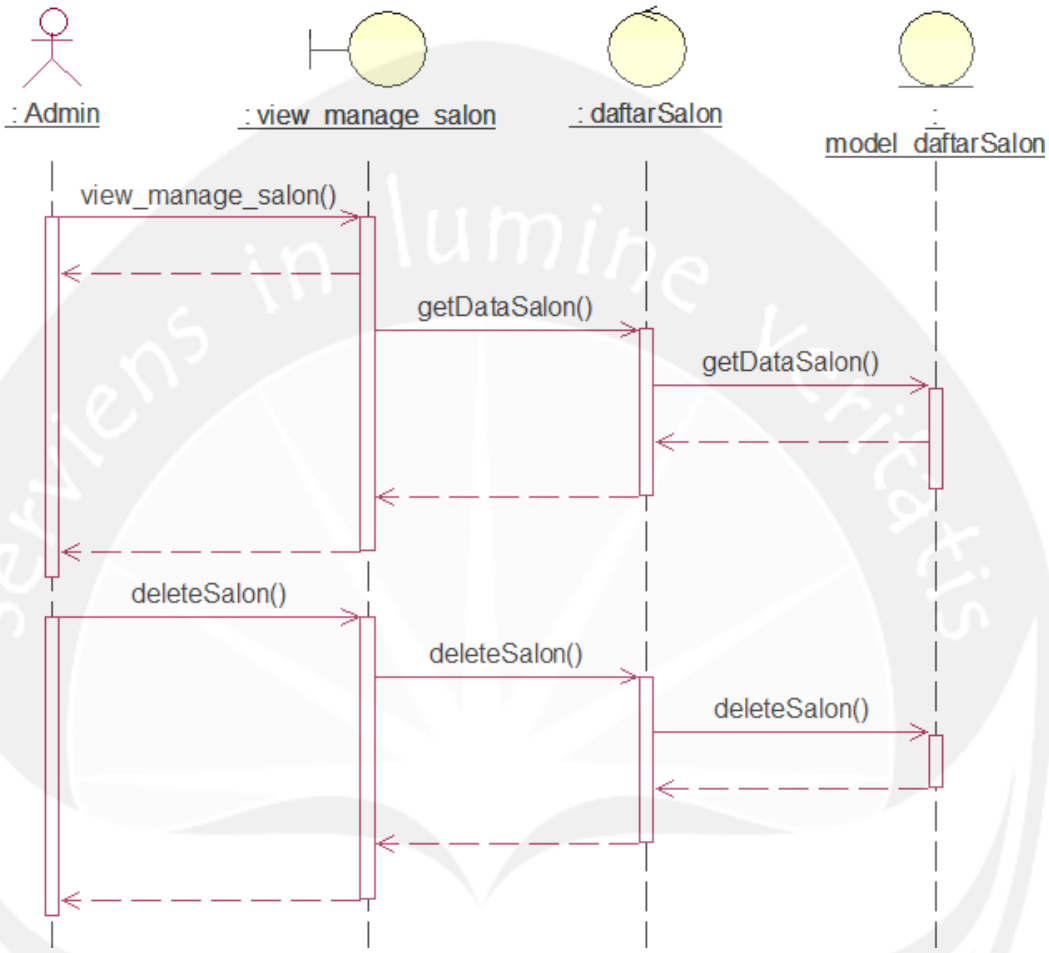
Gambar 2.16 Sequence Diagram : Add Daftar Salon

4.2.2.2.6 Edit Daftar Salon



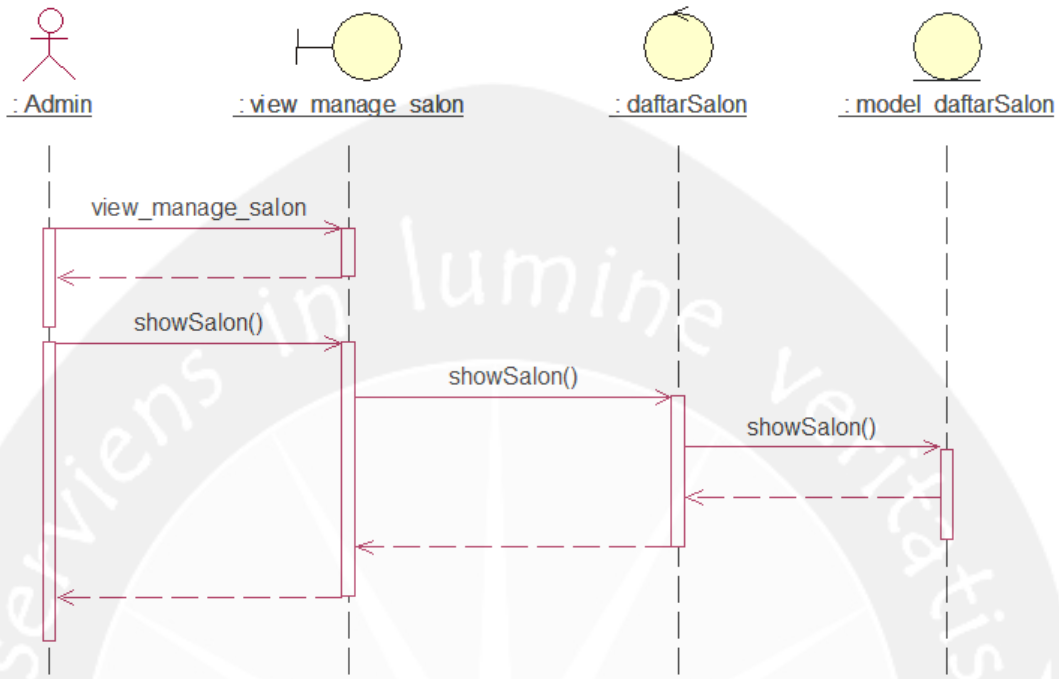
Gambar 2.17 Sequence Diagram : Edit Daftar Salon

4.2.2.2.7 Delete Daftar Salon



Gambar 2.18 Sequence Diagram : Delete Daftar Salon

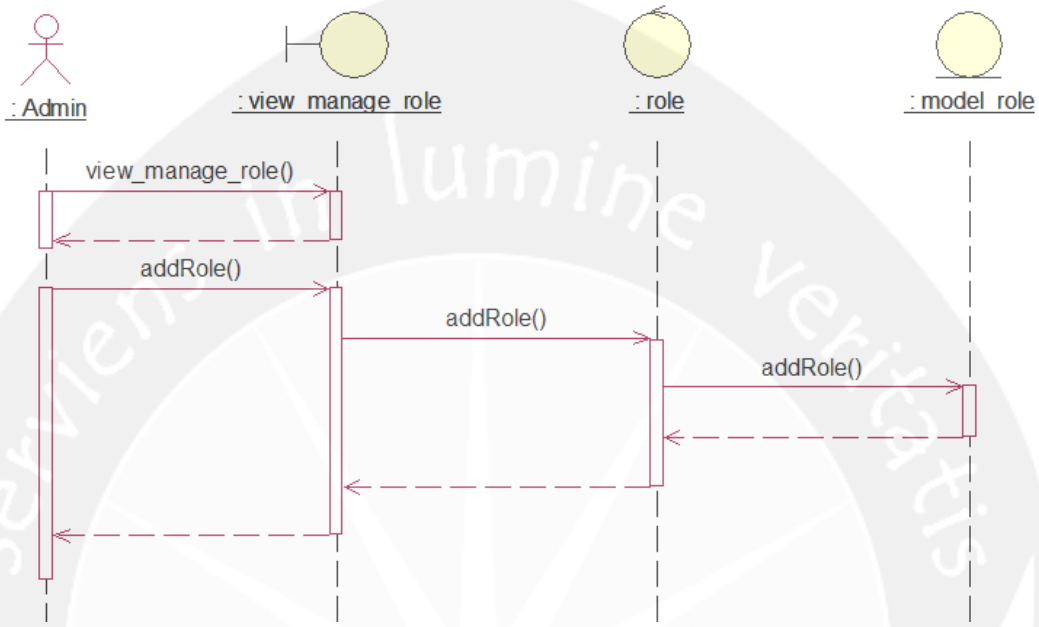
4.2.2.2.8 Show Daftar Salon



Gambar 2.19 Sequence Diagram : Show Daftar Salon

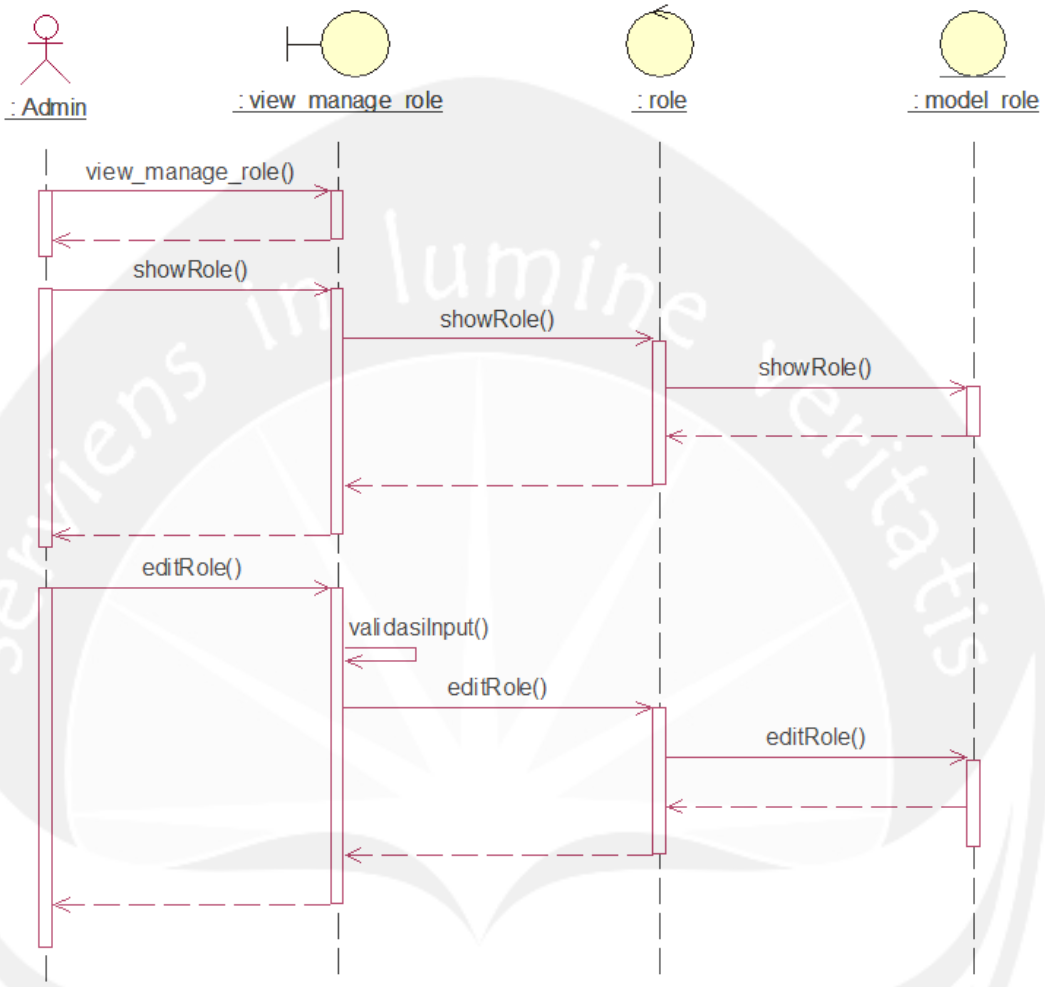
2.2.1.11 Pengelolaan Role

2.2.2.7.1 Add Role



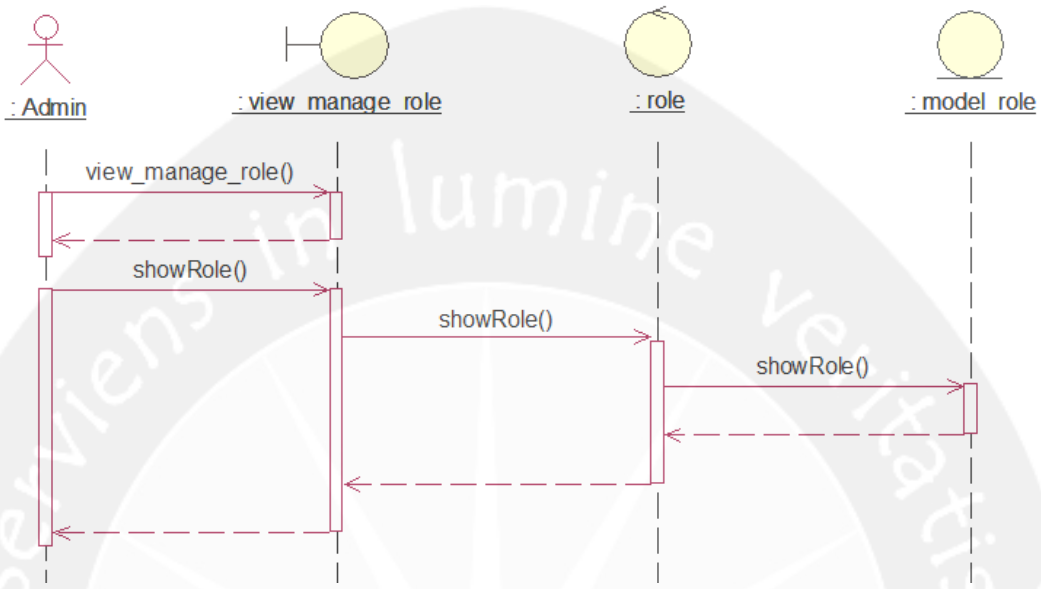
Gambar 2.20 Sequence Diagram : Add Role

2.2.2.7.2 Edit Role



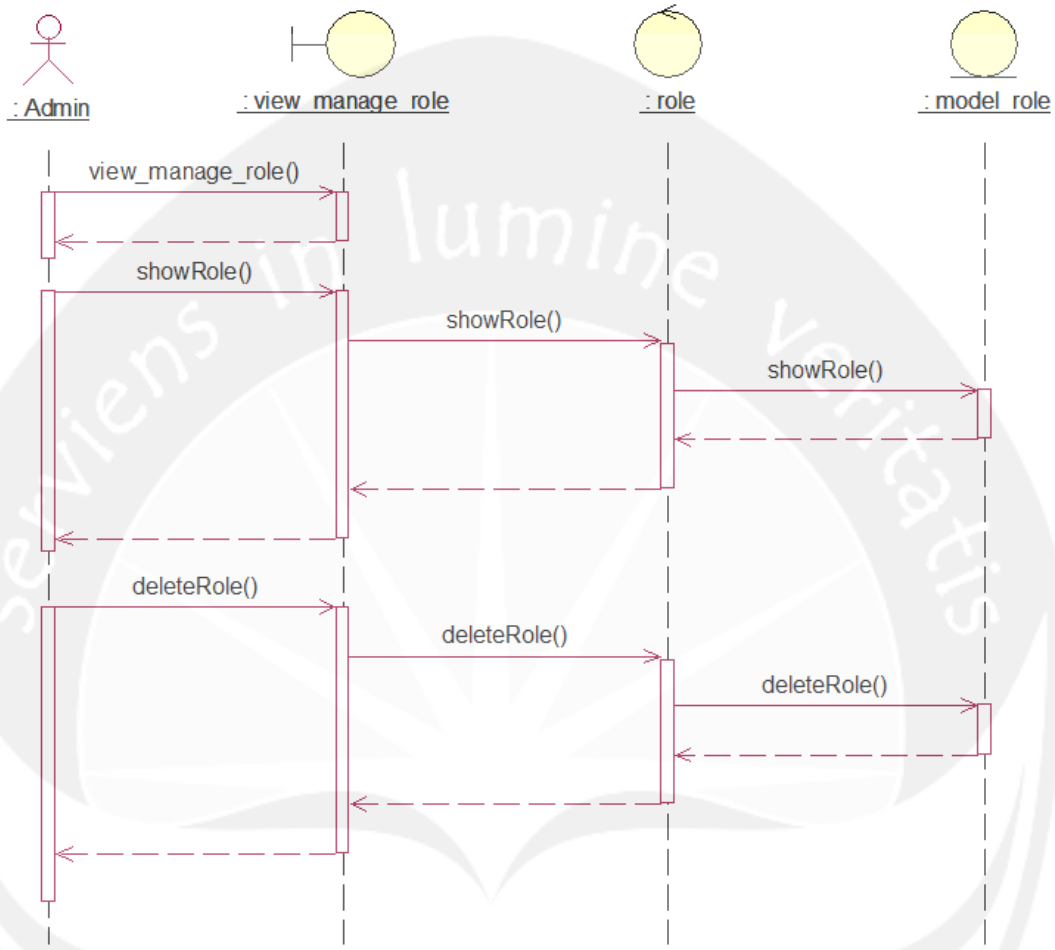
Gambar 2.21 Sequence Diagram : Edit Role

2.2.2.7.3 Show Role



Gambar 2.22 Sequence Diagram : Show Role

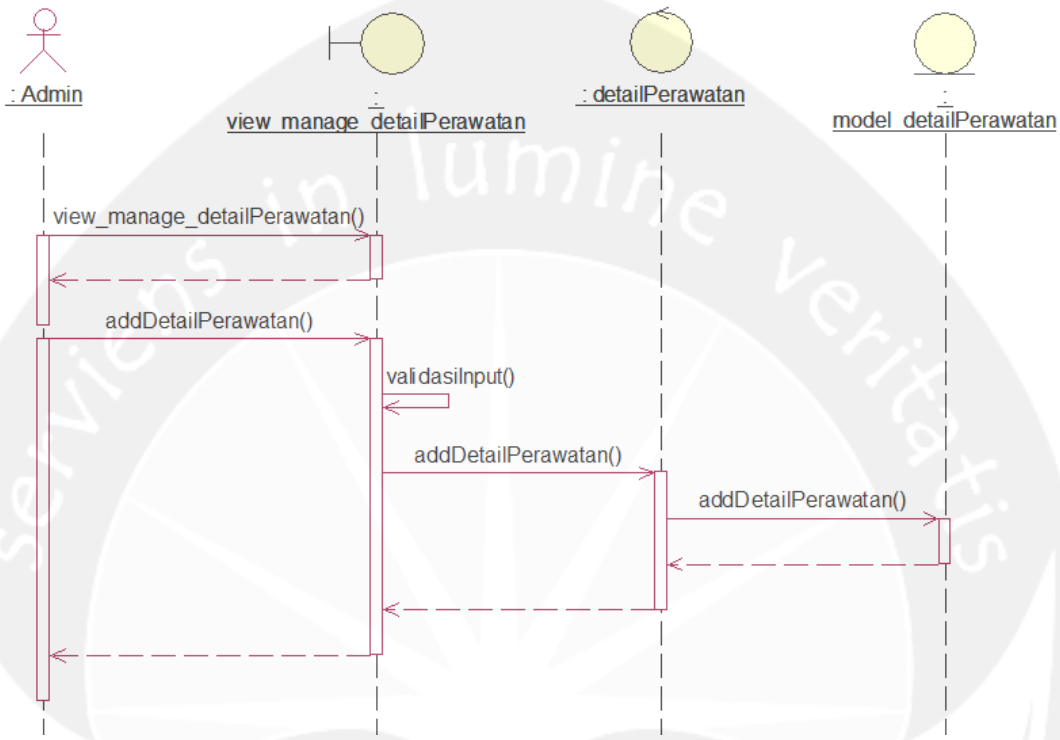
2.2.2.7.4 Delete Role



Gambar 2.23 Sequence Diagram : Delete Role

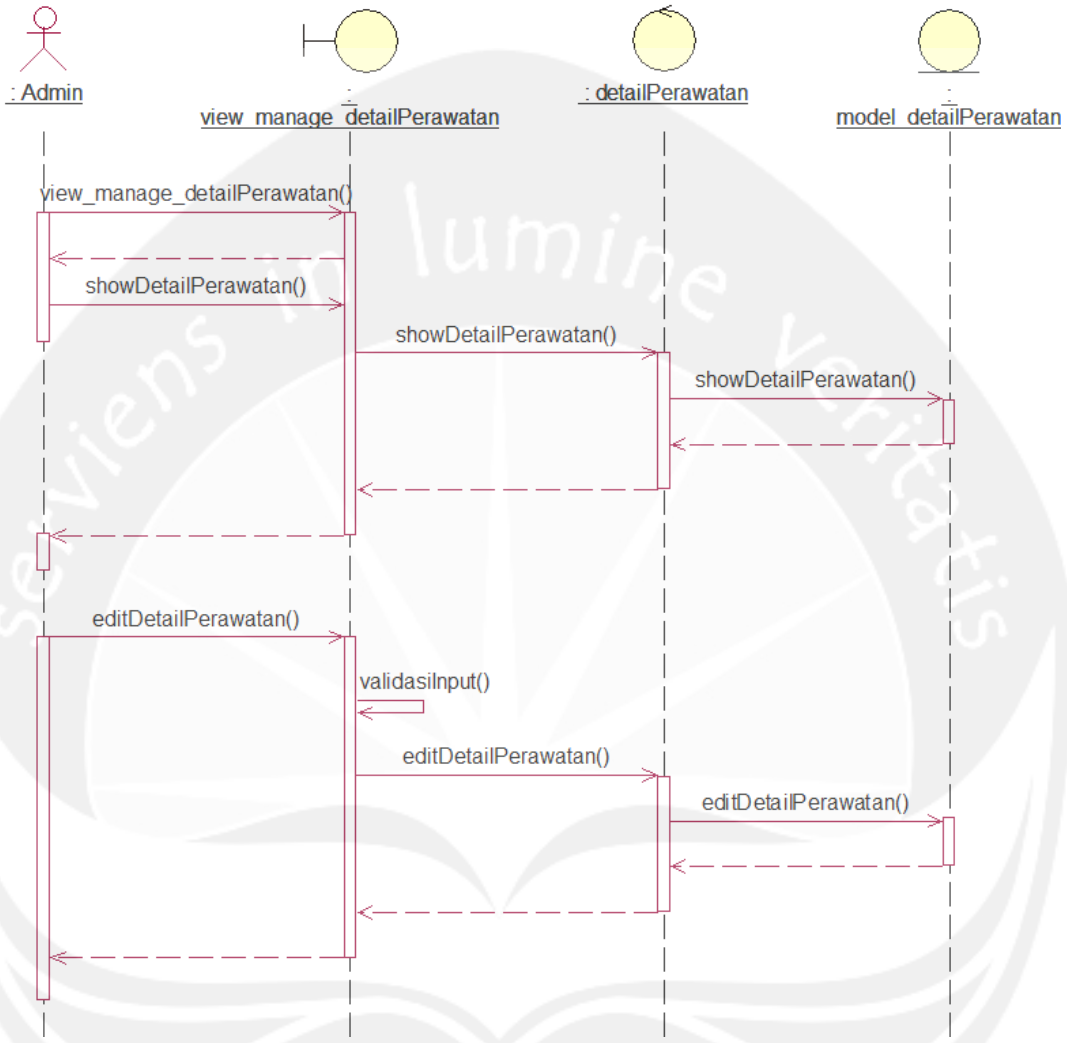
2.2.1.12 Pengelolaan Detail Perawatan

4.2.2.3.5 Add Detail Perawatan



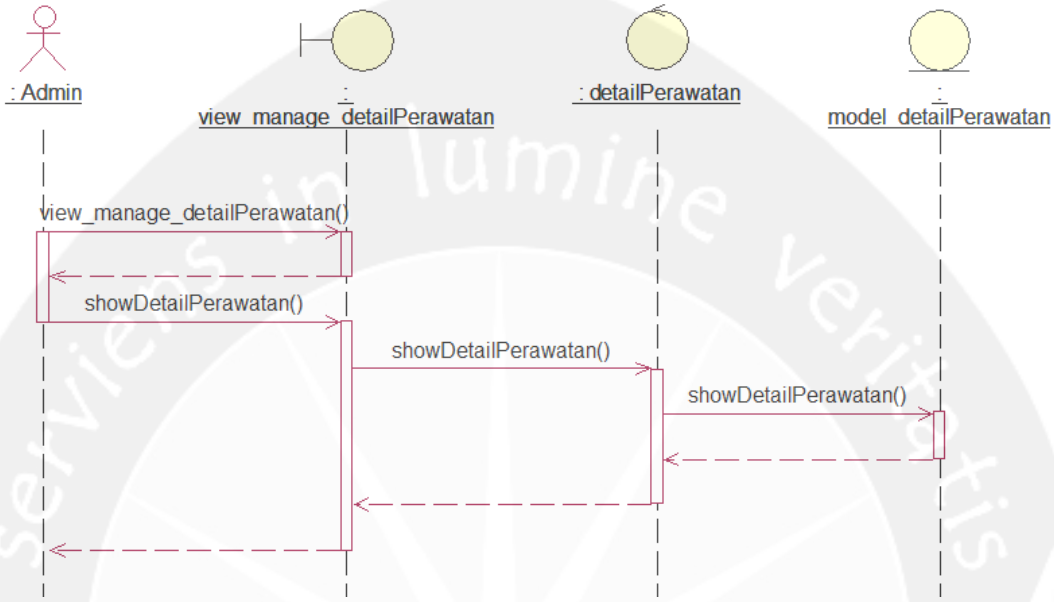
Gambar 2.24 Sequence Diagram : Add Detail Perawatan

4.2.2.3.6 Edit Detail Perawatan



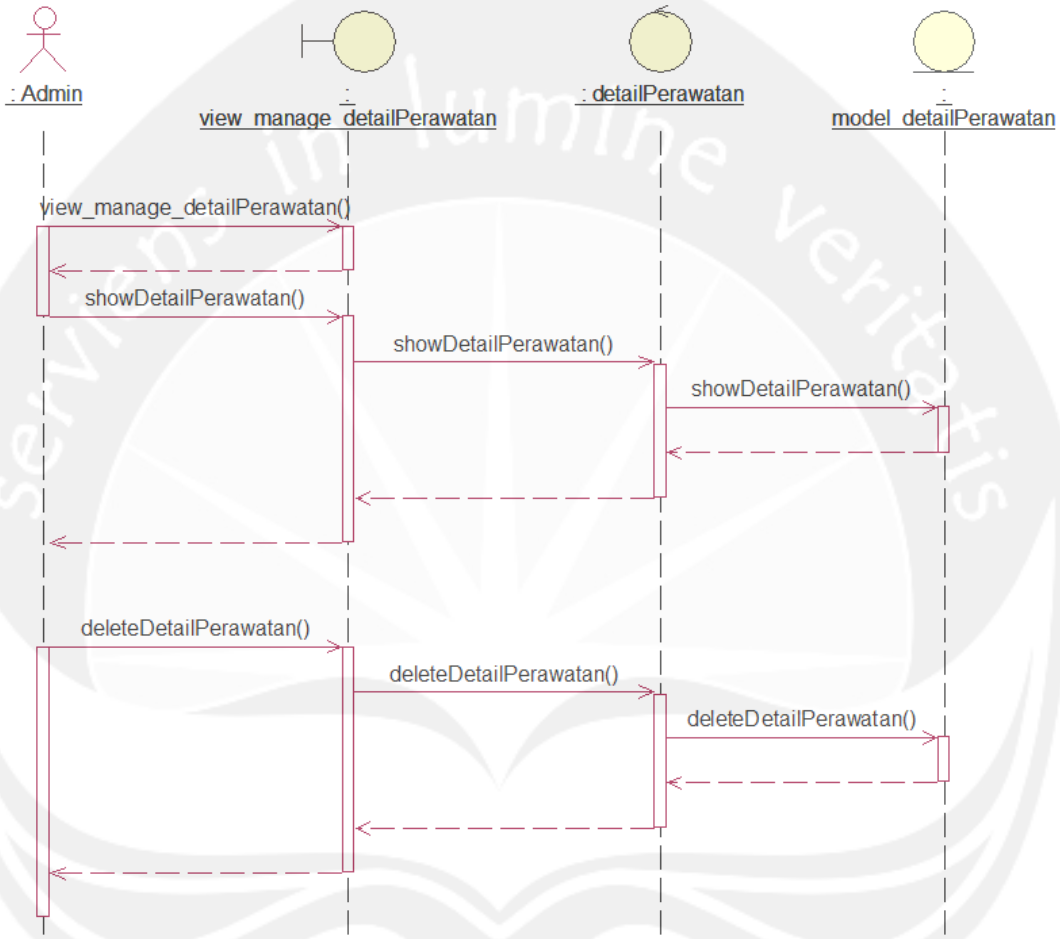
Gambar 2.25 Sequence Diagram : Edit Detail Perawatan

4.2.2.3.7 Show Detail Perawatan



Gambar 2.26 Sequence Diagram : Show Detail Perawatan

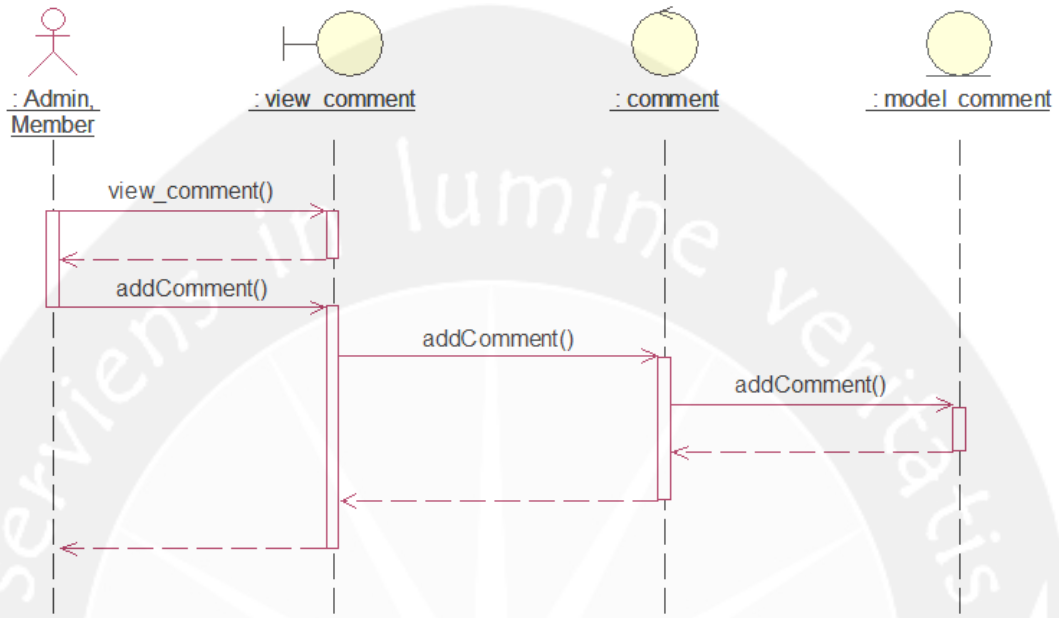
4.2.2.3.8 Delete Detail Perawatan



Gambar 2.27 Sequence Diagram : Delete Detail Perawatan

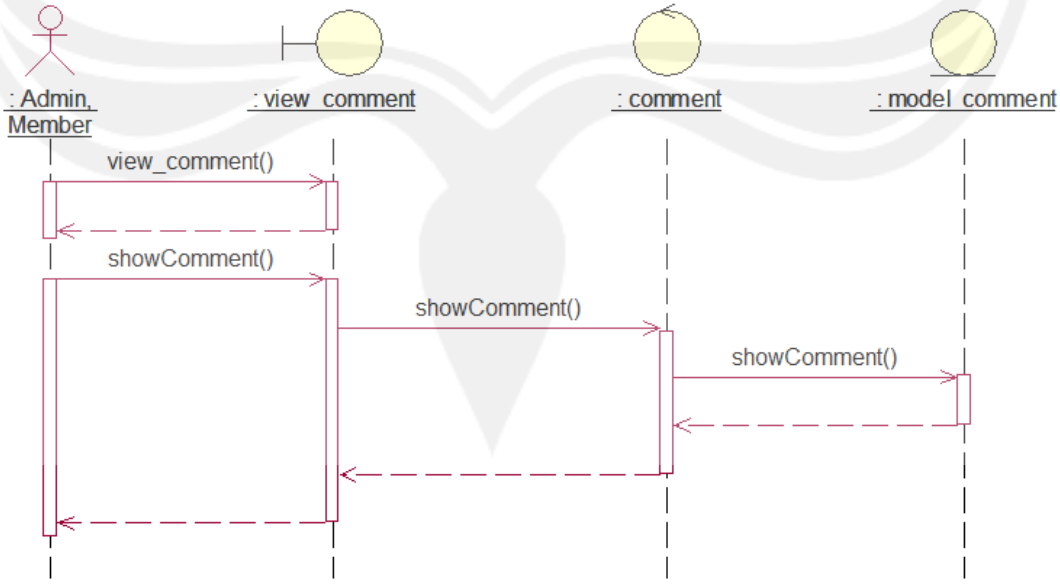
2.2.1.13 Komentar

2.2.2.9.9.1 Show Komentar



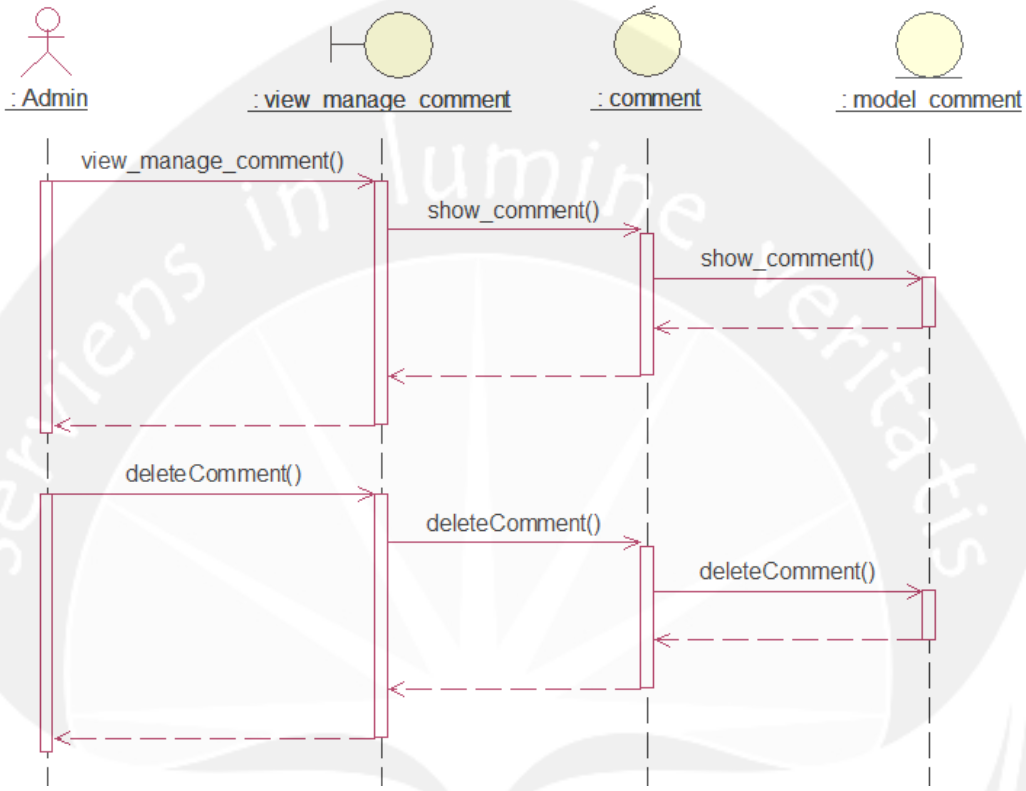
Gambar 2.28 Sequence Diagram : Show Komentar

2.2.2.9.9.2 Isi Komentar



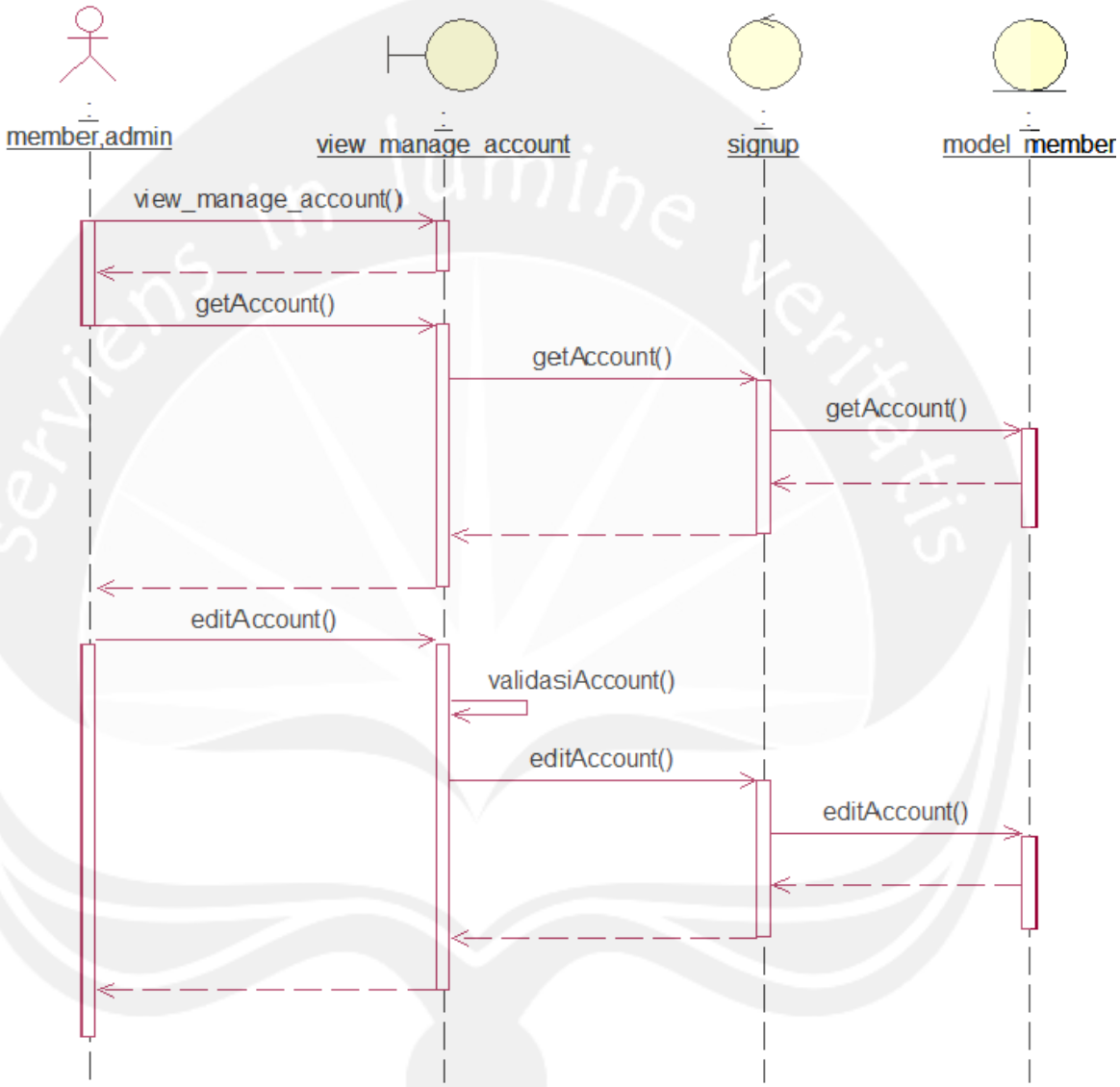
Gambar 2.29 Sequence Diagram : Isi Komentar

2.2.1.14 Pengelolaan Komentar (Hapus komentar oleh admin)



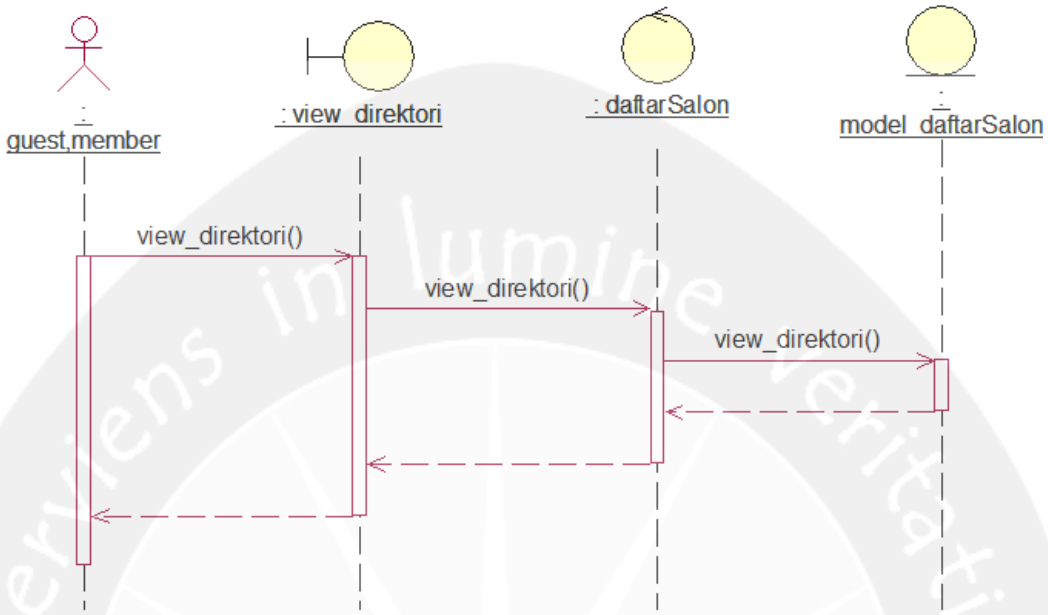
Gambar 2.30 Sequence Diagram : Isi Komentar

2.2.1.15 *Manage Account*



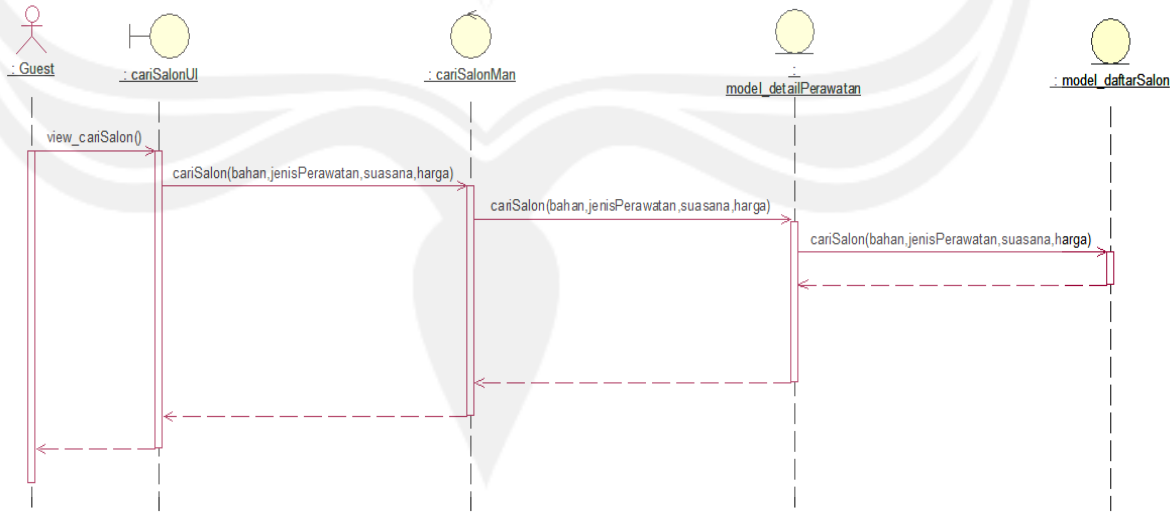
Gambar 2.31 Sequence Diagram : *Manage Account*

2.2.1.16 Show Direktori



Gambar 2.32 Sequence Diagram : Show Direktori

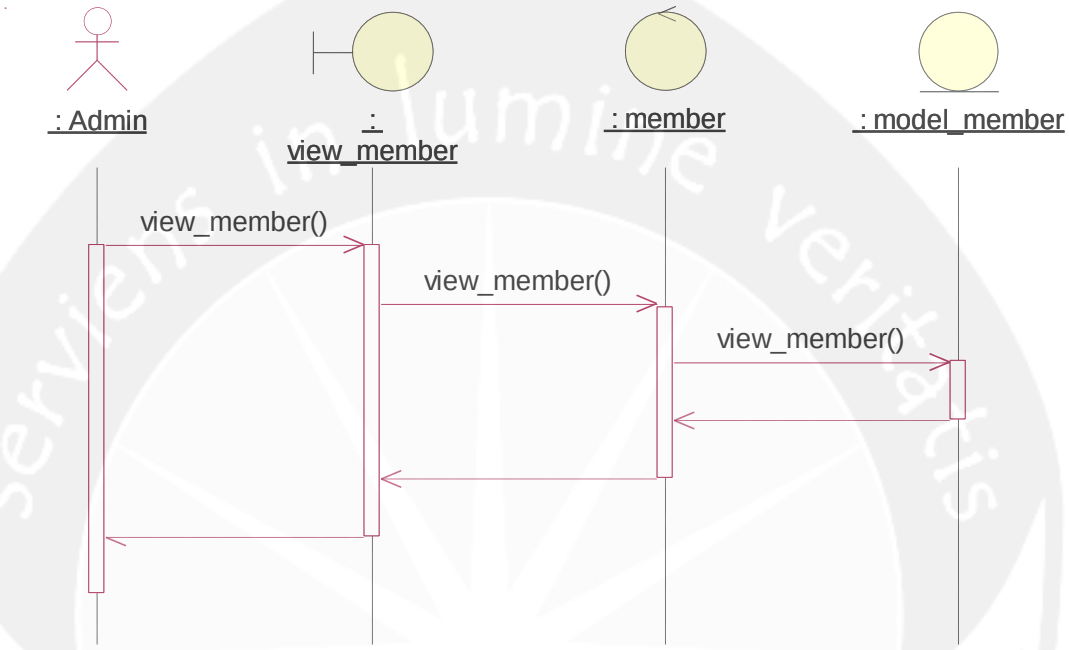
2.2.1.17 Cari Salon



Gambar 2.33 Sequence Diagram : Cari Salon

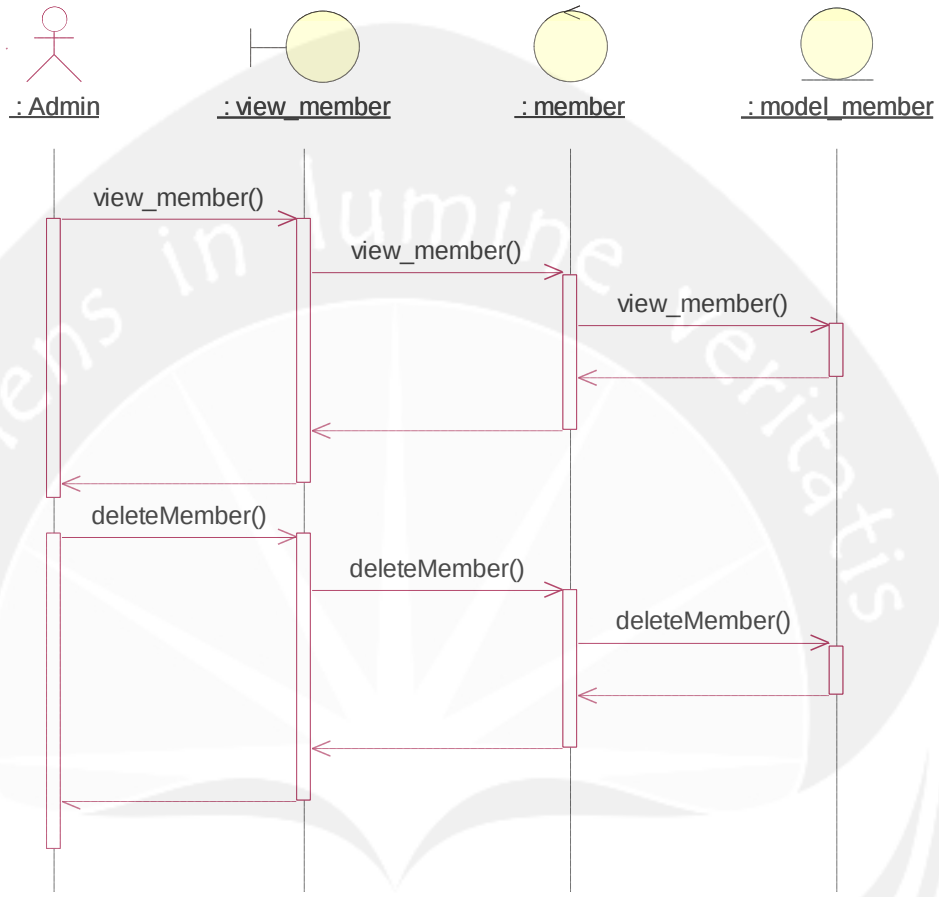
2.2.1.18 Manage Member

2.2.2.9.14.1 *Show Member*



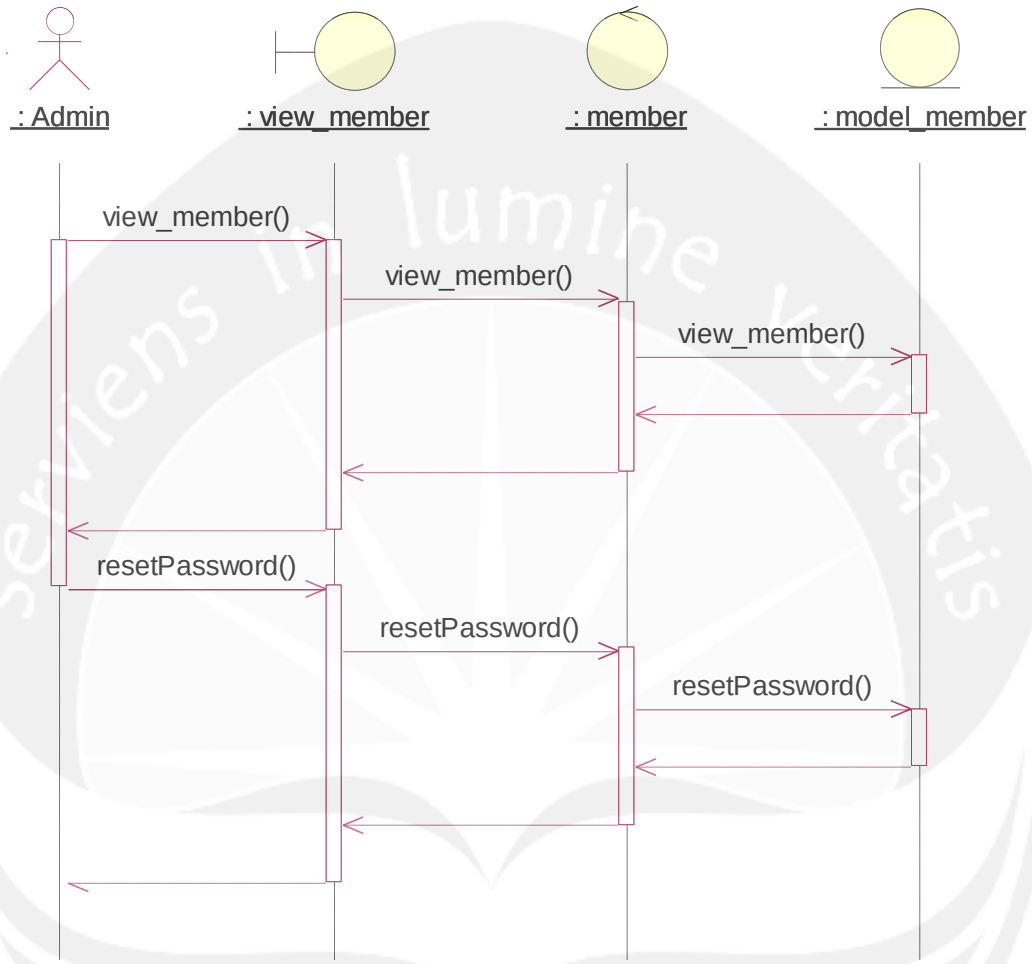
Gambar 2.34 Sequence Diagram : *Show Member*

2.2.2.9.14.2 *Delete Member*



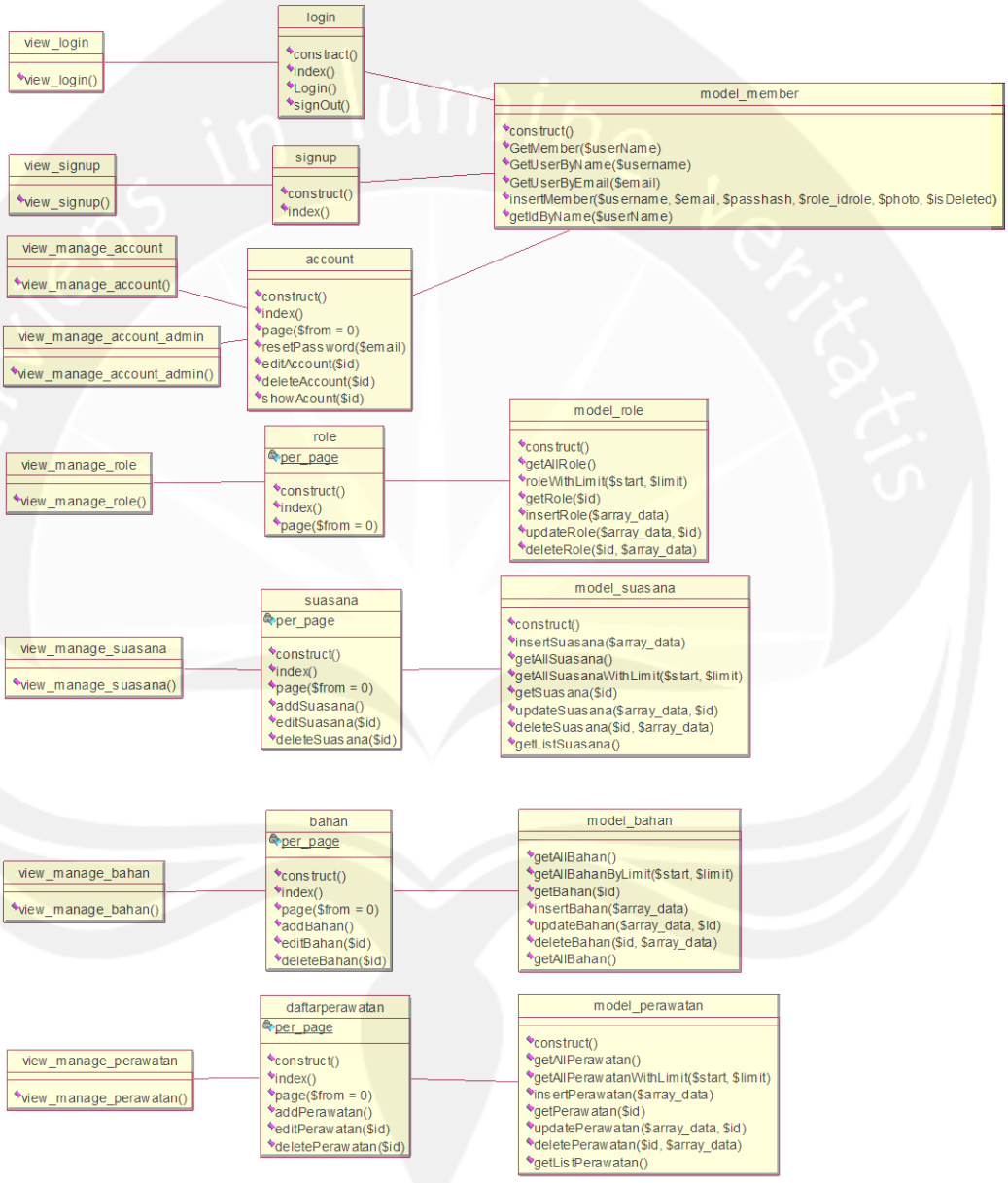
Gambar 2.35 Sequence Diagram : *Delete Member*

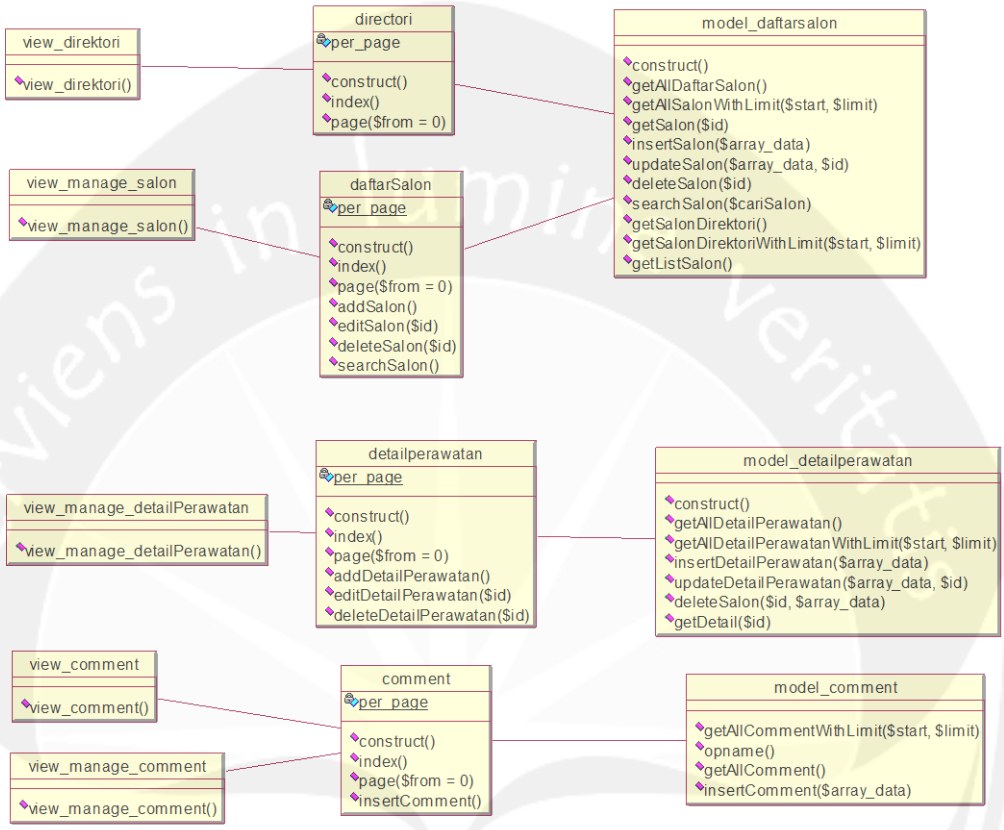
2.2.2.9.14.3 *Reset Password*



Gambar 2.36 Sequence Diagram : *Reset Password*

2.2.2 Diagram Kelas





Gambar 2.37 Class Diagram

2.2.3 Deskripsi Kelas

4.2.3.1 Spesifikasi Design Class view_login

view_login	**<<Boundary>>**
+ view_login ()	
Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_login merupakan sebuah *class* untuk menampilkan antarmuka *login*.	

Antarmuka ini berfungsi supaya *user* dapat masuk ke dalam sistem.

4.2.3.2 Spesifikasi Design Class view_signup

view_signup	**<<Boundary>>**
+ view_signup () Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_signup merupakan sebuah *class* untuk menampilkan antarmuka mendaftar ke dalam sistem. Antarmuka ini berfungsi supaya *user* dapat mendaftar ke dalam sistem.	

2.2.3.3 Spesifikasi Design Class view_manage_role

view_manage_role	**<<Boundary>>**
+ view_manage_role () Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_role merupakan sebuah *class* untuk menampilkan antarmuka mengelola *role*.	

4.2.3.3 Spesifikasi Design Class view_manage_suasana

view_manage_suasana	**<<Boundary>>**
+ view_manage_suasana() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_suasana merupakan sebuah *class* untuk menampilkan antarmuka	

mengelola suasana.

4.2.3.4 Spesifikasi Design Class view_manage_bahan

view_manage_bahan	**<<Boundary>>**
+ view_manage_bahan() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_bahan merupakan sebuah *class* untuk menampilkan antarmuka mengelola bahan.	

4.2.3.5 Spesifikasi Design Class view_manage_perawatan

view_manage_perawatan	**<<Boundary>>**
+ view_manage_perawatan() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_perawatan merupakan sebuah *class* untuk menampilkan antarmuka mengelola perawatan.	

4.2.3.6 Spesifikasi Design Class view_manage_salon

view_manage_salon	**<<Boundary>>**
+ view_manage_salon() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_salon merupakan sebuah *class* untuk menampilkan antarmuka mengelola salon.	

4.2.3.7 Spesifikasi Design Class view_direktori

view_direktori	**<<Boundary>>**
+ view_direktori() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_direktori merupakan sebuah *class* untuk menampilkan antarmuka mengelola direktori.	

4.2.3.8 Spesifikasi Design Class view_manage_detailPerawatan

view_manage_detailPerawatan	<<Boundary>>
+ view_manage_detailPerawatan() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_detailPerawatan merupakan sebuah *class* untuk menampilkan antarmuka mengelola detail perawatan.	

4.2.3.9 Spesifikasi Design Class view_comment

view_comment	<<Boundary>>
+ view_comment() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_comment merupakan sebuah *class* untuk menampilkan antarmuka comment.	

4.2.3.10 Spesifikasi Design Class view_manage_comment

view_manage_comment	<<Boundary>>
+ view_manage_comment()	

Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_comment merupakan sebuah *class* untuk menampilkan antarmuka pengelolaan comment. Pengelolaan *comment* dilakukan oleh admin untuk menghapus *comment*.

4.2.3.11 Spesifikasi Design Class view_manage_account

view_manage_account	<<Boundary>>
+ view_manage_account() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_account merupakan sebuah *class* untuk menampilkan antarmuka pengelolaan account. Pengelolaan account dilakukan oleh admin atau member untuk mengedit *account* mereka.	

4.2.3.12 Spesifikasi Design Class view_manage_account_admin

view_manage_account_admin	<<Boundary>>
+ view_manage_account_admin() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. view_manage_account_admin merupakan sebuah *class* untuk menampilkan antarmuka pengelolaan account oleh admin. Pengelolaan account dilakukan oleh admin untuk mereset *password* dan melihat semua daftar *member.*	

4.2.3.13 Spesifikasi Design Class login

login	<<Control>>

+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. +Login() Operasi ini digunakan untuk melakukan proses login. Operasi ini berisikan pengecekan data *user* yang diinputkan dengan data yang ada di *database*. +signOut() Operasi ini digunakan untuk melakukan proses keluar dari sistem.

4.2.3.14 Spesifikasi Design Class signup

signup	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Pada operasi ini segala proses untuk input data untuk member baru dilakukan.	

4.2.3.15 Spesifikasi Design Class account

account	<<Control>>
+construct()	

Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Pada operasi ini segala proses untuk input data untuk member baru dilakukan.+ addMember() Operasi ini digunakan untuk menambahkan satu data *account* baru sebagai *member* di *database*. + editAccount(int idMember) Operasi ini digunakan untuk mengubah isi data *account member* dengan data *account* yang baru dengan mencocokan id dari data membernya. + resetPassword(int idMember) Operasi ini digunakan untuk mereset *password* member dengan *password* baru.

4.2.3.16 Spesifikasi Design Class role

role	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk pengelolaan *role*. + addRole()	

Operasi ini digunakan untuk menambahkan role baru ke dalam database. + editRole() Operasi ini digunakan untuk mengubah role yang sudah ada di database dengan data role yang baru. + deleteRole() Operasi ini untuk menghapus sebuah role yang ada di dalam database. + page($from=0) Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan dalam satu page.

4.2.3.17 Spesifikasi Design Class suasana

suasana	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk pengelolaan suasana. + addSuasana() Operasi ini digunakan untuk menambahkan suasana baru ke dalam database. + editSuasana() Operasi ini digunakan untuk mengubah suasana yang sudah ada di database dengan data suasana yang baru.	

+ deleteSuasana() Operasi ini untuk menghapus sebuah suasana yang ada di dalam database. + page($from=0) Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan dalam satu page.

4.2.3.18 Spesifikasi Design Class bahan

bahan	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk pengelolaan bahan. + addBahan() Operasi ini digunakan untuk menambahkan jenis bahan baru ke dalam database. + editBahan() Operasi ini digunakan untuk mengubah jenis bahan yang sudah ada di database dengan data jenis bahan yang baru. + deleteBahan() Operasi ini untuk menghapus sebuah jenis bahan yang ada di dalam database. + page($from=0)	

Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan dalam satu page.

4.2.3.19 Spesifikasi Design Class daftarperawatan

daftarperawatan	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk pengelolaan daftar perawatan. + addPerawatan() Operasi ini digunakan untuk menambahkan jenis perawatan baru ke dalam database. + editPerawatan() Operasi ini digunakan untuk mengubah jenis perawatan yang sudah ada di database dengan data jenis perawatan yang baru. + deletePerawatan() Operasi ini untuk menghapus sebuah jenis perawatan yang ada di dalam database. + page($from=0) Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan dalam satu page.	

4.2.3.20 Spesifikasi Design Class daftarSalon

daftarSalon	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk pengelolaan daftar salon. + addSalon () Operasi ini digunakan untuk menambahkan data salon baru ke dalam database. + editSalon() Operasi ini digunakan untuk mengubah data salon yang sudah ada di database dengan data data salon yang baru. + deleteSalon() Operasi ini untuk menghapus sebuah data salon yang ada di dalam database. + searchSalon() Operasi ini untuk mencari sebuah data salon yang ada di dalam database. + page($from=0) Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan dalam satu page.	

4.2.3.21 Spesifikasi Design Class directori

directori	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk direktori salon. + page($from=0) Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan dalam satu page.	

4.2.3.22 Spesifikasi Design Class detailperawatan

detailperawatan	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk pengelolaan detail perawatan. + page($from=0) Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan	

dalam satu page. + addDetailPerawatan() Operasi ini digunakan untuk menambahkan detail perawatan yang baru ke dalam database. + editDetailPerawatan ($id) Operasi ini digunakan untuk mengubah detail perawatan yang sudah ada di database dengan data data detail perawatan yang baru. + deleteSalon($id) Operasi ini untuk menghapus sebuah data detail perawatan yang ada di dalam database.

4.2.3.23 Spesifikasi Design Class comment

comment	<<Control>>
+construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. +index() Operasi ini digunakan sebagai operasi yang akan dipanggil pertama kali saat kelas ini dipanggil. Operasi ini akan menampilkan halaman untuk comment. + page($from=0) Operasi ini digunakan memberikan initial kepada pagination berapa banyak data yang akan ditampilkan dalam satu page. + insertComment() Operasi ini digunakan untuk menambahkan comment yang baru ke dalam database.	

4.2.3.24 Spesifikasi Design Class model_member

model_member	<<Entity>>
+ construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. + GetMember($userName) Operasi ini merupakan operasi untuk mendapatkan data *member* sesuai dengan *username user* dan mendapatkan *role* dari *user.* + GetUserByName($userName) Operasi ini merupakan operasi untuk mendapatkan data *member* sesuai dengan *username user* dan mendapatkan *role* dari *user.* + GetUserByEmail($email) Operasi ini merupakan operasi untuk mendapatkan data *member* sesuai dengan *email user.* Operasi ini digunakan untuk mengecek email yang diinputkan *user* dengan email yang ada di *database.* + insertMember($userName, $email, $passhash, $role_idrole) Operasi ini digunakan untuk menginputkan data *user* baru ke dalam *database.* + getIdByName($userName) Operasi ini digunakan untuk mendapatkan id user sesuai dengan namanya.	

4.2.3.25 Spesifikasi Design Class model_role

model_role	<<Entity>>

+ construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. + GetAllRole() Operasi ini merupakan operasi untuk mendapatkan semua data role. + roleWithLimit($start,$limit) Operasi ini merupakan operasi untuk mendapatkan *role* dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi awal data dan banyaknya data yang akan diambil. + getRole($id) Operasi ini merupakan operasi untuk mendapatkan data *role* sesuai dengan *id* role-nya. + insertRole($array_data) Operasi ini digunakan untuk menginputkan data *role* baru ke dalam *database*. Data yang diinputkan berupa array. + updateRole($array_data, $id) Operasi ini digunakan untuk mengedit data role dengan data role yang baru. + deleteRole($id) Operasi ini digunakan untuk menghapus data role.

4.2.3.26 Spesifikasi Design Class model_suasana

model_suasana	<<Entity>>
+ construct() Default konstruktor, digunakan untuk inisialisasi	

semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. + GetAllSuasana() Operasi ini merupakan operasi untuk mendapatkan semua data suasana. + getAllSuasanaWithLimit($start,$limit) Operasi ini merupakan operasi untuk mendapatkan data suasana dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi awal data dan banyaknya data yang akan diambil. + getSuasana($id) Operasi ini merupakan operasi untuk mendapatkan data suasana sesuai dengan *id* suasana-nya. + insertSuasana($array_data) Operasi ini digunakan untuk menginputkan data suasana baru ke dalam *database.* Data yang diinputkan berupa array. + updateSuasana($array_data, $id) Operasi ini digunakan untuk mengedit data suasana dengan data suasana yang baru. + deleteSuasana($id) Operasi ini digunakan untuk menghapus data suasana. + getListSuasana() Operasi ini digunakan untuk menampilkan data yang ada di table suasana kemudian akan ditampilkan dalam *combobox.*

4.2.3.27 Spesifikasi Design Class model_bahan

model_bahan	<<Entity>>

+ construct()

Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini.

+ GetAllBahan()

Operasi ini merupakan operasi untuk mendapatkan semua data bahan.

+ getAllBahanWithLimit($start,$limit)

Operasi ini merupakan operasi untuk mendapatkan data bahan dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi awal data dan banyaknya data yang akan diambil.

+ getBahan($id)

Operasi ini merupakan operasi untuk mendapatkan data bahan sesuai dengan *id* bahan-nya.

+ insertBahan($array_data)

Operasi ini digunakan untuk menginputkan data bahan baru ke dalam *database.* Data yang diinputkan berupa array.

+ updateBahan($array_data, $id)

Operasi ini digunakan untuk mengedit data bahan dengan data bahan yang baru.

+ deleteBahan($id)

Operasi ini digunakan untuk menghapus data bahan.

+ getAllBahan()

Operasi ini digunakan untuk menampilkan data yang ada di table bahan kemudian akan ditampilkan dalam *combobox.*

4.2.3.28 Spesifikasi Design Class model_perawatan

model_perawatan	<<Entity>>
+ construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. + GetAllPerawatan() Operasi ini merupakan operasi untuk mendapatkan semua data perawatan. + getAllPerawatanWithLimit($start,$limit) Operasi ini merupakan operasi untuk mendapatkan data perawatan dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi awal data dan banyaknya data yang akan diambil. + getPerawatan($id) Operasi ini merupakan operasi untuk mendapatkan data perawatan sesuai dengan *id* perawatan-nya. + insertPerawatan($array_data) Operasi ini digunakan untuk menginputkan data perawatan baru ke dalam *database*. Data yang diinputkan berupa array. + updatePerawatan($array_data, $id) Operasi ini digunakan untuk mengedit data perawatan dengan data perawatan yang baru. + deletePerawatan($id) Operasi ini digunakan untuk menghapus data perawatan. + getAllPerawatan() Operasi ini digunakan untuk menampilkan data yang ada di table perawatan kemudian akan ditampilkan dalam *combobox*.	

4.2.3.29 Spesifikasi Design Class model_daftarsalon

model_daftarsalon	<<Entity>>
+ construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. + GetAllDaftarSalon() Operasi ini merupakan operasi untuk mendapatkan semua daftar salon. + getAllSalonWithLimit($start,$limit) Operasi ini merupakan operasi untuk mendapatkan data salon dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi awal data dan banyaknya data yang akan diambil. + getSalon($id) Operasi ini merupakan operasi untuk mendapatkan data salon sesuai dengan *id* salon-nya. + insertSalon($array_data) Operasi ini digunakan untuk menginputkan data salon baru ke dalam *database.* Data yang diinputkan berupa array. + updateSalon($array_data, $id) Operasi ini digunakan untuk mengedit data salon dengan data salon yang baru. + deleteSalon($id) Operasi ini digunakan untuk menghapus data salon. + getListSalon() Operasi ini digunakan untuk menampilkan data salon	

yang ada di tabel salon kemudian akan ditampilkan dalam *combobox.* + searchSalon($cariSalon) Operasi ini digunakan untuk menampilkan data salon sesuai dengan pencarian *user.* + getSalonDirektori() Operasi ini digunakan untuk menampilkan data salon sebagai sebuah direktori. + getSalonDirektoriWithLimit($start,$limit) Operasi ini merupakan operasi untuk mendapatkan data salon untuk direktori dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi awal data dan banyaknya data yang akan diambil.

4.2.3.30 Spesifikasi Design Class model_detailperawatan

model_detailperawatan	<<Entity>>
+ construct() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini dan *library* yang diperlukan dalam kelas ini. + getAlldetailPerawatan() Operasi ini merupakan operasi untuk mendapatkan semua detail perawatan. + getAlldetailPerawatanWithLimit($start,$limit) Operasi ini merupakan operasi untuk mendapatkan data detail perawatan dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi	

awal data dan banyaknya data yang akan diambil. + insertDetailPerawatan() Operasi ini digunakan untuk menambahkan detail perawatan baru ke dalam *database*. + updateDetailPerawatan() Operasi ini digunakan untuk mengubah detail perawatan yang sudah ada di *database* dengan data detail perawatan yang baru. + deleteDetailPerawatan Operasi ini untuk menghapus sebuah detail perawatan yang ada di dalam *database*. + getDetail($id) Operasi ini digunakan untuk mendapatkan detail peratawan sesuai dengan id-nya.

4.2.3.31 Spesifikasi Design Class model_comment

model_comment	<<Entity>>
+ getAllCommentWithLimit($start,$limit) Operasi ini merupakan operasi untuk mendapatkan daftar comment dengan batasan tertentu. Operasi ini digunakan dalam pagination yaitu berguna untuk mengambil data dalam page tertentu tergantung posisi awal data dan banyaknya data yang akan diambil. + getAllComment() Operasi ini merupakan operasi untuk mendapatkan semua comment. + insertComment() Operasi ini digunakan untuk menambahkan *comment* baru ke dalam sistem.	

3 Perancangan Data

3.1 Dekomposisi Data

3.1.1 Deskripsi Entitas Data Role

Nama	Tipe	Panjang	Keterangan
IDRole	**Integer**	**-**	**Id dari Role, primary key**
deskRole	Variable character	20	Deskipsi dari id role

3.1.2 Deskripsi Entitas Data Member

Nama	Tipe	Panjang	Keterangan
idMember	**Integer**	**-**	**Id dari Member, *primary key***
idRole	Integer	-	Id dari role, *foreign key*
username	Variable character	50	Nama untuk masuk sistem
password	Variable character	50	Password (kunci) untuk masuk sistem
email	Variable character	50	Alamat *email member*
information	Text	-	Informasi yang dimiliki *member*
photo	Variable character	300	Foto dari *member*

3.1.3 Deskripsi Entitas Data Comment

Nama	Tipe	Panjang	Keterangan

idComment	**Integer**	**-**	**Id dari Comment, Primary Key**
idMember	Integer	-	Id dari member, *foreign key*
isiComment	Text	-	Komentar dari *member*
tglComment	Datetime	-	Waktu saat komentar disimpan

3.1.4 Deskripsi Entitas Data Status

Nama	Tipe	Panjang	Keterangan
idStatus	**Integer**	**-**	**Id dari status, primary key**
deskStatus	Variable Character	20	Deskripsi dari status

3.1.5 Deskripsi Entitas Data Suasana

Nama	Tipe	Panjang	Keterangan
idSuasana	**Integer**	**-**	**Id dari suasana, *primary key***
jenisSuasana	Variable Character	30	Macam-macam jenis dari suasana

3.1.6 Deskripsi Entitas Data BahanSalon

Nama	Tipe	Panjang	Keterangan
idBahan	**Integer**	**-**	**Id dari bahan-bahan salon, *primary key***
jenisBahan	Variable Character	20	Macam-macam jenis dari bahan salon

3.1.7 Deskripsi Entitas Data Perawatan

Nama	Tipe	Panjang	Keterangan
idPerawatan	**Integer**	**-**	**Id dari perawatan di salon, *primary key***
namaPerawatan	Variable Character	30	Macam-macam jenis dari perawatan salon

3.1.8 Deskripsi Entitas Data Detail Perawatan

Nama	Tipe	Panjang	Keterangan
idDetailPerawatan	**Integer**	**-**	**Id dari detail perawatan di salon, *primary key***
idStatus	Integer	-	Id dari status, *foreign key*
idPerawatan	Integer	-	Id dari perawatan di salon, *foreign key*
deskripsi	Text	-	Keterangan khusus tentang suatu perawatan
Harga	float	-	Harga dari satu macam perawatan
Gambar	Variable Character	300	Gambar dari satu macam perawatan

3.1.9 Deskripsi Entitas Data DaftarSalon

Nama	Tipe	Panjang	Keterangan
idDaftarSalon	**Integer**	**-**	**Id dari nama-nama salon, *primary key***

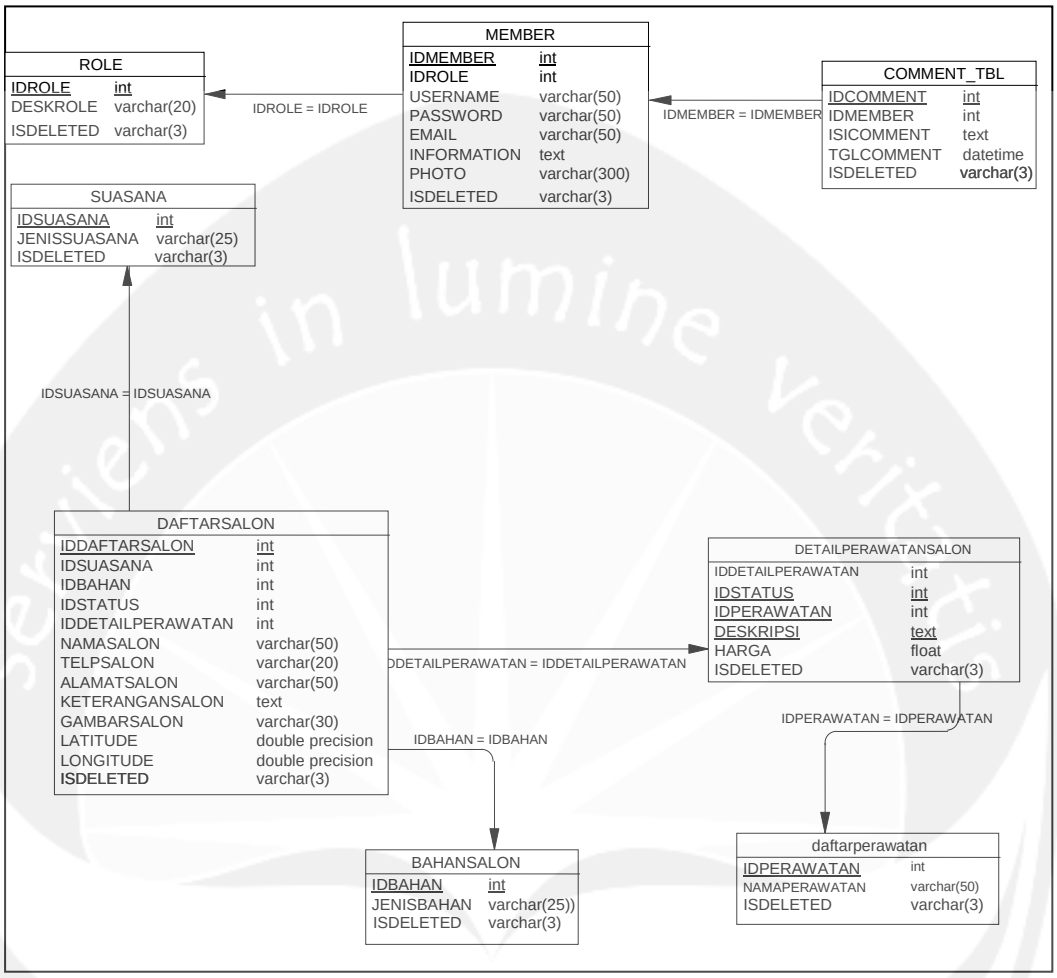
idSuasana	Integer	-	Id dari suasana, *foreign key*
idBahan	Integer	-	Id dari bahan-bahan salon, *foreign key*
idStatus	Integer	-	Id dari status, *foreign key*
idDetailPerawatan	Integer	-	Id dari detail perawatan di salon, *foreign key*
namaSalon	Variable Character	50	Nama dari suatu salon
telpSalon	Variable Character	20	Telepon dari suatu salon
alamatSalon	Variable Character	70	Alamat dari suatu salon
keteranganSalon	Text	-	Keterangan khusus dari suatu salon
gambarSalon	Variable Character	300	Gambar dari sebuah salon
Latitude	Double	-	Koordinat lintang lokasi salon
Langitude	Double	-	Koordinat bujur lokasi salon

3.1.10 Deskripsi Entitas Data Paket

Nama	Tipe	Panjang	Keterangan
idPaket	**Integer**	**-**	**Id dari paket milik suatu salon, *primary***

			key
idDaftarSalon	Integer	-	Id dari nama-nama salon, *foreign key*
namaPaket	Variable Character	50	Nama paket yang ditawarkan salon
isiPaket	Text	-	Isi dari paket yang ada
gambarPaket	Variable Character	300	Gambar dari paket yang dimiliki salon
ketPaket	Text	-	Keterangan dari paket yang ada

3.2 Physical Data Model



Gambar 3.1 *Physical Data Model*

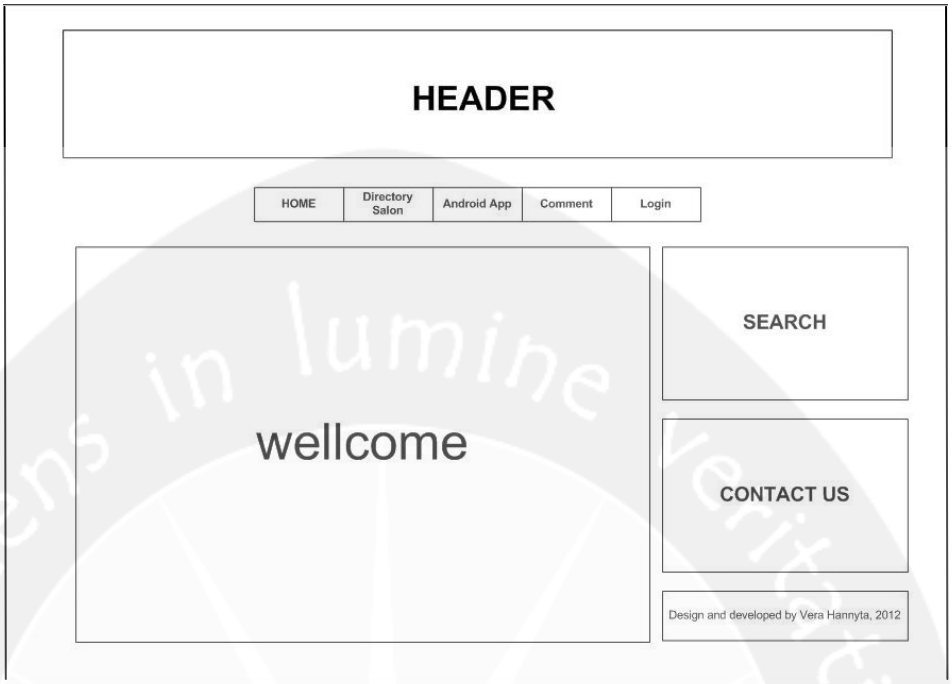
5 Perancangan Antarmuka

5.1 Sketsa UI dan deskripsinya

4.2.4.1. Antarmuka Halaman Utama

Gambar 4.1 Halaman utama pada ponsel

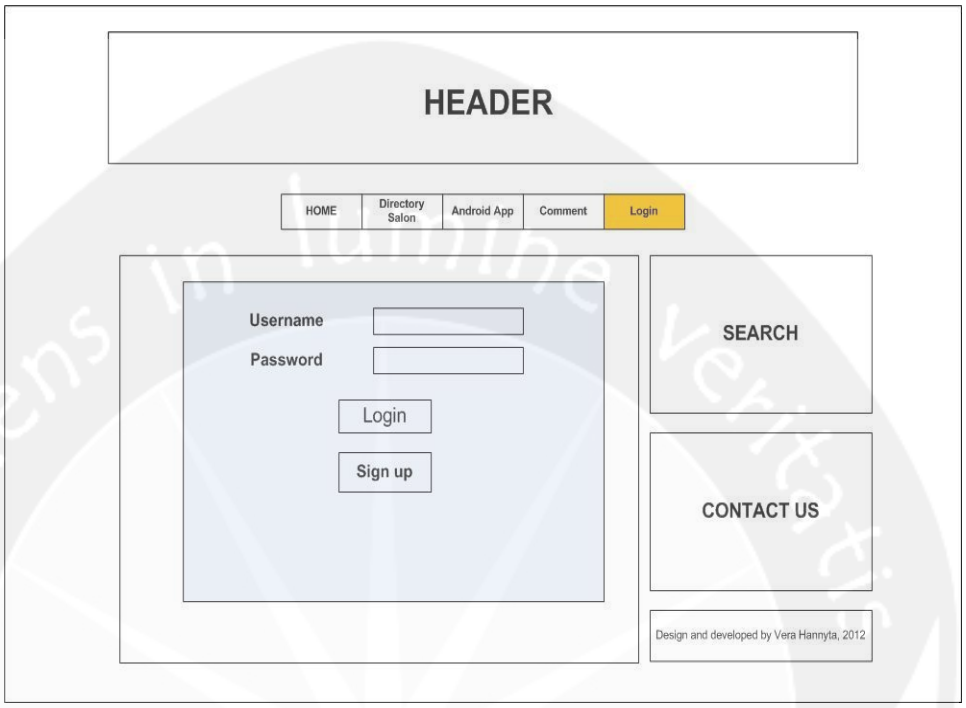
Antarmuka pada gambar 4.1 merupakan antarmuka saat pertama kali penggunan membuka aplikasi *getYourSalon* pada ponsel Android. Antarmuka ini terdiri dari tiga bagian *header*, *body*, dan *footer.* Bagian *body* terdiri dari dua menu, yaitu menu untuk masuk ke sistem pendukung keputusan (*getYourSalon*) dan menu directory salon untuk masuk ke daftar direktori salon yang berisikan data-data salon di Yogyakarta.



Gambar 4.2 Halaman utama pada *website*

Antarmuka ini merupakan antarmuka pada halaman pertamakali *user* memasuki aplikasi *website.* Antarmuka ini juga disebut sebagai halaman *home.* Halaman *home* ini terdiri dari menu-menu yang akan mengarahkan *user* kepada halaman-halaman tertentu. Menu tersebut adalah menu *Directory salon*, *android app*, *comment*, dan *login*.

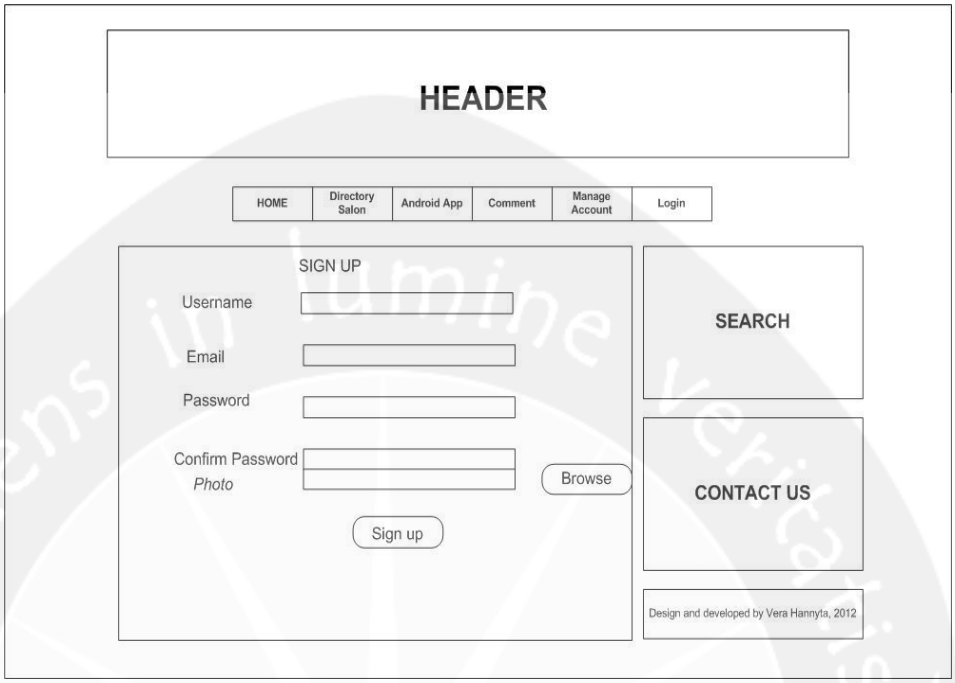
4.2.4.2. Antarmuka Halaman *Login*



Gambar 4.3 Halaman *Login* pada halaman *website*

Antarmuka ini digunakan untuk melakukan proses *login* ke dalam aplikasi. Untuk mendapat akses masuk ke dalam aplikasi, pengguna harus menginputkan data *login* berupa *username* dan *password* dengan benar pada *textbox* yang telah disediakan. Pada saat tombol *login* ditekan, sistem akan mengecek *username* dan *password* yang diinputkan dengan data *username* dan *password* yang ada di dalam *database*. Jika data *username* dan *password* benar atau cocok maka pengguna akan masuk ke dalam aplikasi, kemudian sistem akan memberikan hak akses sesuai peranan yaitu sebagai administrator atau sebagai *member.*

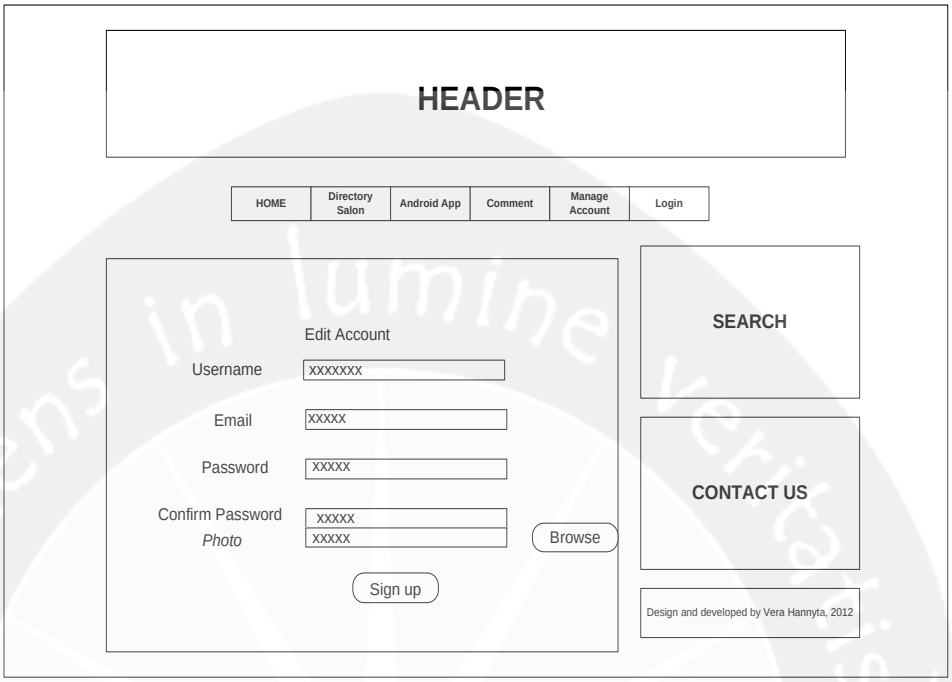
4.2.4.3. Antarmuka *Join Member (Sign Up)*



Gambar 4.4 Halaman *Sign up* pada halaman *website*

Antarmuka ini digunakan untuk melakukan pendaftaran menjadi *member.* Pertama pengguna harus mengisikan data sesuai dengan *textbox* yang sudah disediakan, juga disediakan untuk fasilitas memberi foto. Kemudian ketika ditekan tombol *register* maka data baru dari pengguna akan disimpan ke *database.*

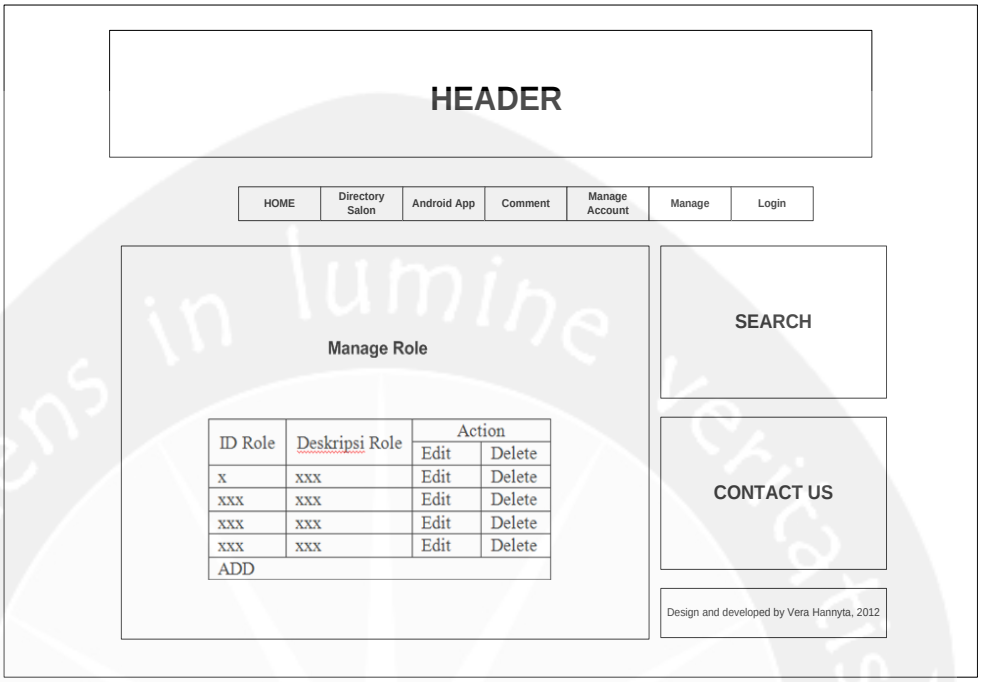
4.2.4.4. Antarmuka *Manage Account*



Gambar 4.5 Halaman *Edit Account* pada halaman *website*

Antarmuka ini merupakan antarmuka untuk *manage* atau men*edit* sebuah *account.* Menu yang disedikan adalah untuk merubah *password* dan foto, sedangkan *username* dan *email* tidak bisa diubah. *Manage account* dapat dilakukan melalui *website*, kemudian data baru akan disimpan ke dalam *database.*

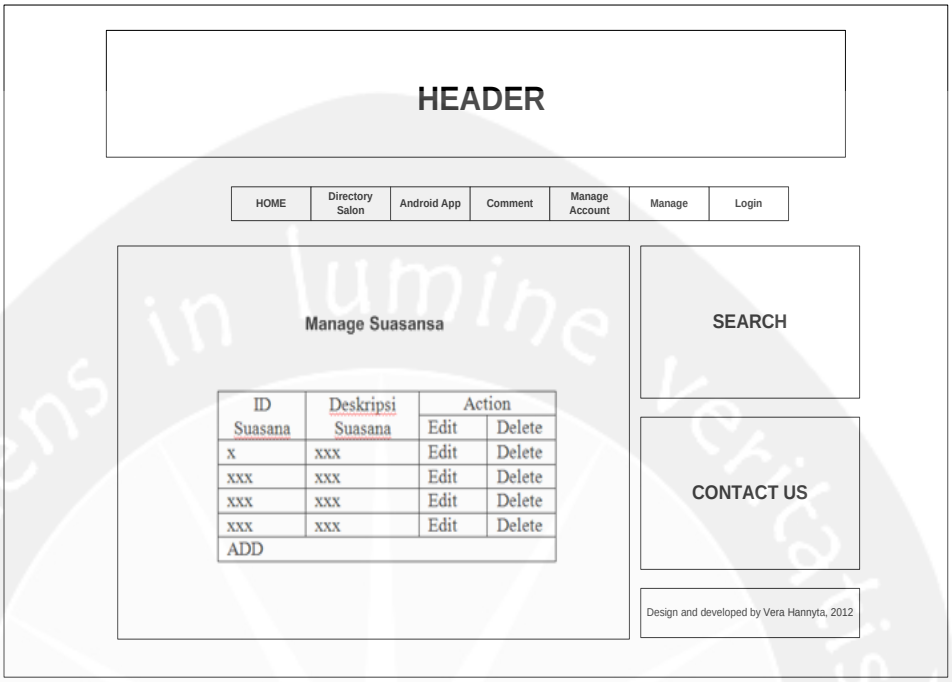
4.2.4.5. Antarmuka *Manage Role*



Gambar 4.6 Halaman *Manage role* pada halaman *website*

Antarmuka ini merupakan antarmuka untuk melakukan pengelolaan *role*, ada tiga bagian penting dalam proses pengelolaan ini, pertama adalah menu add di dalam table, yang berfungsi untuk memasukan data *role* yang baru kedalam *database*. Kedua adalah menu edit yang berada di masing-masing baris setiap variabel, yang berfungsi untuk mengubah data *role*, kemudian akan disimpan ke dalam *database*. Bagian ketiga adalah fungsi *delete* yang juga berada di setiap baris, yang berfugsi untuk menghapus data dari *database.* Perintah add dan edit pada dasarnya menggunakan *form* yang sama, hanya berbeda *fungsionalitas* sesuai dengan *button* yang dijalankan.

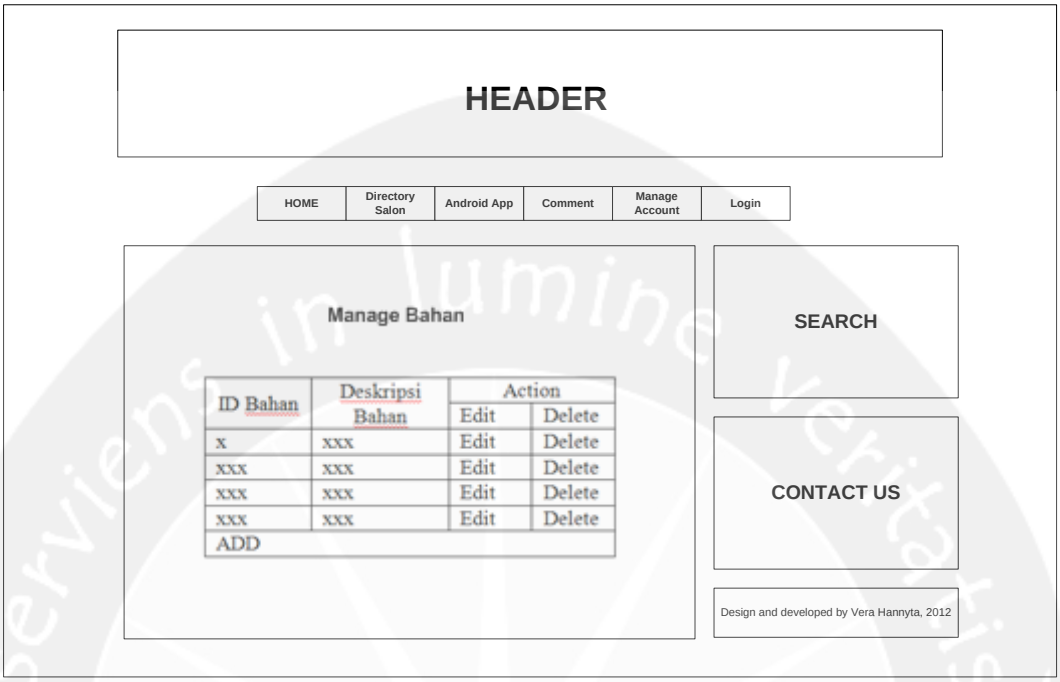
4.2.4.6. Antarmuka *Manage* Suasana



Gambar 4.7 Halaman *Manage* suasana pada halaman *website*

Antarmuka ini merupakan antarmuka untuk melakukan pengelolaan *suasana*, ada tiga bagian penting dalam proses pengelolaan ini, pertama adalah menu add di dalam table, yang berfungsi untuk memasukan data *role* yang baru kedalam *database*. Kedua adalah menu edit yang berada di masing-masing baris setiap variabel, yang berfungsi untuk mengubah data *suasana*, kemudian akan disimpan ke dalam *database*. Bagian ketiga adalah fungsi *delete* yang juga berada di setiap baris, yang berfugsi untuk menghapus data dari *database.* Perintah add dan edit pada dasarnya menggunakan *form* yang sama, hanya berbeda *fungsionalitas* sesuai dengan *button* yang dijalankan.

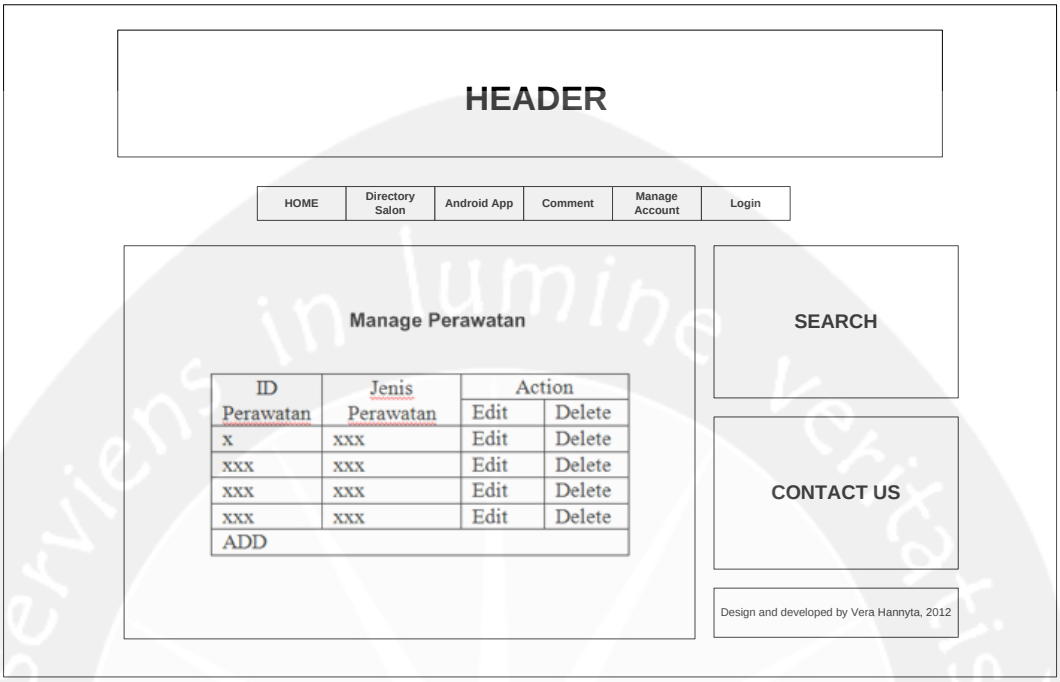
4.2.4.7. Antarmuka *Manage* Bahan



Gambar 4.8 Halaman *Manage* bahan pada halaman *website*

Antarmuka ini merupakan antarmuka untuk melakukan pengelolaan *bahan*, ada tiga bagian penting dalam proses pengelolaan ini, pertama adalah menu add di dalam table, yang berfungsi untuk memasukan data *role* yang baru kedalam *database*. Kedua adalah menu edit yang berada di masing-masing baris setiap variabel, yang berfungsi untuk mengubah data *bahan*, kemudian akan disimpan ke dalam *database*. Bagian ketiga adalah fungsi *delete* yang juga berada di setiap baris, yang berfugsi untuk menghapus data dari *database*. Perintah add dan edit pada dasarnya menggunakan *form* yang sama, hanya berbeda *fungsionalitas* sesuai dengan *button* yang dijalankan.

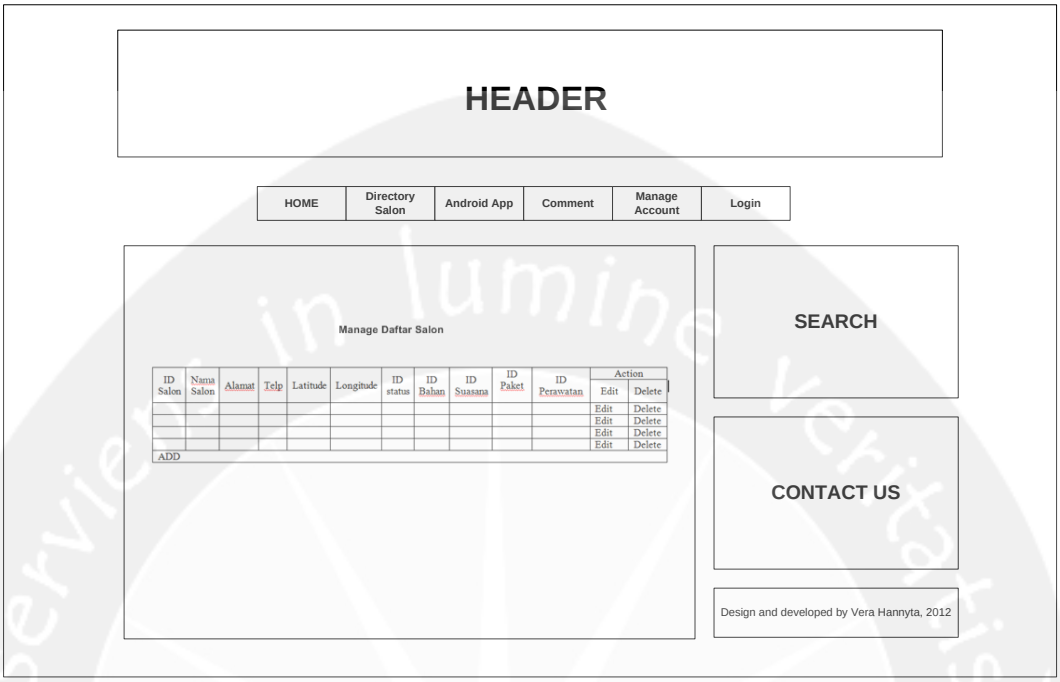
4.2.4.8. Antarmuka *Manage* Daftar Perawatan



Gambar 4.9 Halaman *Manage* daftar perawatan pada halaman *website*

Antarmuka ini merupakan antarmuka untuk melakukan pengelolaan *daftar perawatan*, ada tiga bagian penting dalam proses pengelolaan ini, pertama adalah menu add di dalam table, yang berfungsi untuk memasukan data *role* yang baru kedalam *database*. Kedua adalah menu edit yang berada di masing-masing baris setiap variabel, yang berfungsi untuk mengubah data *daftar perawatan*, kemudian akan disimpan ke dalam *database*. Bagian ketiga adalah fungsi *delete* yang juga berada di setiap baris, yang berfugsi untuk menghapus data dari *database*. Perintah add dan edit pada dasarnya menggunakan *form* yang sama, hanya berbeda *fungsionalitas* sesuai dengan *button* yang dijalankan.

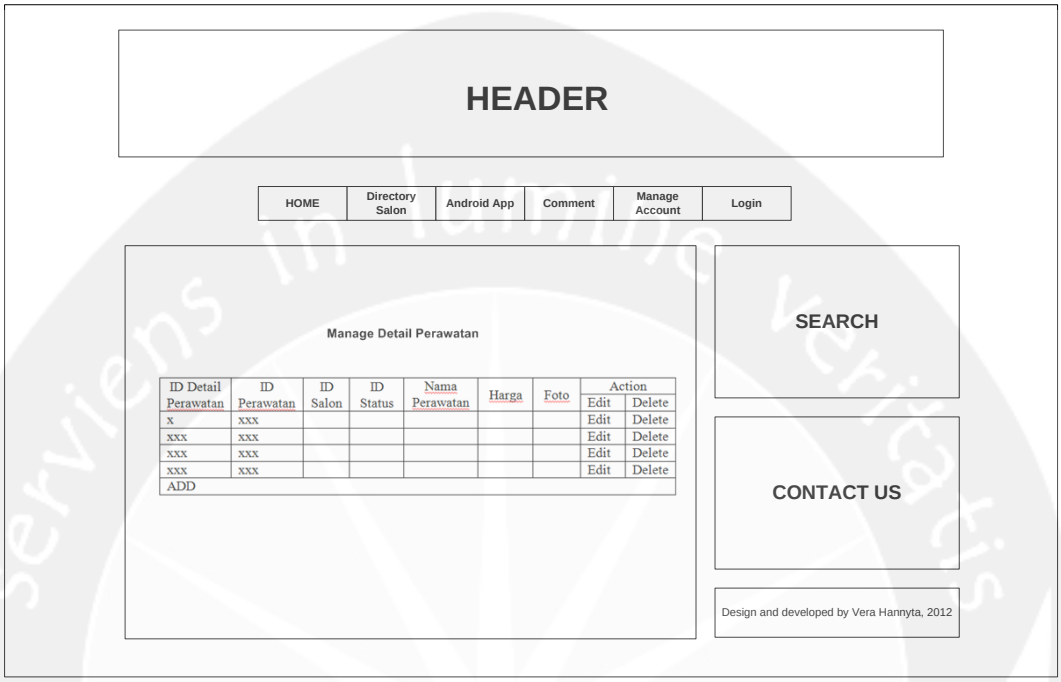
4.2.4.9. Antarmuka *Manage* Daftar Salon



Gambar 4.10 Halaman *Manage* suasana pada halaman *website*

Antarmuka ini merupakan antarmuka untuk melakukan pengelolaan *suasana*, ada tiga bagian penting dalam proses pengelolaan ini, pertama adalah menu add di dalam table, yang berfungsi untuk memasukan data *role* yang baru kedalam *database*. Kedua adalah menu edit yang berada di masing-masing baris setiap variabel, yang berfungsi untuk mengubah data *suasana*, kemudian akan disimpan ke dalam *database*. Bagian ketiga adalah fungsi *delete* yang juga berada di setiap baris, yang berfugsi untuk menghapus data dari *database*. Perintah add dan edit pada dasarnya menggunakan *form* yang sama, hanya berbeda *fungsionalitas* sesuai dengan *button* yang dijalankan.

4.2.4.10. Antarmuka *Manage* Detail Perawatan



Gambar 4.11 Halaman *Manage* detail perawatan pada halaman *website*

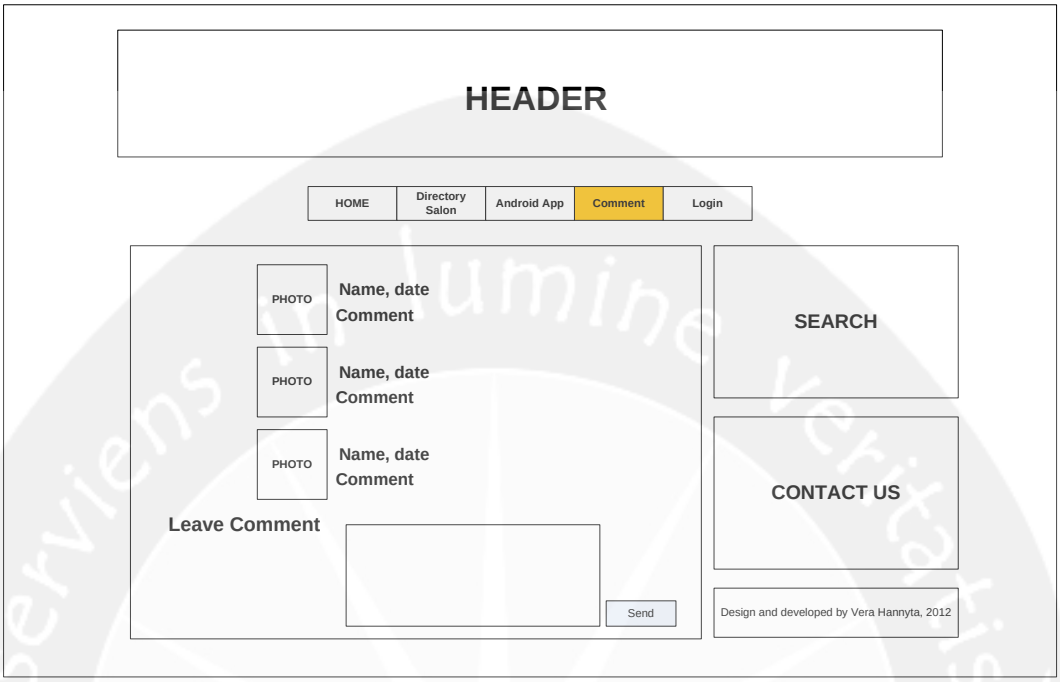
Antarmuka ini merupakan antarmuka untuk melakukan pengelolaan *detail perawatan*, ada tiga bagian penting dalam proses pengelolaan ini, pertama adalah menu add di dalam table, yang berfungsi untuk memasukan data *role* yang baru kedalam *database*. Kedua adalah menu edit yang berada di masing-masing baris setiap variabel, yang berfungsi untuk mengubah data *detail perawatan*, kemudian akan disimpan ke dalam *database*. Bagian ketiga adalah fungsi *delete* yang juga berada di setiap baris, yang berfugsi untuk menghapus data dari *database*. Perintah add dan edit pada dasarnya menggunakan *form* yang sama, hanya berbeda *fungsionalitas* sesuai dengan *button* yang dijalankan.

4.2.4.11. Antarmuka Cari Salon Berdasarakan Kriteria Tertentu

Gambar 4.12 Halaman *Manage* cari salon berdasar kriteria tertentu

Antarmuka ini digunakan untuk mencari salon berdasarkan kriteria tertentu, *user* hanya perlu menginputkan *budget* dan memilih melalui *combobox* yang tersedia. *Button* cari digunakan untuk memproses *input* dengan mencari di *database.*

4.2.4.12. Antarmuka untuk *Comment*



Gambar 4.13 Halaman Antarmuka *comment* pada *website*

Antarmuka ini digunakan untuk menampilkan komentar terdahulu dari *member* serta untuk memberikan *comment* baru, setelah mengetikan komentar lalu komentar akan dimasukan ke dalam *database* dengan menekan tomobol Post. Antarmuka ini juga memunculkan comment yang ada *di database*.