BAB 6
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari pembangunan Sistem Pakar Diagnosa Hama dan Penyakit Tanaman Kentang (PERKASA) dengan Metode *Certainty Factor* bahwa PERKASA telah selesai dibangun sebagai solusi dari rumusan masalah yang diangkat yaitu membangun sistem pakar yang dapat digunakan untuk mengetahui Hama dan Penyakit pada tanaman kentang berdasarkan gejala yang diberikan dan dapat memberikan solusi penanganan terhadap hama dan penyakit yang menyerang tanaman kentang.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan Sistem Pakar Diagnosa Hama dan Penyakit Tanaman Kentang (PERKASA) dengan Metode *Certainty Factor* adalah dengan mengembangkan sistem agar dapat juga diakses dengan menggunakan web dan fasilitas mobile lainnya, dan juga sistem dapat dikembangkan agar dapat lebih interaktif dan informatif.

SKPL

SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK

PERKASA

(Sistem Pakar Penanggulangan Hama dan Penyakit Pada Tanaman Kentang)

Untuk :

Universitas Atma Jaya Yogyakarta

Dipersiapkan oleh:

Goza Mauser / 5698

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

1 Pendahuluan 6
1.1 Tujuan 6
1.2 Lingkup Masalah 6
1.3 Definisi, Akronim dan Singkatan 7
1.4 Referensi 8
1.5 Deskripsi umum (Overview) 8
2 Deskripsi Kebutuhan 9
2.1 Perspektif produk 9
2.2 Fungsi Produk 10
2.3 Karakteristik Pengguna 16
2.4 Batasan-batasan 16
2.5 Asumsi dan Ketergantungan 16
3 Kebutuhan khusus 16
3.1 Kebutuhan antarmuka eksternal 16
3.2 Kebutuhan fungsionalitas Perangkat Lunak 18
4 Spesifikasi Rinci Kebutuhan 19
4.1 Spesifikasi Kebutuhan Fungsionalitas 19
5 Entity Relationship Diagram (ERD) 38

Daftar Gambar

1. Arsitektur Perangkat Lunak PERKASA 10
2. Use Case Diagram 18
3. Entity Relationship Diagram 38

1 Pendahuluan

1.1 Tujuan

Dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) ini merupakan dokumen spesifikasi kebutuhan perangkat lunak PERKASA (Sistem Pakar Penanggulangan Hama dan Penyakit Pada Tanaman Kentang) untuk mendefinisikan kebutuhan perangkat lunak yang meliputi antarmuka perangkat lunak (antarmuka antara sistem dengan pengguna), mendefinisikan perspektif perangkat lunak, mendefinisikan fungsionalitas perangkat lunak. SKPL-PERKASA ini juga mendefinisikan batasan perancangan perangkat lunak.

Dokumen ini digunakan oleh pengembang perangkat lunak sebagai acuan teknis pengembangan perangkat lunak pada tahap selanjutnya.

1.2 Lingkup Masalah

Perangkat Lunak PERKASA dibangun dengan tujuan untuk:

1. Membantu user untuk menentukan jenis hama dan penyakit pada tanaman kentang.
2. Membantu user untuk menemukan solusi terhadap jenis hama dan penyakit yang dihadapi tanaman kentang.
3. Membantu user untuk mendapatkan informasi mengenai keadaan tanaman kentang .

Dan berjalan pada lingkungan dengan platform Windows.

1.3 Definisi, Akronim dan Singkatan

Daftar definisi akronim dan singkatan :

Keyword/Phrase	Definisi
SKPL	Merupakan spesifikasi kebutuhan dari perangkat lunak yang akan dikembangkan.
SKPL-PERKASA-XXX	Kode yang merepresentasikan kebutuhan pada PERKASA (Sistem Pakar Penanggulangan Hama dan Penyakit Pada Tanaman Kentang) dimana XXX merupakan nomor fungsi produk.
PERKASA	Perangkat lunak yang dapat membantu keputusan user untuk menemukan solusi terhadap jenis hama dan penyakit pada tanaman kentang.
DBMS	Data Base Management Sistem merupakan tempat penyimpan dan pengelolaan database.
Server	Komputer yang menyediakan sumber daya bagi klien yang terhubung melalui jaringan.
Sistem Pakar	Sistem Pakar merupakan sistem informasi yang berisi dengan pengetahuan dari pakar sehingga dapat digunakan untuk konsultasi.
CF	Certainty Factor merupakan suatu metode untuk membuktikan apakah suatu fakta itu pasti ataukah tidak pasti yang biasanya digunakan dalam sistem pakar.

1.4 Referensi

Referensi yang digunakan pada perangkat lunak tersebut adalah:

1. Deitel, *C# How to Program*, Prentice-Hall Inc, 2002.
2. MSDN Library-October 2005, Microsoft, 2005.
3. Boggs Wendy, Boggs Michael, Mastering UML with Rational Rose 2002, SYBEX Inc,2002.
4. Bennet Simon, McRobb Steve, Farmer Ray, Object-Oriented System Analysis and Design Using UML, McGraw-Hill Companies, 2002.

1.5 Deskripsi umum (Overview)

Secara umum dokumen SKPL ini terbagi atas 3 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 PERKASA 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 PERKASA tersebut.

Bagian ketiga berisi penjelasan secara lebih rinci tentang kebutuhan perangkat lunak PERKASA yang akan dikembangkan.

2 Deskripsi Kebutuhan

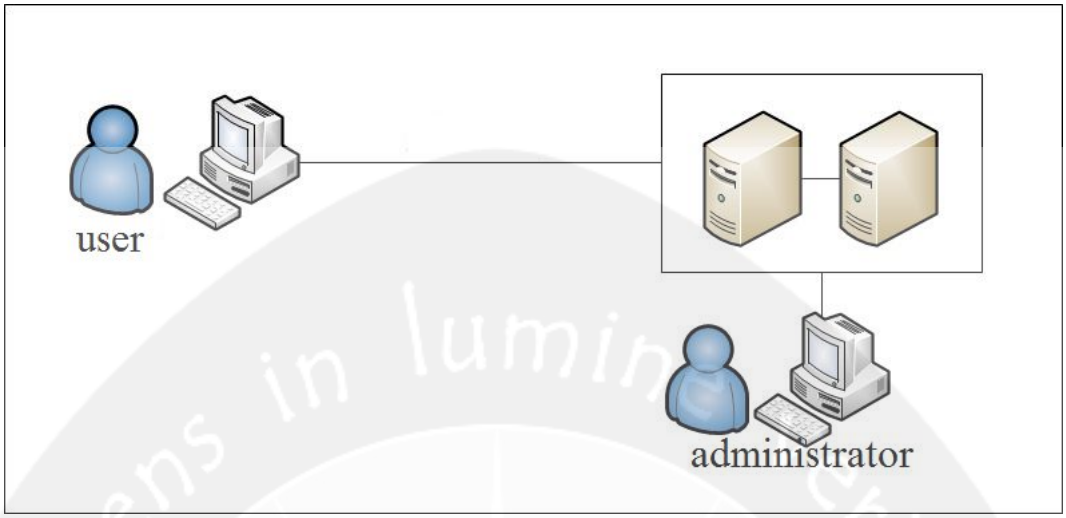
2.1 Perspektif produk

PERKASA merupakan perangkat lunak yang dibangun untuk membantu menemukan solusi dan penanggulangan penyakit dan hama pada tanaman kentang berdasarkan inputan dari user.

Perangkat lunak PERKASA ini berjalan pada platform Windows XP, Vista, dan 7 untuk perangkat PC, dan dibuat menggunakan bahasa pemrograman Microsoft Visual C#. Sedangkan untuk lingkungan pemrogramannya menggunakan Microsoft Visual Studio 2005.

Pengguna akan berinteraksi dengan sistem melalui antarmuka GUI (Graphical User Interface). Pada sistem ini, seperti terlihat pada gambar 1, arsitektur perangkat lunak yang digunakan berupa client server, di mana semua data disimpan di server. User dapat mengakses data yang ada di server tersebut.

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 merequest melalui komputer client.



Gambar 1. Arsitektur Perangkat lunak PERKASA

2.2 Fungsi Produk

Fungsi produk perangkat lunak PERKASA adalah sebagai berikut :

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

   Merupakan fungsi yang digunakan oleh user untuk dapat masuk dalam sistem yang akan digunakan.

2. Fungsi Edit Password (**SKPL-PERKASA-002**).

   Merupakan fungsi yang digunakan oleh user untuk mengedit password mereka masing-masing.

3. Fungsi *Pengelolaan Data User* (**SKPL-PERKASA-003**).

   Merupakan fungsi yang digunakan untuk mengelola petugas PERKASA yang akan menggunakan sistem ini.

   Fungsi Pengelolaan Data User mencakup :

   a. Fungsi *Entry Data User* (**SKPL-PERKASA-003-01**).
      Merupakan fungsi yang digunakan untuk menambahkan data user yang baru.

b. Fungsi *Edit Data User* (**SKPL-PERKASA-003-02).** Merupakan fungsi yang digunakan untuk mengubah data user yang diinginkan kecuali password.

c. Fungsi *Delete Data User* (**SKPL-PERKASA-003-03**). Merupakan fungsi yang digunakan untuk menghapus data user yang diinginkan.

d. Fungsi *Display Data User* (**SKPL-PERKASA-003-04**).

Merupakan fungsi yang digunakan untuk menampilkan data user yang diinginkan.

e. Fungsi *Cari Data User* (**SKPL-PERKASA-003-05**).

Merupakan fungsi yang digunakan untuk mencari pegawai berdasarkan nama user dan role.

4. Fungsi *Pengelolaan Gejala*(**SKPL-PERKASA-004**).

Merupakan fungsi yang digunakan untuk mengelola gejala-gejala apa saja yang dialami tanaman kentang.

Fungsi pengelolaan Data Gejala meliputi:

a. Fungsi *Entry Gejala* (**SKPL-PERKASA-004-01**).

Merupakan fungsi yang digunakan untuk menambah gejala baru.

b. Fungsi *Edit Gejala* (**SKPL-PERKASA-004-02**).

Merupakan fungsi untuk mengedit gejala yang diinginkan.

c. Fungsi *Delete Gejala* (**SKPL-PERKASA-004-03**).

Merupakan fungsi untuk menghapus gejala yang diinginkan.

d. Fungsi *Display Gejala* (**SKPL-PERKASA-004-04**).

Merupakan fungsi yang digunakan untuk menampilkan gejala yang diinginkan.

e. Fungsi Search Gejala **(SKPL-PERKASA-004-05).**

Merupakan fungsi yang digunakan untuk mencari gejala yang diinginkan.

5. Fungsi *Display Gambar* (**SKPL-PERKASA-005**)

Merupakan fungsi yang digunakan untuk mengetahui gejala tanaman melalui gambar.

Fungsi Display Gambar meliputi :

a. Fungsi *Display Gambar* (**SKPL-PERKASA-005-01**).

Merupakan fungsi yang digunakan untuk menampilkan gambar yang mirip dengan gejala yang diinputakan user.

6. Fungsi *Display Solusi* (**SKPL-PERKASA-006**).

Merupakan fungsi yang digunakan untuk memberikan solusi dan penanggulangan berdasarkan gejala yang diinputkan user.

a. Fungsi *Display Data Solusi* (**SKPL-PERKASA-006-01**).

Merupakan fungsi untuk menampilkan data solusi berdasarkan gejala yang diinputkan user untuk menemukan solusi untuk penanggulangan hama dan penyakit pada tanaman kentang.

7. Fungsi Pengelolaan Penyakit (**SKPL-PERKASA-007**).

Merupakan fungsi yang digunakan untuk mengelola penyakit yang diinputkan user.

Fungsi pengelolaan penyakit meliputi:

a. Fungsi *Entry Penyakit* (**SKPL-PERKASA-007-01**).

   Merupakan fungsi yang digunakan untuk menambah penyakit baru.

b. Fungsi *Edit Penyakit* (**SKPL-PERKASA-007-02**).

   Merupakan fungsi untuk mengedit penyakit yang diinginkan.

c. Fungsi *Delete Penyakit* (**SKPL-PERKASA-007-03**).

   Merupakan fungsi untuk menghapus penyakit yang diinginkan.

d. Fungsi *Display Penyakit* (**SKPL-PERKASA-007-04**).

   Merupakan fungsi yang digunakan untuk menampilkan penyakit yang diinginkan.

e. Fungsi Search Penyakit **(SKPL-PERKASA-007-05).**

   Merupakan fungsi yang digunakan untuk mencari penyakit yang diinginkan.

8. Fungsi Pengelolaan Hama **(SKPL-PERKASA-008).**

   Merupakan fungsi yang digunakan untuk mengelola hama yang diinputkan user.

   Fungsi pengelolaan Hama meliputi:

   a. Fungsi *Entry Hama* (**SKPL-PERKASA-008-01**).

      Merupakan fungsi yang digunakan untuk menambah hama baru.

   b. Fungsi *Edit Hama* (**SKPL-PERKASA-008-02**).

      Merupakan fungsi untuk mengedit hama yang diinginkan.

c. Fungsi *Delete Hama* (**SKPL-PERKASA-008-03**).

Merupakan fungsi untuk menghapus hama yang diinginkan.

d. Fungsi *Display Hama* (**SKPL-PERKASA-008-04**).

Merupakan fungsi yang digunakan untuk menampilkan hama yang diinginkan.

e. Fungsi Search Hama **(SKPL-PERKASA-008-05).**

Merupakan fungsi yang digunakan untuk mencari hama yang diinginkan.

9. Fungsi Pengelolaan Aturan **(SKPL-PERKASA-009).**

Merupakan fungsi yang digunakan untuk mengelola aturan berdasarkan hama dan penyakit.

Fungsi pengelolaan Aturan meliputi:

a. Fungsi *Entry Aturan* (**SKPL-PERKASA-009-01**).

Merupakan fungsi yang digunakan untuk menambah aturan baru.

b. Fungsi *Edit Aturan* (**SKPL-PERKASA-009-02**).

Merupakan fungsi untuk mengedit aturan yang diinginkan.

c. Fungsi *Delete Aturan* (**SKPL-PERKASA-009-03**).

Merupakan fungsi untuk menghapus aturan yang diinginkan.

d. Fungsi *Display Aturan* (**SKPL-PERKASA-009-04**).

Merupakan fungsi yang digunakan untuk menampilkan aturan yang diinginkan.

e. Fungsi Search Aturan **(SKPL-PERKASA-009-05).**

Merupakan fungsi yang digunakan untuk mencari aturan yang diinginkan.

10. Fungsi *Display Gejala* (**SKPL-PERKASA-011**)

Merupakan fungsi yang digunakan untuk mengetahui gejala-gejala apa saja yang dialami tanaman kentang.

Fungsi Display Gambar meliputi :

a. Fungsi *Display Gejala* (**SKPL-PERKASA-011-01**).

Merupakan fungsi yang digunakan untuk menampilkan gejala yang diinputkan user.

11. Fungsi Display Hama (**SKPL-PERKASA-012**)

Merupakan fungsi yang digunakan untuk mengetahui Hama apa saja yang dialami tanaman kentang.

Fungsi Display Gambar meliputi :

a. Fungsi *Display Hama* (**SKPL-PERKASA-012-01**).

Merupakan fungsi yang digunakan untuk menampilkan Hama yang diinputkan user.

12. Fungsi Display Penyakit (**SKPL-PERKASA-013**)

Merupakan fungsi yang digunakan untuk mengetahui Penyakit apa saja yang dialami tanaman kentang.

Fungsi Display Gambar meliputi :

a. Fungsi *Display Penyakit* (**SKPL-PERKASA-013-01**).

Merupakan fungsi yang digunakan untuk menampilkan Penyakit yang diinputkan user.

2.3 Karakteristik Pengguna

Karakteristik dari pengguna perangkat lunak PERKASA adalah sebagai berikut :

1. Memahami pengoperasian PC dan fitur-fitur aplikasinya.

2.4 Batasan-batasan

Batasan-batasan dalam pengembangan perangkat lunak PERKASA tersebut adalah :

1. Kebijaksanaan Umum

   Berpedoman pada tujuan dari pengembangan perangkat lunak PERKASA.

2. Keterbatasan perangkat keras

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

2.5 Asumsi dan Ketergantungan

Sistem ini dapat dijalankan pada perangkat PC (Komputer, Laptop, dll) yang menggunakan system operasi Windows XP, Vista, 7.

3 Kebutuhan khusus

3.1 Kebutuhan antarmuka eksternal

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

3.1.1 Antarmuka pemakai

Pengguna berinteraksi dengan antarmuka yang ditampilkan dalam bentuk form-form.

3.1.2 Antarmuka perangkat keras

Antarmuka perangkat keras yang digunakan dalam perangkat lunak PERKASA adalah:

1. Perangkat PC (Komputer, Laptop, dll).

3.1.3 Antarmuka perangkat lunak

Perangkat lunak yang dibutuhkan untuk mengoperasikan perangkat lunak PERKASA adalah sebagai berikut :

1. Nama : SQL Server 2005
   Sumber : Microsoft
   Sebagai database management system (DBMS) yang digunakan untuk penyimpan data di sisi server.

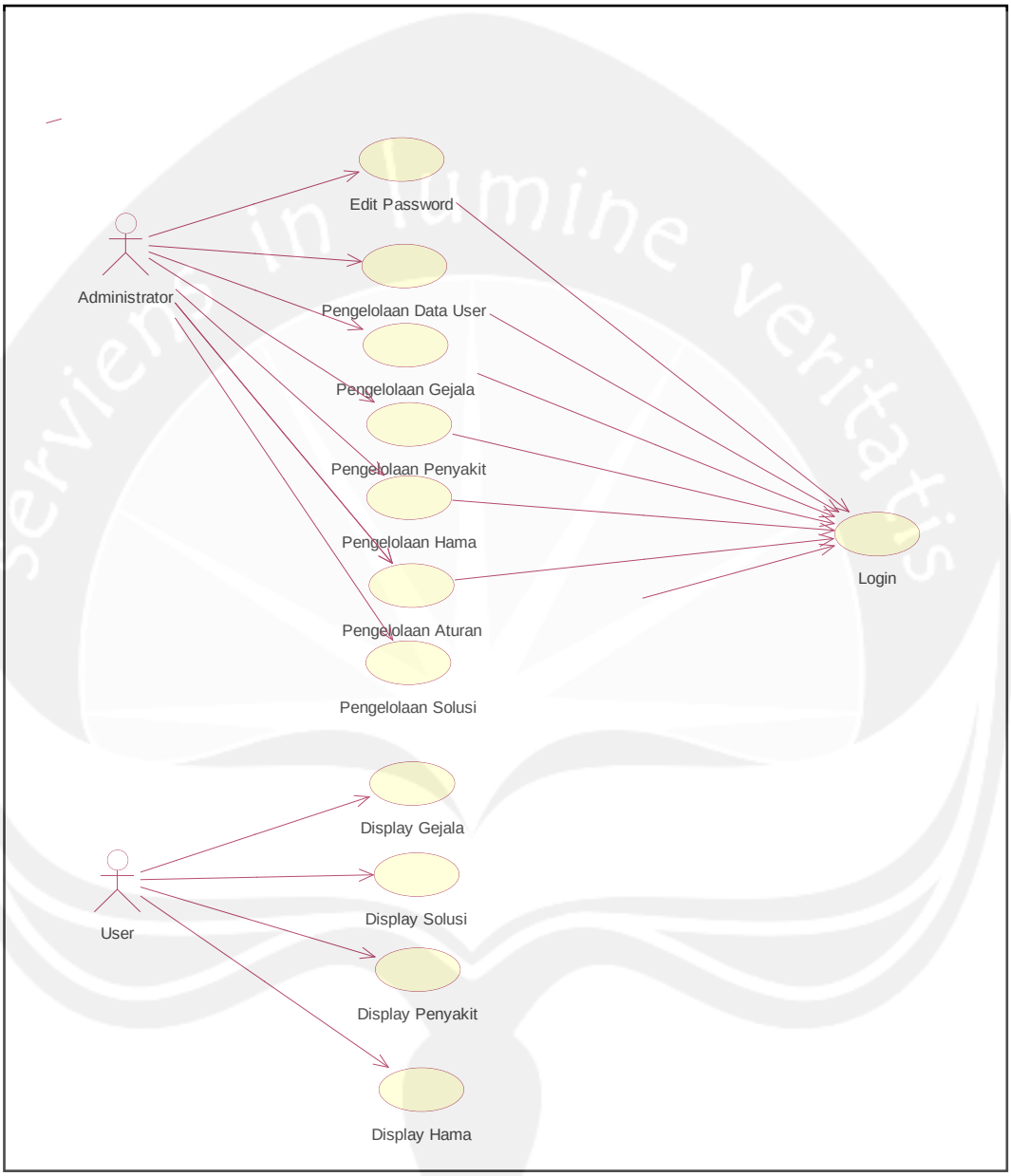
2. Nama : Windows XP, Vista, 7
   Sumber : Microsoft.
   Sebagai sistem operasi untuk perangkat PC.

3.1.4 Antarmuka Komunikasi

Antarmuka komunikasi perangkat lunak PERKASA menggunakan protocol HTTP.

3.2 Kebutuhan fungsionalitas Perangkat Lunak

3.2.1 Use Case Diagram



Gambar 2. Use Case Diagram

4 Spesifikasi Rinci Kebutuhan

4.1 Spesifikasi Kebutuhan Fungsionalitas

4.1.1 Use case Spesification : Login

1. Brief Description

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

2. Primary Actor

1. Administrator
2. Petugas Pencatat Transaksi Barang
3. Petugas Pengelola Status Barang
4. Petugas Pengelola Gambar

3. Supporting Actor

none

4. Basic Flow

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

5. Alternative Flow

none

6. 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

7. PreConditions

none

8. PostConditions

1. Aktor memasuki sistem dan dapat menggunakan fungsi-fungsi pada sistem.

4.1.2 **Use case Spesification :** Pengelolaan data User

1. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data user. Aktor dapat melakukan entry data user, edit data user, delete data user, display data user, atau search data user.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

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

   A-1 Aktor memilih untuk melakukan edit data user

   A-2 Aktor memilih untuk melakukan delete data user

   A-3 Aktor memilih untuk melakukan display data user

   A-4 Aktor memilih untuk melakukan search data user
4. Aktor menginputkan data user

5. Aktor meminta sistem untuk menyimpan data user yang telah diinputkan
6. Sistem mengecek data user yang telah diinputkan
   E-1 Data user yang diinputkan aktor salah
7. Sistem menyimpan data user ke database
8. Use Case selesai

5. Alternative Flow

A-1 Aktor memilih untuk melakukan edit data user
1. Sistem menampilkan data user
2. Aktor mengedit data user yang sudah ditampilkan
3. Aktor meminta sistem untuk menyimpan data user yang telah diedit
4. Sistem melakukan pengecekan terhadap data user yang telah diedit
   E-2 Data user yang telah diedit salah
5. Sistem meyimpan data user yang telah diedit ke database
6. Berlanjut ke Basic Flow langkah ke 8

A-2 Aktor memilih untuk melakukan delete data user
1. Sistem menampilkan data user
2. Aktor mengklik tombol delete yang ada pada sistem
3. Sistem menghapus data user dari database
4. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan display data user
1. Sistem menampilkan data user
2. Berlanjut ke Basic Flow langkah ke 8

A-4 Aktor memilih untuk melakukan search data user
1. Sistem menampilkan form untuk menginputkan nama dan role untuk diisi dengan nama dan role yang diinginkan aktor

2. Aktor menginputkan nama dan role user yang dicari
3. Sistem mencari dan mencocokkan dengan database
   E3 User tidak ditemukan
4. Sistem menampilkan data user yang dicari
5. Berlanjut ke Basic Flow langkah ke 8

6. Error Flow

E-1 Data user 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 user yang diinputkan aktor salah
1. Sistem memberikan pesan peringatan bahwa data yang diedit salah
2. Kembali ke Alternative Flow A-1 Langkah ke 2

E-3 Data user yang dicari tidak ditemukan
1. Sistem memberikan pesan peringatan bahwa data yang dicari tidak ditemukan
2. Kembali ke Alternative Flow A-4 Langkah ke 2

7. PreConditions

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

8. PostConditions

1. Data user di database telah terupdate

4.1.3 **Use case Spesification :** Pengelolaan Gejala

1. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data gejala. Aktor dapat melakukan entry data gejala, edit data gejala, delete data gejala, display data gejala, atau search data gejala.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk melakukan pengelolaan data gejala.
2. Sistem memberikan pilihan untuk melakukan entry data gejala, edit data gejala, delete data gejala, display data gejala, atau search data gejala.
3. Aktor memilih untuk melakukan entry data gejala
   A-1 Aktor memilih untuk melakukan edit data gejala
   A-2 Aktor memilih untuk melakukan delete data gejala
   A-3 Aktor memilih untuk melakukan display data gejala
   A-4 Aktor memilih untuk melakukan search data gejala
4. Aktor menginputkan data gejala
5. Aktor meminta sistem untuk menyimpan data gejala yang telah diinputkan
6. Sistem mengecek data gejala yang telah diinputkan
   E-1 Data gejala yang diinputkan aktor salah
7. Sistem menyimpan data gejala ke database
8. Use Case selesai

5. Alternative Flow

A-1 Aktor memilih untuk melakukan edit data gejala

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

4. Sistem melakukan pengecekan terhadap data gejala yang telah diedit

E-2 Data gejala yang telah diedit salah

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

A-2 Aktor memilih untuk melakukan delete data gejala

1. Sistem menampilkan data gejala
2. Aktor mengklik tombol delete yang ada pada sistem
3. Sistem menghapus data gejala dari database
4. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan display data gejala

1. Sistem menampilkan data gejala
2. Berlanjut ke Basic Flow langkah ke 8

A-4 Aktor memilih untuk melakukan search data gejala

6. Sistem menampilkan form untuk menginputkan nama gejala untuk diisi dengan nama gejala yang diinginkan aktor
7. Aktor menginputkan nama gejala yang dicari
8. Sistem mencari dan mencocokkan dengan database

E3 Gejala tidak ditemukan

9. Sistem menampilkan data gejala yang dicari
10. Berlanjut ke Basic Flow langkah ke 8

6. Error Flow

E-1 Data gejala 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 gejala yang diinputkan aktor salah

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

E-3 Data gejala yang dicari tidak ditemukan

3. Sistem memberikan pesan peringatan bahwa data yang dicari tidak ditemukan
4. Kembali ke Alternative Flow A-4 Langkah ke 2

7. PreConditions

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

8. PostConditions

1. Data gejala di database telah terupdate

4.1.4 Use case Spesification : Display Solusi

1. Brief Description

Use Case ini digunakan oleh aktor untuk melakukan display solusi. Aktor dapat melakukan display solusi.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk melakukan display solusi.
2. Sistem memberikan pilihan untuk melakukan display solusi.
3. Aktor meminta sistem untuk menampilkan gejala yang telah diinputkan user.
4. Sistem mengecek gejala yang diinputkan

   E-1 gejala yang diinputkan aktor salah
5. Use Case selesai
6.

5. Alternative Flow

none

6. Error Flow

E-1 Data tarif yang diinputkan aktor salah

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

7. PreConditions

1. Use Case Login telah dilakukan
2. Aktor telah memasuki system

8. PostConditions

Solusi di database telah ditampilkan.

4.1.5 **Use case Spesification :** Pengelolaan Hama

1. Brief Description

Use Case ini digunakan oleh aktor untuk mengelola data hama. Aktor dapat melakukan entry data hama, edit data hama, delete data hama, display data hama, atau search data hama.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

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

   A-1 Aktor memilih untuk melakukan edit data hama

A-2 Aktor memilih untuk melakukan delete data hama

A-3 Aktor memilih untuk melakukan display data hama

A-4 Aktor memilih untuk melakukan search data hama

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

E-1 Data hama yang diinputkan aktor salah

7. Sistem menyimpan data hama ke database
8. Use Case selesai

5. Alternative Flow

A-1 Aktor memilih untuk melakukan edit data hama

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

E-2 Data hama yang telah diedit salah

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

A-2 Aktor memilih untuk melakukan delete data hama

5. Sistem menampilkan data hama
6. Aktor mengklik tombol delete yang ada pada sistem
7. Sistem menghapus data hama dari database
8. Berlanjut ke Basic Flow langkah ke 8

A-3 Aktor memilih untuk melakukan display data hama

1. Sistem menampilkan data hama
2. Berlanjut ke Basic Flow langkah ke 8

A-4 Aktor memilih untuk melakukan search data hama

11. Sistem menampilkan form untuk menginputkan nama hama untuk diisi dengan nama hama yang diinginkan aktor
12. Aktor menginputkan nama hama yang dicari
13. Sistem mencari dan mencocokkan dengan database

E3 Gejala tidak ditemukan

14. Sistem menampilkan data hama yang dicari
15. Berlanjut ke Basic Flow langkah ke 8

6. Error Flow

E-1 Data hama yang diinputkan aktor salah

7. Sistem memberikan pesan peringatan bahwa data yang diinputkan salah
8. Kembali ke Basic Flow Langkah ke 4

E-2 Data hama yang diinputkan aktor salah

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

E-3 Data hama yang dicari tidak ditemukan

5. Sistem memberikan pesan peringatan bahwa data yang dicari tidak ditemukan
6. Kembali ke Alternative Flow A-4 Langkah ke 2

7. PreConditions

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

8. PostConditions

1. Data hama di database telah terupdate.
2.

4.1.7 Use case Spesification : Pengelola Penyakit

1. Primary Actor

1. Administrator

2. Supporting Actor

None

3. Brief Description

Use case ini digunakan oleh administrator untuk mengelola penyakit. Penyakit yang dikelola meliputi : Entri penyakit, Edit penyakit, Hapus penyakit, Display penyakit, Search penyakit.

4. PreConditions

1. Administrator telah berhasil memasuki sistem

5. Basic Flow

1. Use Case ini dimulai ketika administrator memilih untuk mengelola penyakit.
2. Sistem memberikan pilihan untuk entry penyakit, edit penyakit, hapus penyakit , display penyakit , dan search penyakit.
3. Administrator memilih untuk melakukan entry penyakit.

   A-1 administrator memilih untuk melakukan edit penyakit.

   A-2 administrator memilih untuk melakukan hapus penyakit.

   A-3 administrator memilih untuk melakukan display penyakit.

   A-4 administrator memilih untuk melakukan search penyakit.
4. Administrator menginputkan penyakit.
5. Administrator meminta sistem menyimpan penyakit yang telah diinputkan.
6. Sistem mengecek penyakit yang telah diinputkan

   E-1 Penyakit yang diinputkan ada yang salah.

7. Sistem menyimpan penyakit ke database
8. Use Case selesai

6. Alternative Flow

A-1 administrator memilih untuk melakukan edit penyakit (setelah Basic Flow langkah ke-2)

1. Sistem menampilkan penyakit
2. Administrator memilih penyakit yang akan diedit
3. Sistem menampilkan penyakit yang telah dipilih
4. Administrator mengedit penyakit yang dipilih
5. Sistem melakukan pengecekan terhadap penyakit yang telah diedit

   E-2 Penyakit yang telah diedit salah
6. Sistem meyimpan penyakit yang telah diedit ke database
7. Kembali ke Basic Flow langkah ke-8

A-2 administrator memilih untuk melakukan hapus penyakit(setelah Basic Flow langkah ke-2)

1. Sistem menampilkan penyakit
2. Administrator memilih penyakit yang akan didelete
3. Administrator meminta sistem untuk mendelete penyakit yang telah dipilih
4. Sistem melakukan penghapusan penyakit yang dipilih
5. Kembali ke Basic Flow langkah ke-8.

A-3 administrator memilih untuk melakukan display penyakit(setelah Basic Flow langkah ke-2)

1. Sistem menampilkan penyakit
2. Kembali ke Basic Flow langkah ke 8

A-4 administrator memilih melakukan search penyakit(setelah Basic Flow langkah ke-2)

1.Sistem menampilkan kembali penyakit
2.Administrator melakukan pencarian penyakit

3.Kembali ke Basic Flow langkah ke-8

7. Error Flow

E-1 penyakit yang diinputkan ada yang salah.

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

8. PostConditions

1. Penyakit di database telah terisi sesuai inputan administrator.

4.1.8 Use case Spesification : Pengelolaan Aturan

1. Primary Actor

1. Administrator.

2. Supporting Actor

None

3. Brief Description

Use case ini digunakan oleh administrator untuk mengelola aturan. Aturan yang dikelola meliputi : Entri aturan, Edit aturan, Hapus aturan, Display aturan, Search aturan.

4. PreConditions

1. Administrator telah berhasil memasuki sistem

5. Basic Flow

1. Use Case ini dimulai ketika administrator memilih untuk mengelola aturan.
2. Sistem memberikan pilihan untuk entry aturan, edit aturan, hapus aturan , display aturan , dan search aturan.
3. Administrator memilih untuk melakukan entry aturan.

A-1 administrator memilih untuk melakukan edit aturan.

A-2 administrator memilih untuk melakukan hapus aturan.

A-3 administrator memilih untuk melakukan display aturan.

A-4 administrator memilih untuk melakukan search aturan.

4. Administrator menginputkan aturan.
5. Administrator meminta sistem menyimpan aturan yang telah diinputkan.
6. Sistem mengecek aturan yang telah diinputkan

E-1 Aturan yang diinputkan ada yang salah.

7. Sistem menyimpan aturan ke database
8. Use Case selesai

6. Alternative Flow

A-1 administrator memilih untuk melakukan edit aturan (setelah Basic Flow langkah ke-2)

8. Sistem menampilkan aturan
9. Administrator memilih aturan yang akan diedit
10. Sistem menampilkan aturan yang telah dipilih
11. Administrator mengedit aturan yang dipilih
12. Sistem melakukan pengecekan terhadap aturan yang telah diedit

E-2 Aturan yang telah diedit
salah

13. Sistem meyimpan aturan yang telah diedit ke database
14. Kembali ke Basic Flow langkah ke-8

A-2 administrator memilih untuk melakukan hapus aturan(setelah Basic Flow langkah ke-2)

5. Sistem menampilkan aturan

6. Administrator memilih aturan yang akan didelete

7. Administrator meminta sistem untuk mendelete aturan yang telah dipilih

8. Sistem melakukan penghapusan aturan yang dipilih

6. Kembali ke Basic Flow langkah ke-8.

A-3 administrator memilih untuk melakukan display aturan(setelah Basic Flow langkah ke-2)

3. Sistem menampilkan aturan
4. Kembali ke Basic Flow langkah ke 8

A-4 administrator memilih melakukan search aturan(setelah Basic Flow langkah ke-2)

1.Sistem menampilkan kembali aturan
2.Administrator melakukan pencarian aturan
3.Kembali ke Basic Flow langkah ke-8

7. Error Flow

E-1 aturan yang diinputkan ada yang salah.

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

8. PostConditions

1. Aturan di database telah terisi sesuai inputan administrator.

4.1.9 Use case Spesification : Display Gejala

1. Brief Description

Use Case ini digunakan oleh aktor untuk melakukan display gejala. Aktor dapat melakukan display display gejala.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk melakukan display gejala.
2. Sistem memberikan pilihan untuk melakukan display gejala.

3. Aktor meminta sistem untuk menampilkan gejala sesuai dengan gejala yang diinputkan user
4. Sistem mengecek gejala yang telah diinputkan
   E-1 gejala yang diinputkan aktor salah
5. Use Case selesai

5. Alternative Flow

none

6. Error Flow

E-1 gejala yang diinputkan aktor salah
1. Sistem memberikan pesan peringatan bahwa gejala yang diinputkan salah
2. Kembali ke Basic Flow Langkah ke 1

7. PreConditions

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

8. PostConditions

1. gejala di database telah ditampilkan.

4.1.10 **Use case Spesification :** Display Hama

1. Brief Description

Use Case ini digunakan oleh aktor untuk melakukan display hama. Aktor dapat melakukan display display hama.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk melakukan display hama.
2. Sistem memberikan pilihan untuk melakukan display hama.
3. Aktor meminta sistem untuk menampilkan hama sesuai dengan hama yang diinputkan user
4. Sistem mengecek hama yang telah diinputkan

E-1 hama yang diinputkan aktor salah

5. Use Case selesai

5. Alternative Flow

none

6. Error Flow

E-1 hama yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa hama yang diinputkan salah
2. Kembali ke Basic Flow Langkah ke 1

7. PreConditions

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

8. PostConditions

1. hama di database telah ditampilkan.

4.1.11 Use case Spesification : Display Penyakit

1. Brief Description

Use Case ini digunakan oleh aktor untuk melakukan display penyakit. Aktor dapat melakukan display display penyakit.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk melakukan display penyakit.
2. Sistem memberikan pilihan untuk melakukan display penyakit.
3. Aktor meminta sistem untuk menampilkan penyakit sesuai dengan penyakit yang diinputkan user
4. Sistem mengecek penyakit yang telah diinputkan

   E-1 penyakit yang diinputkan aktor salah

5. Use Case selesai

5. Alternative Flow

none

6. Error Flow

E-1 penyakit yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa penyakit yang diinputkan salah
2. Kembali ke Basic Flow Langkah ke 1

7. PreConditions

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

8. PostConditions

1. penyakit di database telah ditampilkan.

4.1.12 Use case Spesification : Display Solusi

1. Brief Description

Use Case ini digunakan oleh aktor untuk melakukan display hama. Aktor dapat melakukan display display solusi.

2. Primary Actor

1. Administrator

3. Supporting Actor

none

4. Basic Flow

1. Use Case ini dimulai ketika aktor memilih untuk melakukan display solusi.
2. Sistem memberikan pilihan untuk melakukan display solusi.
3. Aktor meminta sistem untuk menampilkan solusi sesuai dengan solusi yang diinputkan user
4. Sistem mengecek solusi yang telah diinputkan

   E-1 solusi yang diinputkan aktor salah
5. Use Case selesai

5. Alternative Flow

none

6. Error Flow

E-1 solusi yang diinputkan aktor salah

1. Sistem memberikan pesan peringatan bahwa solusi yang diinputkan salah
2. Kembali ke Basic Flow Langkah ke 1

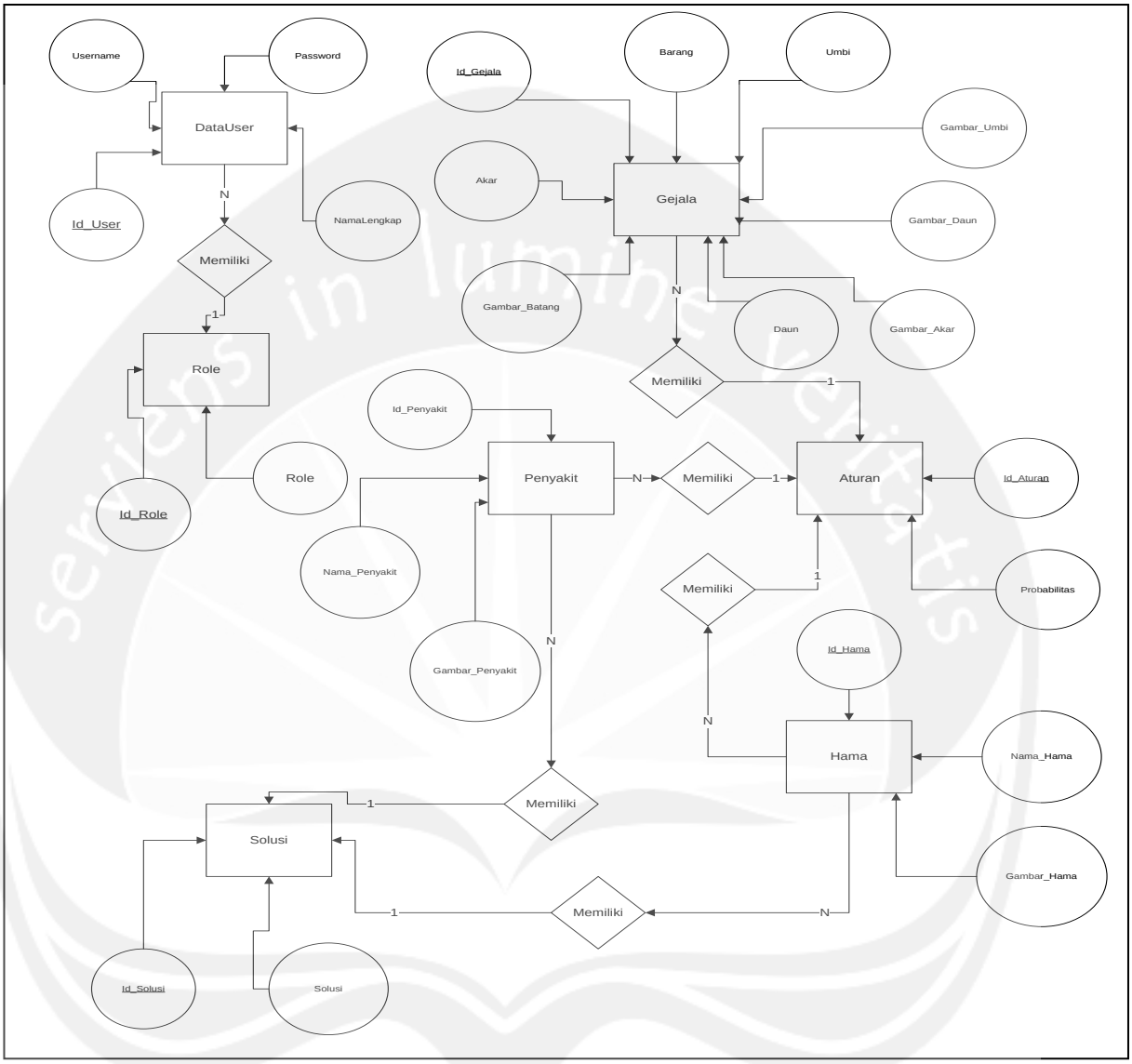
7. PreConditions

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

8. PostConditions

1. solusi di database telah ditampilkan.

5 Entity Relationship Diagram (ERD)



Gambar 3. Entity Relationship Diagram

DPPL

DESKRIPSI PERANCANGAN PERANGKAT LUNAK

PERKASA

(Sistem Pakar Penanggulangan Hama dan Penyakit Pada Tanaman Kentang)

Untuk :

Universitas Atma Jaya Yogyakarta

Dipersiapkan oleh:

Goza Mauser / 08 07 05698

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

1 Pendahuluan 6
1.1 *Tujuan* 6
1.2 *Ruang Lingkup* 6
1.3 *Definisi dan Akronim* 6
1.4 *Referensi* 7
2 Perancangan Sistem 9
2.1 *Perancangan Arsitektur* 9
2.2 *Perancangan Rinci* 10
2.2.1 Sequence Diagram 10
2.2.2 Class Diagram 37
2.2.3 Spesifikasi Deskripsi Kelas Diagram 38
3 Perancangan Data 48
3.1 Dekomposisi Data 48
3.1.1 Deskripsi Entitas Data User 48
3.1.2 Deskripsi Entitas Gejala 48
3.1.3 Deskripsi Entitas Penyakit 48
3.1.4 Deskripsi Entitas Hama 48
3.1.5 Deskripsi Entitas Aturan 49
3.2 *Physical Data Model* 49
4 Perancangan Antarmuka 50
4.1 *Sketsa Antarmuka dan Deskripsinya* 50
4.1.1 Antarmuka Login 50
4.1.2 Antarmuka Pengelolaan User 51
4.1.3 Antarmuka Edit Password 52
4.1.4 Antarmuka Pengelolaan Gejala 53
4.1.5 Antarmuka Pengelolaan Penyakit 54
4.1.6 Antarmuka Pengelolaan Hama 56
4.1.7 Antarmuka Pengelolaan Aturan 57
4.1.8 Antarmuka Display Gejala 59
4.1.9 Antarmuka Display Penyakit dan Hama ... 61
4.1.10 Antarmuka Display Solusi 62

Daftar Gambar

Gambar 2.28 Class Diagram Perkasa 37
Gambar 3.1 Physical Data Model 49
Gambar 4.1 Rancangan Antarmuka Login 50
Gambar 4.5 Rancangan Antarmuka Pengelolaan Gejala ... 53
Gambar 4.6 Rancangan Antarmuka Pengelolaan Penyakit . 54
Gambar 4.7 Rancangan Antarmuka Pengelolaan Hama 56
Gambar 4.8 Rancangan Antarmuka Pengelolaan Aturan ... 57
Gambar 4.9 Rancangan Antarmuka Display Gejala 59
Gambar 4.10 Rancangan Antarmuka Display Penyakit dan Hama 61
Gambar 4.11 Rancangan Antarmuka Display Solusi 62

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 Perkasa dibangun dengan tujuan untuk :

Membantu user untuk menentukan jenis hama dan penyakit pada tanaman kentang.

Membantu user untuk menemukan solusi terhadap jenis hama dan penyakit yang dihadapi tanaman kentang.

Membantu user untuk mendapatkan informasi mengenai keadaan tanaman kentang.

Dan berjalan pada lingkungan dengan platform Windows.

1.3 *Definisi dan Akronim*

Daftar definisi akronim dan singkatan :

Keyword/Phrase	Definisi
DPPL	Deskripsi Perancangan Perangkat Lunak disebut juga *Software Design Description* (SDD) merupakan deskripsi dari perancangan produk/perangkat lunak yang akan dikembangkan.

Perkasa	Perangkat lunak yang dapat membantu keputusan user untuk menentukan dan mendapatkan solusi penanggulangan hama dan penyakit pada tanaman kentang berdasarkan gejala yang diinputkan user.
Server	Komputer yang menyediakan sumber daya bagi klien yang terhubung melalui jaringan.
DBMS	Data Base Management Sistem merupakan tempat penyimpan dan pengelolaan database.
Sistem Pakar	Sistem Pakar merupakan sistem informasi yang berisi dengan pengetahuan dari pakar sehingga dapat digunakan untuk konsultasi.
CF	Certainty Factor merupakan suatu metode untuk membuktikan apakah suatu fakta itu pasti ataukah tidak pasti yang biasanya digunakan dalam sistem pakar.

1.4 *Referensi*

Referensi yang digunakan pada perangkat lunak tersebut adalah:

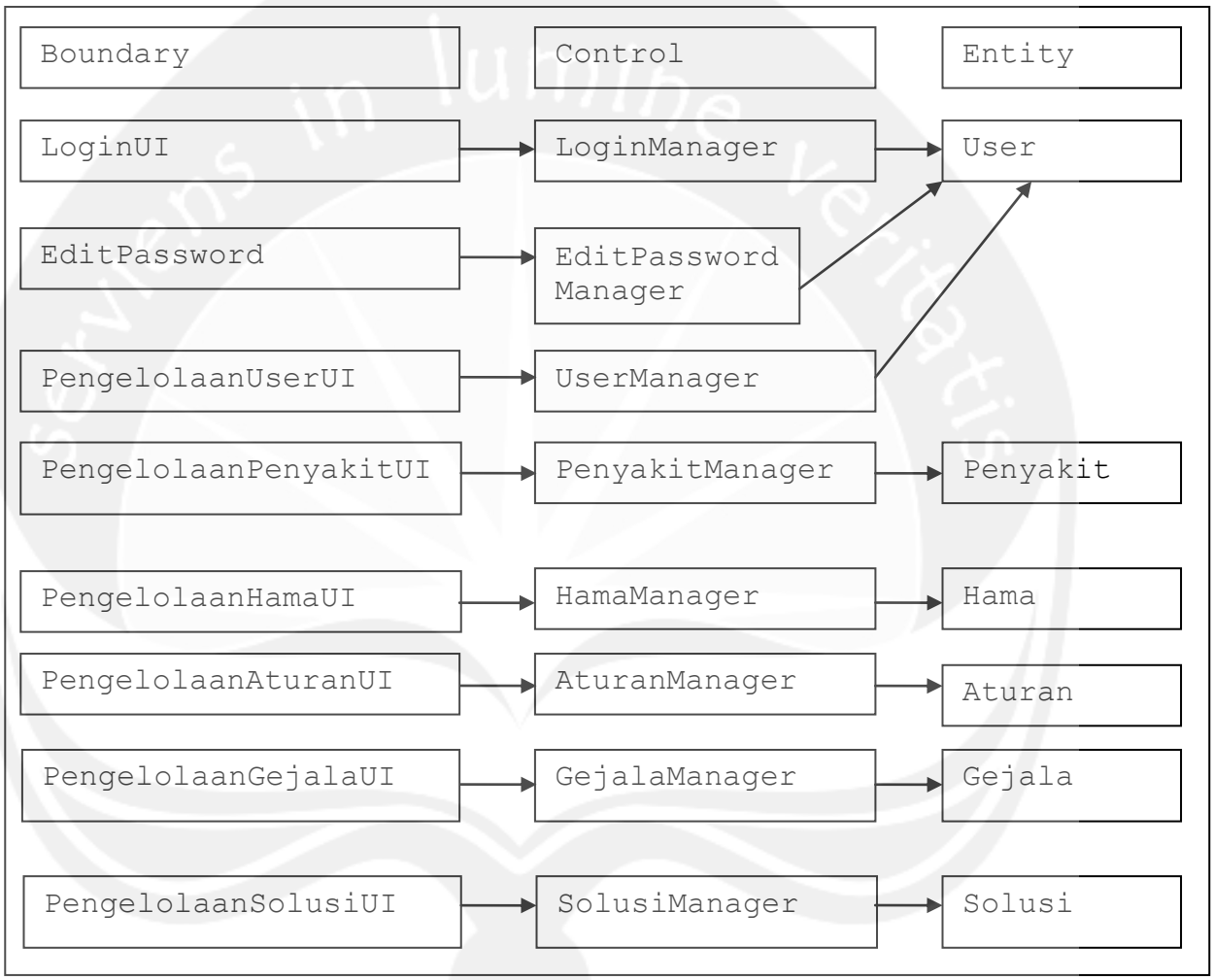
1. Bennet Simon, McRobb Steve, Farmer Ray, *Object-Oriented System Analysis and Design Using UML*, McGraw-Hill Companies, 2002.
2. Boggs Wendy, Boggs Michael, *Mastering UML with Rational Rose 2002*, SYBEX Inc, 2002.
3. Mauser , Goza. *Spesifikasi Kebutuhan Perangkat Lunak Perkasa*, Universitas Atma Jaya Yogyakarta, 2013.

4. Sapta, Juli. *Deskripsi Perancangan Perangkat Lunak SC3*, Universitas Atma Jaya Yogyakarta, 2006.
5. Ditya, Emmanuel. *Aplikasi Wisata Jogja, Universitas Atma Jaya Yogyakarta, 2013.*

2. Perancangan Sistem

1.5 *Perancangan Arsitektur*

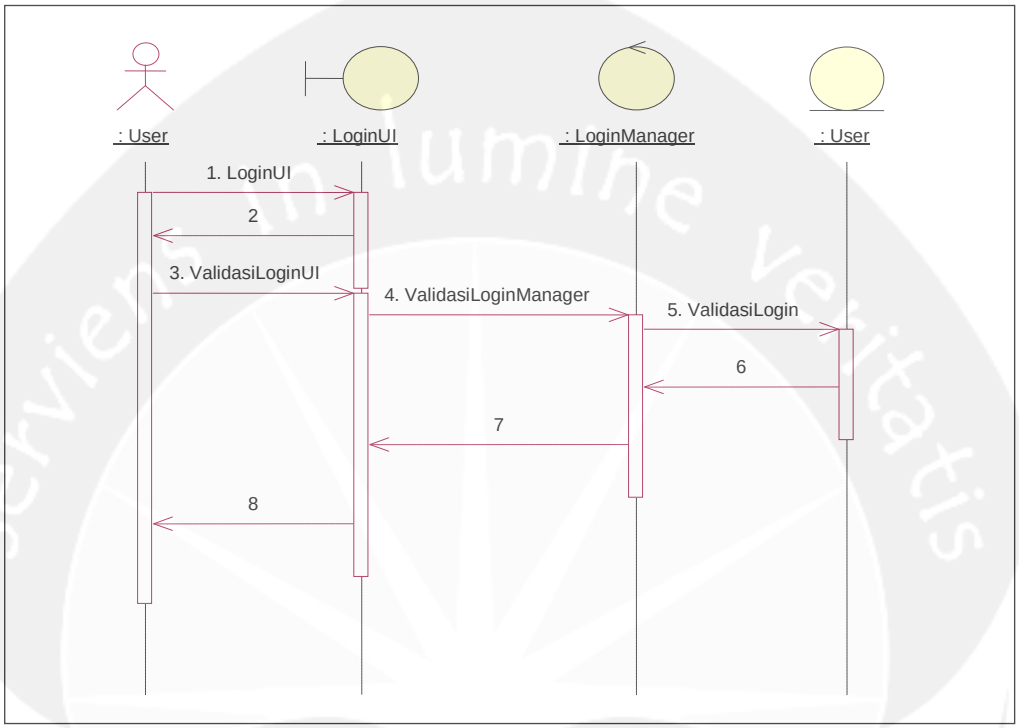
Gambar 2.1 Rancangan Arsitektur Perkasa



1.6 *Perancangan Rinci*

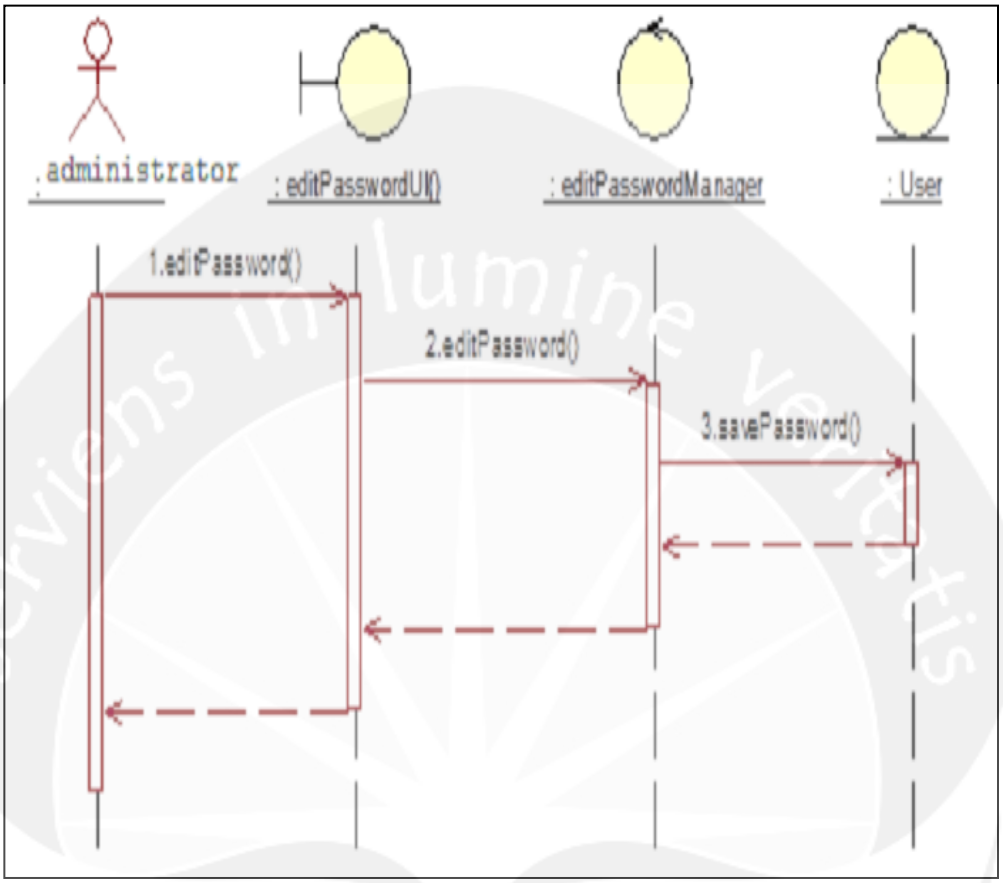
1.6.1 Sequence Diagram

2.2.1.1 Login User



Gambar 2.1 Sequence Diagram: Login User

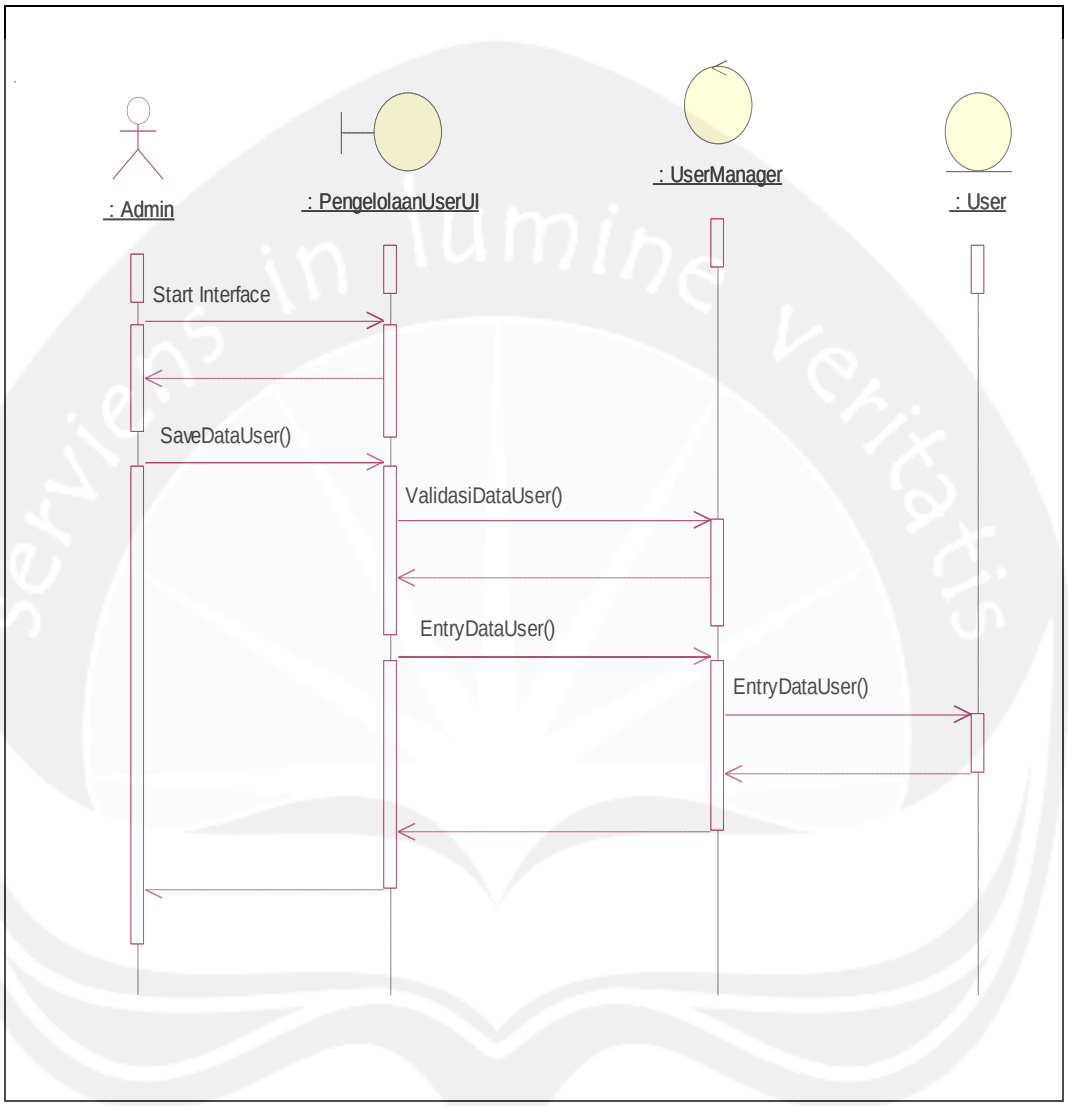
2.2.1.2 Edit Password



Gambar 2.2 Sequence Diagram: Edit Password User

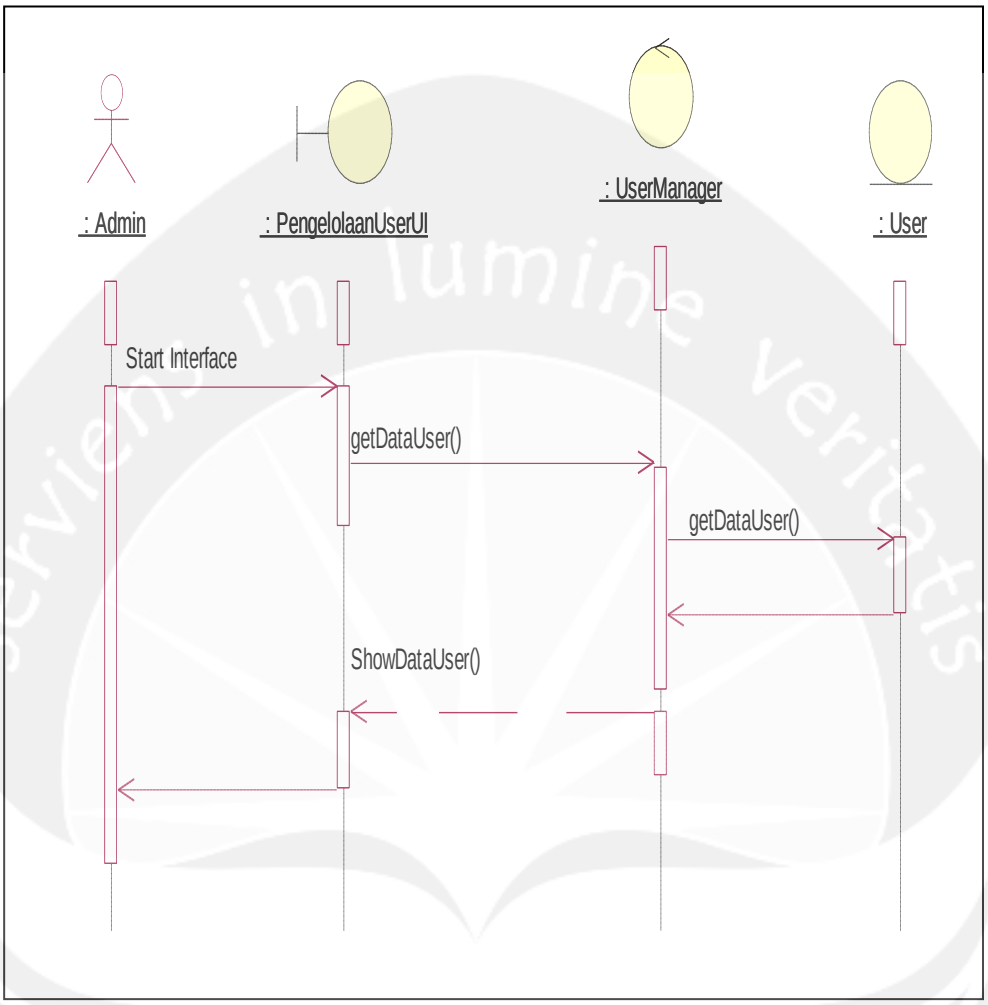
2.2.1.2 Pengelolaan Data User

2.2.1.2.1 Input Data User



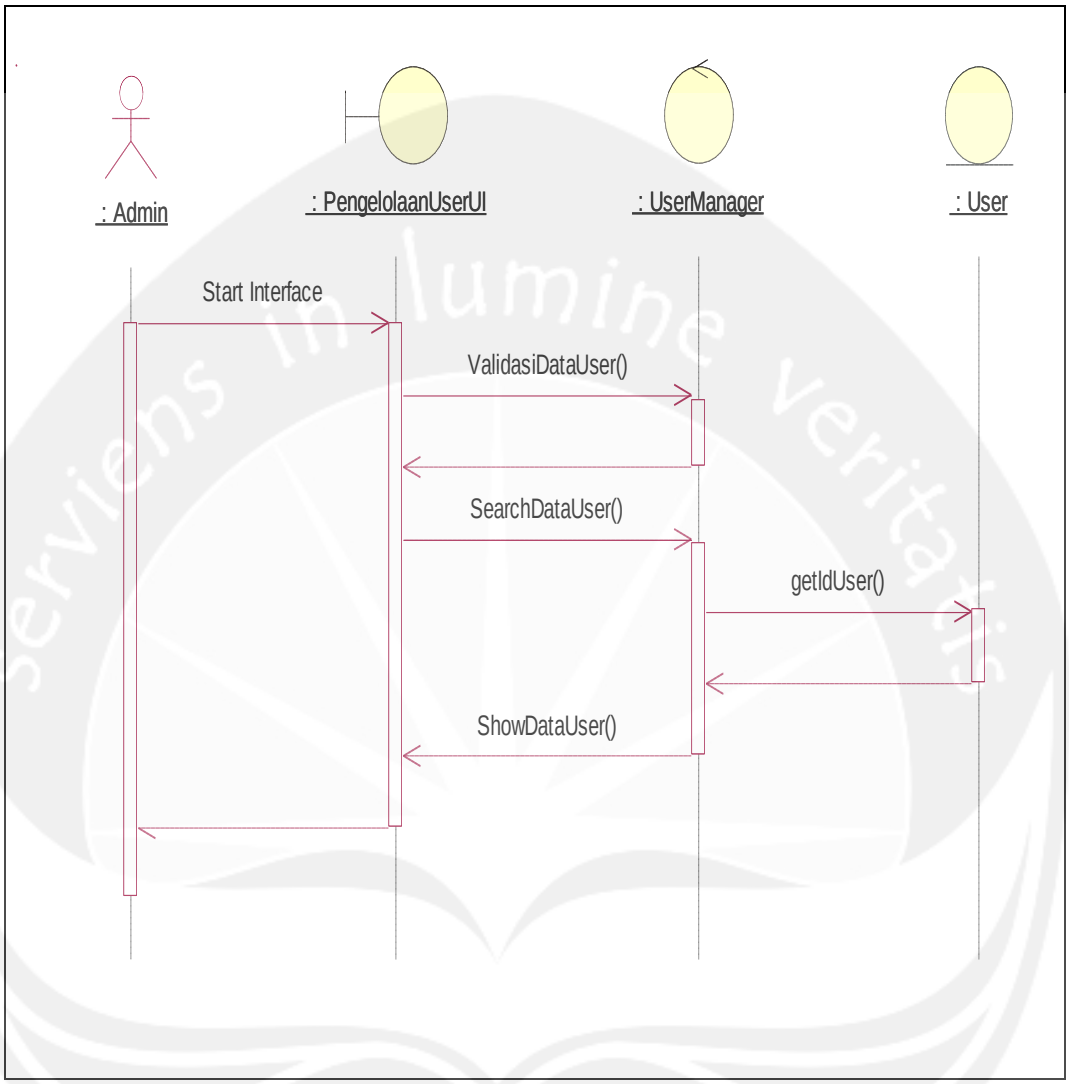
Gambar 2.3 Sequence Diagram: Input Data User

2.2.1.2.2 Display Data User



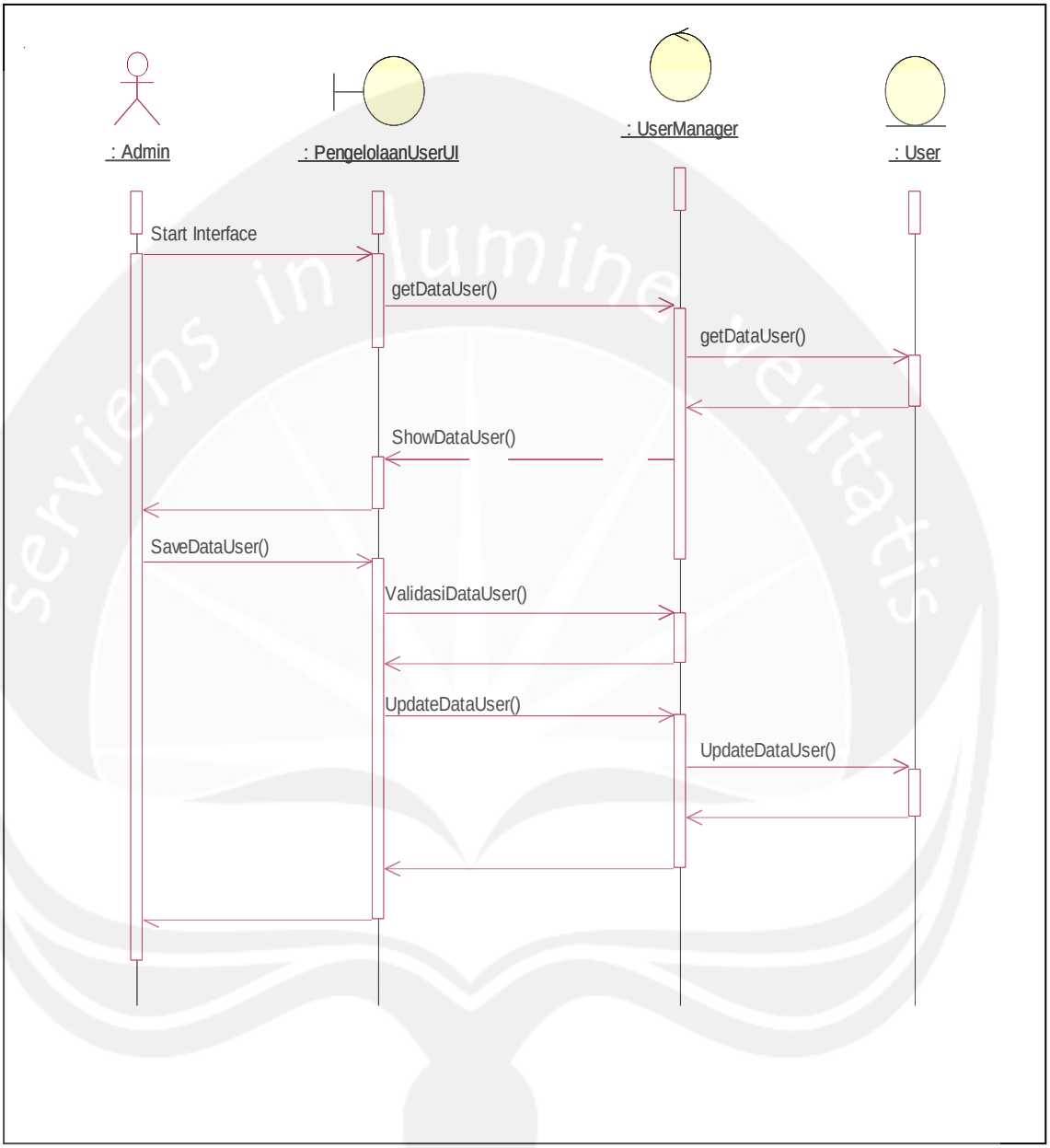
Gambar 2.4 Sequence Diagram: Display Data User

2.2.1.2.3 Search Data User



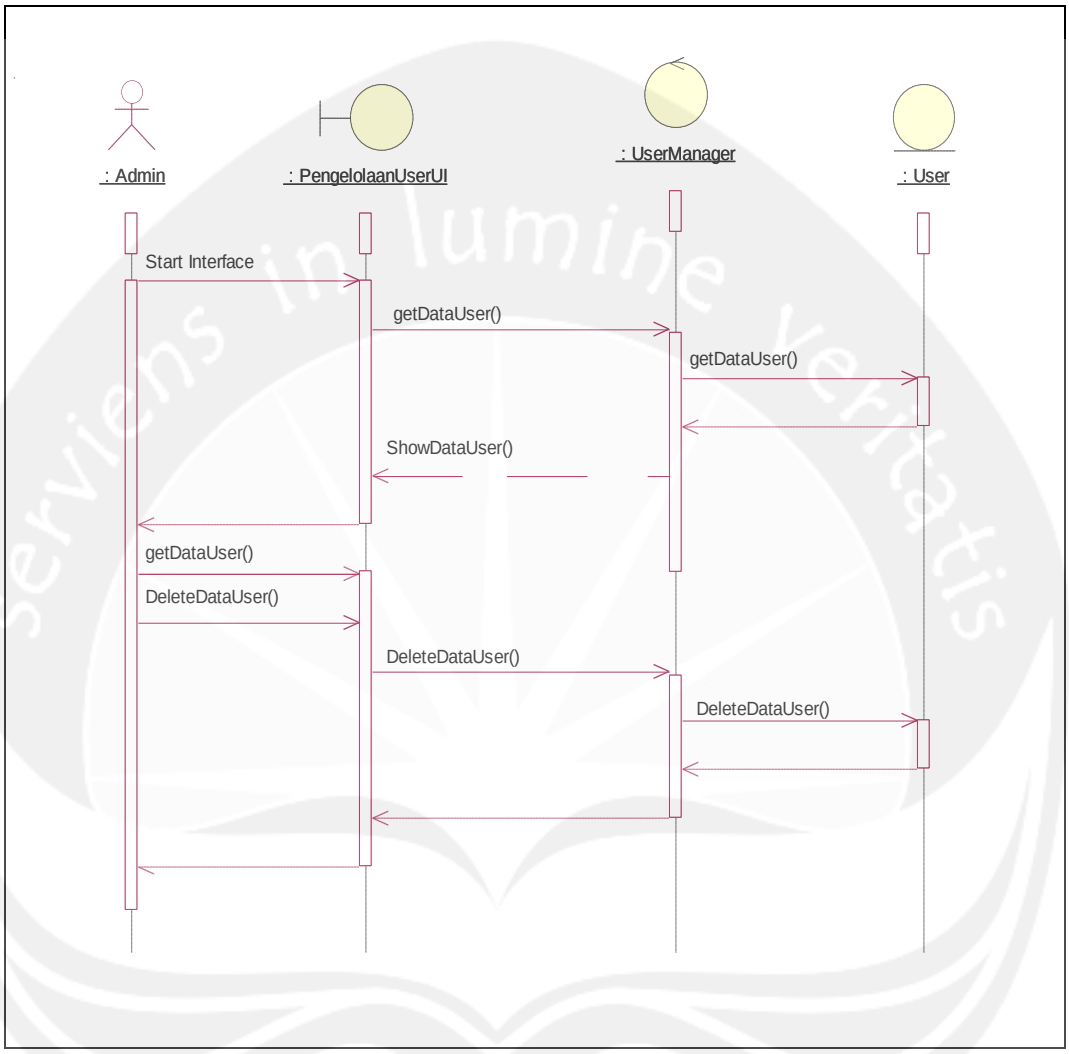
Gambar 2.5 Sequence Diagram: Search Data User

2.2.1.2.4 Update Data User



Gambar 2.6 Sequence Diagram: Update Data User

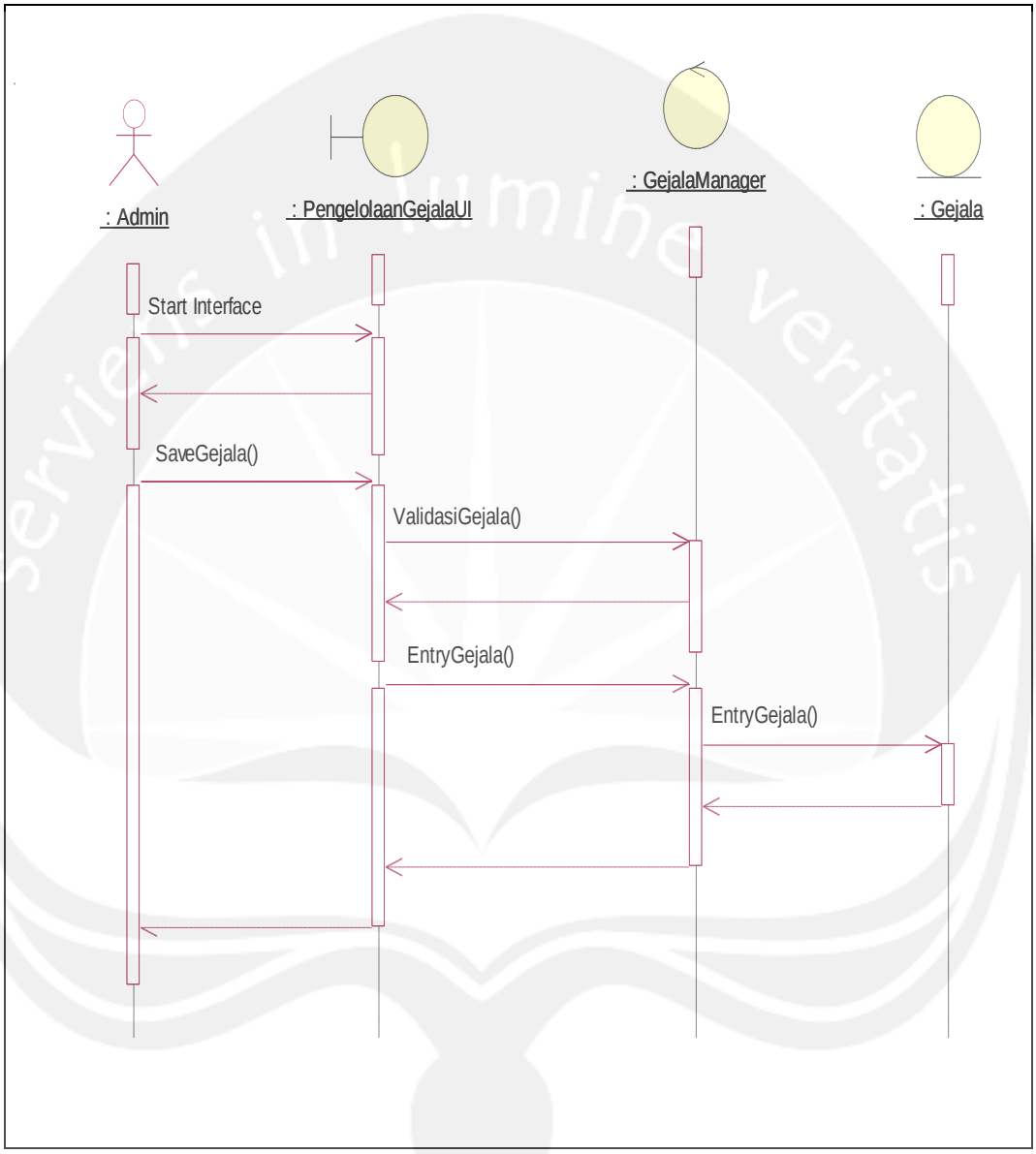
2.2.1.2.5 Delete Data User



Gambar 2.7 Sequence Diagram: Delete Data User

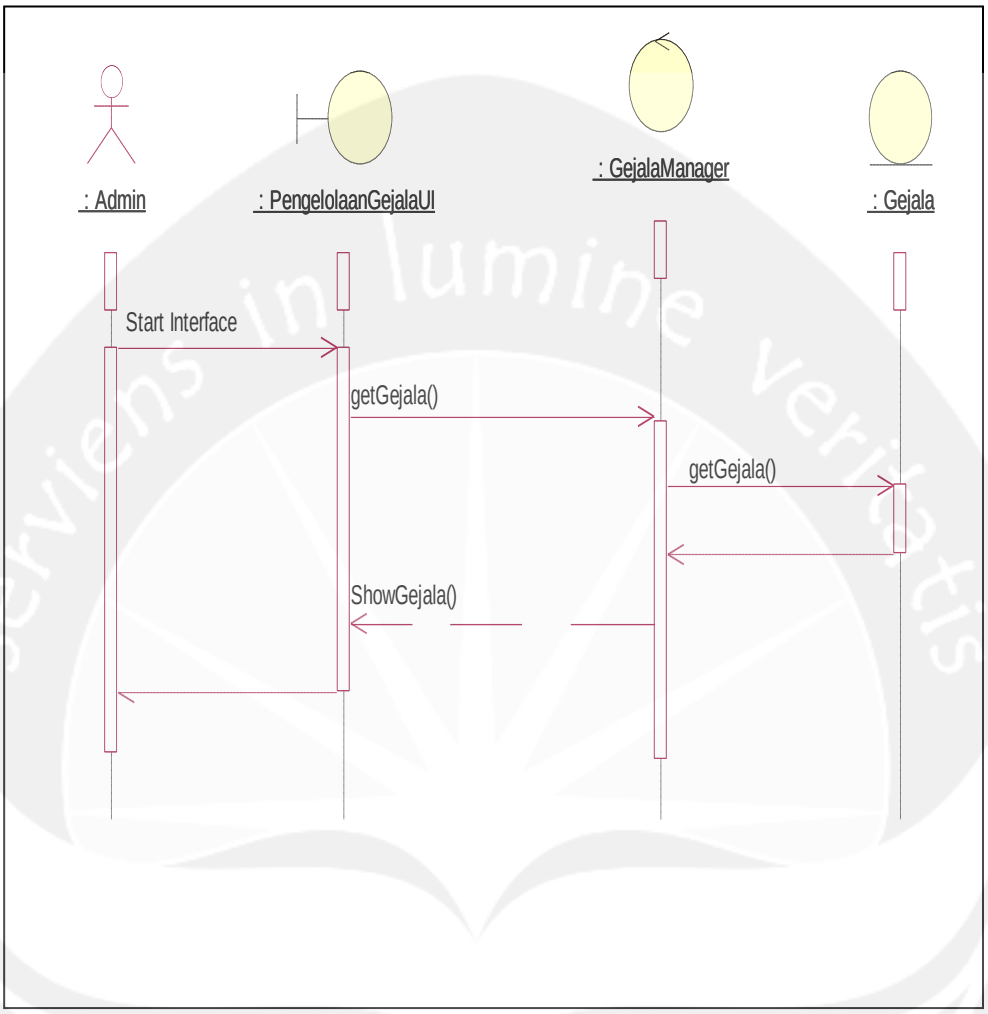
2.2.1.3 Pengelolaan Gejala

2.2.1.3.1 Input Gejala



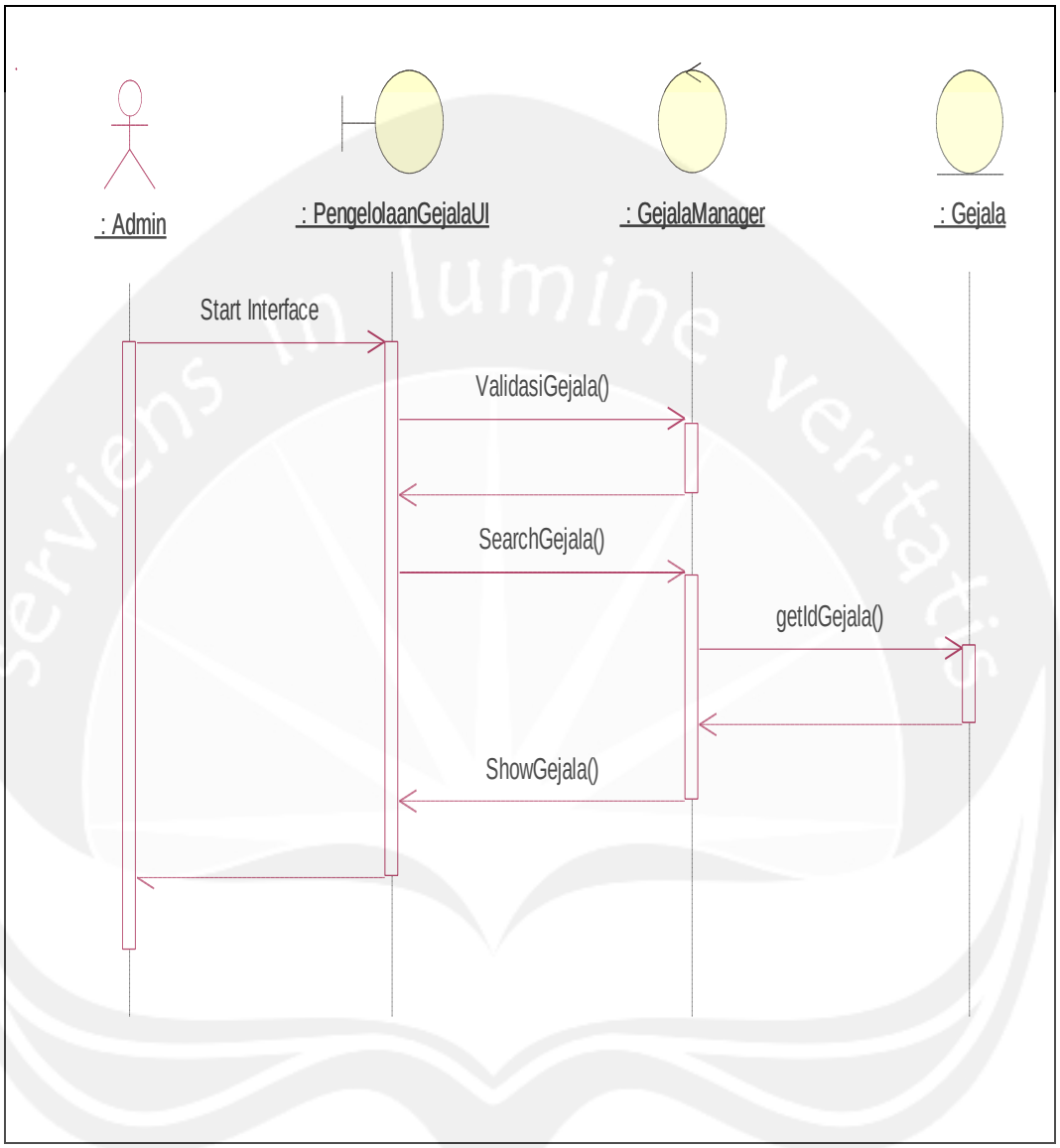
Gambar 2.8 Sequence Diagram: Input Gejala

2.2.1.3.2 Display Gejala



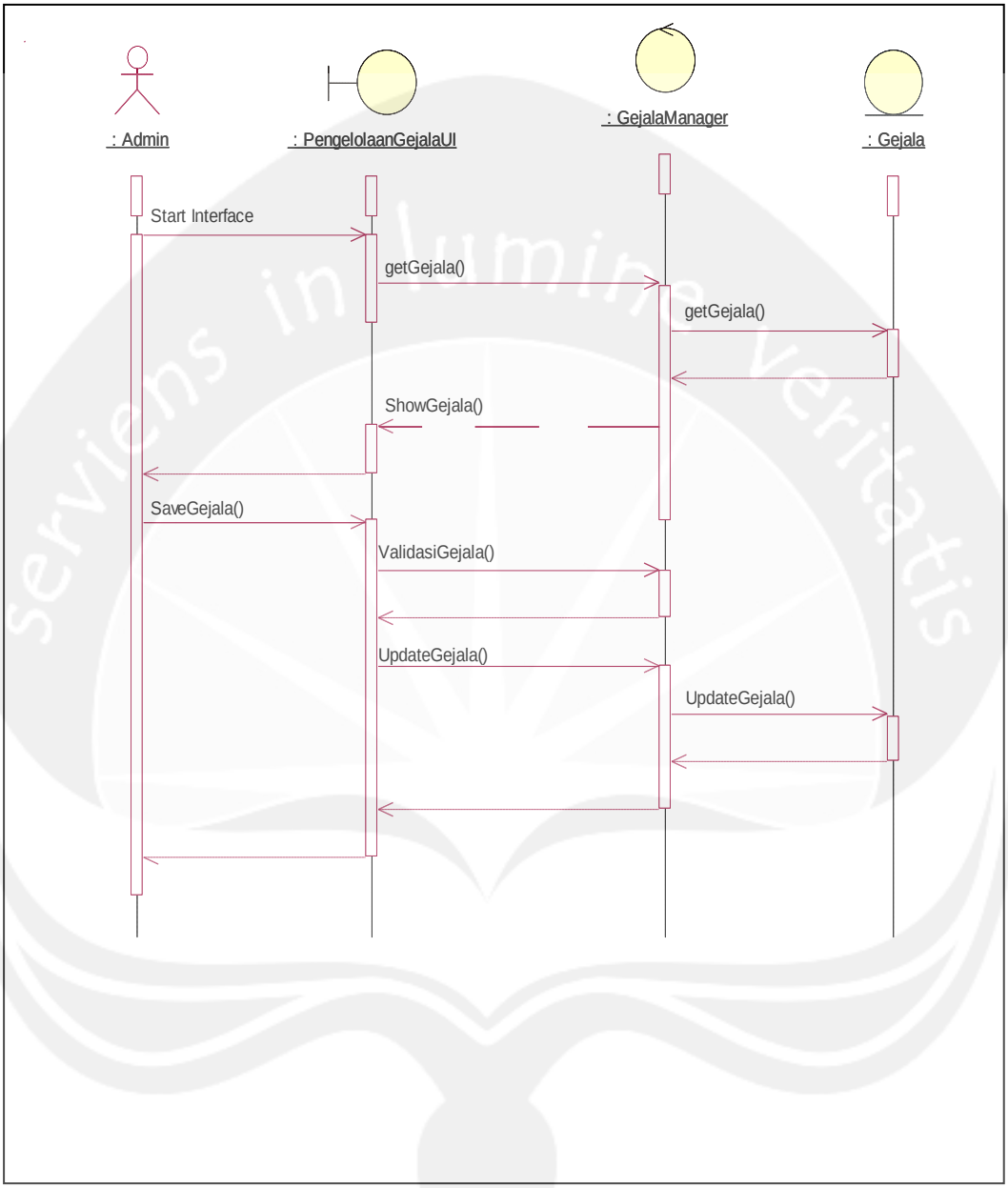
Gambar 2.9 Sequence Diagram: Display Gejala

2.2.1.3.3 Search Gejala



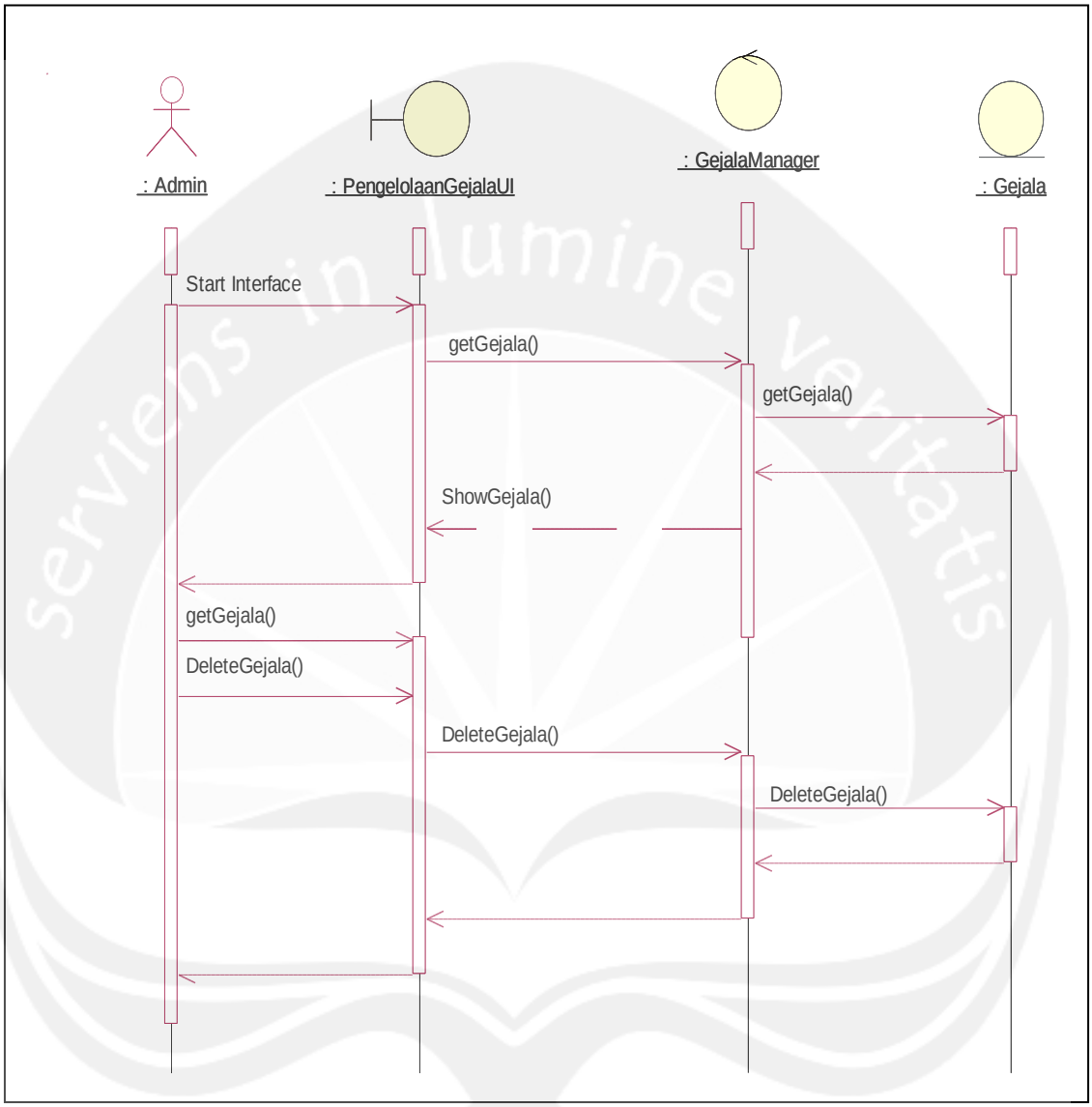
Gambar 2.10 Sequence Diagram: Search Gejala

2.2.1.3.4 Update Gejala



Gambar 2.11 Sequence Diagram: Update Gejala

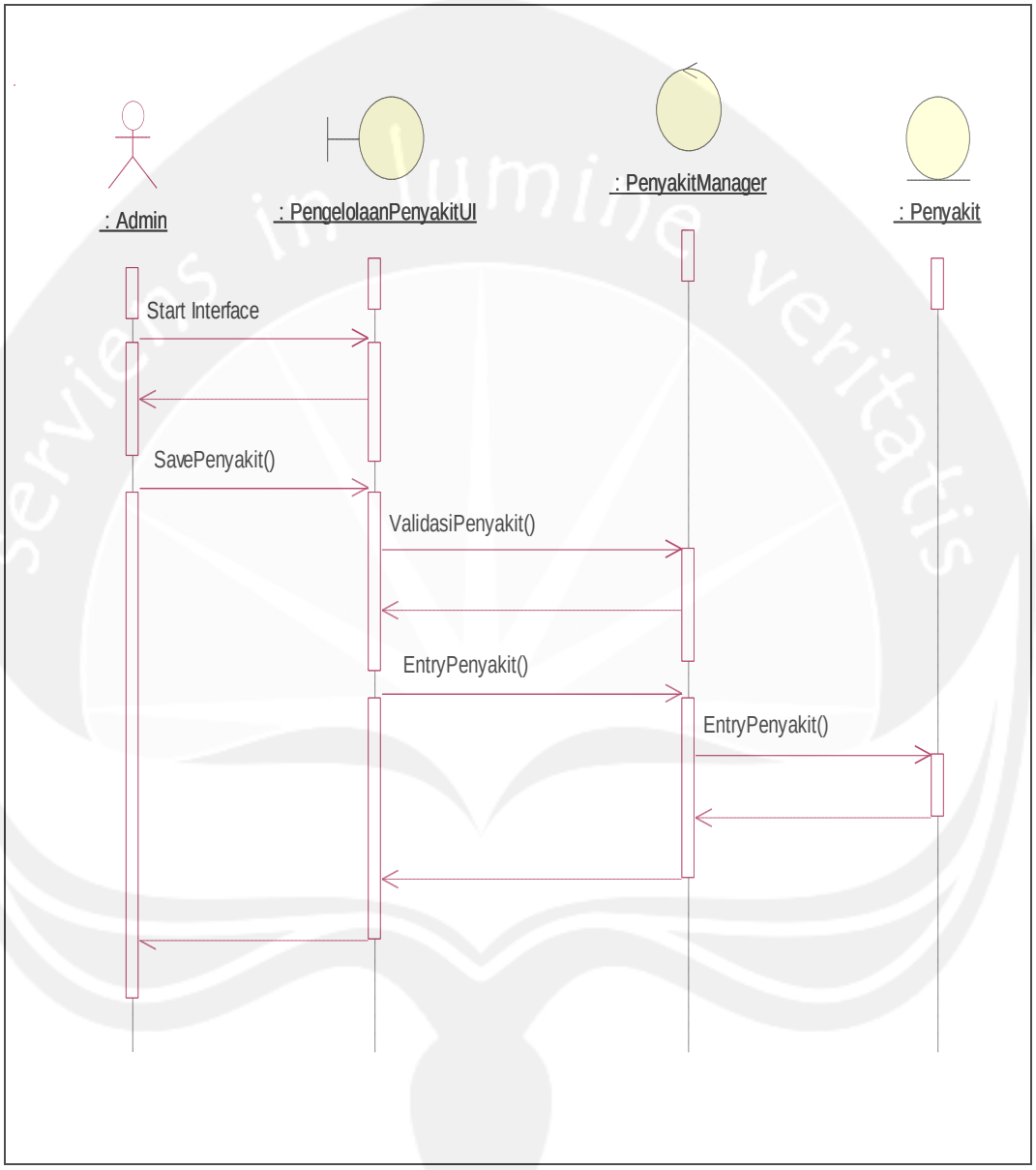
2.2.1.3.5 Delete Gejala



Gambar 2.12 Sequence Diagram: Delete Gejala

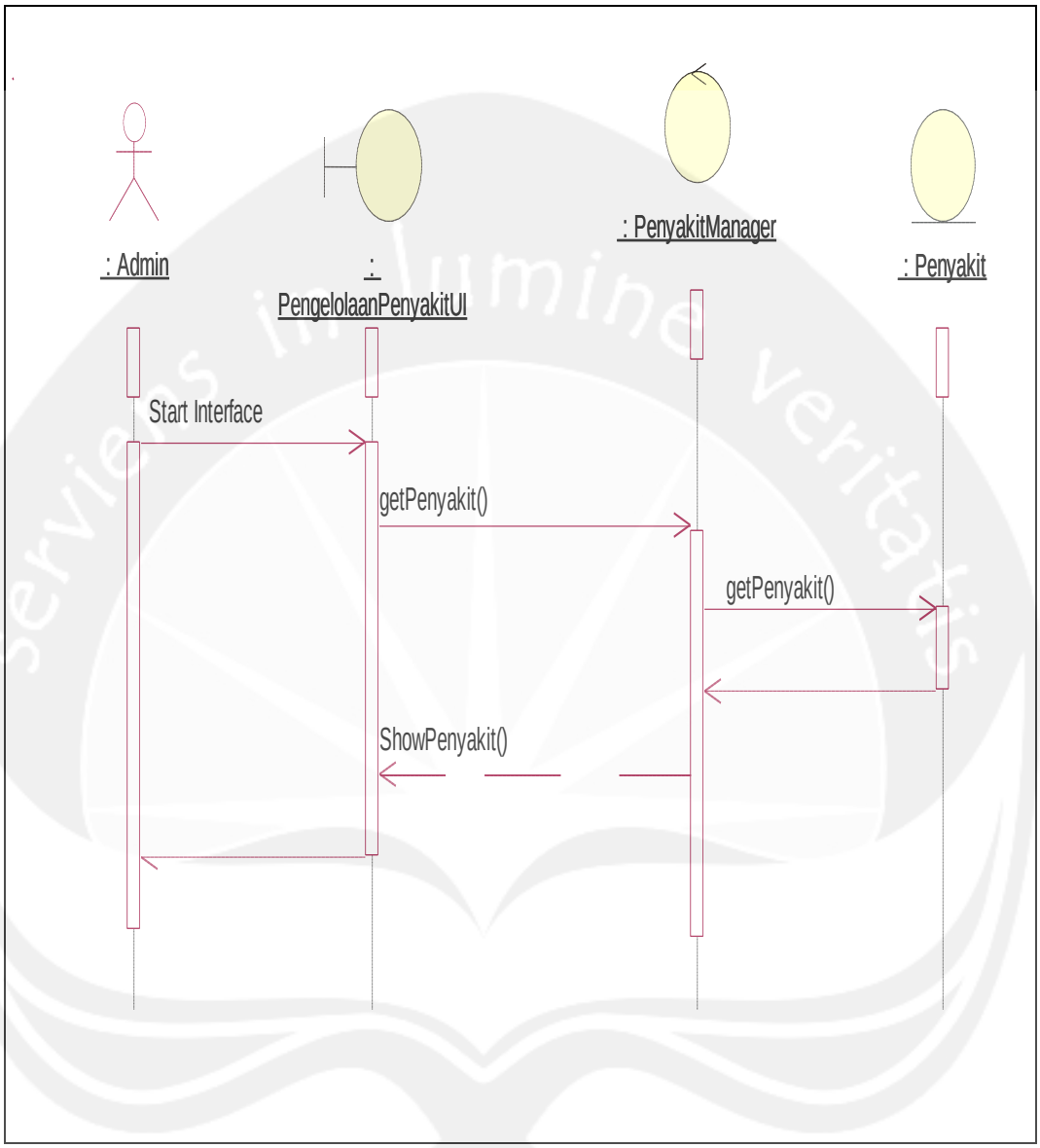
2.2.1.4 Pengelolaan Penyakit

2.2.1.4.1 Input Penyakit



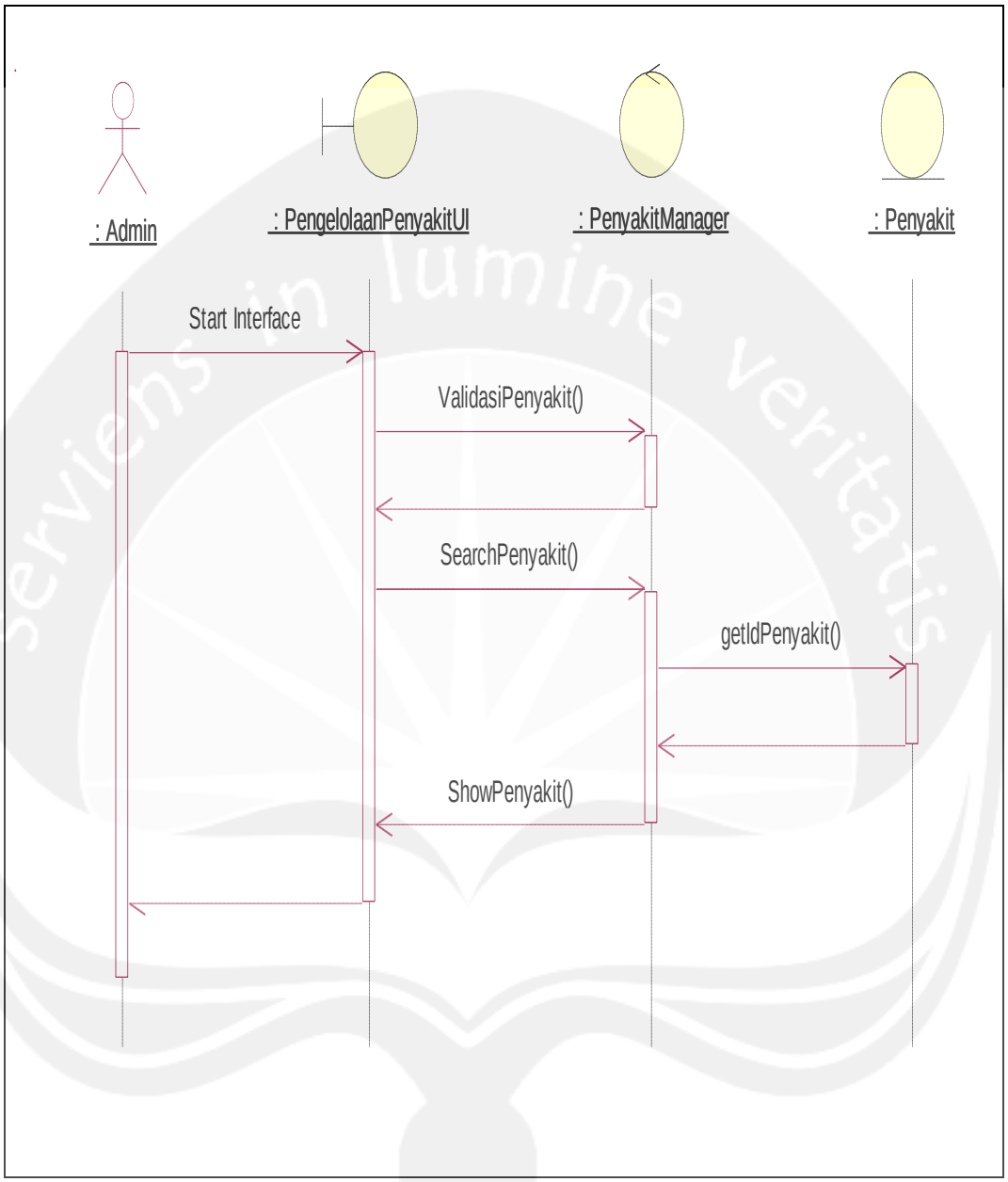
Gambar 2.13 Sequence Diagram: Input Penyakit

2.2.1.4.2 Display Penyakit



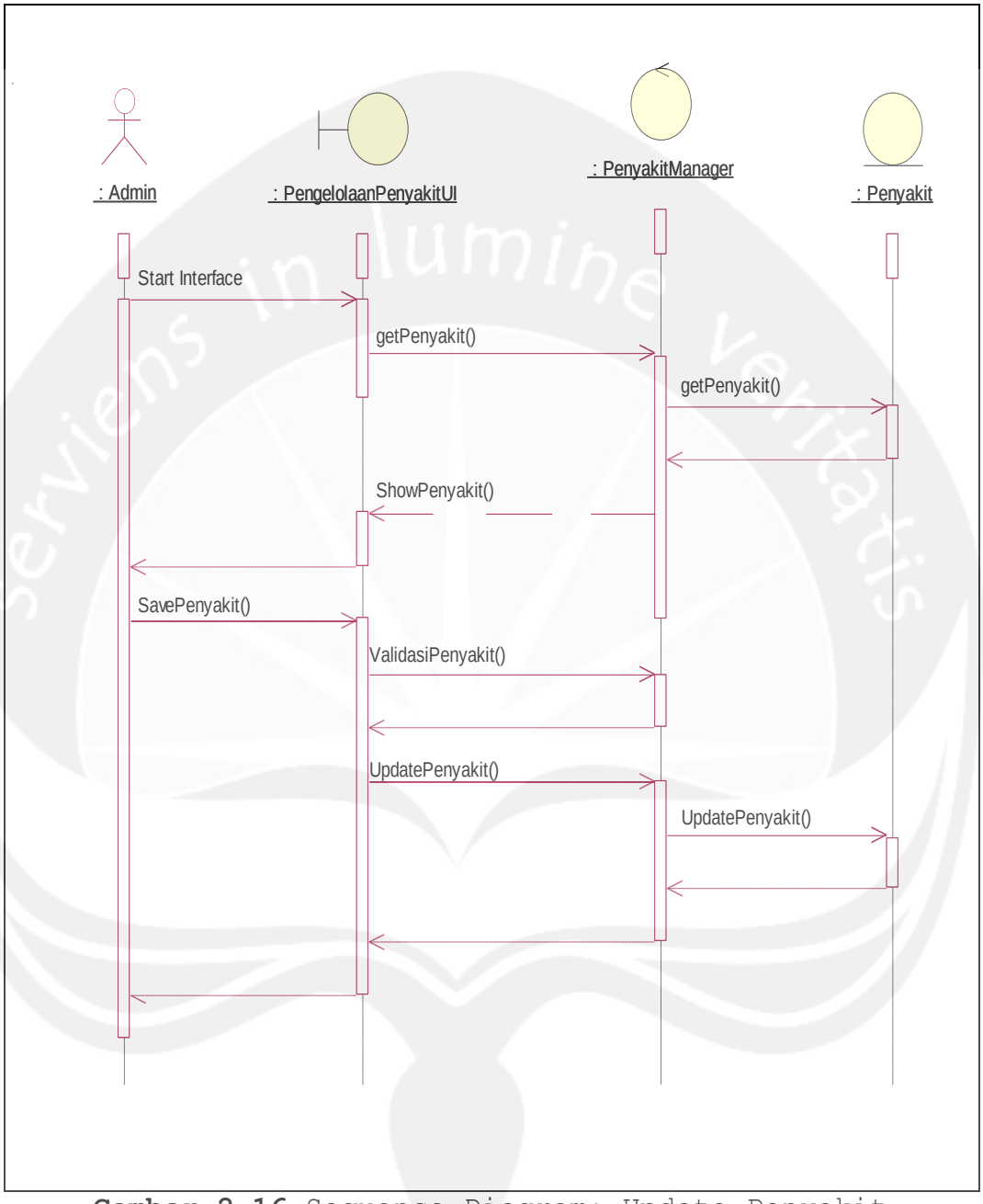
Gambar 2.14 Sequence Diagram: Display Penyakit

2.2.1.4.3 Search Penyakit



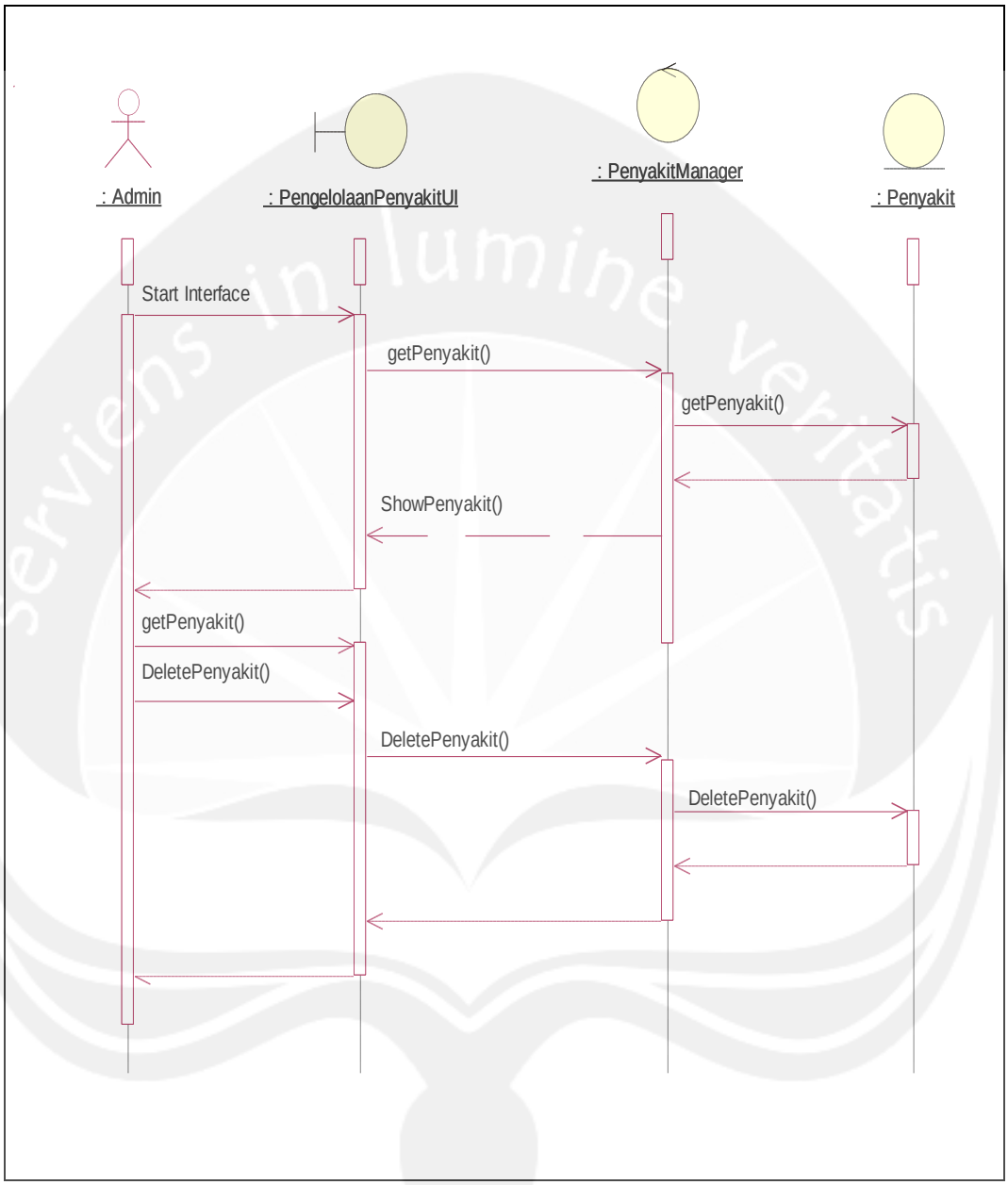
Gambar 2.15 Sequence Diagram: Search Penyakit

2.2.1.4.4 Update Penyakit



Gambar 2.16 Sequence Diagram: Update Penyakit

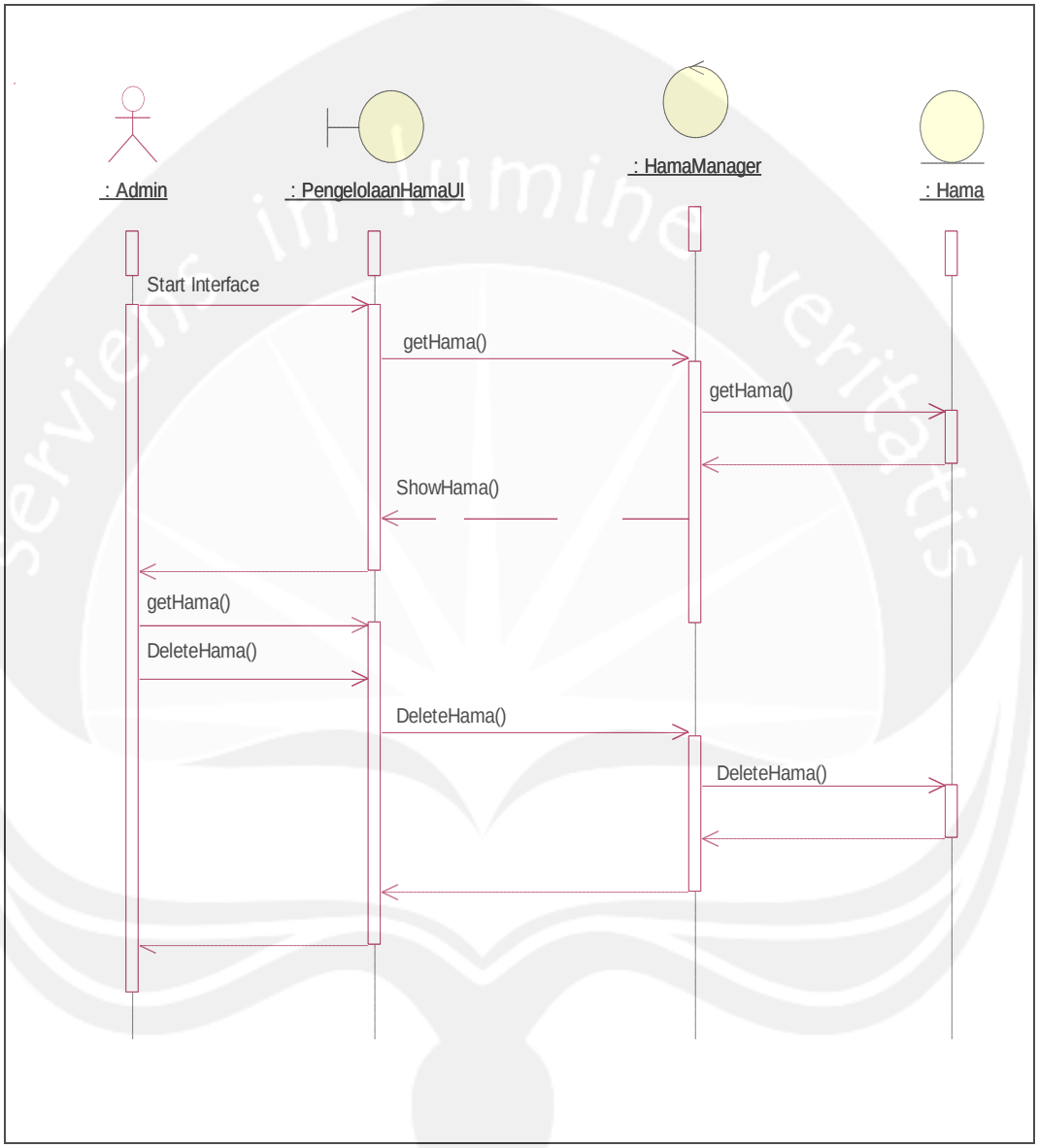
2.2.1.4.5 Delete Penyakit



Gambar 2.17 Sequence Diagram: Delete Penyakit

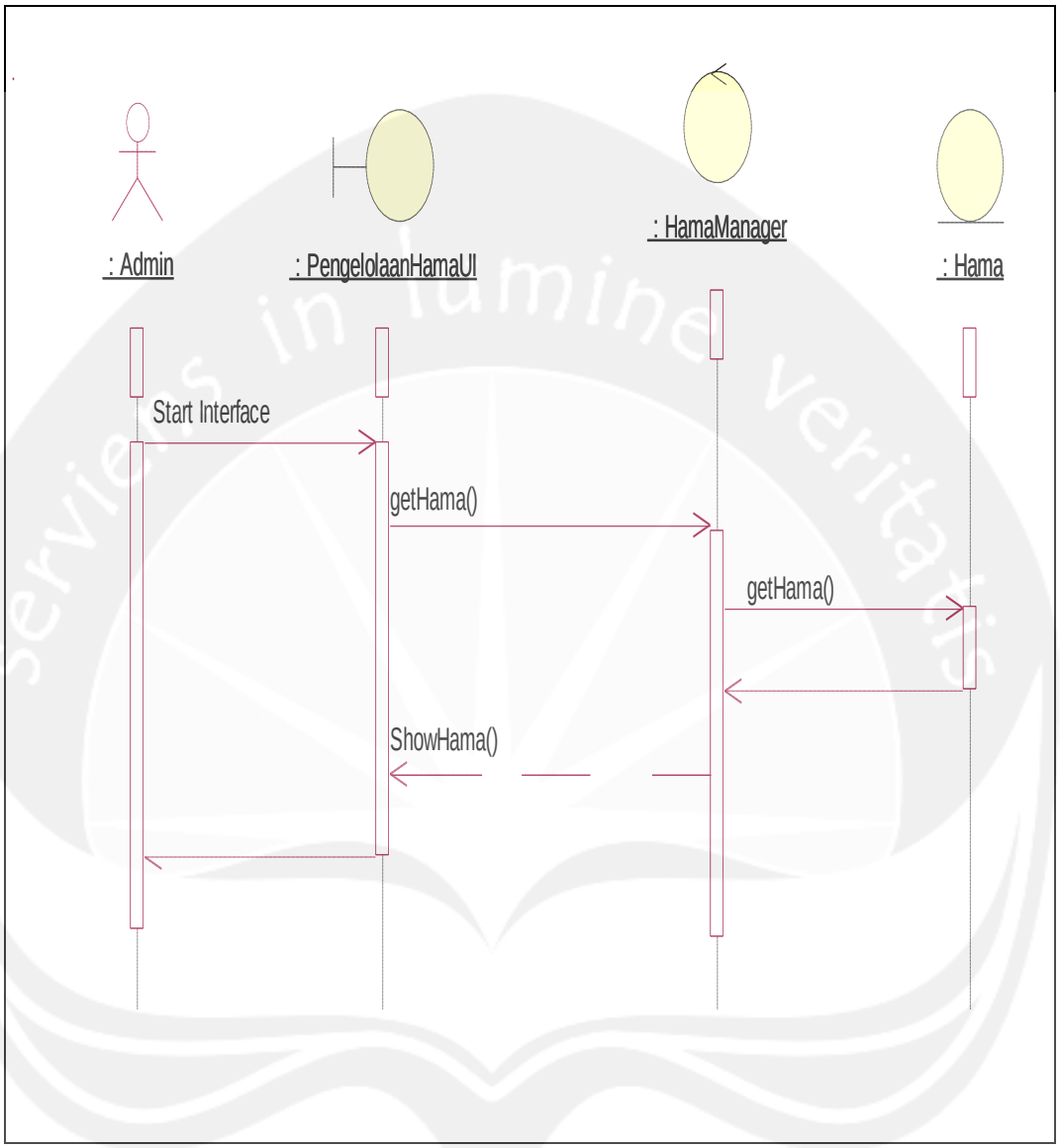
2.2.1.5 Pengelolaan Hama

2.2.1.5.1 Input Hama



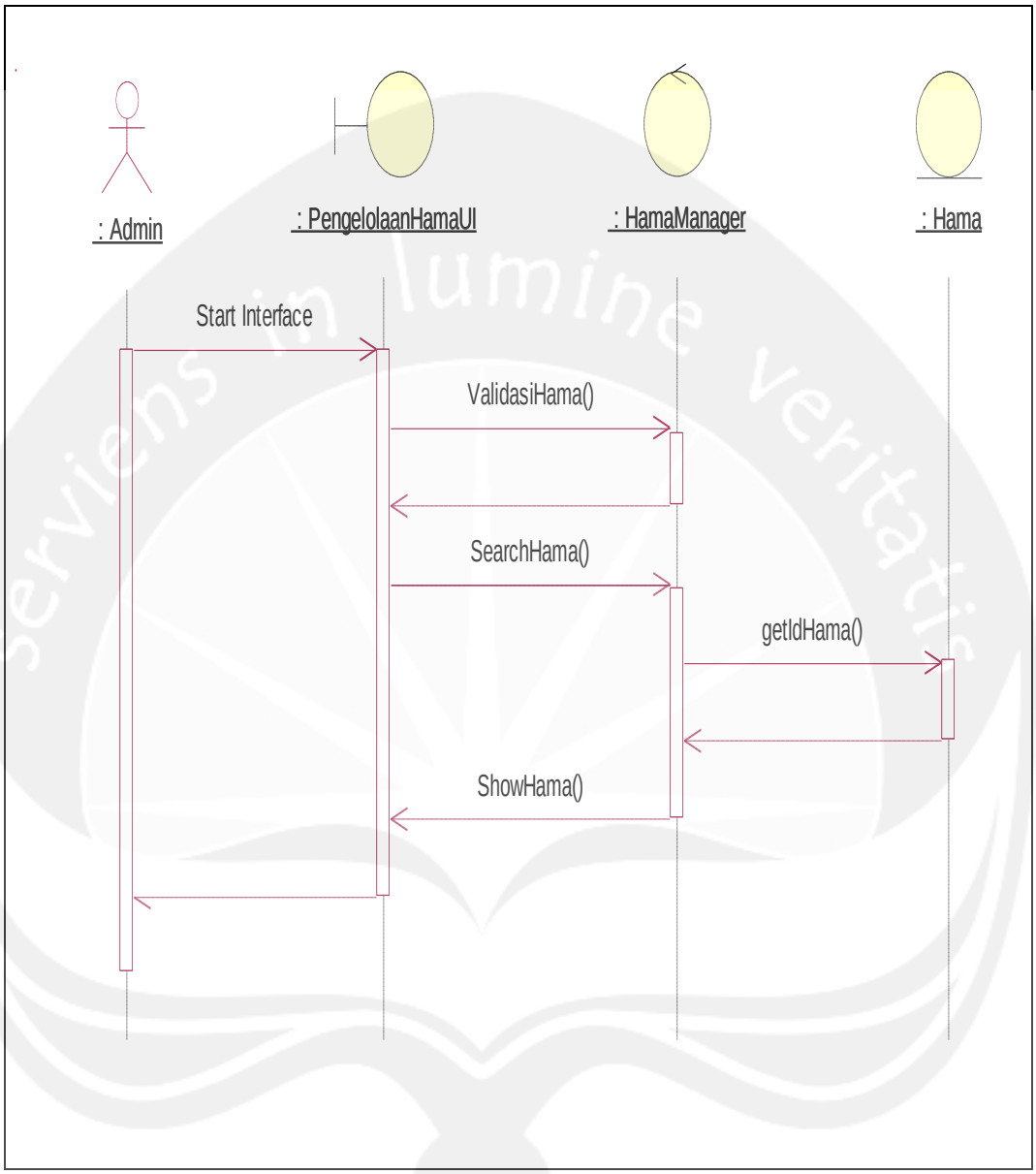
Gambar 2.18 Sequence Diagram: Input Hama

2.2.1.5.2 Display Hama



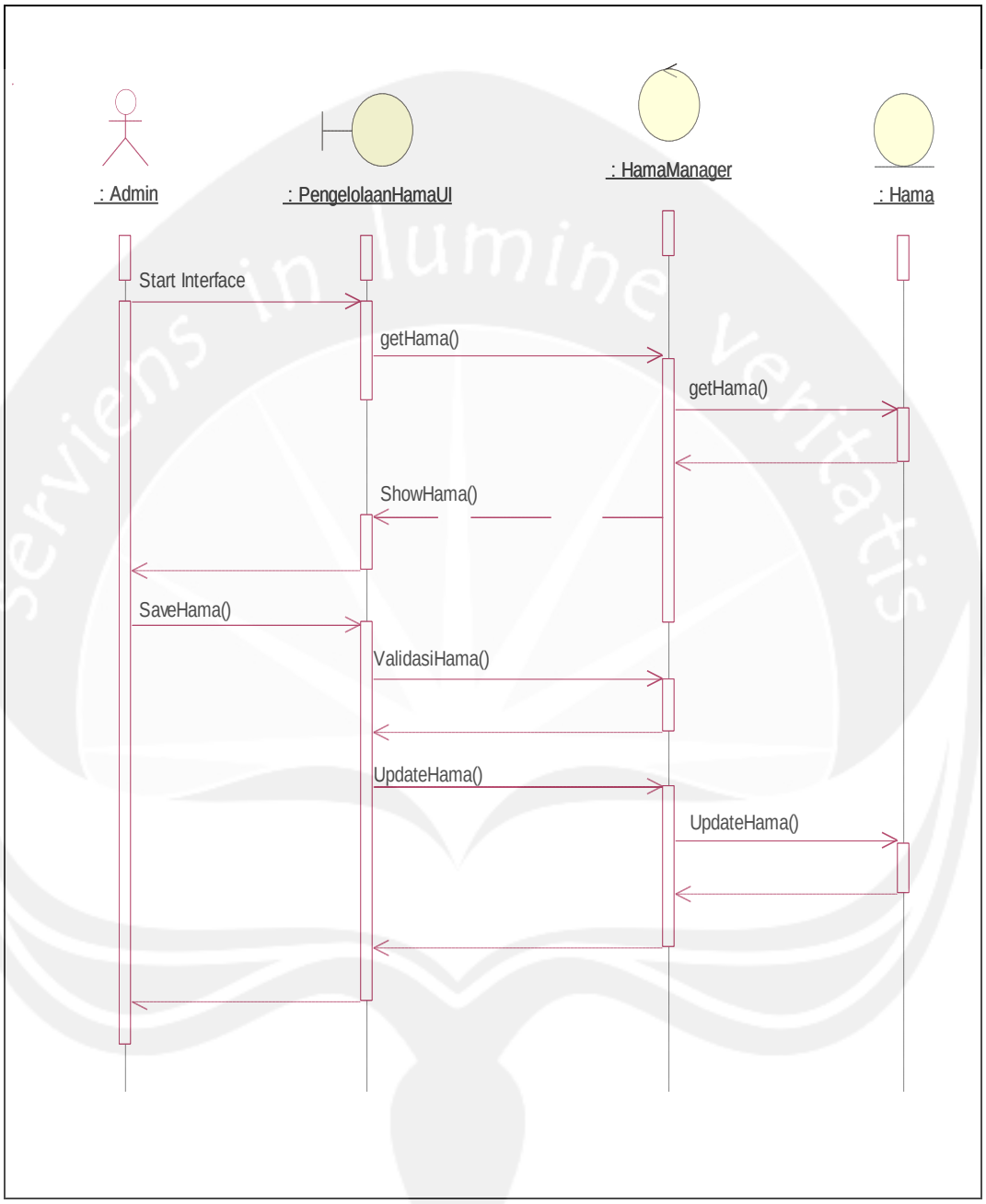
Gambar 2.19 Sequence Diagram: Display Hama

2.2.1.5.3 Search Hama



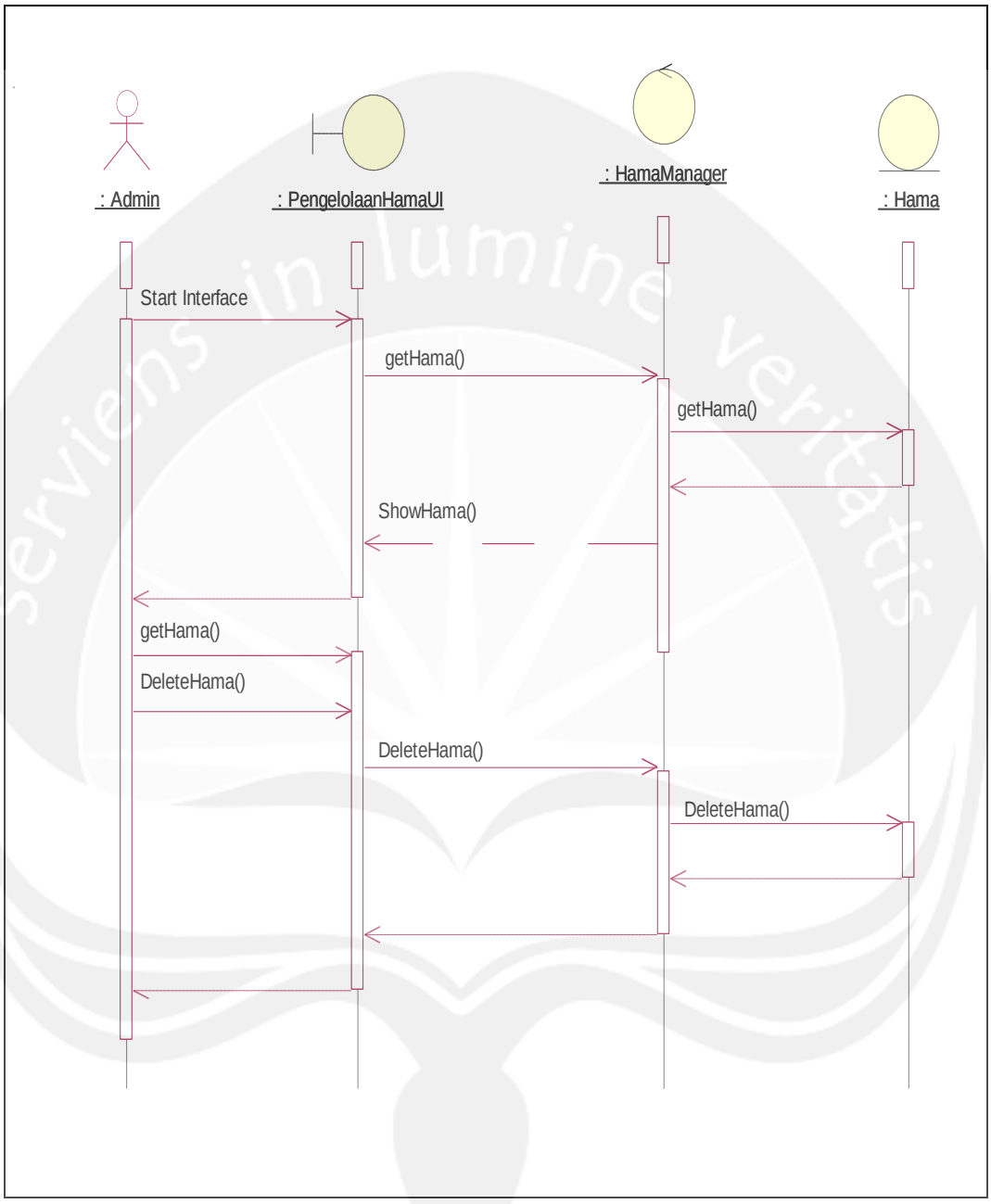
Gambar 2.20 Sequence Diagram: Search Hama

2.2.1.5.4 Update Hama



Gambar 2.21 Sequence Diagram: Update Hama

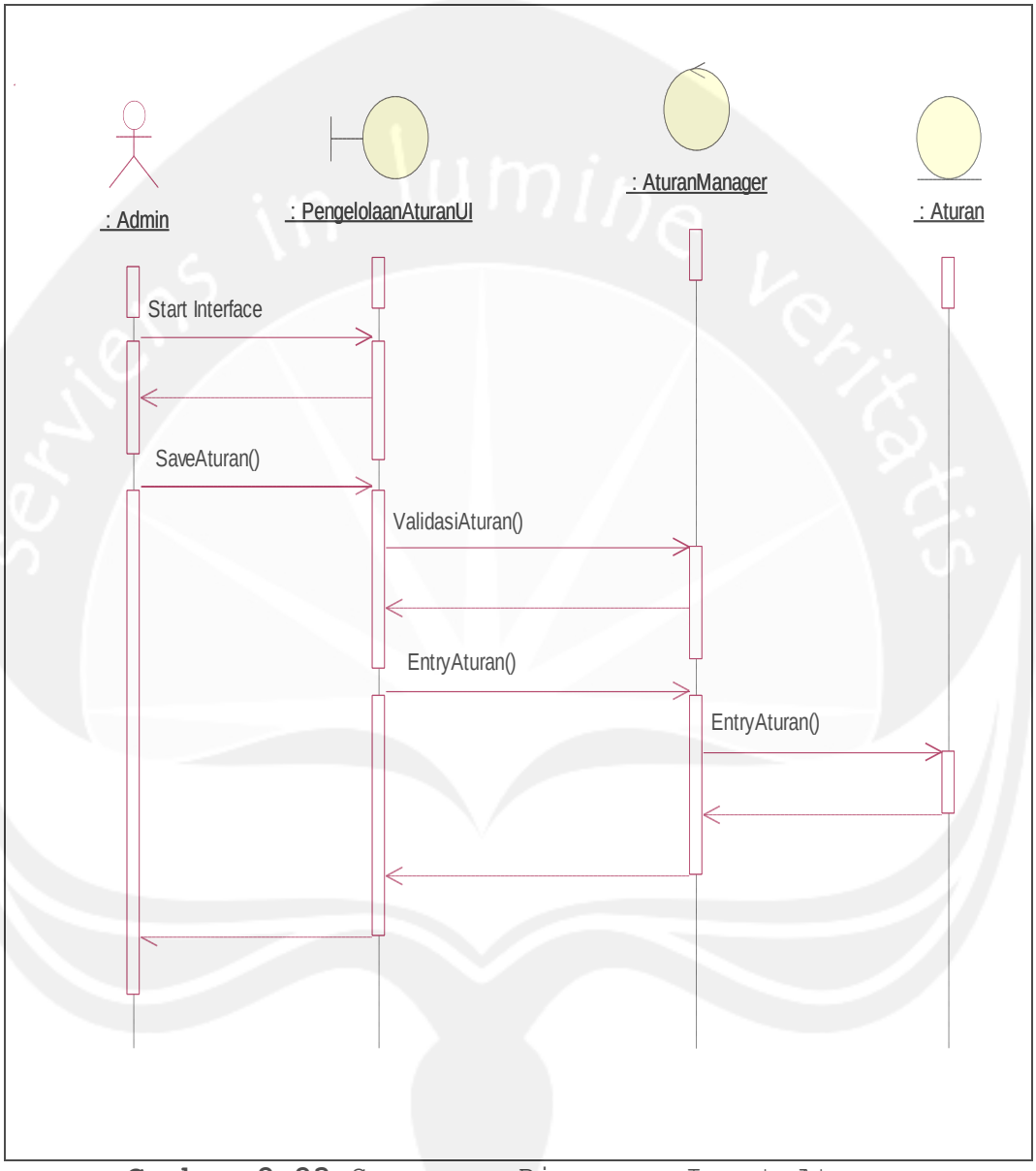
2.2.1.5.5 Delete Hama



Gambar 2.22 Sequence Diagram: Delete Hama

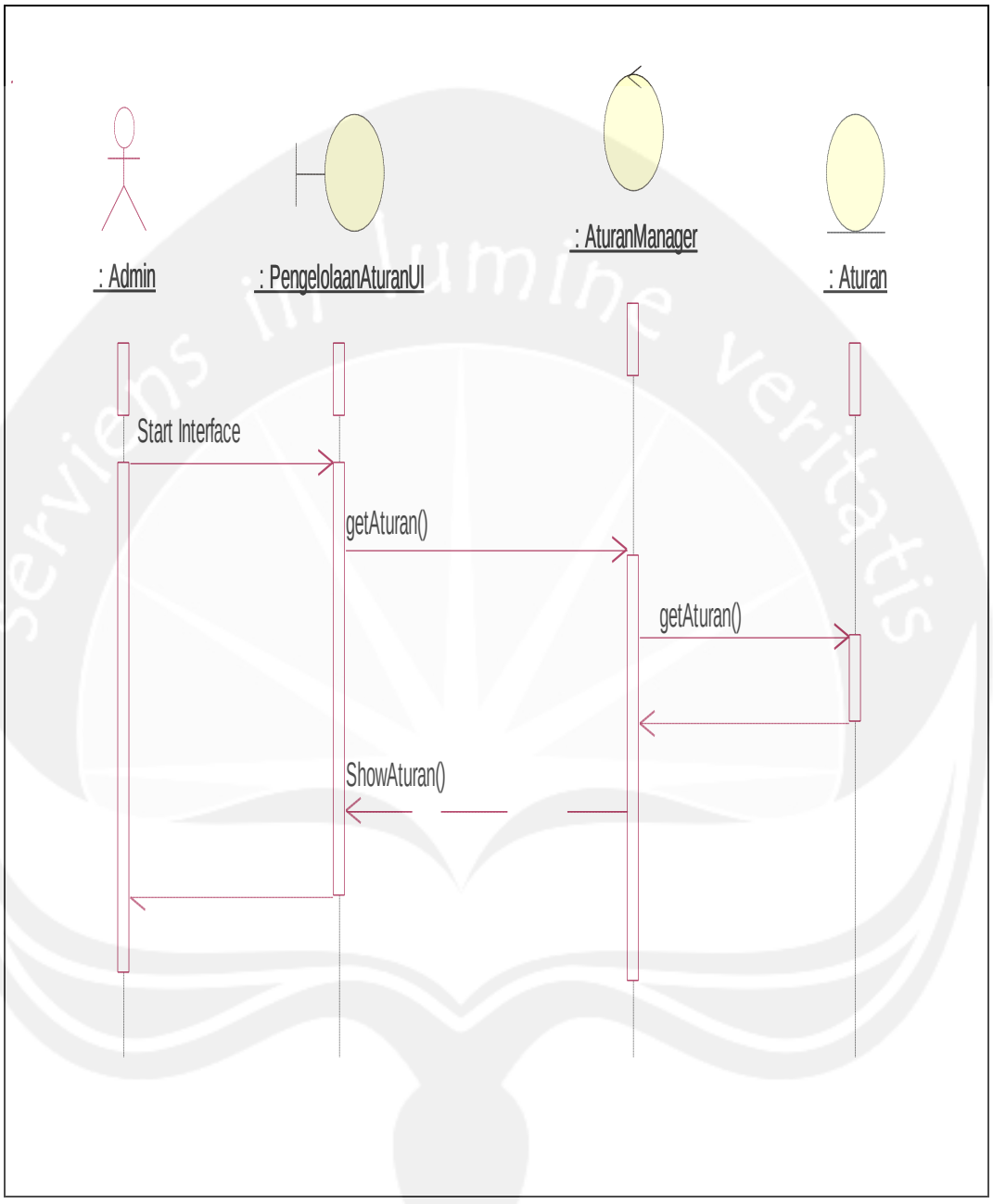
2.2.1.6 Pengelolaan Aturan

2.2.1.6.1 Input Aturan



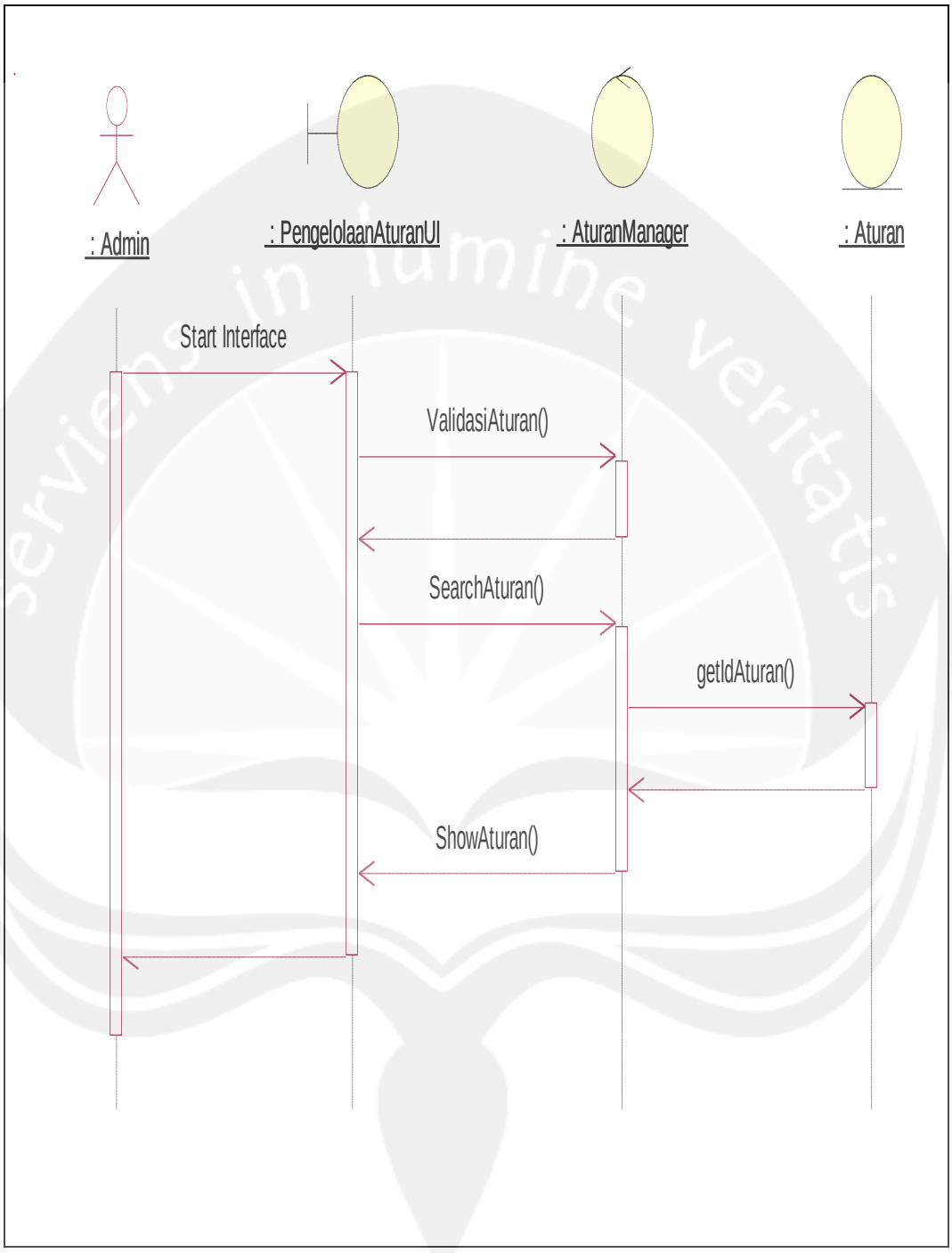
Gambar 2.23 Sequence Diagram: Input Aturan

2.2.1.4.2 Display Aturan



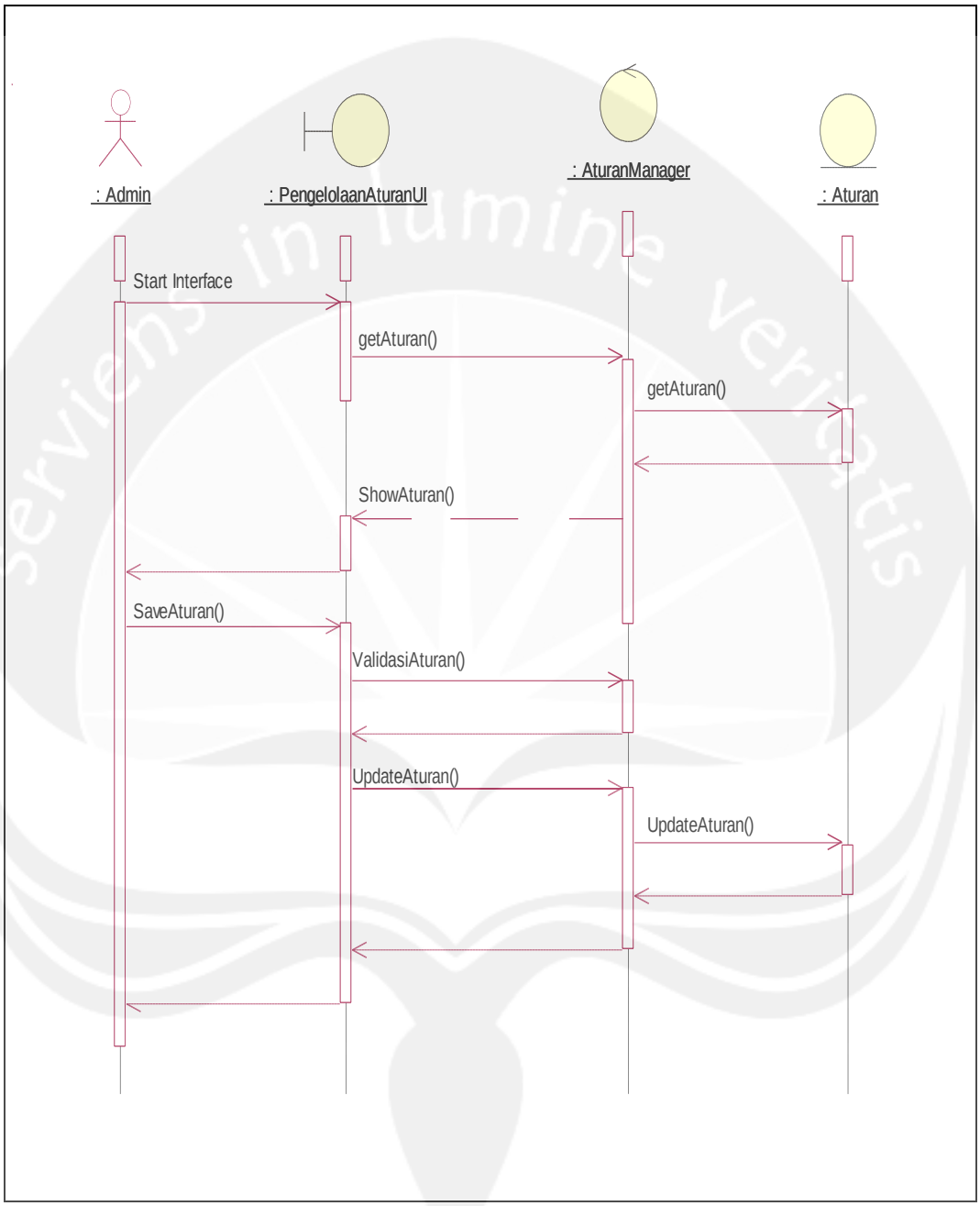
Gambar 2.24 Sequence Diagram: Display Aturan

2.2.1.4.3 Search Aturan



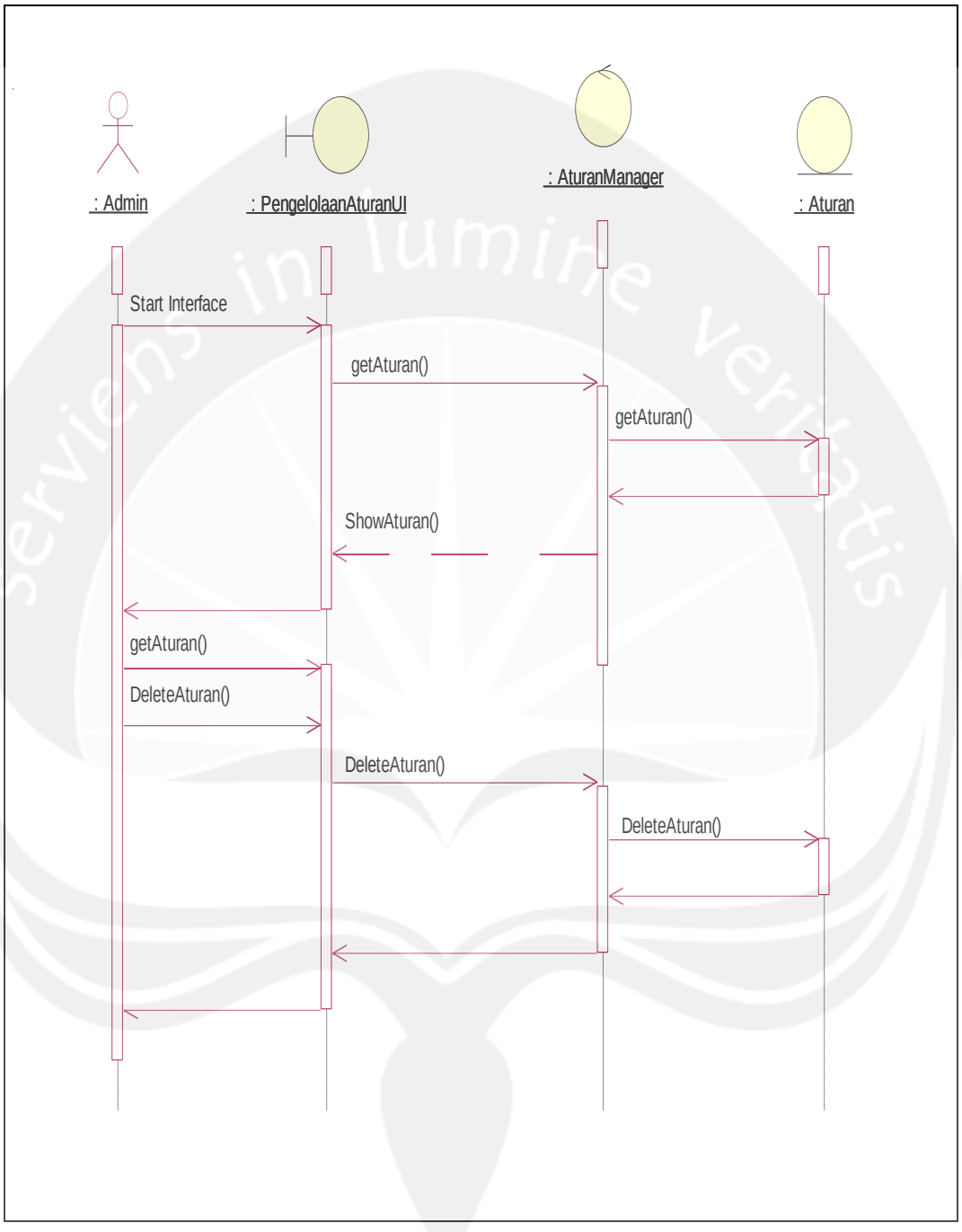
Gambar 2.25 Sequence Diagram: Search Aturan

2.2.1.4.4 Update Aturan



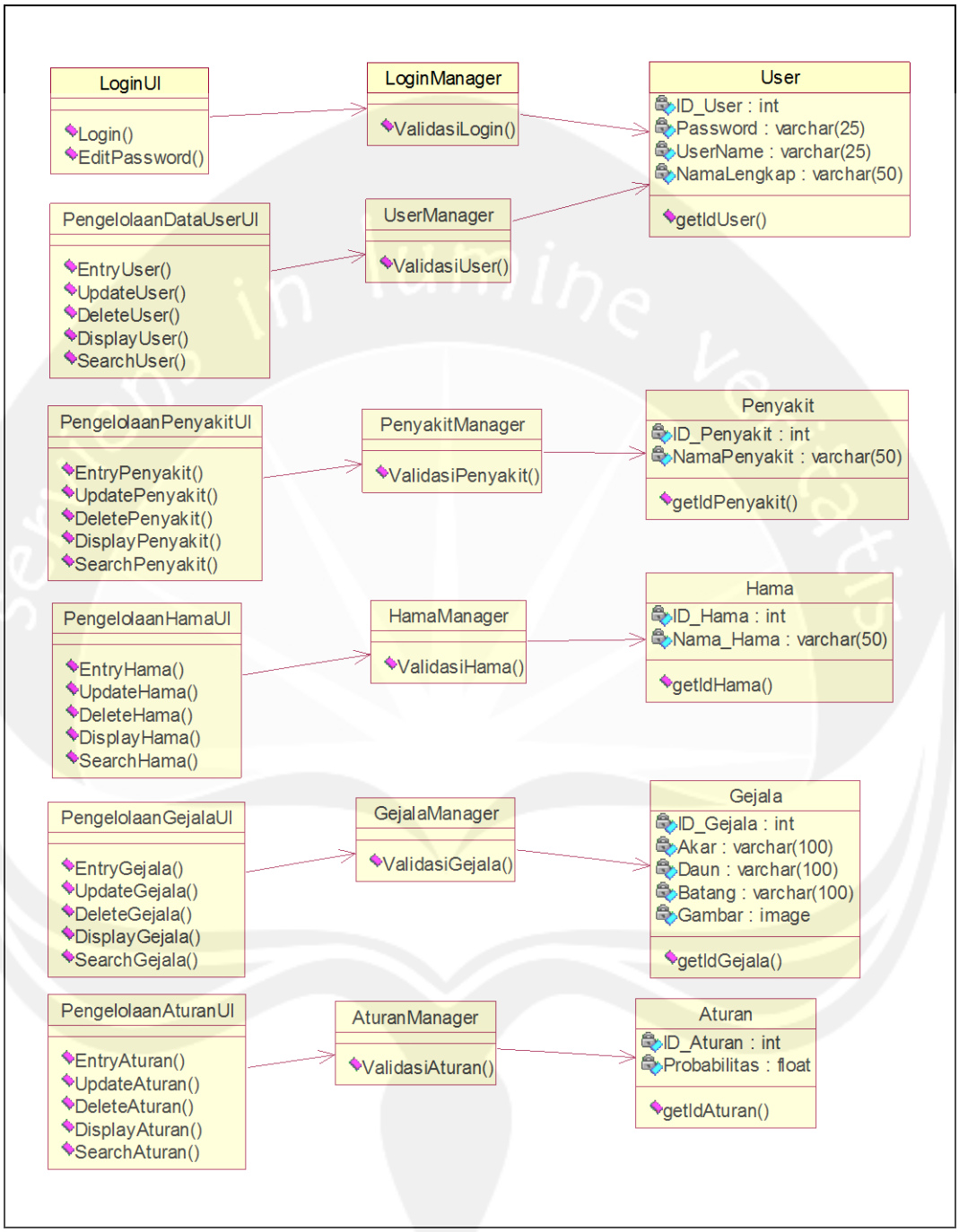
Gambar 2.26 Sequence Diagram: Update Aturan

2.2.1.4.5 Delete Aturan



Gambar 2.27 Sequence Diagram: Delete Aturan

1.6.2 Class Diagram



Gambar 2.28 Class Diagram Perkasa

2.2.3 Spesifikasi Deskripsi Kelas Diagram

Spesifikasi Design Kelas LoginUI

LoginUI	**<<boundary>>**
+LoginUI() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. +inputLogin(username,password) Operasi ini digunakan untuk mengambil data login yang diinputkan oleh user, yaitu username dan password.	

Spesifikasi Design Kelas PengelolaanDataUserUI

PengelolaanUserUI	**<<boundary>>**
+PengelolaanUserUI() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. +entryDataUser() Operasi ini digunakan untuk memasukkan data user. +validasiDataUser() :bool Operasi ini digunakan untuk melakukan validasi data user yang diinputkan +saveDataUser(userManager) :void Operasi ini digunakan untuk menyimpan data user kedalam database. +getDataUser() :void Operasi ini digunakan untuk mengambil data user dari database. +showDataUser(User) :void Operasi ini digunakan untuk menampilkan data user +editDataUser() Operasi ini digunakan untuk mengedit data user yang sudah ada di dalam database.	

+deleteDataUser() Operasi ini digunakan untuk menghapus data user yang ada didalam database. +displayDataUser() Operasi ini digunakan untuk menampilkan data user yang ada di dalam database. +searchDataUser() Operasi ini digunakan untuk mencari data user didalam database.

Spesifikasi Design Kelas PengelolaanGejalaUI

PengelolaanGejalaUI	**<<boundary>>**
+PengelolaanGejalaUI() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. +entryGejala() Operasi ini digunakan untuk memasukkan gejala. +validasiGejala() :bool Operasi ini digunakan untuk melakukan validasi gejala yang diinputkan +saveGejala(GejalaManager) :void Operasi ini digunakan untuk menyimpan gejala kedalam database. +getGejala() :void Operasi ini digunakan untuk mengambil gejala dari database. +showGejala(Gejala) :void Operasi ini digunakan untuk menampilkan gejala +editGejala() Operasi ini digunakan untuk mengedit gejala yang sudah ada di dalam database. +deleteGejala() Operasi ini digunakan untuk menghapus gejala yang ada didalam database.	

+displayGejala() Operasi ini digunakan untuk menampilkan gejala yang ada di dalam database. +searchGejala() Operasi ini digunakan untuk mencari gejala didalam database.

Spesifikasi Design Kelas PengelolaanPenyakitUI

PengelolaanPenyakitUI	**<<boundary>>**
+PengelolaanPenyakitUI() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. +entryPenyakit() Operasi ini digunakan untuk memasukkan penyakit. +validasiPenyakit() :bool Operasi ini digunakan untuk melakukan validasi penyakit yang diinputkan +savePenyakit(PenyakitManager) :void Operasi ini digunakan untuk menyimpan penyakit kedalam database. +getPenyakit() :void Operasi ini digunakan untuk mengambil penyakit dari database. +showPenyakit(Penyakit) :void Operasi ini digunakan untuk menampilkan penyakit +editPenyakit() Operasi ini digunakan untuk mengedit penyakit yang sudah ada di dalam database. +deletePenyakit() Operasi ini digunakan untuk menghapus penyakit yang ada didalam database. +displayPenyakit()	

Operasi ini digunakan untuk menampilkan penyakit yang ada di dalam database. +searchPenyakit() Operasi ini digunakan untuk mencari penyakit didalam database.

Spesifikasi Design Kelas PengelolaanHamaUI

PengelolaanHamaUI	**<<boundary>>**
+PengelolaanHamaUI() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. +entryHama() Operasi ini digunakan untuk memasukkan hama. +validasiHama() :bool Operasi ini digunakan untuk melakukan validasi hama yang diinputkan +saveHama(HamaManager) :void Operasi ini digunakan untuk menyimpan hama kedalam database. +getHama() :void Operasi ini digunakan untuk mengambil hama dari database. +showHama(Hama) :void Operasi ini digunakan untuk menampilkan hama +editHama() Operasi ini digunakan untuk mengedit hama yang sudah ada di dalam database. +deleteHama() Operasi ini digunakan untuk menghapus hama yang ada didalam database. +displayHama() Operasi ini digunakan untuk menampilkan hama yang ada di dalam database. +searchHama() Operasi ini digunakan untuk mencari hama didalam database.	

Spesifikasi Design Kelas PengelolaanAturanUI

PengelolaanAturanUI	**<<boundary>>**
+PengelolaanAturanUI() Default konstruktor, digunakan untuk inisialisasi semua atribut dari kelas ini. +entryAturan() Operasi ini digunakan untuk memasukkan aturan. +validasiAturan() :bool Operasi ini digunakan untuk melakukan validasi aturan yang diinputkan +saveAturan(AturanManager) :void Operasi ini digunakan untuk menyimpan aturan kedalam database. +getAturan() :void Operasi ini digunakan untuk mengambil aturan dari database. +showAturan(Aturan) :void Operasi ini digunakan untuk menampilkan aturan +editAturan() Operasi ini digunakan untuk mengedit aturan yang sudah ada di dalam database. +deleteAturan() Operasi ini digunakan untuk menghapus aturan yang ada didalam database. +displayAturan() Operasi ini digunakan untuk menampilkan aturan yang ada di dalam database. +searchAturan() Operasi ini digunakan untuk mencari aturan didalam database.	

Spesifikasi Design Kelas LoginManager

LoginManager	**<<control>>**
+LoginManager() Default konstruktor, digunakan untuk inisialisasi semua	

attribute dari kelas ini. +ValidasiLogin() Operasi ini digunakan untuk mengecek data login yang diinputkan. Data login yang diinputkan user akan dibandingkan dengan data yang sudah tersimpan di database,apabila data login yang diinputkan benar maka akan direturnkan nilai true,jika sebaliknya akan direturnkan nilai false.

Spesifikasi Design Kelas UserManager

UserManager	**<<control>>**
+UserManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini.	

Spesifikasi Design Kelas GejalaManager

GejalaManager	**<<control>>**
+GejalaManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini.	

Spesifikasi Design Kelas PenyakitManager

PenyakitManager	**<<control>>**
+PenyakitManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini.	

Spesifikasi Design Kelas HamaManager

HamaManager	**<<control>>**
+HamaManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini.	

Spesifikasi Design Kelas AturanManager

AturanManager	**<<control>>**
+AturanManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini.	

Spesifikasi Design Kelas User

User	**<<entity>>**
-Username : varchar(25) Atribut ini digunakan untuk mempermudah dikenali, dan juga digunakan untuk username ketika login untuk masuk ke sistem -Password : String Atribut ini digunakan untuk menyimpan data nama user -Id_User: Int Atribut Ini digunakan sebagai tanda pengenal	
+User() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +getDataUser() : User Operasi ini digunakan untuk mengambil data user dari database. +updateDataUser() Operasi ini digunakan untuk mengupdate data user di database. +insertDataUser() Operasi ini digunakan untuk menyimpan data user ke database. +searchDataUser() Operasi ini digunakan untuk mencari data user di database. +deleteDataUser() Operasi ini digunakan untuk menghapus data user di database.	

Spesifikasi Design Kelas Gejala

Gejala	**<<entity>>**

-Akar : varchar(100) Atribut ini digunakan untuk menyimpan data akar. -Daun : varchar(100) Atribut ini digunakan untuk menyimpan data daun -Batang : varchar(100) Atribut ini digunakan untuk menyimpan data batang. -Id_Gejala: Int Atribut Ini digunakan untuk menyimpan data id dari gejala.
+Gejala() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +getDataGejala() : Gejala Operasi ini digunakan untuk mengambil data gejala dari database. +updateDataGejala() Operasi ini digunakan untuk mengupdate data gejala di database. +insertDataGejala() Operasi ini digunakan untuk menyimpan data gejala ke database. +searchDataGejala() Operasi ini digunakan untuk mencari data gejala di database. +deleteDataGejala() Operasi ini digunakan untuk menghapus data gejala di database.

Spesifikasi Design Kelas Penyakit

Penyakit	**<<entity>>**
-Id_Penyakit: Int Atribut Ini digunakan untuk menyimpan data id dari penyakit. -Nama_Penyakit: varchar(50) Atribut ini digunakan untuk menyimpan nama penyakit. -Probabilitas_P: float	

Atribut ini digunakan untuk perbandingan gejala dengan penyakit yang ada.
+Penyakit() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +getDataPenyakit() : Penyakit Operasi ini digunakan untuk mengambil data penyakit dari database. +updateDataPenyakit() Operasi ini digunakan untuk mengupdate data penyakit di database. +insertDataPenyakit() Operasi ini digunakan untuk menyimpan data penyakit ke database. +searchDataPenyakit() Operasi ini digunakan untuk mencari data penyakit di database. +deleteDataPenyakit() Operasi ini digunakan untuk menghapus data penyakit di database.

Spesifikasi Design Kelas Hama

Hama	**<<entity>>**
-Id_Hama: Int Atribut Ini digunakan untuk menyimpan data id dari hama. -Nama_Hama: varchar(50) Atribut ini digunakan untuk menyimpan nama hama. -Probabilitas_H: float Atribut ini digunakan untuk perbandingan gejala dengan hama yang ada.
+Hama() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +getDataHama() : Hama

Operasi ini digunakan untuk mengambil data hama dari database. +updateDataHama() Operasi ini digunakan untuk mengupdate data hama di database. +insertDataHama() Operasi ini digunakan untuk menyimpan data hama ke database. +searchDataHama() Operasi ini digunakan untuk mencari data hama di database. +deleteDataHama() Operasi ini digunakan untuk menghapus data hama di database.

Spesifikasi Design Kelas Aturan

Aturan	**<<entity>>**
-Id_Aturan: Int Atribut Ini digunakan untuk menyimpan data id dari aturan.	
+Aturan() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +getDataAturan() : Aturan Operasi ini digunakan untuk mengambil data aturan dari database. +updateDataAturan() Operasi ini digunakan untuk mengupdate data aturan di database. +insertDataAturan() Operasi ini digunakan untuk menyimpan data aturan ke database. +searchDataAturan() Operasi ini digunakan untuk mencari data aturan di database. +deleteDataAturan() Operasi ini digunakan untuk menghapus data aturan di database.	

2 Perancangan Data

2.1 Dekomposisi Data

2.1.1 Deskripsi Entitas Data User

Nama	Tipe	Panjang	Keterangan
IDUSER	INT	-	Id user, primary key
USERNAME	Varchar	25	Nama login untuk pengguna
PASSWORD	Varchar	25	Password yang dimiliki pengguna
NAMAASLI	Integer	-	Nama asli pengguna

2.1.2 Deskripsi Entitas Gejala

Nama	Tipe	Panjang	Keterangan
IDGEJALA	INT	-	Id gejala, primary key
AKAR	Varchar	100	Akar
DAUN	Varchar	100	Daun
BATANG	Varchar	100	Batang
GAMBAR	Image	-	gambar

2.1.3 Deskripsi Entitas Penyakit

Nama	Tipe	Panjang	Keterangan
IDPENYAKIT	INT	-	Id penyakit, primary key
NAMAPENYAKIT	Varchar	50	Nama Penyakit

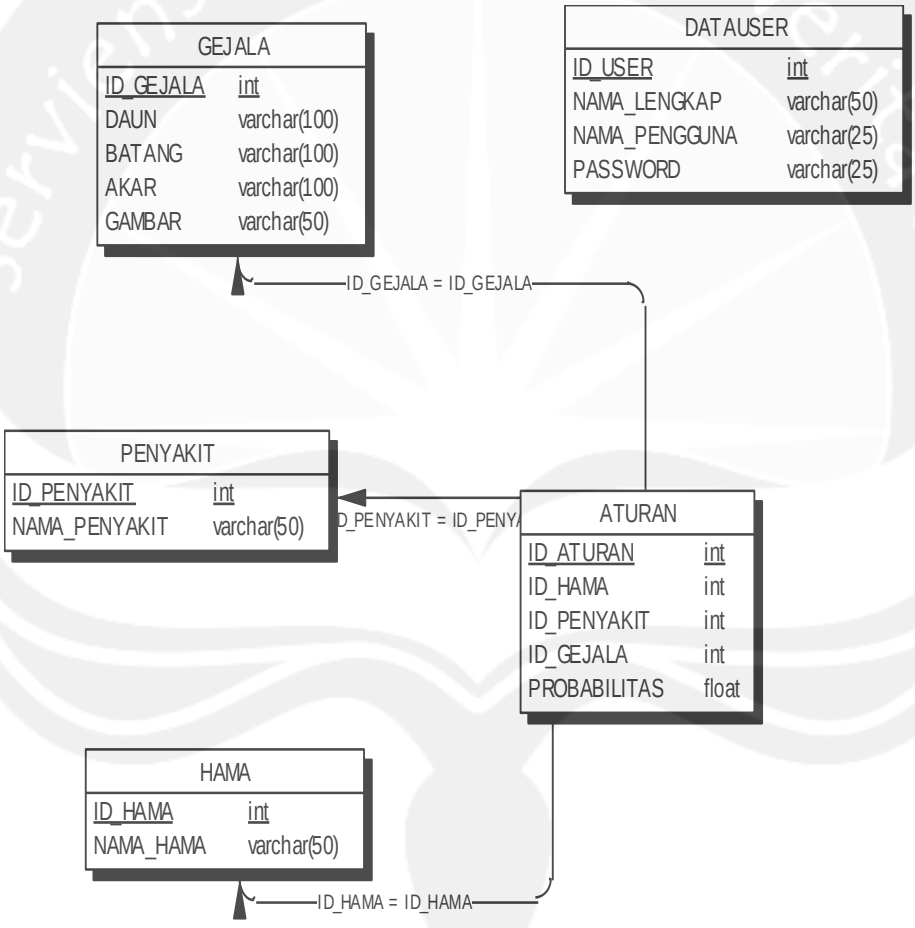
2.1.4 Deskripsi Entitas Hama

IDHAMA	INT	-	Id hama, primary key
NAMAHAMA	Varchar	50	Nama Hama

2.1.5 Deskripsi Entitas Aturan

IDATURAN	INT	-	Id aturan, primary key
PROBABILITAS	FLOAT	-	Probabilitas

2.2 *Physical Data Model*

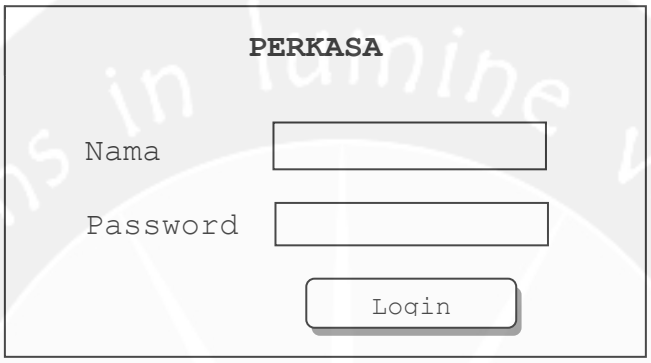


Gambar 3.1 Physical Data Model

3 Perancangan Antarmuka

3.1 *Sketsa Antarmuka dan Deskripsinya*

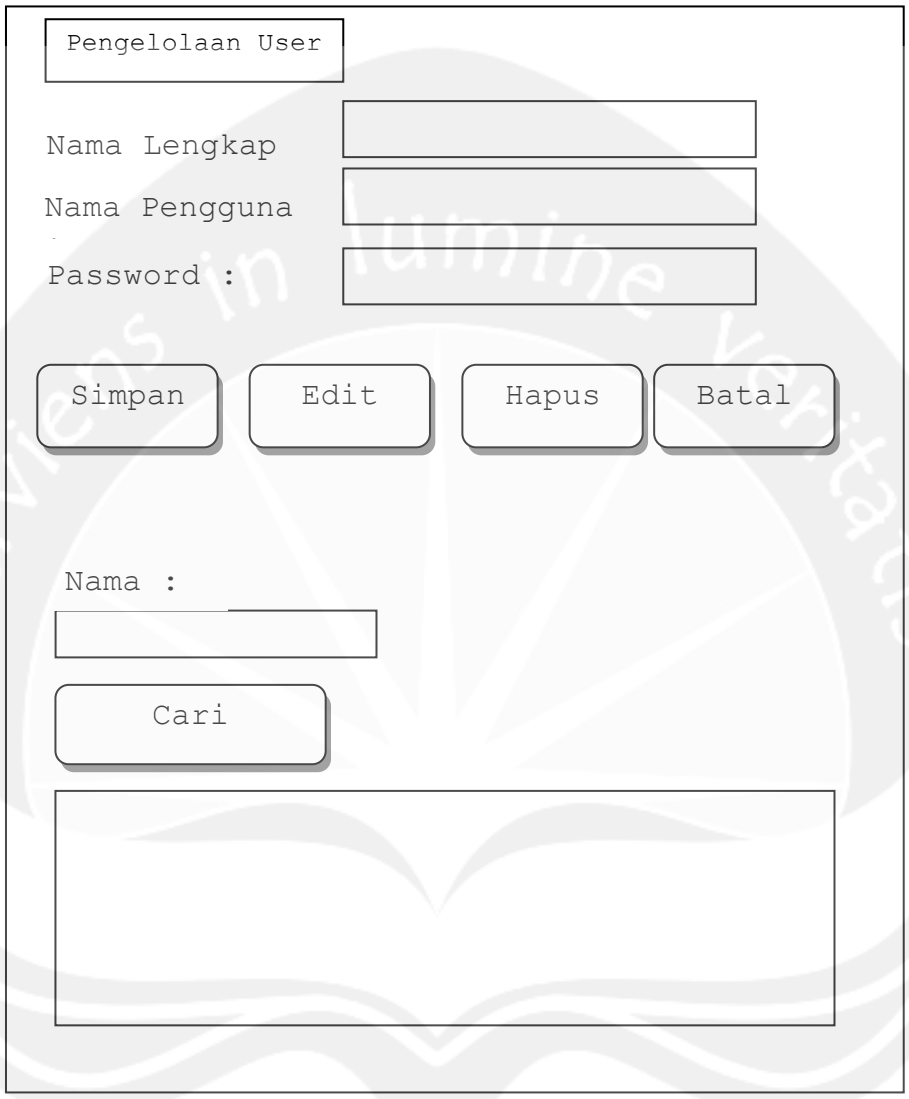
3.1.1 Antarmuka Login



Gambar 4.1 Rancangan Antarmuka Login

Antar muka yang digunakan oleh user agar dapat memasuki sistem. User diminta memasukkan nama login dan passwordnya untuk memasuki sistem. Sebelumnya data sudah tersimpan di database. Setelah user memasukkan username dan login, user mengklik tombol login. Jika valid maka user akan masuk ke sistem , jika tidak, maka user dapat memasukkan kembali username dan password yang benar.

3.1.2 Antarmuka Pengelolaan User

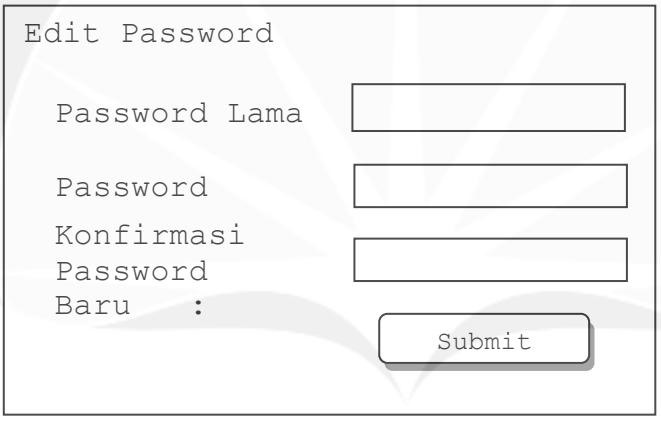


Gambar 4.2 Antar Muka Pengelolaan User

Antar muka ini digunakan administrator untuk mengelola data user berupa input, edit, dan delete. Untuk input admin akan memasukkan data-data kedalam field. Setelah memasukkan data, admin mengklik tombol Simpan dan data akan masuk ke dalam database dan muncul di dalam datagrid. Tombol Ubah dan Hapus di set disable, kecuali admin telah memilih salah satu data yang ada di

dalam datagrid. Jika admin memilih ubah, maka data user yang dipilih akan masuk kedalam field-field, lalu admin mengubahnya. Jika admin memilih hapus, maka data yang dipilih admin akan dihapus dari database dan dari datagrid. Untuk mencari data user, admin memasukkan nama user yang akan dicari di dalam field yang tersedia, lalu mengklik tombol cari. Lalu data akan muncul pada datagrid.

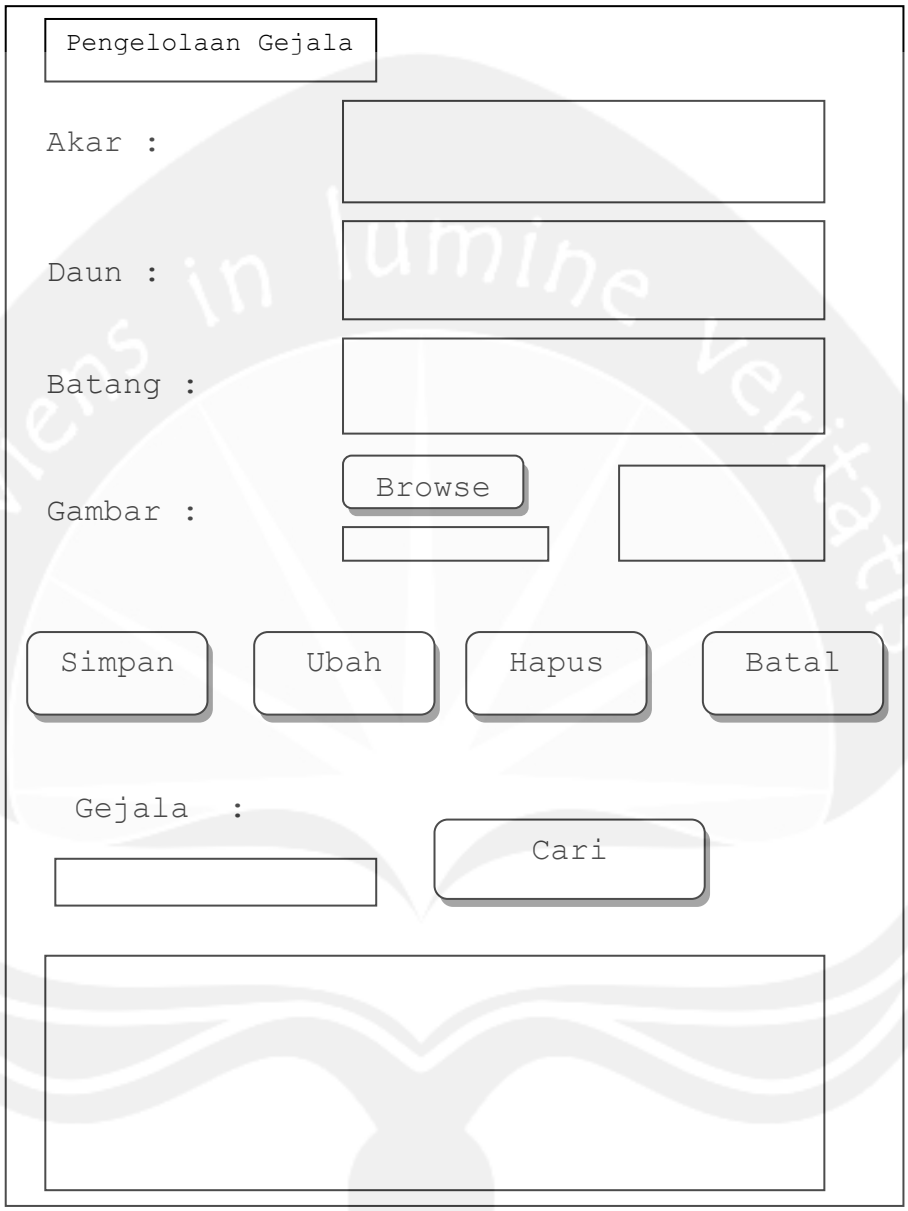
3.1.3 Antarmuka Edit Password



Gambar 4.3 Antar Muka Edit Password

Ini merupakan antar muka yang digunakan oleh seluruh user untuk mengubah password.

3.1.4 Antarmuka Pengelolaan Gejala

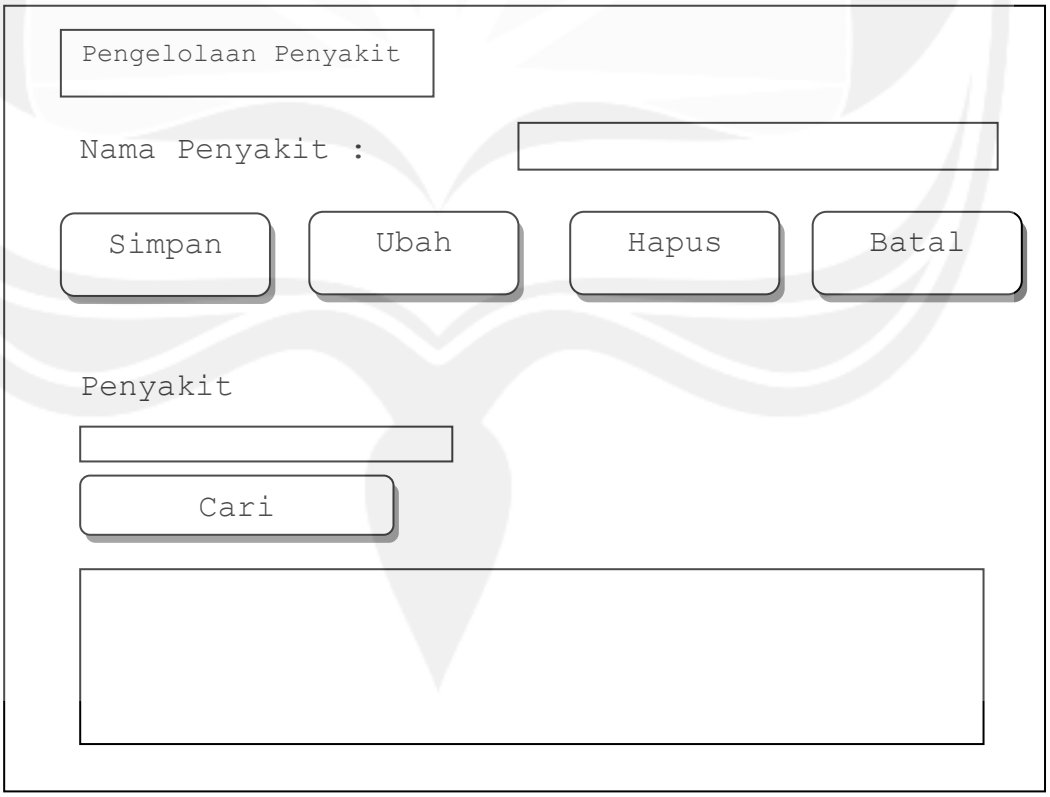


Gambar 4.5 Rancangan Antarmuka Pengelolaan Gejala

Antar muka ini digunakan administrator untuk mengelola gejala berupa input, edit, dan delete. Untuk input admin akan memasukkan data-data kedalam field. Setelah memasukkan data, admin

mengklik tombol Simpan dan data akan masuk ke dalam database dan muncul di dalam datagrid. Tombol Ubah dan Hapus di set disable, kecuali admin telah memilih salah satu data yang ada di dalam datagrid. Jika admin memilih ubah, maka data user yang dipilih akan masuk kedalam field-field, lalu admin mengubahnya. Jika admin memilih hapus, maka data yang dipilih admin akan dihapus dari database dan dari datagrid. Untuk mencari data gejala, admin memasukkan gejala yang akan dicari di dalam field yang tersedia, lalu mengklik tombol cari. Lalu data akan muncul pada datagrid.

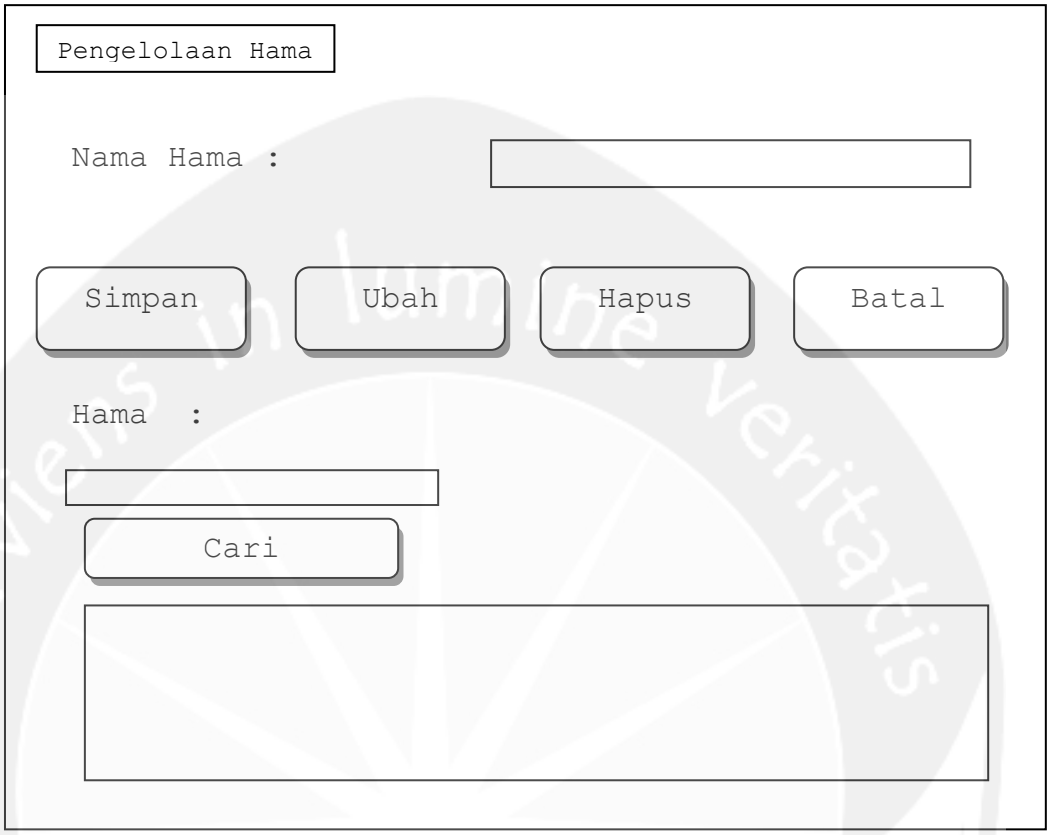
3.1.5 Antarmuka Pengelolaan Penyakit



Gambar 4.6 Rancangan Antarmuka Pengelolaan Penyakit

Antar muka ini digunakan administrator untuk mengelola penyakit berupa input, edit, dan delete. Untuk input admin akan memasukkan data-data kedalam field. Setelah memasukkan data, admin mengklik tombol Simpan dan data akan masuk ke dalam database dan muncul di dalam datagrid. Tombol Ubah dan Hapus di set disable, kecuali admin telah memilih salah satu data yang ada di dalam datagrid. Jika admin memilih ubah, maka data penyakit yang dipilih akan masuk kedalam field-field, lalu admin mengubahnya. Jika admin memilih hapus, maka data yang dipilih admin akan dihapus dari database dan dari datagrid. Untuk mencari data penyakit, admin memasukkan penyakit yang akan dicari di dalam field yang tersedia, lalu mengklik tombol cari. Lalu data akan muncul pada datagrid.

3.1.6 Antarmuka Pengelolaan Hama

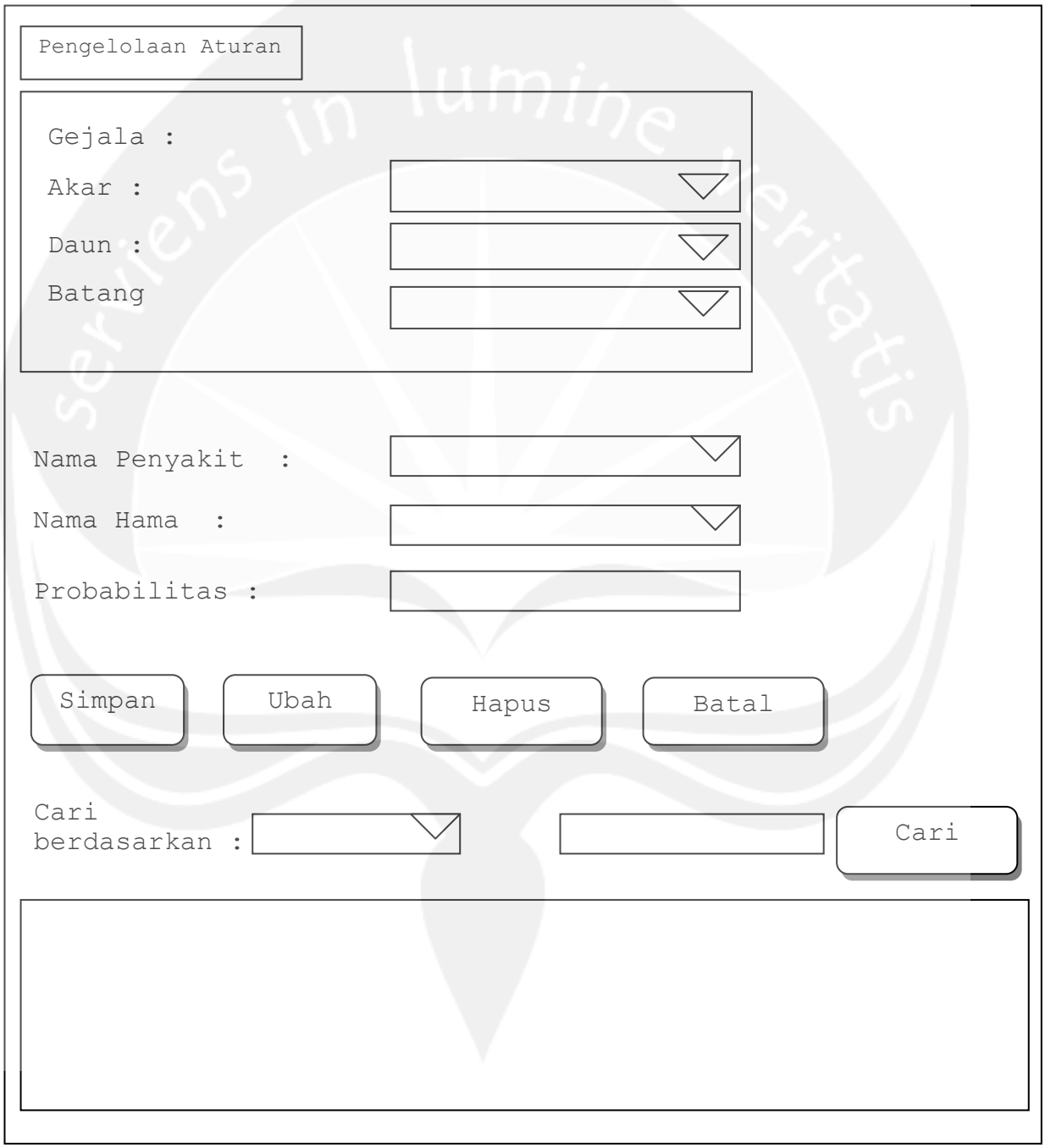


Gambar 4.7 Rancangan Antarmuka Pengelolaan Hama

Antar muka ini digunakan administrator untuk mengelola hama berupa input, edit, dan delete. Untuk input admin akan memasukkan data-data kedalam field. Setelah memasukkan data, admin mengklik tombol Simpan dan data akan masuk ke dalam database dan muncul di dalam datagrid. Tombol Ubah dan Hapus di set disable, kecuali admin telah memilih salah satu data yang ada di dalam datagrid. Jika admin memilih ubah, maka data hama yang dipilih akan masuk kedalam field-field, lalu admin mengubahnya. Jika admin memilih hapus, maka data yang dipilih admin akan dihapus dari database dan dari datagrid. Untuk mencari data hama, admin memasukkan

hama yang akan dicari di dalam field yang tersedia, lalu mengklik tombol cari. Lalu data akan muncul pada datagrid.

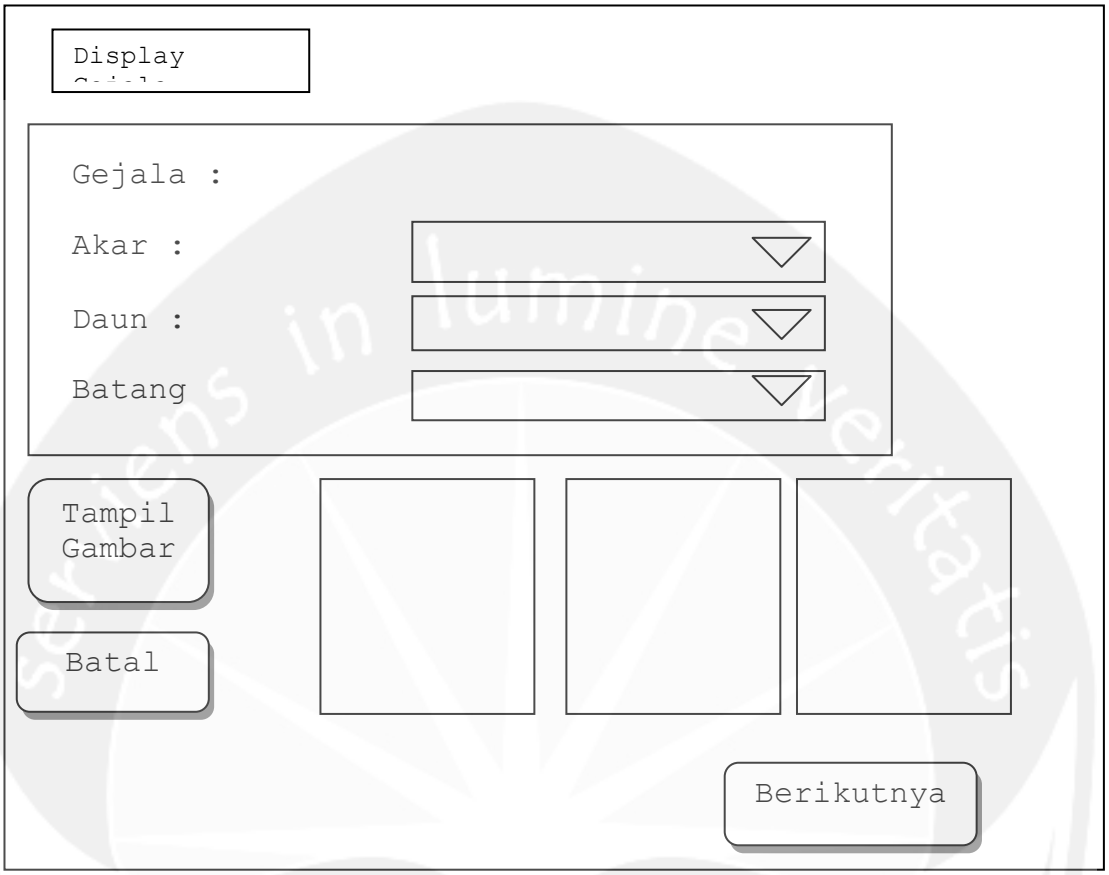
3.1.7 Antarmuka Pengelolaan Aturan



Gambar 4.8 Rancangan Antarmuka Pengelolaan Aturan

Antar muka ini digunakan administrator untuk mengelola aturan berupa input, edit, dan delete. Untuk input admin akan memasukkan data-data kedalam field. Setelah memasukkan data, admin mengklik tombol Simpan dan data akan masuk ke dalam database dan muncul di dalam datagrid. Tombol Ubah dan Hapus di set disable, kecuali admin telah memilih salah satu data yang ada di dalam datagrid. Jika admin memilih ubah, maka data aturan yang dipilih akan masuk kedalam field-field, lalu admin mengubahnya. Jika admin memilih hapus, maka data yang dipilih admin akan dihapus dari database dan dari datagrid. Untuk mencari data aturan, admin memasukkan aturan yang akan dicari di dalam field yang tersedia, lalu mengklik tombol cari. Lalu data akan muncul pada datagrid.

3.1.8 Antarmuka Display Gejala



Gambar 4.9 Rancangan Antarmuka Display Gejala

Antarmuka Gambar 4.9 merupakan antarmuka display gejala. User harus memilih minimal salah satu dari gejala yang diinginkan yang tersedia disini. Kemudian user memencet tombol Tampil Gambar untuk menampilkan gambar sesuai gejala yang diinputkan *oleh user, setelah gambar muncul dan sesuai dengan gejala yang diinputkan user , user melanjutkan ke langkah selanjutnya dengan memencet tombol Berikutnya.*

```
if(((penyakitAkar == penyakitBatang AND penyakitAkar.Length !=0)OR
    (penyakitAkar == penyakitDaun AND penyakitAkar.Length !=0) OR
    (penyakitAkar == penyakitUmbi AND penyakitAkar.Length !=0) OR
  (penyakitBatang == penyakitDaun AND penyakitBatang.Length !=0)OR
 (penyakitBatang == penyakitUmbi AND penyakitBatang.Length !=0)OR
   (penyakitDaun == penyakitUmbi AND penyakitDaun.Length !=0) OR
  (hamaAkar == hamaBatang AND hamaAkar.Length !=0) OR
  (hamaAkar == hamaDaun AND hamaAkar.Length !=0) OR
  (hamaAkar == hamaUmbi AND hamaAkar.Length !=0) OR
  (hamaBatang == hamaDaun AND hamaBatang.Length !=0) OR
  (hamaBatang == hamaUmbi AND hamaBatang.Length !=0) OR
  (hamaDaun == hamaUmbi  AND hamaDaun.Length !=0) )
then
{
 combine();
}
else
{
single();
}
end if
```

Implementasi prosedur combined()

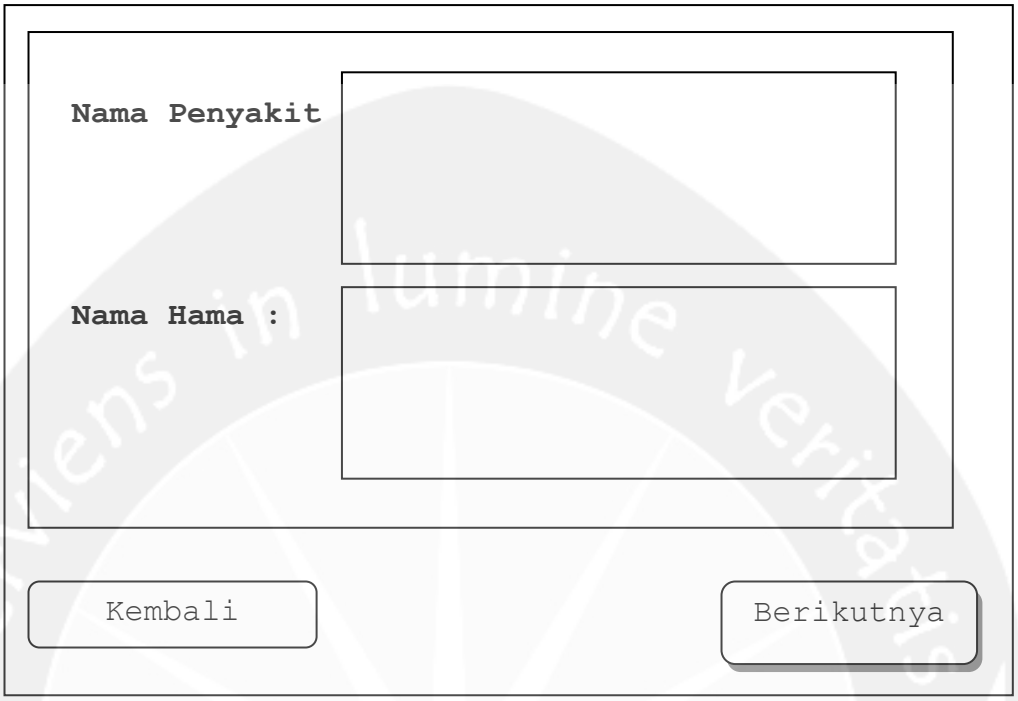
```
double nilai1 = (Convert.ToDouble ( tbcfAkar.Text ) / 100 ) *
Convert.ToDouble ( cfPakarAkar );
double nilai2 = (Convert.ToDouble ( tbcfBatang.Text ) / 100 ) *
Convert.ToDouble ( cfPakarBatang );

hasil1 = nilai1 + (nilai2 * (1 - nilai1));
hasil1 = hasil1 * 100 ;
```

Implementasi prosedur single()

```
hasilsingle1 = (probUserAkar * probPakarAkar) * 100;
hasilsingle2 = (probUserBatang * probPakarBatang) * 100;
hasilsingle3 = (probUserDaun * probPakarDaun) * 100;
hasilsingle4 = (probUserUmbi * probPakarUmbi) * 100;
```

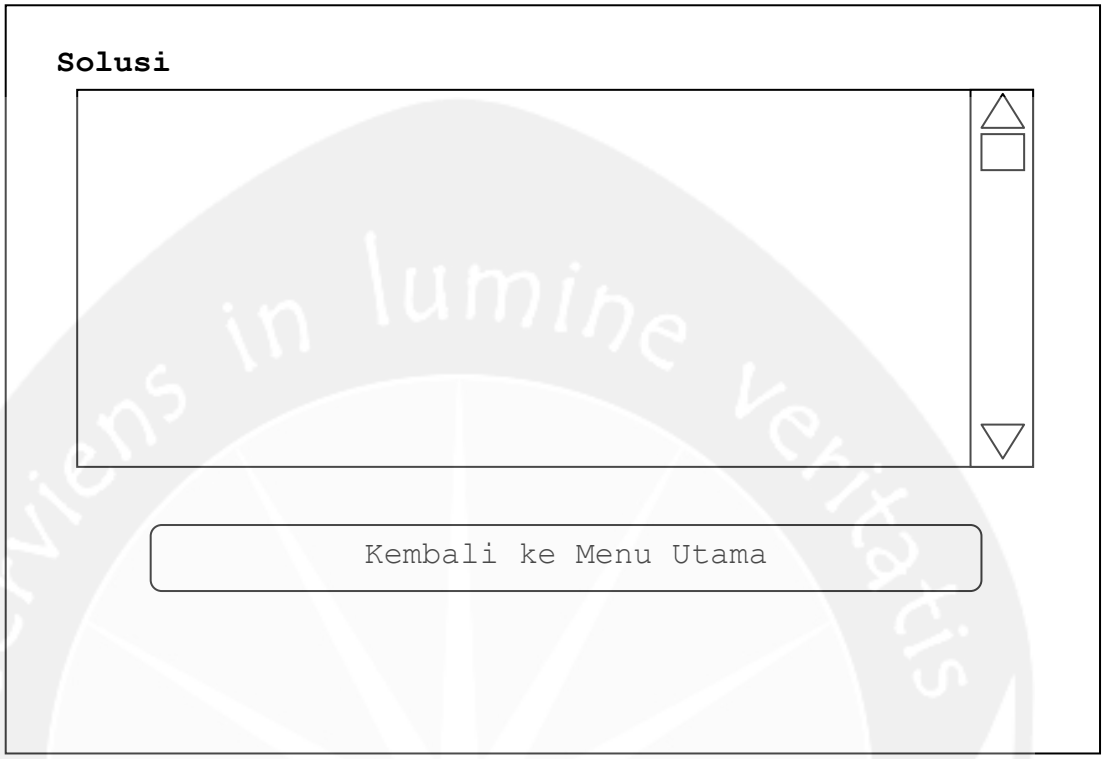
3.1.9 Antarmuka Display Penyakit dan Hama



Gambar 4.10 Rancangan Antarmuka Display Penyakit dan Hama

Antarmuka Gambar 4.10 merupakan antarmuka yang berisi kemungkinan penyakit dan hama yang merupakan hasil dari pilihan-pilihan user dari pertanyaan-pertanyaan yang diajukan oleh sistem sebelumnya. Untuk melanjutkan, user mengklik tombol Berikutnya.

3.1.10 Antarmuka Display Solusi



Gambar 4.11 Rancangan Antarmuka Display Solusi

Antarmuka Gambar 4.11 merupakan antarmuka yang berisi solusi yang merupakan hasil dari pilihan-pilihan user dari pertanyaan-pertanyaan yang diajukan oleh sistem sebelumnya,untuk mengakhiri , user menekan tombol Kembali ke Menu Utama.

Daftar Pustaka

Artanti, F. R. 2004. Perancangan dan Pembuatan Sistem Pakar Hama dan Pengendaliannya untuk Tanaman Hortikultura. Skripsi. Surabaya : Universitas Kristen Petra.

Budhi, G. S., Intan, R. 2010. Probabilitas Penggunaan Premis untuk Menentukan Certainty Factor dari Rule.

Daniel, Gloria Virginia. 2010. Implementasi Sistem Pakar untuk Mendiagnosis Penyakit Dengan Gejala Demam Menggunakan Metode Certainty Factor. Jurnal Informatika, Volume 5 Nomor 1, April 2010.

Dhany, Safia. 2009. Perancangan Sistem Pakar untuk Diagnosa Penyakit Anak. Skripsi. Medan : Universitas Sumatera Utara.

Durkin, John; “Expert Systems Design and Development”, Pretince Hall, 1994.

Feri Fahrur Rohman, Ami Fauziah. “Rancang Bangun Aplikasi Sistem Pakar Untuk Menentukan Jenis Gangguan pada Anak”. Universitas Islam Indonesia.

Kusrini. 2008. Aplikasi Sistem Pakar. Yogyakarta: Andi Offset.

Kusrini. 2006. Sistem Pakar Teori dan Aplikasi. Yogyakarta: Andi Offset.

Kusrini. 2009. Penggunaan Certainty Factor dalam Sistem Pakar untuk Melakukan Diagnosis dan Memberikan

terapi Penyakit Epilepsi dan Keluarganya. Yogyakarta: STMIK AMIKOM.

Kusumadewi, S. 2003. Artificial Inteligence(Teknik dan Aplikasinya). Yogyakarta: Graha Ilmu.

Ruben, Milka Widyasari. 2012. Pengembangan Sistem Pakar Diagnosa Hama Dan Penyakit Tanaman Padi Dengan Metode Certainty Factor. Skripsi. Yogyakarta: Universitas Atma Jaya Yogyakarta.

Sasmito, Ginanjar Wiro. 2010. Aplikasi Sistem Pakar untuk Simulasi Diagnosa Hama dan Penyakit Tanaman Bawang Merah dan Cabai Menggunakan Forward Chaining dan Pendekatan Berbasis Aturan. Tesis. Semarang: Universitas Diponegoro.

Suyamto. 2007. Masalah Lapang Hama Penyakit Hara pada Padi. Pusat Penelitian dan Pengembangan Tanaman Pangan.

Tim Penilis PS. 1995. Hama Penyakit Sayur dan Palawija(Gejala, Jenis, dan Pengendalian). Jakarta: Penebar Swadaya.

Ditya, Emmanuel Dakris. 2013. Aplikasi Wisata Jogja Berbasis Web Dan Mobile. Skripsi. Yogyakarta: Universitas Atma Jaya Yogyakarta.