BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan pembahasan pada bab-bab sebelumnya, serta hasil pengujian program Pembangunan Sistem Pakar untuk *Troubleshooting* Perangkat Keras Komputer *(SiPaTrou)* yang telah dilakukan, maka dapat ditarik kesimpulan bahwa: Perangkat lunak Sistem Pakar untuk *Troubleshooting* Perangkat Keras Komputer *(SiPaTrou)* berhasil dibangun dan dapat berjalan dengan baik.

5.2. Saran

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

1. Perangkat lunak *SiPaTrou* ini masih dapat dikembangkan lagi dengan menambahkan macam kerusakan dan jenis kerusakan yang lebih banyak lagi.
2. Perangkat lunak *SiPaTrou* ini dapat dibuat menjadi sistem yang *on-line* sehingga dapat diakses oleh banyak pengguna setiap saat.

DAFTAR PUSTAKA

Arhami, Muhammad, Konsep Dasar Sistem Pakar, C.V ANDI OFFSET, Jl. Beo 38-40, Yogyakarta.

Budiharto, Widodo, dan Sukmadi, Dodi Yogi, 2004, *Panduan Bagi Programmer .NET Aplikasi e-Commerce Menggunakan Visual C#.NET*, Penerbit Andi, Yogyakarta.

Kaliandra, Jayatri, Nino, 2007, Sistem Pakar untuk Pengaturan Diet Sehat Berdasarkan Tinggi Badan, Berat Badan, dan Golongan Darah, UAJY, Yogyakarta.

Kusrini, S.Kom, SISTEM PAKAR Teori dan Aplikasi, C.V ANDI OFFSET, Jl. Beo 38-40, Yogyakarta.

Oktavianti, F. Wina, 2007, Pengembangan Sistem Pakar Model Rambut dan Visualisasinya Berdasarkan Bentuk Wajah Wanita Indonesia, UAJY, Yogyakarta.

Sholiq, 2006, Pemodelan Sistem Informasi Berorientasi Objek dengan dengan UML, Graha Ilmu, Jakarta.

Suyoto, M, 2003, Intelegensi Buatan : Teori dan Pemrograman, Gava Media, Yogyakarta.

Wahyono, Teguh, 2005, PC Troubleshooting Plus, Gava Media, Klitren Lor GK III/15, Yogyakarta.

_____. http://id.wikipedia.org/wiki/Sistem_pakar : sistem pakar diakses pada tanggal 29 September 2007.

_____.http://staf.uum.edu.my/ruzinoor/Troubleshooting.html diakses pada tanggal 12 September 2007.

DPPL

DESKRIPSI PERANCANGAN PERANGKAT LUNAK

Sistem Pakar Untuk *Troubleshooting* Perangkat Keras Komputer

(SiPaTrou)

Disusun oleh:

Rika Novenawati

03 07 03695

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
Ditulis oleh							
Diperiksa oleh							
Disetujui oleh							

DAFTAR HALAMAN PERUBAHAN

Halaman	Revisi	Halaman	Revisi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

1 Pendahuluan

1.1 Tujuan

Dokumen Deskripsi Perancangan Perangkat Lunak (DPPL) *SiPaTrou* (Sistem Pakar Untuk *Troubleshooting* Perangkat Keras Komputer) ini bertujuan untuk mendefinisikan perancangan perangkat lunak yang akan dibangun. Dokumen DPPL ini digunakan oleh pengembang perangkat lunak sebagai acuan untuk implementasi pada tahap selanjutnya.

1.2 Ruang Lingkup

Perangkat Lunak *SiPaTrou* dikembangkan dengan tujuan untuk:

1. Menangani ***login,*** **ubah password, pengelolaan pakar, pengelolaan data macam kerusakan, data jenis kerusakan, dan data penyebab kerusakan beserta solusi** oleh pakar.
2. Menangani proses **penelusuran kerusakan hardware** oleh pakar dan user biasa.

Perangkat lunak *SiPaTrou* berjalan pada komputer *stand alone* dan *DBMS* yang digunakan adalah *Microsoft SQL Server 2000*.

1.3 Definisi, Akronim, dan Singkatan

Daftar definisi dan akronim yang digunakan:

Tabel 1. Tabel 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.
SiPaTrou	Perangkat lunak yang berguna untuk memberikan informasi tentang *troubleshooting* perangkat keras komputer.

1.4 Referensi

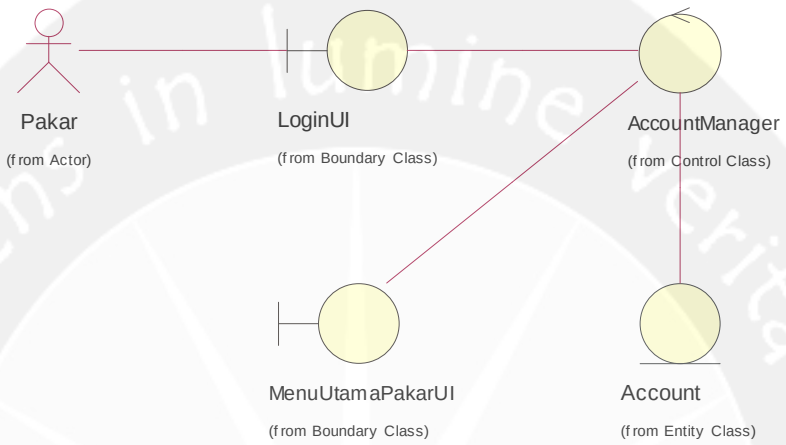
Dokumen yang digunakan sebagai acuan dalam rencana pembangunan perangkat lunak ini adalah:

- Novenawati, Rika, **SKPL-SiPaTrou**, Program Studi Teknik Informatika Universitas Atma Jaya Yogyakarta, 2007.
- Dewi, Findra Kartika Sari. **GL-FINGERS-03**, *Deskripsi Perancangan Perangkat Lunak*, Jurusan Teknik Informatika – UAJY.
- Nurdini, Ratna, **DPPL-Chosen**, Program Studi Teknik Informatika Universitas Atma Jaya Yogyakarta, 2007.
- http://ilmukomputer.com:81/umum/yanti-uml.php

2 Analysis Model

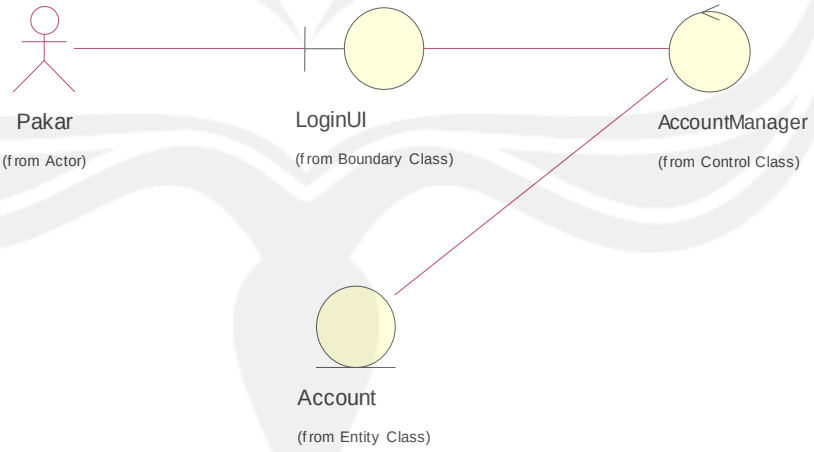
2.1 Realisasi Class Diagram

2.1.1 Login



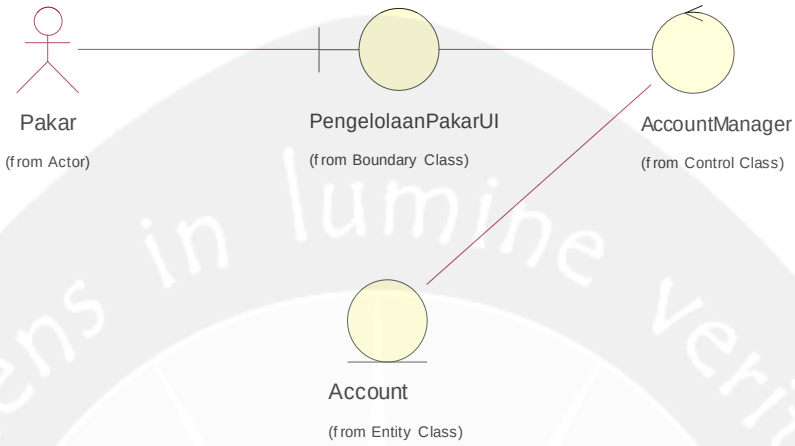
Gambar 1. Realisasi Class Diagram: Login

2.1.2 Ubah Password



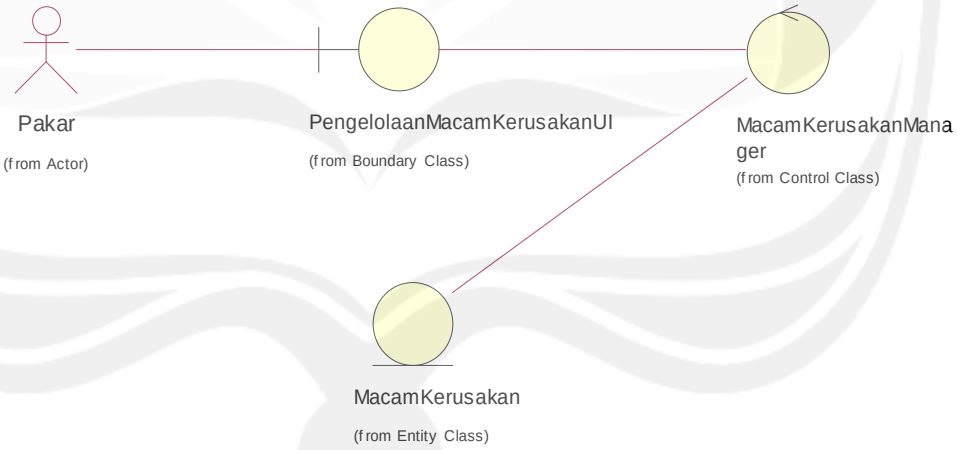
Gambar 2. Realisasi Class Diagram: Ubah Password

2.1.3 Pengelolaan Pakar



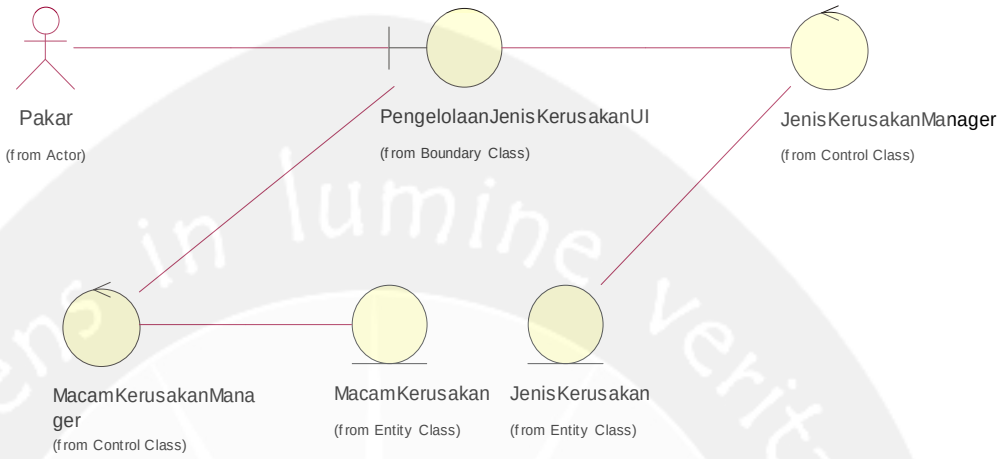
Gambar 3. Realisasi Class Diagram: Pengelolaan Pakar

2.1.4 Pengelolaan Macam Kerusakan



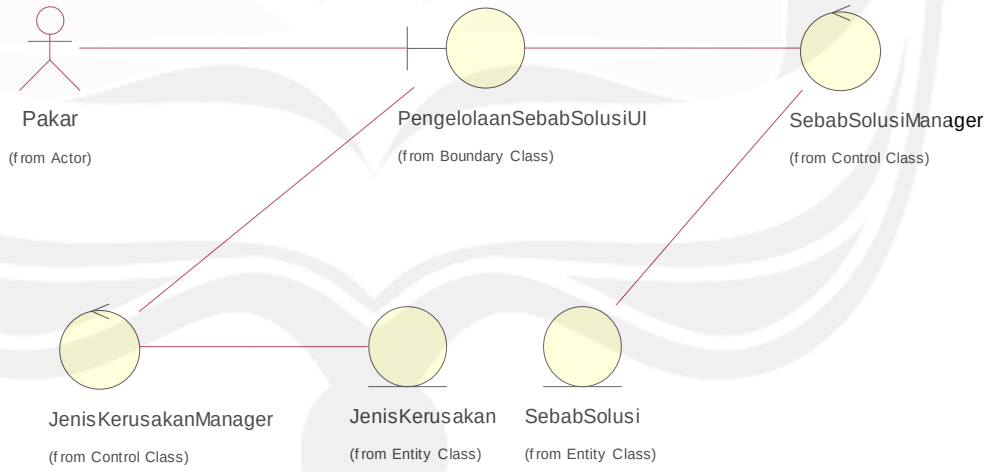
Gambar 4. Realisasi Class Diagram: Pengelolaan Macam Kerusakan

2.1.5 Pengelolaan Jenis Kerusakan



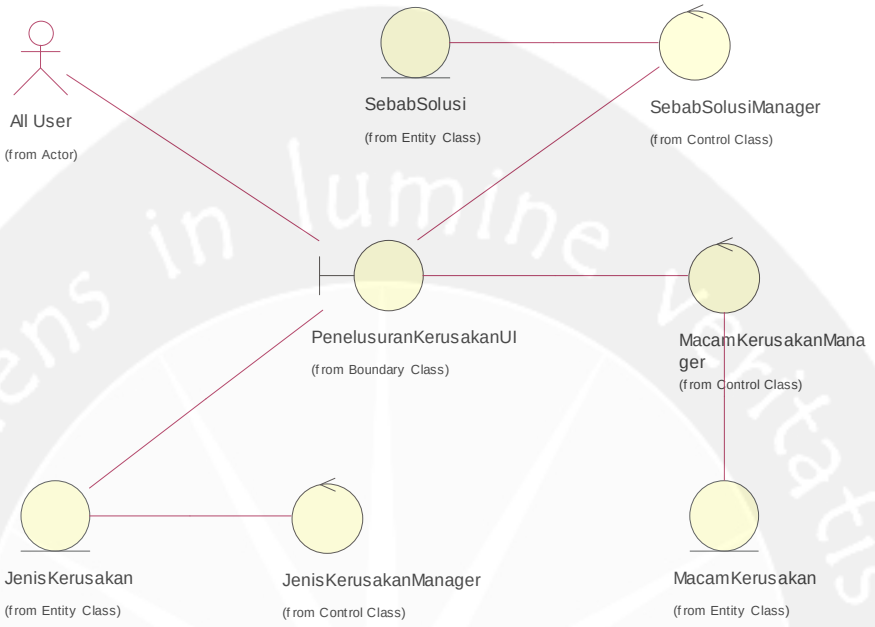
Gambar 5. Realisasi Class Diagram: Pengelolaan Jenis Kerusakan

2.1.6 Pengelolaan Penyebab dan Solusi



Gambar 6. Realisasi Class Diagram: Pengelolaan Penyebab dan Solusi

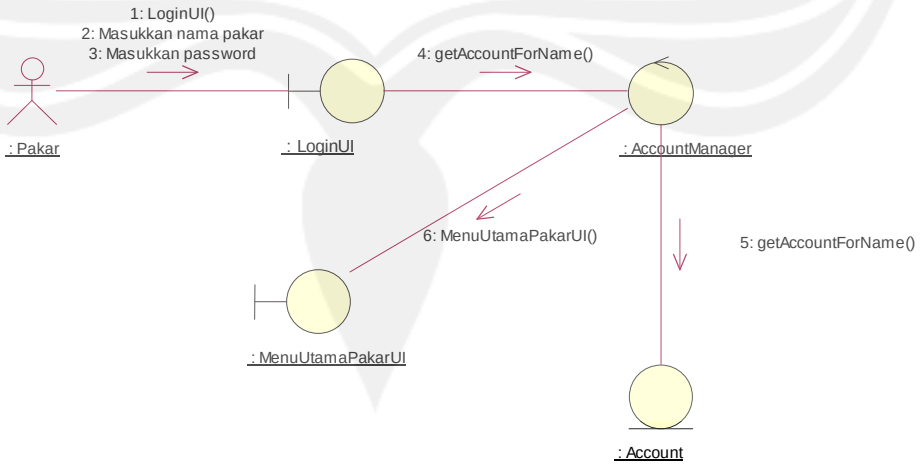
2.1.7 Penelusuran Kerusakan



Gambar 7. Realisasi Class Diagram: Penelusuran Kerusakan

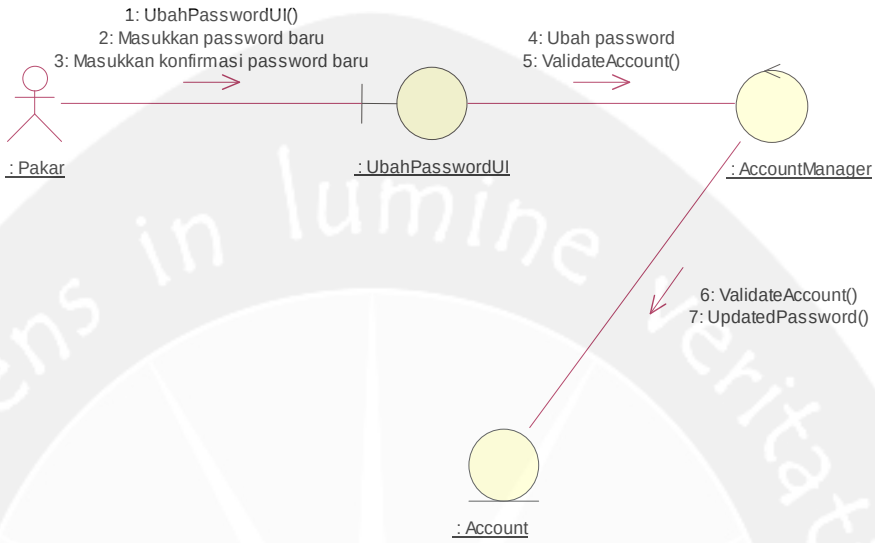
2.2 Collaboration Diagram

2.2.1 Login



Gambar 8. Collaboration Diagram: Login

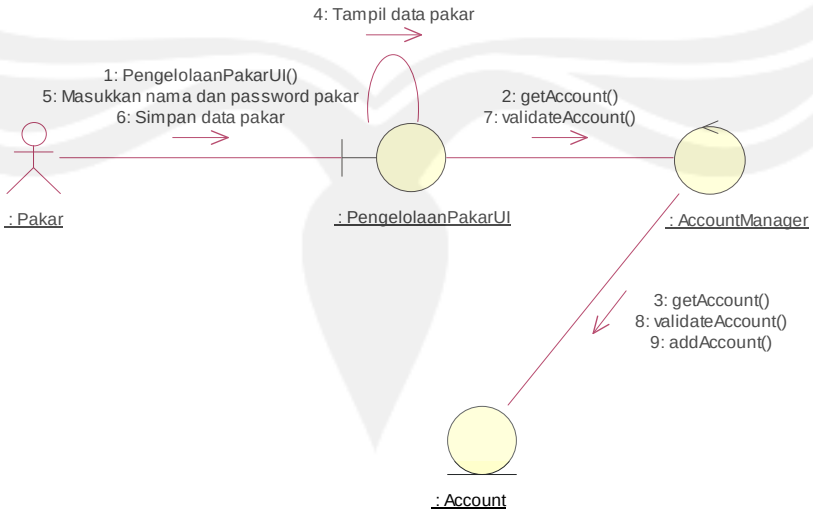
2.2.2 Ubah Password



Gambar 9. Collaboration Diagram: Ubah Password

2.2.3 Pengelolaan Pakar

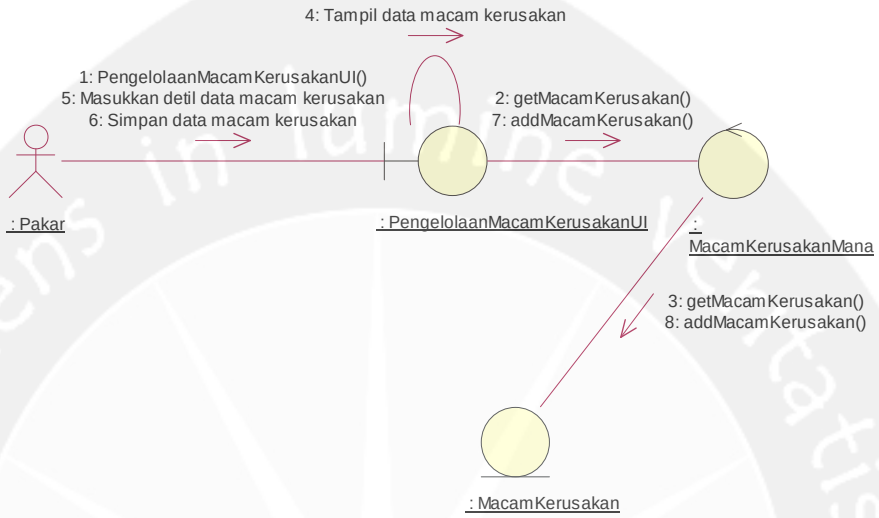
2.2.3.1 Tambah Pakar



Gambar 10. Collaboration Diagram: Pengelolaan Pakar – Tambah Pakar

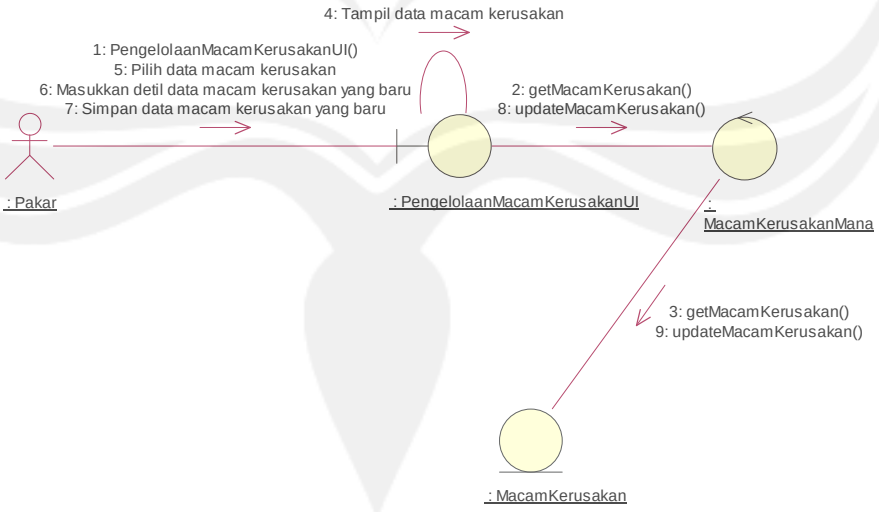
2.2.4 Pengelolaan Macam Kerusakan

2.2.4.1 Tambah Macam Kerusakan



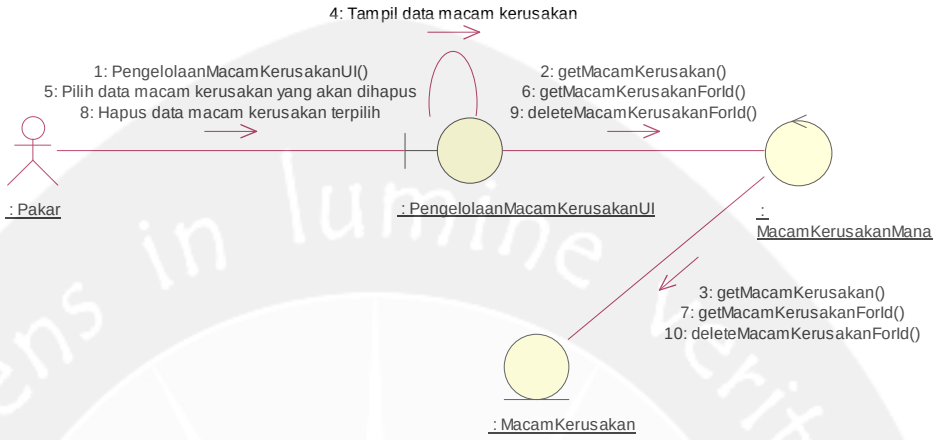
Gambar 11. Collaboration Diagram: Pengelolaan Macam Kerusakan – Tambah Macam Kerusakan

2.2.4.2 Ubah Macam Kerusakan



Gambar 12. Collaboration Diagram: Pengelolaan Macam Kerusakan – Ubah Macam Kerusakan

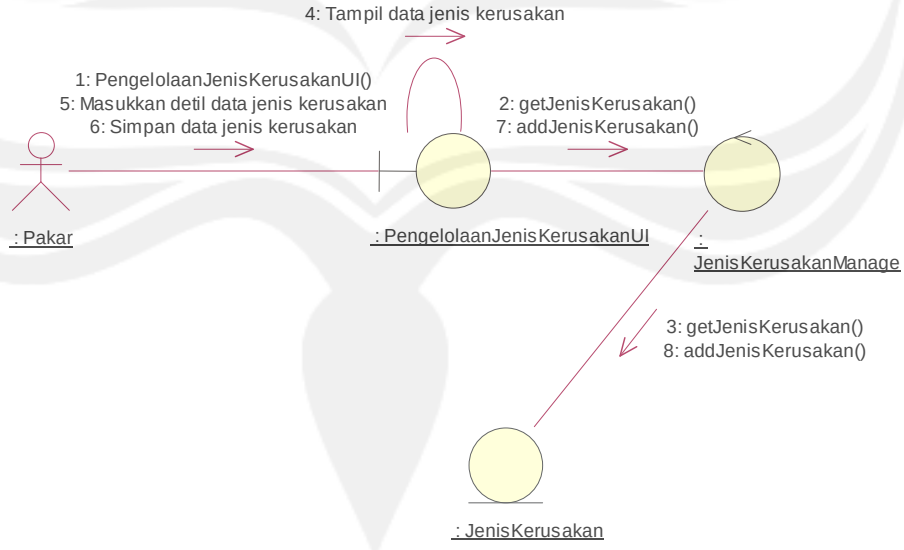
2.2.4.3 Hapus Macam Kerusakan



Gambar 13. Collaboration Diagram: Pengelolaan Macam Kerusakan – Hapus Macam Kerusakan

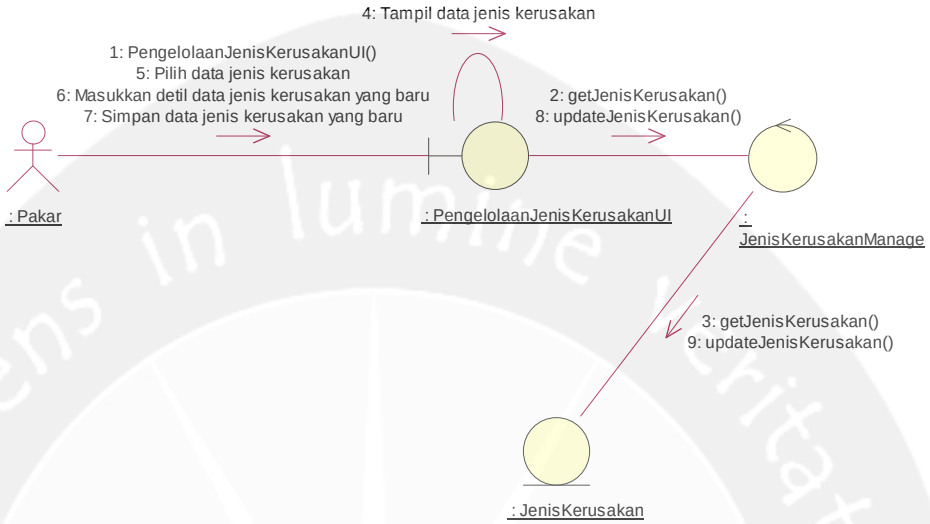
2.2.5 Pengelolaan Jenis Kerusakan

2.2.5.1 Tambah Jenis Kerusakan



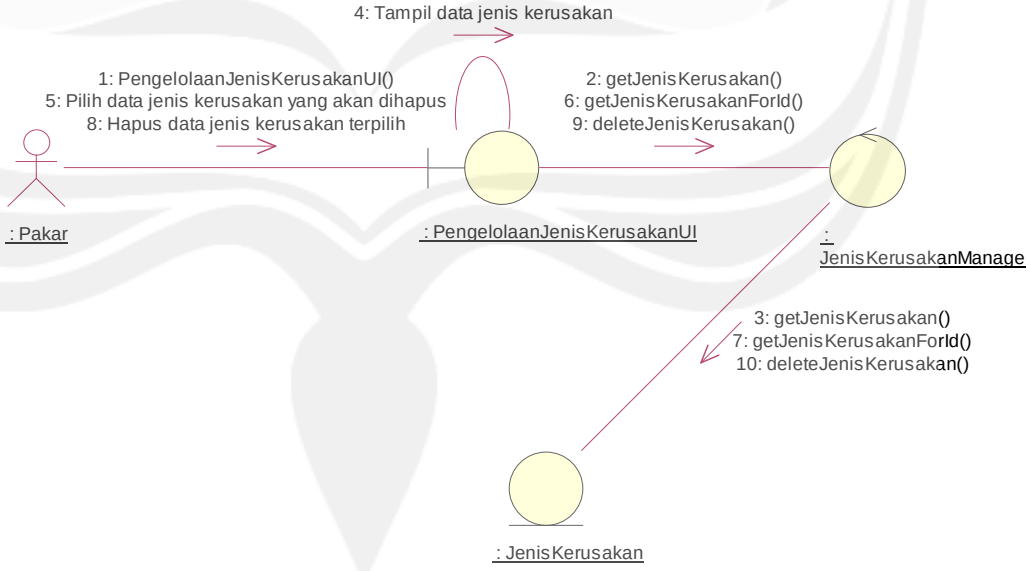
Gambar 14. Collaboration Diagram: Pengelolaan Jenis Kerusakan – Tambah Jenis Kerusakan

2.2.5.2 Ubah Jenis Kerusakan



Gambar 15. Collaboration Diagram: Pengelolaan Jenis Kerusakan – Ubah Jenis Kerusakan

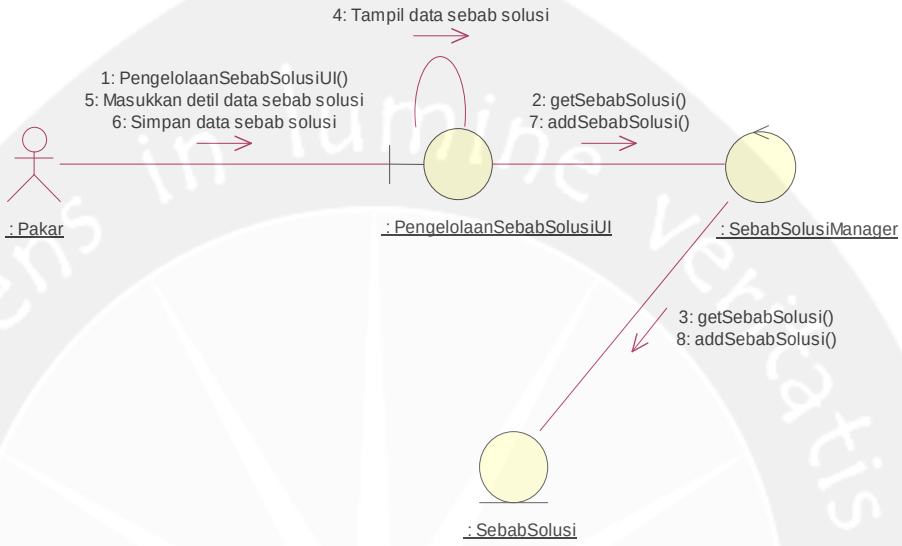
2.2.5.3 Hapus Jenis Kerusakan



Gambar 16. Collaboration Diagram: Pengelolaan Jenis Kerusakan – Hapus Jenis Kerusakan

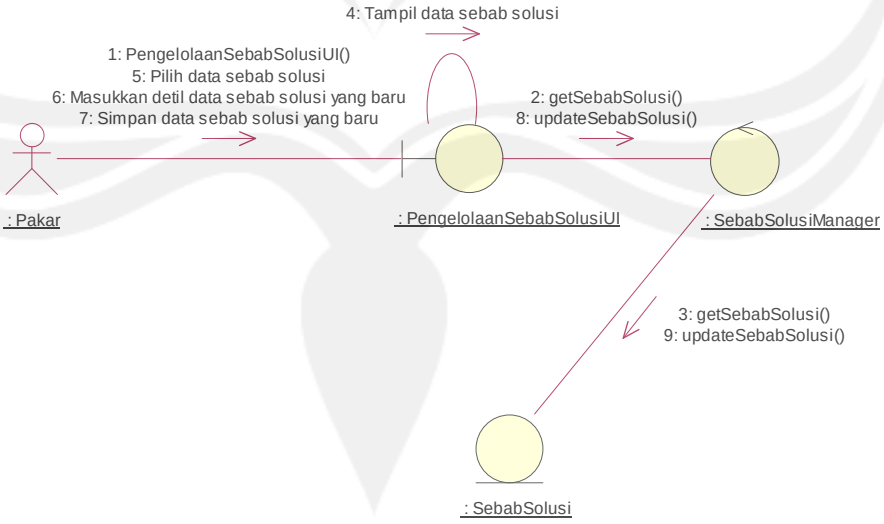
2.2.6 Pengelolaan Penyebab dan Solusi

2.2.6.1 Tambah Penyebab dan Solusi



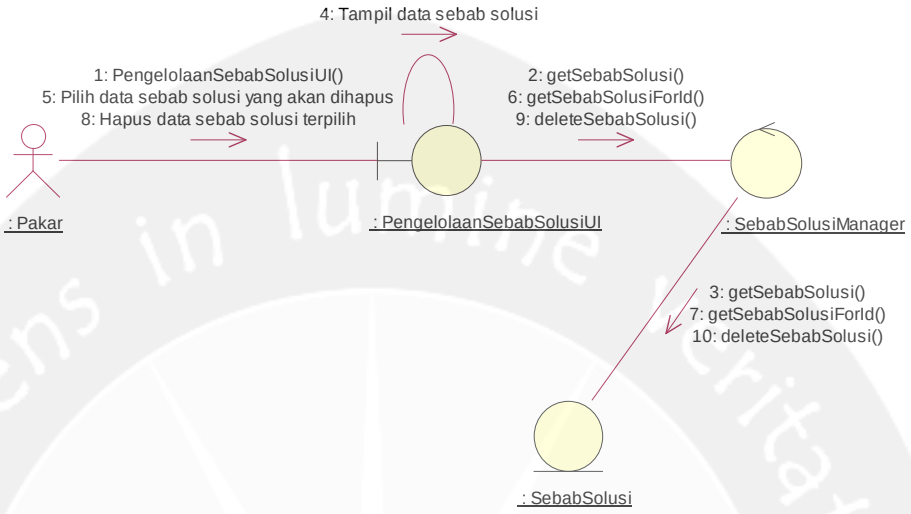
Gambar 17. Collaboration Diagram: Pengelolaan Penyebab dan Solusi – Tambah Penyebab dan Solusi

2.2.6.2 Ubah Penyebab dan Solusi



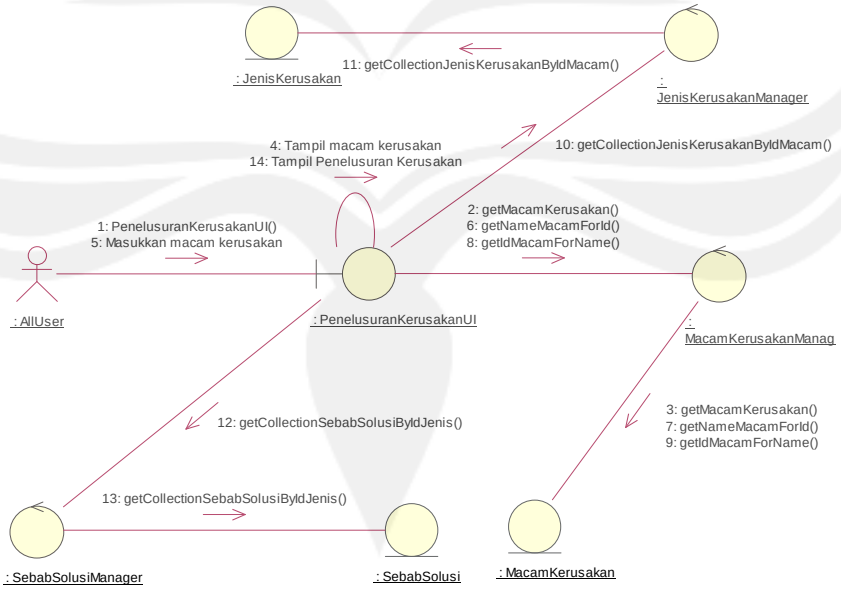
Gambar 18. Collaboration Diagram: Pengelolaan Penyebab dan Solusi – Ubah Penyebab dan Solusi

2.2.6.3 Hapus Penyebab dan Solusi



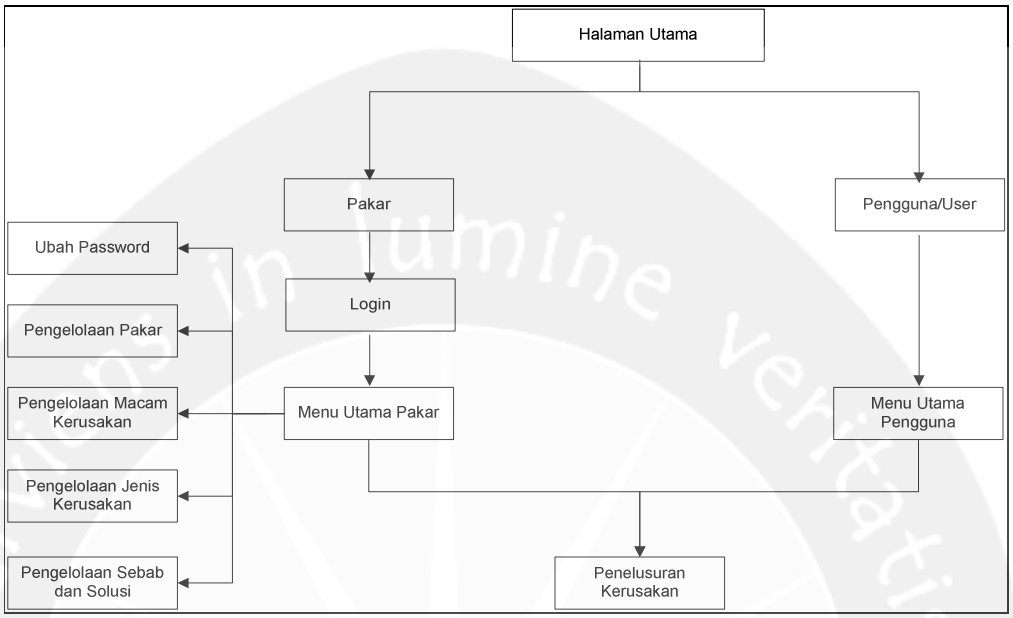
Gambar 19. Collaboration Diagram: Pengelolaan Penyebab dan Solusi – Hapus Penyebab dan Solusi

2.2.7 Penelusuran Kerusakan



Gambar 20. Collaboration Diagram: Penelusuran Kerusakan

3 Rancangan Arsitektur



Gambar 1. Rancangan Arsitektur SiPaTrou

4 Deskripsi Dekomposisi

4.1 Dekomposisi Data

4.1.1 Deskripsi Entitas Data Pakar

Nama	Tipe	Panjang	Keterangan
Nama_Pakar	**Character**	**20**	**Nama pakar, Primary Key**
Password	Character	5	Password pakar

Tabel 1. Deskripsi Entitas Data Pakar

4.1.2 Deskripsi Entitas Data Macam Kerusakan

Nama	Tipe	Panjang	Keterangan
Id_Macam	**Character**	**10**	**Id macam, Primary Key**
Macam	Varchar	75	Nama macam kerusakan

Tabel 2. Deskripsi Entitas Data Macam_Kerusakan

4.1.3 Deskripsi Entitas Data Jenis Kerusakan

Nama	Tipe	Panjang	Keterangan
Id_Jenis	**Character**	**10**	**Id jenis, Primary Key**
Jenis	Varchar	50	Nama jenis kerusakan
Id_Macam	**Character**	**10**	**Id macam, Foreign Key**

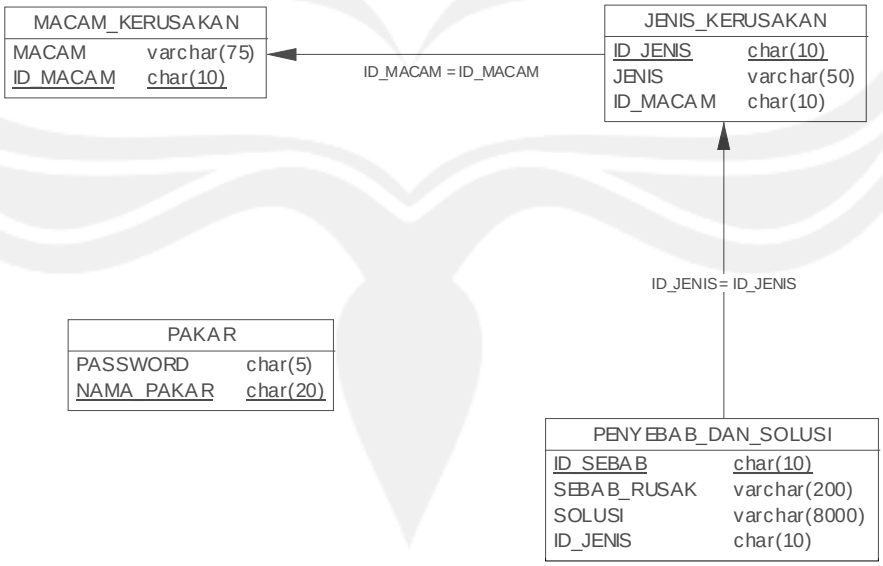
Tabel 3. Deskripsi Entitas Data Jenis_Kerusakan

4.1.4 Deskripsi Entitas Data Penyebab dan Solusi

Nama	Tipe	Panjang	Keterangan
Id_Sebab	**Character**	**10**	**Id sebab, Primary Key**
Sebab	Varchar	200	Sebab kerusakan
Solusi	Varchar	8000	Solusi kerusakan
Id_Jenis	**Character**	**10**	**Id jenis, Foreign Key**

Tabel 4. Deskripsi Entitas Data Penyebab_dan_Solusi

4.2 Conceptual Data Model

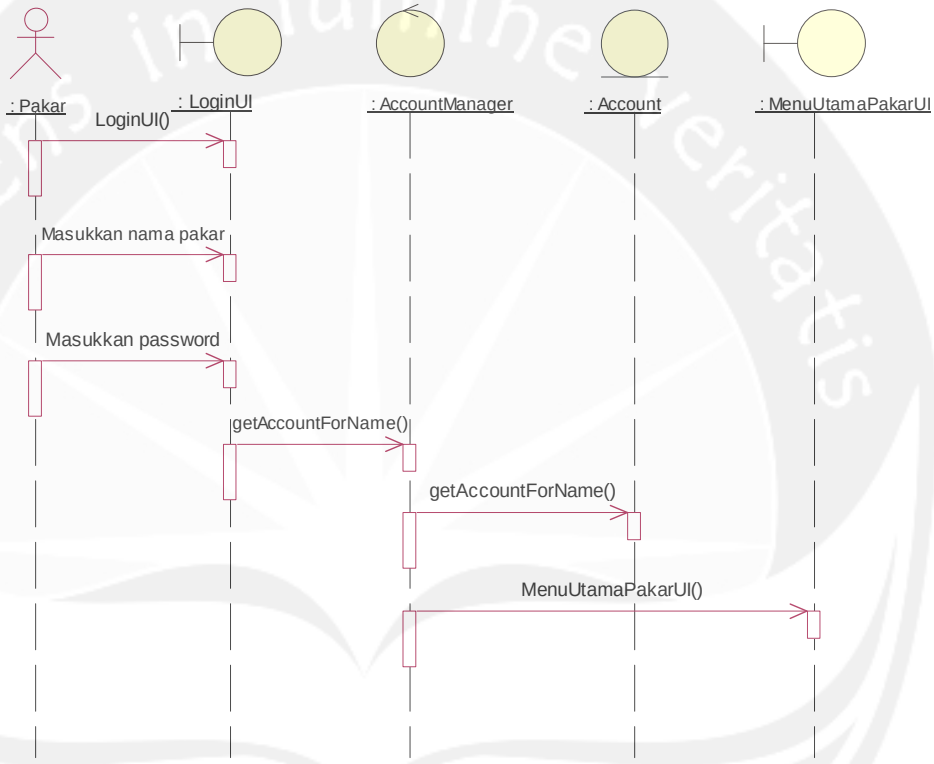


Gambar 1. Conceptual Data Model

5 Design Model

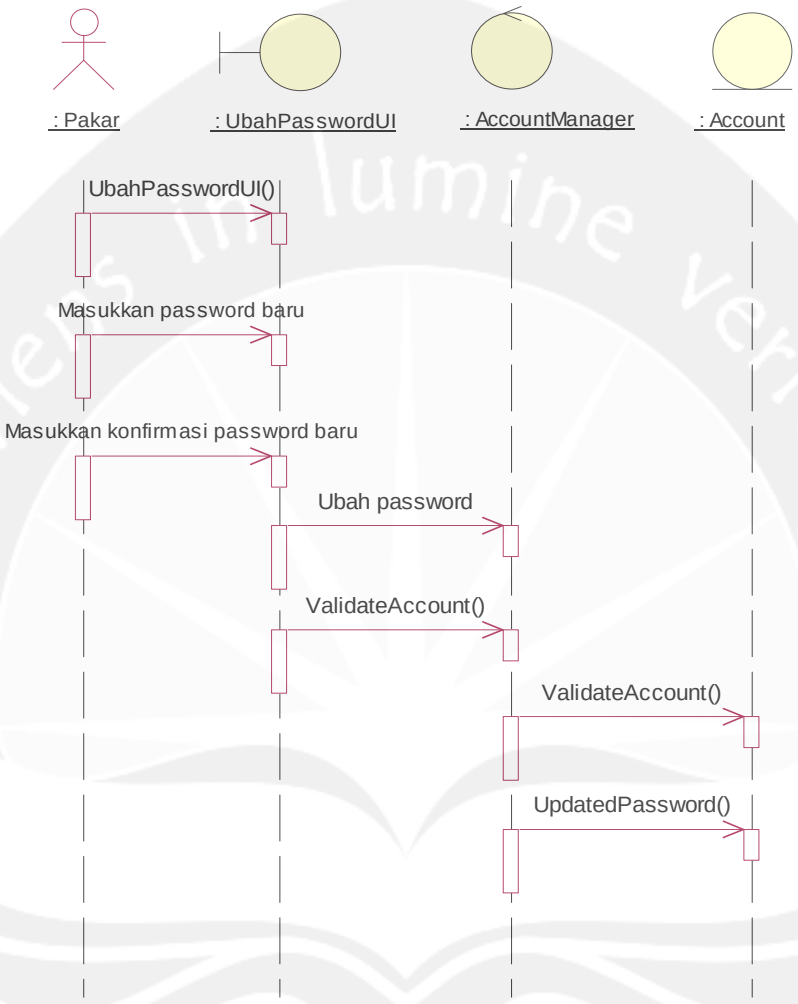
5.1 Sequence diagram

5.1.1 Login



Gambar 1. Sequence Diagram: Login

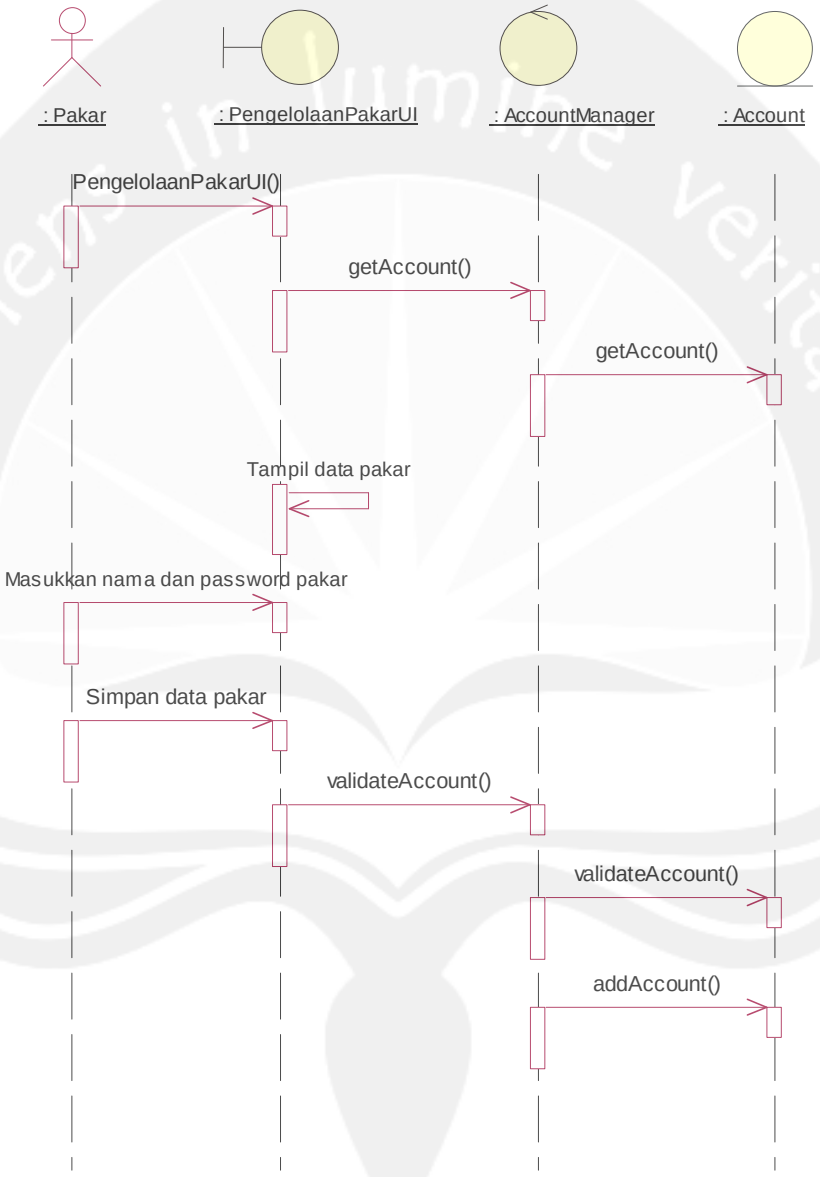
5.1.2 Ubah Password



Gambar 2. Sequence Diagram: Ubah Password

5.1.3 Pengelolaan Pakar

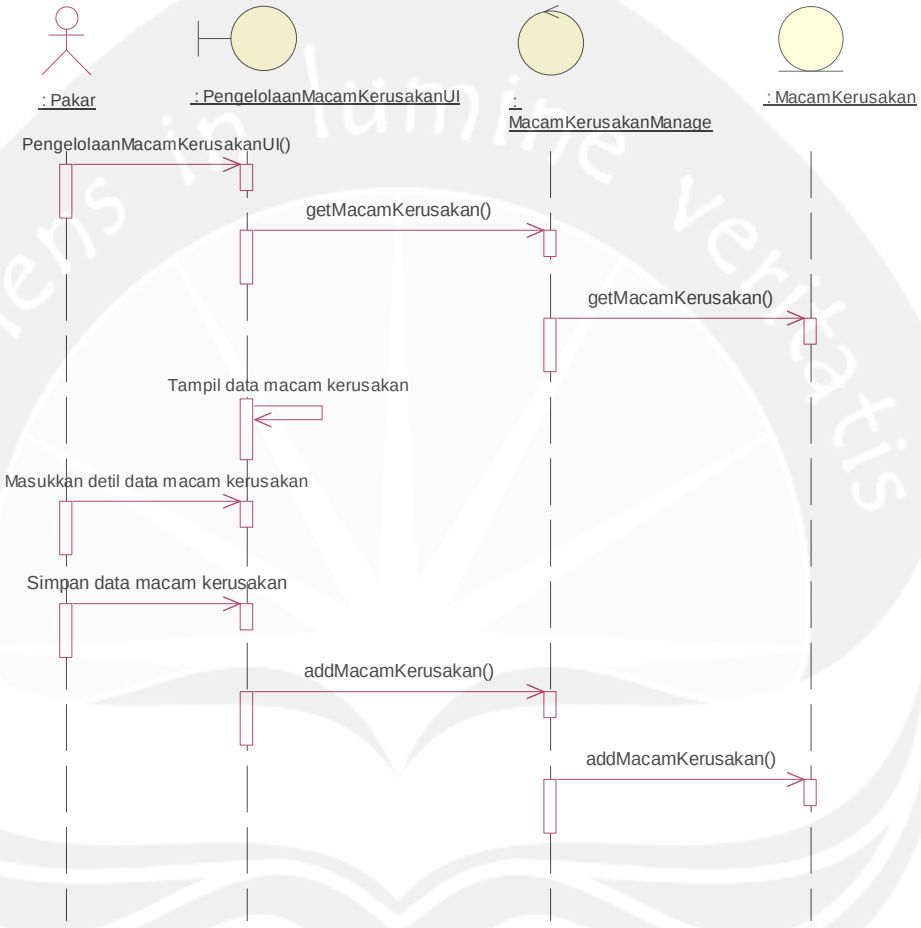
5.1.3.1 Tambah Pakar



Gambar 3. Sequence Diagram: Pengelolaan Pakar – Tambah Pakar

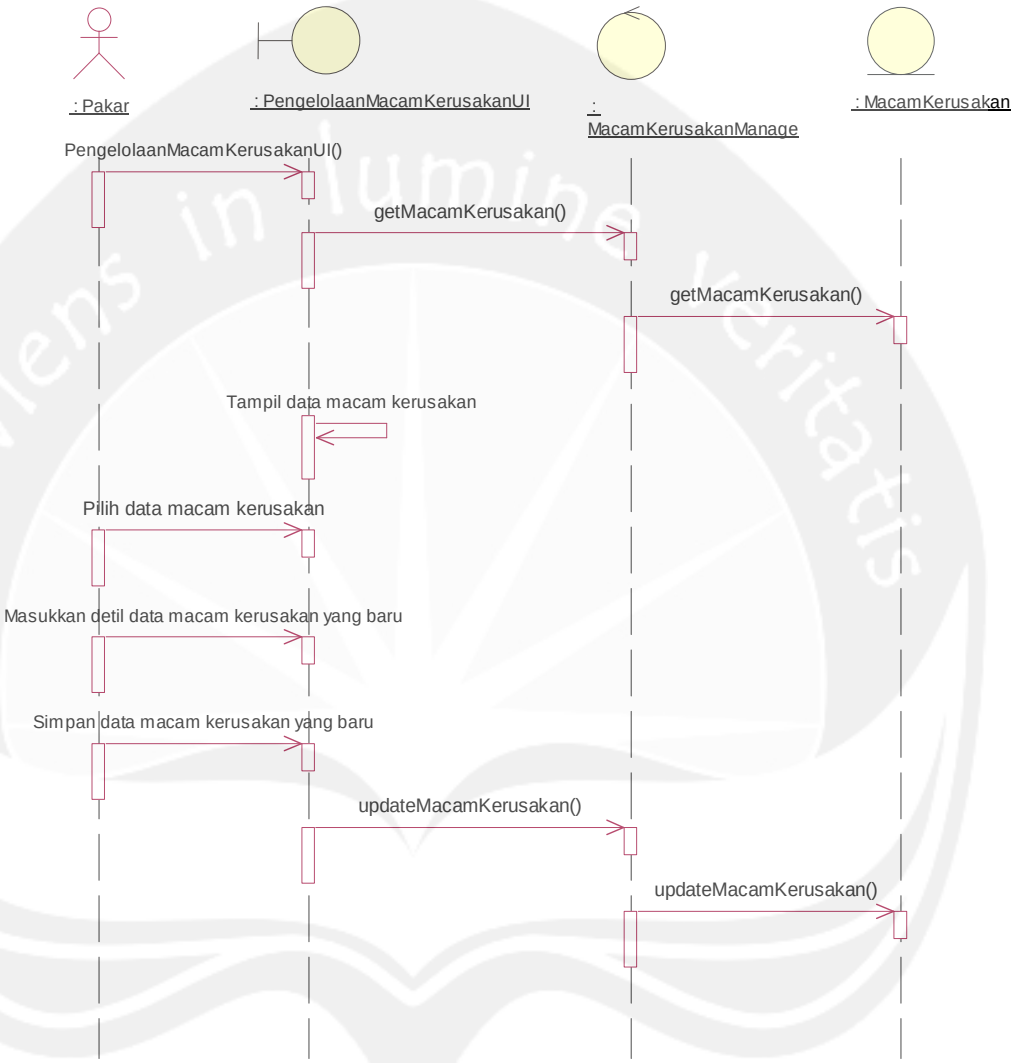
5.1.4 Pengelolaan Macam Kerusakan

5.1.4.1 Tambah Macam Kerusakan



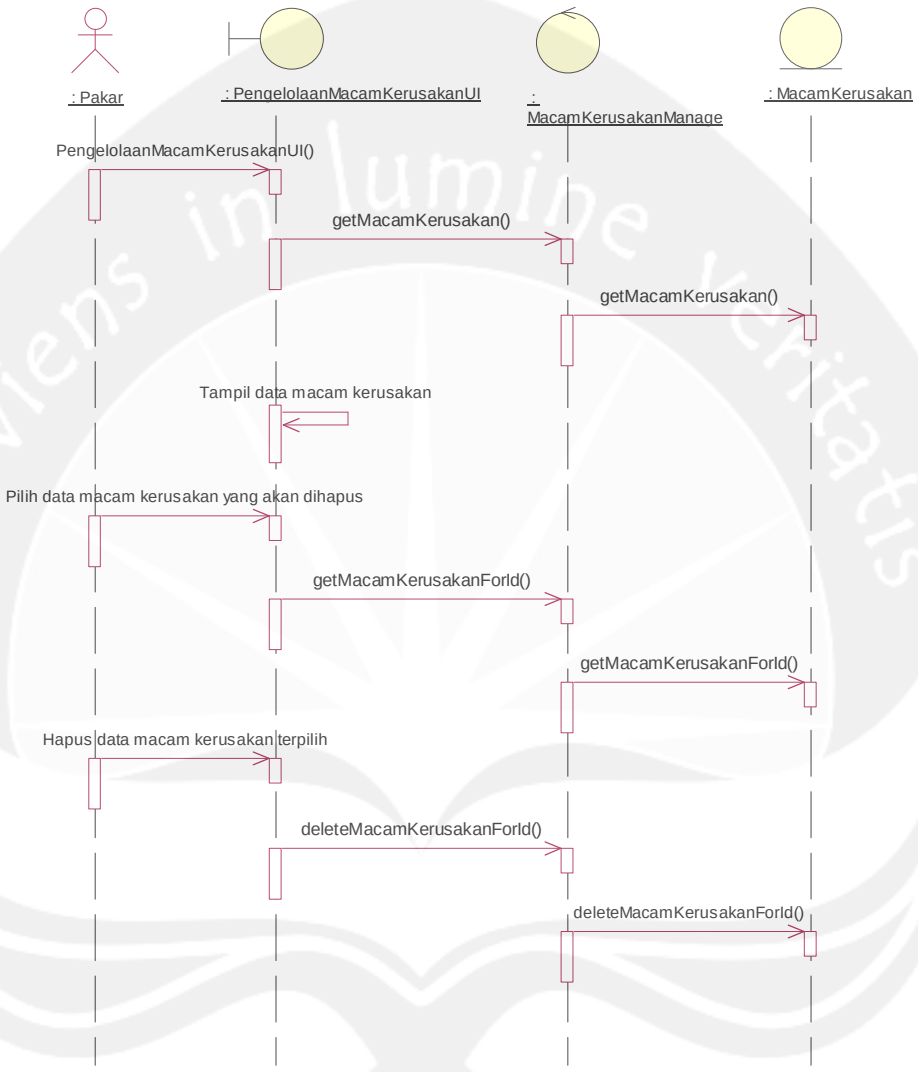
Gambar 4. Sequence Diagram: Pengelolaan Macam Kerusakan – Tambah Macam Kerusakan

5.1.4.2 Ubah Macam Kerusakan



Gambar 5. Sequence Diagram: Pengelolaan Macam Kerusakan – Ubah Macam Kerusakan

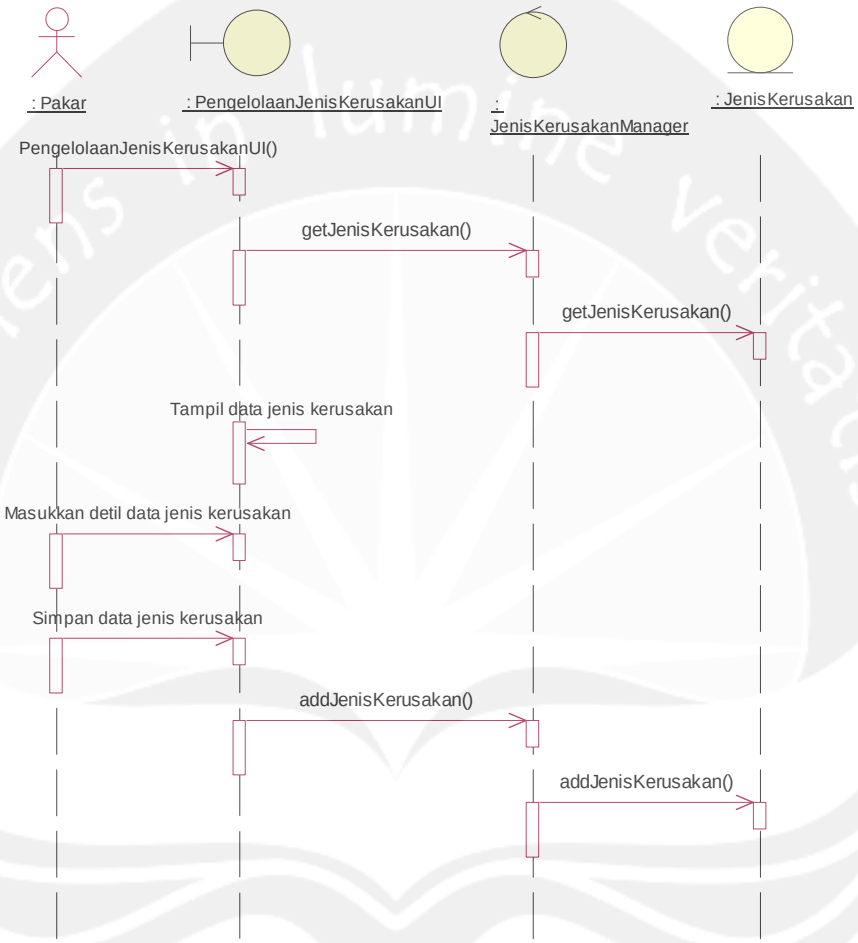
5.1.4.3 Hapus Macam Kerusakan



Gambar 6. Sequence Diagram: Pengelolaan Macam Kerusakan – Hapus Macam Kerusakan

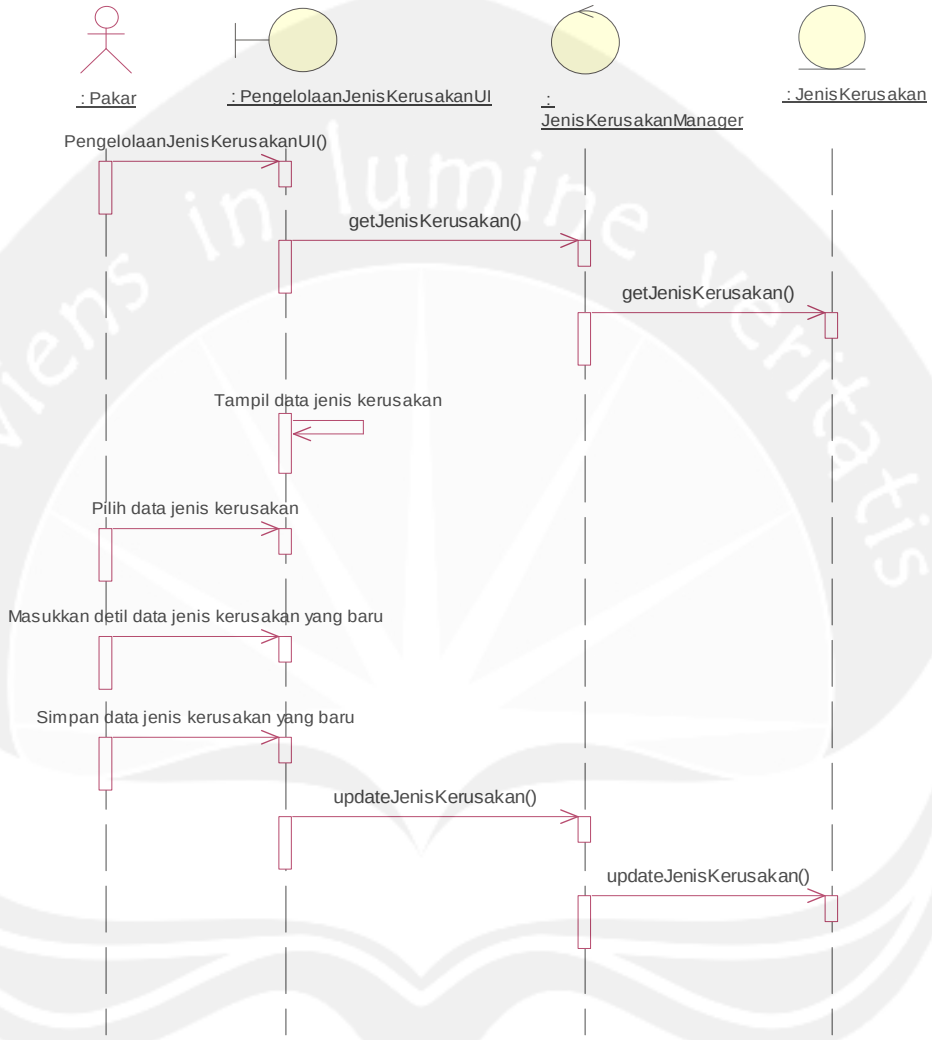
5.1.5 Pengelolaan Jenis Kerusakan

5.1.5.1 Tambah Jenis Kerusakan



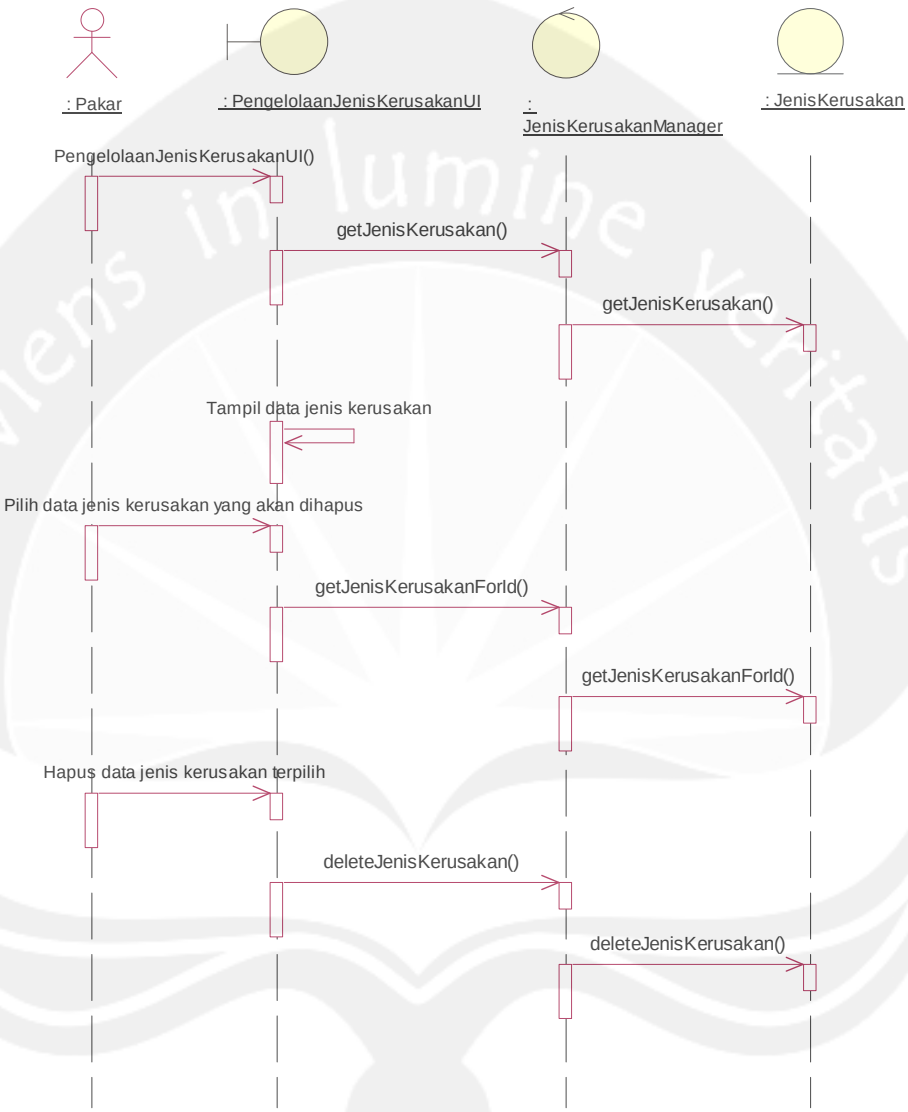
Gambar 7. Sequence Diagram: Pengelolaan Jenis Kerusakan – Tambah Jenis Kerusakan

5.1.5.2 Ubah Jenis Kerusakan



Gambar 8. Sequence Diagram: Pengelolaan Jenis Kerusakan – Ubah Jenis Kerusakan

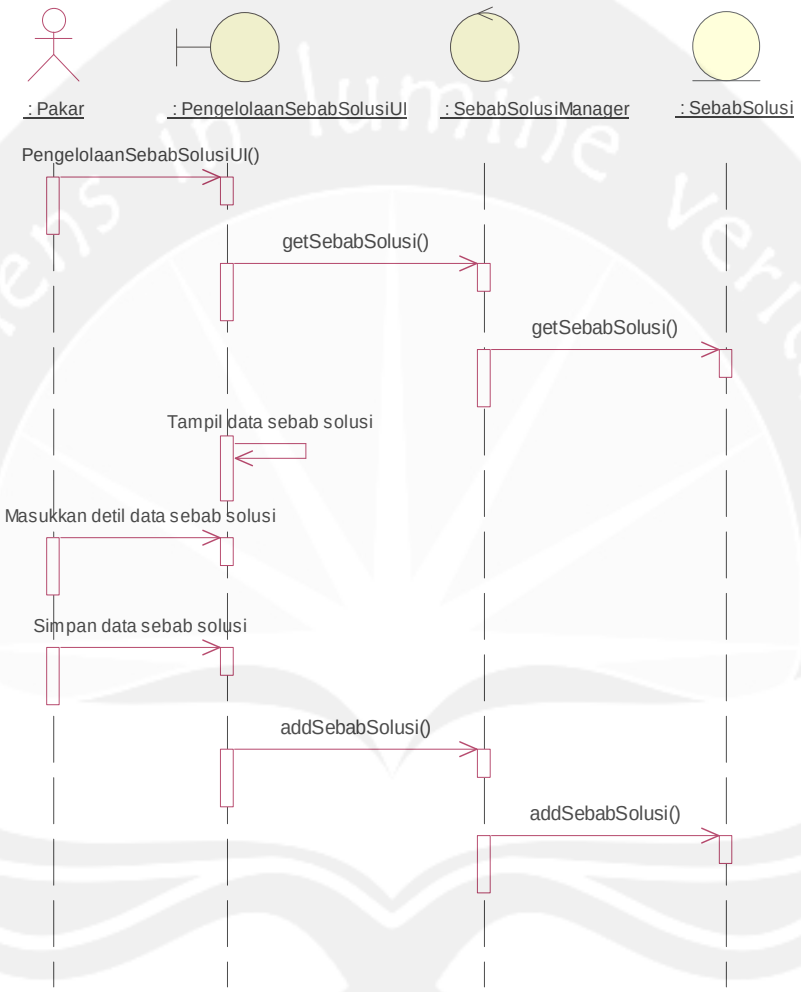
5.1.5.3 Hapus Jenis Kerusakan



Gambar 9. Sequence Diagram: Pengelolaan Jenis Kerusakan – Hapus Jenis Kerusakan

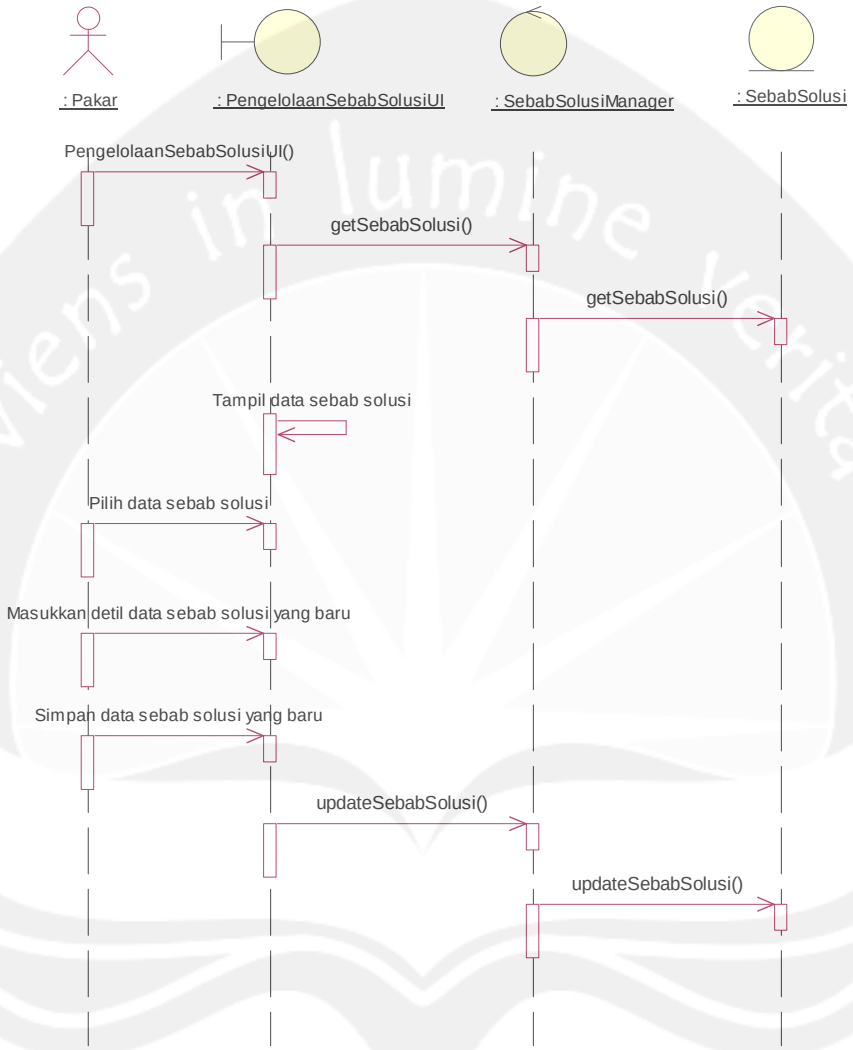
5.1.6 Pengelolaan Penyebab dan Solusi

5.1.6.1 Tambah Penyebab dan Solusi



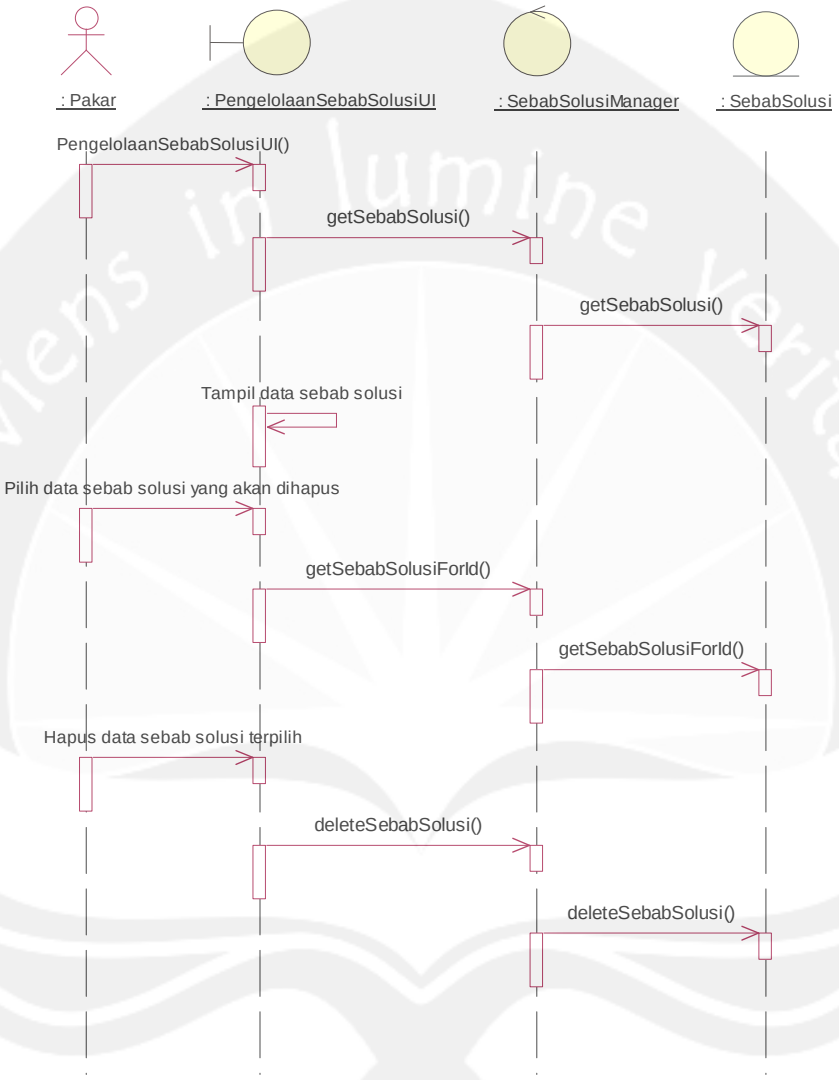
Gambar 10. Sequence Diagram: Pengelolaan Penyebab dan Solusi – Tambah Penyebab dan Solusi

5.1.6.2 Ubah Penyebab dan Solusi



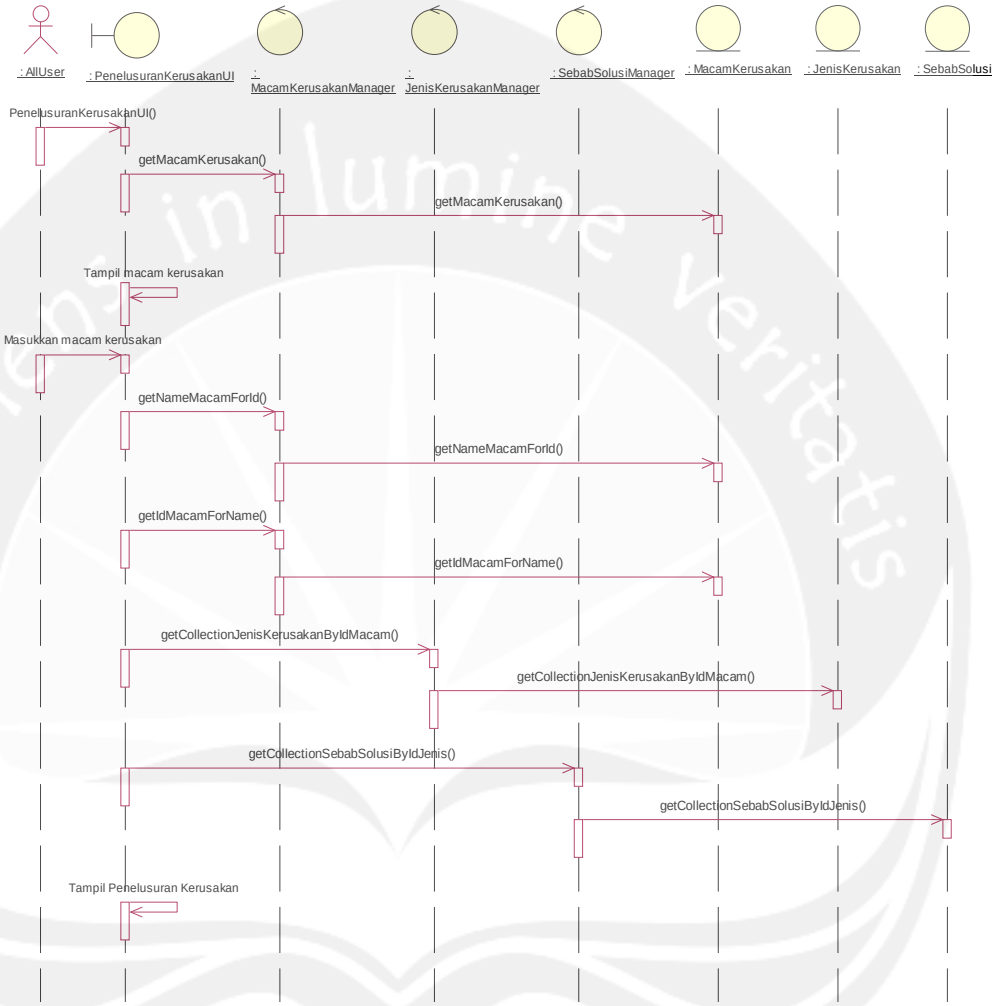
Gambar 11. Sequence Diagram: Pengelolaan Penyebab dan Solusi – Ubah Penyebab dan Solusi

5.1.6.3 Hapus Penyebab dan Solusi



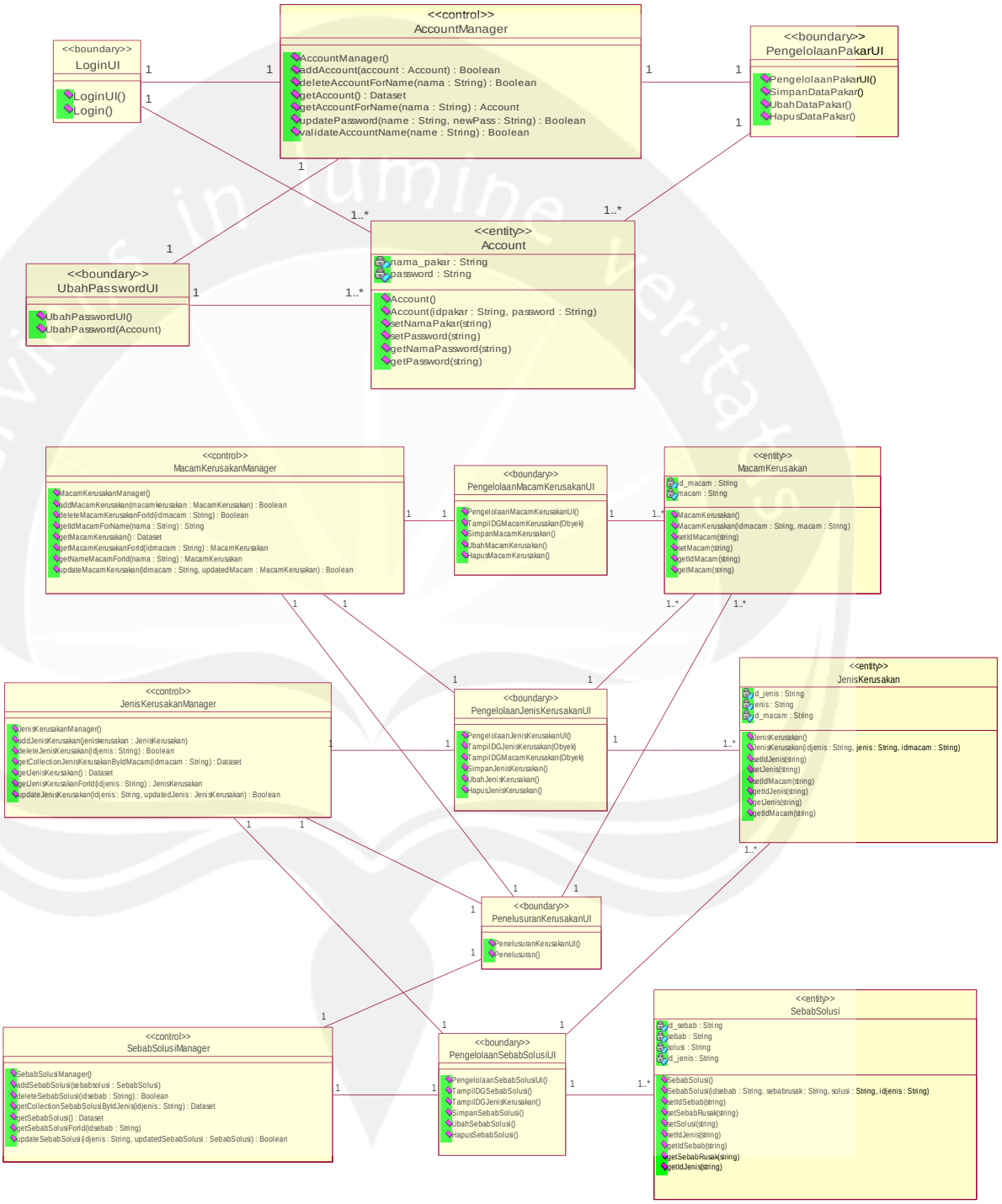
Gambar 12. Sequence Diagram: Pengelolaan Penyebab dan Solusi – Hapus Penyebab dan Solusi

5.1.7 Penelusuran Kerusakan



Gambar 13. Sequence Diagram: Penelusuran Kerusakan

5.2 Class Diagram



Gambar 14. Class Diagram

5.3 Diagram Specific Description

5.3.1 Specific Design Class LoginUI

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

5.3.2 Specific Design Class UbahPasswordUI

UbahPasswordUI	**<<boundary>>**
+UbahPasswordUI() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +UbahPassword() Operasi ini digunakan untuk mengubah password.	

5.3.3 Specific Design Class PengelolaanPakarUI

PengelolaanPakarUI	**<<boundary>>**
+PengelolaanPakarUI() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +SimpanPakar() Operasi ini digunakan untuk menyimpan data pakar baru. +HapusPakar() Operasi ini digunakan untuk menghapus data pakar yang dipilih. +TampilPakar() Operasi ini digunakan untuk menampilkan data pakar.	

5.3.4 Specific Design Class PengelolaanMacamKerusakanUI

PengelolaanMacamKerusakanUI	<<boundary>>
+PengelolaanMacamKerusakanUI() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +SimpanMacamKerusakan() Operasi ini digunakan untuk menyimpan data macam kerusakan baru. +UbahMacamKerusakan() Operasi ini digunakan untuk mengubah data macam kerusakan. +HapusMacamKerusakan() Operasi ini digunakan untuk menghapus data macam kerusakan yang dipilih. +TampilMacamKerusakan () Operasi ini digunakan untuk menampilkan data macam kerusakan.	

5.3.5 Specific Design Class PengelolaanJenisKerusakanUI

PengelolaanJenisKerusakanUI	<<boundary>>
+PengelolaanJenisKerusakanUI() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +SimpanJenisKerusakan() Operasi ini digunakan untuk menyimpan data jenis kerusakan baru. +UbahJenisKerusakan() Operasi ini digunakan untuk mengubah data jenis kerusakan. +HapusJenisKerusakan() Operasi ini digunakan untuk menghapus data jenis kerusakan yang dipilih.	

+TampilJenisKerusakan () Operasi ini digunakan untuk menampilkan data macam kerusakan.

5.3.6 Specific Design Class PengelolaanPenyebabSolusiUI

PengelolaanSebabSolusiUI	<<boundary>>
+PengelolaanPenyebabSolusiUI() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +SimpanSebabSolusi() Operasi ini digunakan untuk menyimpan data sebab dan solusi kerusakan baru. +UbahSebabSolusi() Operasi ini digunakan untuk mengubah data sebab dan solusi kerusakan. +HapusSebabSolusi() Operasi ini digunakan untuk menghapus data sebab dan solusi kerusakan yang dipilih. +TampilSebabSolusi() Operasi ini digunakan untuk menampilkan data sebab dan solusi kerusakan.	

5.3.7 Specific Design Class PenelusuranKerusakanUI

PenelusuranKerusakanUI	<<boundary>>
+PenelusuranKerusakanUI() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +Penelusuran() Operasi ini digunakan untuk melakukan penelusuran kerusakan.	

5.3.8 Specific Design Class AccountManager

AccountManager	**<<control>>**
+AccountManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +addAccount(Account) Operasi ini digunakan untuk menyimpan data account(pakar) ke basis data. +deleteAccountForName(string) Operasi ini digunakan untuk menghapus data account(pakar) yang tersimpan di basis data. +getAccount() Operasi ini digunakan untuk mengambil data account(pakar) yang tersimpan di basis data. +getAccountForName(string) Operasi ini digunakan untuk mengambil data account(pakar) berdasarkan nama pakar. +updatePassword(string) Operasi ini digunakan untuk mengubah password pada account berdasarkan nama account(pakar). +validateAccountName(string) Operasi ini digunakan untuk mengecek data account yang dimasukkan oleh pakar.	

5.3.9 Specific Design Class MacamKerusakanManager

MacamKerusakanManager	**<<control>>**
+MacamKerusakanManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +addMacamKerusakan(MacamKerusakan) Operasi ini digunakan untuk menyimpan data macam kerusakan ke basis data. +deleteMacamKerusakanForId(string) Operasi ini digunakan untuk menghapus data macam kerusakan berdasarkan id macam. +getIdMacamForName(string)	

Operasi ini digunakan untuk mengambil id macam kerusakan berdasarkan macam kerusakan. +getMacamKerusakan() Operasi ini digunakan untuk mengambil data macam kerusakan yang tersimpan di basis data. +getMacamKerusakanForId(string) Operasi ini digunakan untuk mengambil data macam kerusakan berdasarkan id macam. +getNameMacamForId(string) Operasi ini digunakan untuk mengambil data macam kerusakan berdasarkan macam kerusakan. +updateMacamKerusakan(Obyek) Operasi ini digunakan untuk mengubah data macam kerusakan yang tersimpan di basis data.

5.3.10 Specific Design Class JenisKerusakanManager

JenisKerusakanManager	**<<control>>**
+JenisKerusakanManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +addJenisKerusakan(JenisKerusakan) Operasi ini digunakan untuk menyimpan data jenis kerusakan ke basis data. +deleteJenisKerusakan(string) Operasi ini digunakan untuk menghapus data jenis kerusakan berdasarkan id jenis. +getCollectionJenisKerusakanByIdMacam(string) Operasi ini digunakan untuk mengambil data jenis kerusakan berdasarkan id macam. +getJenisKerusakan() Operasi ini digunakan untuk mengambil data jenis kerusakan yang tersimpan di basis data. +getJenisKerusakanForId(string) Operasi ini digunakan untuk mengambil data jenis kerusakan berdasarkan id jenis. +updateJenisKerusakan(Obyek)	

Operasi ini digunakan untuk mengubah data jenis kerusakan yang tersimpan di basis data.

5.3.11 Specific Design Class SebabSolusiManager

SebabSolusiManager	**<<control>>**
+SebabSolusiManager() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +addSebabSolusi(SebabSolusi) Operasi ini digunakan untuk menyimpan data sebab dan solusi kerusakan ke basis data. +deleteSebabSolusi(string) Operasi ini digunakan untuk menghapus data sebab dan solusi kerusakan yang tersimpan di basis data berdasarkan id sebab. +getCollectionSebabSolusiByIdJenis(string) Operasi ini digunakan untuk mengambil data sebab dan solusi kerusakan yang tersimpan di basis data berdasarkan id jenis. +getSebabSolusi() Operasi ini digunakan untuk mengambil data sebab dan solusi kerusakan yang tersimpan di basis data. +getSebabSolusiForId(string) Operasi ini digunakan untuk mengambil data sebab dan solusi kerusakan yang tersimpan di basis data berdasarkan id sebab. +updateSebabSolusi(Obyek) Operasi ini digunakan untuk mengubah data sebab dan solusi kerusakan yang tersimpan di basis data.	

5.3.12 Specific Design Class Account

Account	**<<entity>>**
- Nama_pakar: String Atribut ini digunakan untuk menyimpan nama pakar.	

- Password: String Atribut ini digunakan untuk menyimpan password pakar.
+Account() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +setNamaPakar(string) +setPassword(string) Merupakan mutator dari class Account. +getNamaPakar(string) +getPassword(string) Merupakan accesor dari class Account.

5.3.13 Specific Design Class MacamKerusakan

MacamKerusakan	**<<entity>>**
- Id_macam: String Atribut ini digunakan untuk menyimpan id macam. - Macam: String Atribut ini digunakan untuk menyimpan macam kerusakan.	
+MacamKerusakan() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +setIdMacam(string) +setMacam(string) Merupakan mutator dari class MacamKerusakan. +getIdMacam(string) +getMacams(string) Merupakan accesor dari class MacamKerusakan.	

5.3.14 Specific Design Class JenisKerusakan

JenisKerusakan	**<<entity>>**
- Id_jenis: String Atribut ini digunakan untuk menyimpan id jenis. - Jenis: String Atribut ini digunakan untuk menyimpan jenis kerusakan. - Id_macam: String Atribut ini digunakan untuk menyimpan id macam.	
+JenisKerusakan()	

Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +setIdJenis(string) +setJenis(string) +setIdMacam(string) Merupakan mutator dari class JenisKerusakan. +getIdJenis(string) +getJenis(string) +getIdMacam(string) Merupakan accesor dari class JenisKerusakan.

5.3.15 Specific Design Class SebabSolusi

SebabSolusi	**<<entity>>**
- Id_sebab: String Atribut ini digunakan untuk menyimpan id sebab. - Sebab_rusak: String Atribut ini digunakan untuk menyimpan sebab kerusakan. - Solusi: String Atribut ini digunakan untuk menyimpan solusi kerusakan. - Id_jenis: String Atribut ini digunakan untuk menyimpan id jenis.	
+SebabSolusi() Default konstruktor, digunakan untuk inisialisasi semua attribute dari kelas ini. +setIdSebab(string) +setSebabRusak(string) +setSolusi(string) +setIdJenis(string) Merupakan mutator dari class SebabSolusi. +getIdSebab(string) +getSebabRusak(string) +getSolusi(string) +getIdJenis(string) Merupakan accesor dari class SebabSolusi.	

6 Deskripsi Perancangan Antarmuka

6.1 Form Pilihan Menu

Gambar 1. PilihanMenu.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk melakukan pemilihan menu.

1) *RadioButton* rbYa dan rbTidak untuk menerima masukkan pilihan menu.
2) Tombol Lanjut untuk menampilkan menu sesuai pilihan menu yang dipilih.
3) Tombol Batal untuk keluar dari sistem.

Event

Lanjut:

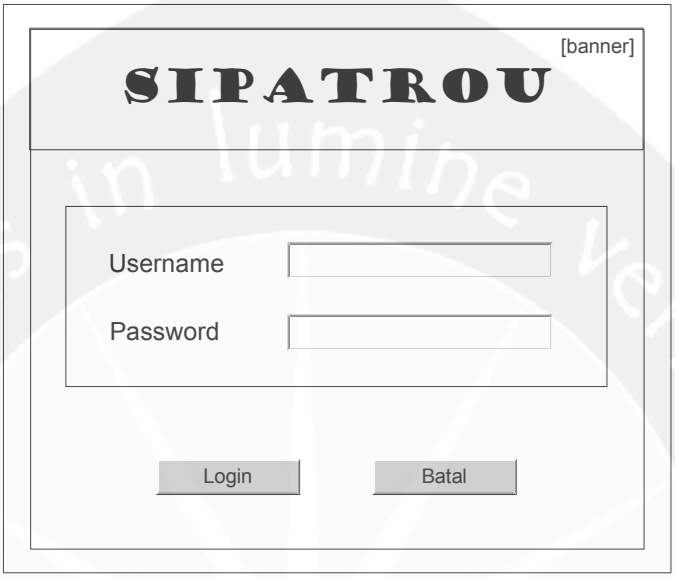
1. Pengguna memilih menu pilihan.
2. Pengguna menekan tombol Lanjut.
3. Sistem akan menampilkan menu utama sesuai dengan menu pilihan yang dipilih.

Batal:

1. Pengguna menekan tombol Batal.

2. Pengguna keluar dari sistem.

6.2 Form Login



Gambar 2. Login.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk *use case Login*.

1) *Textbox* txtUsername untuk menerima masukan nama pakar.
2) *Textbox* txtPassword untuk menerima masukan *password*.
3) Tombol *Login* untuk *event Login* pengguna bersangkutan (Pakar).
4) Tombol Batal untuk membatalkan proses Login.

Event

Login:

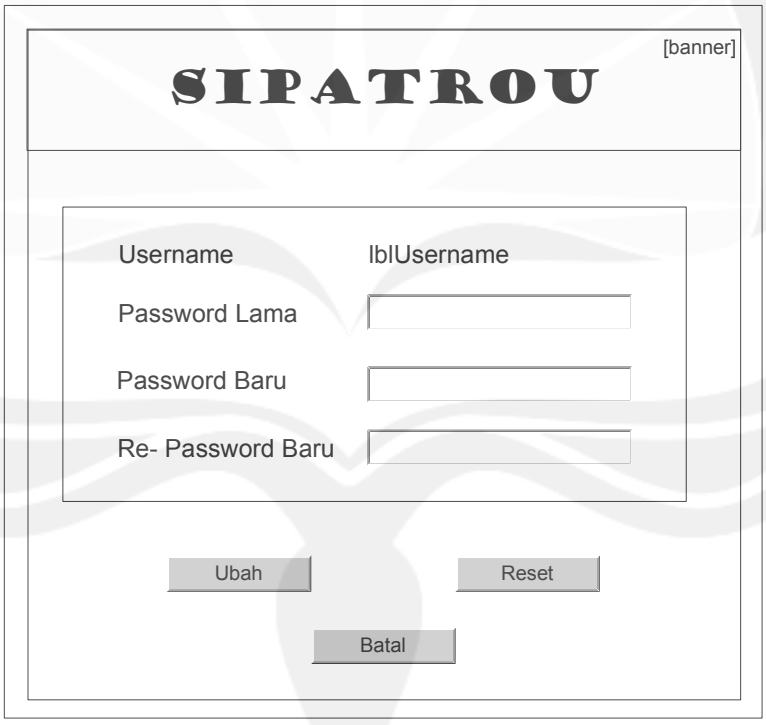
1. Pakar memasukkan nama pakar dan *password*.
2. Pakar menekan tombol Login.

3. Sistem akan membuat sebuah *object* dari *class* Account, kemudian memasukkan nama pakar dan *password*. Kemudian mengecek dalam basis data apakah pakar tersebut valid.
4. Sistem akan mengambil semua informasi pakar tersebut. Kemudian akan menampilkan menu utama pakar.

Batal:

1. Pakar menekan tombol Batal.
2. Pakar keluar dari sistem.

6.3 Form Ubah Password



Gambar 3. Ubah Pasword.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk mengubah password.

1) Label lblUsername untuk menampilkan nama pakar.
2) *Textbox* txtPassLama untuk menerima masukan *password* lama.
3) *Textbox* txtPassBaru untuk menerima masukan *password* baru.
4) *Textbox* txtRePass untuk menerima masukan konfirmasi *password* baru.
5) Tombol Ubah untuk mengubah password pakar.
6) Tombol Reset untuk mengosongkan semua TextBox.
7) Tombol Batal untuk membatalkan proses pengubahan *password.*

Event

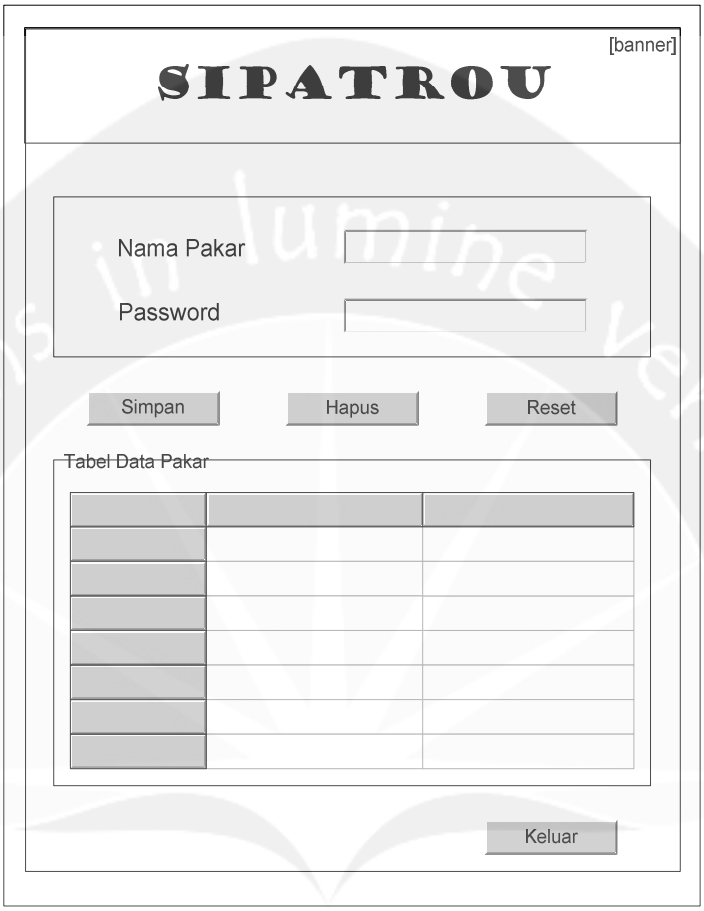
Ubah:

1. Pakar memasukkan *password* lama.
2. Pakar memasukkan *password* lama.
3. Pakar memasukkan *password* baru dan konfirmasi *password* baru.
4. Pakar menekan tombol Ubah.
5. Sistem akan melakukan perubahan *password* lama dengan *password* baru dengan membuat *object* dari *class* Account, kemudian memasukkan *password* baru, yang sebelumnya dilakukan pengecekan terlebih dahulu untuk *password* lama.

Batal:

1. Pakar menekan tombol Batal.
2. Sistem akan menampilkan halaman menu utama pakar.

6.4 Form Pengelolaan Pakar



Gambar 4. PengelolaanPakar.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk mengelola pakar.

1) *Textbox* txtNama untuk menerima masukan nama pakar.
2) *Textbox* txtPassword untuk menerima masukan *password*.
3) DataGrid untuk menampilkan seluruh data pakar yang ada dalam basis data.

4) Tombol Simpan untuk menyimpan data pakar.
5) Tombol Hapus untuk menghapus data pakar.
6) Tombol Reset untuk mengosongkan semua TextBox serta merefresh DataGrid.
7) Tombol Keluar untuk mengembalikan ke halaman menu utama pakar.

Event

Simpan:

1. Pakar memasukkan nama pakar dan *password*.
2. Pakar menekan tombol Simpan.
3. Sistem akan membuat sebuah *object* dari *class* Account, kemudian memasukkan nama dan *password* pakar. Kemudian mengecek apakah data sudah ada dalam basis data. Jika belum sistem akan memasukkan data baru ke dalam basis data.

Hapus:

1. Pakar memilih data pakar yang akan dihapus.
2. Pakar menekan tombol Hapus.
3. Sistem akan membuat sebuah *object* dari *class* Account. Jika data tidak digunakan, maka sistem akan menghapus data pakar dari dalam basis data berdasarkan nama pakar.

Reset:

1. Pakar menekan tombol Reset.
2. Sistem akan mengosongkan semua TextBox dan dan merefresh data yang ditampilkan pada DataGrid.

Keluar:

1. Pakar menekan tombol Keluar.
2. Sistem akan menampilkan halaman menu utama pakar.

6.5 Form Pengelolaan Macam Kerusakan

Gambar 5. PengelolaanMacamKerusakan.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk mengelola macam kerusakan.

1) *Textbox* txtIdMacam untuk menerima masukan id macam.
2) *Textbox* txtMacam untuk menerima masukan macam kerusakan.

3) DataGrid DGMacam untuk menampilkan seluruh data macam kerusakan yang ada dalam basis data.
4) Tombol Simpan untuk menyimpan data macam kerusakan.
5) Tombol Ubah untuk mengubah data macam kerusakan.
6) Tombol Hapus untuk menghapus data macam kerusakan.
7) Tombol Reset untuk mengosongkan semua TextBox serta merefresh DataGrid.
8) Tombol Keluar untuk mengembalikan ke halaman menu utama pakar.

Event

Simpan:

1. Pakar memasukkan id macam dan macam kerusakan.
2. Pakar menekan tombol Simpan.
3. Sistem akan membuat sebuah *object* dari *class* MacamKerusakan, kemudian memasukkan data macam kerusakan. Kemudian mengecek apakah data sudah ada dalam basis data. Jika belum sistem akan memasukkan data baru ke dalam basis data.

Ubah:

1. Pakar memilih data macam kerusakan yang akan diubah.
2. Pakar menekan tombol Ubah.
3. Sistem akan membuat sebuah *object* dari *class* MacamKerusakan, kemudian memasukkan data macam kerusakan. Kemudian mengecek apakah data sudah ada dalam basis data, jika belum maka sistem akan mengubah data macam kerusakan yang bersangkutan dalam basis data.

Hapus:

1. Pakar memilih data macam kerusakan yang akan dihapus.
2. Pakar menekan tombol Hapus.
3. Sistem akan membuat sebuah *object* dari *class* MacamKerusakan. Jika data tidak sedang digunakan, maka sistem akan menghapus data macam kerusakan dari dalam basis data berdasarkan id macam.

Reset:

1. Pakar menekan tombol Reset.
2. Sistem akan mengosongkan semua TextBox dan dan merefresh data yang ditampilkan pada DataGrid.

Keluar:

1. Pakar menekan tombol Keluar.
2. Sistem akan menampilkan halaman menu utama pakar.

6.6 Form Pengelolaan Jenis Kerusakan

Gambar 6. PengelolaanJenisKerusakan.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk mengelola jenis kerusakan.

1) *Textbox* txtIdJenis untuk menerima masukan id jenis.
2) *Textbox* txtJenis untuk menerima masukan jenis kerusakan.
3) *Combobox* cbIdMacam untuk menerima masukan id macam.

4) DataGrid DGJenis untuk menampilkan seluruh data jenis kerusakan yang ada dalam basis data.
5) DataGrid DGMacam untuk menampilkan seluruh data macam kerusakan yang ada dalam basis data.
6) Tombol Simpan untuk menyimpan data jenis kerusakan.
7) Tombol Ubah untuk mengubah data jenis kerusakan.
8) Tombol Hapus untuk menghapus data jenis kerusakan.
9) Tombol Reset untuk mengosongkan semua TextBox, ComboBox serta merefresh DataGrid.
10) Tombol Keluar untuk mengembalikan ke halaman menu utama pakar.

Event

Simpan:

1. Pakar memasukkan id jenis, jenis kerusakan dan id macam.
2. Pakar menekan tombol Simpan.
3. Sistem akan membuat sebuah *object* dari *class* JenisKerusakan, kemudian memasukkan data jenis kerusakan. Kemudian mengecek apakah data sudah ada dalam basis data. Jika belum sistem akan memasukkan data baru ke dalam basis data.

Ubah:

1. Pakar memilih data macam kerusakan yang akan diubah.
2. Pakar menekan tombol Ubah.
3. Sistem akan membuat sebuah *object* dari *class* JenisKerusakan, kemudian memasukkan data jenis

kerusakan. Kemudian mengecek apakah data sudah ada dalam basis data, jika belum maka sistem akan mengubah data jenis kerusakan yang bersangkutan dalam basis data.

Hapus:

1. Pakar memilih data jenis kerusakan yang akan dihapus.
2. Pakar menekan tombol Hapus.
3. Sistem akan membuat sebuah *object* dari *class* JenisKerusakan. Jika data tidak sedang digunakan, maka sistem akan menghapus data jenis kerusakan dari dalam basis data berdasarkan id jenis.

Reset:

1. Pakar menekan tombol Reset.
2. Sistem akan mengosongkan semua TextBox, ComboBox dan merefresh data yang ditampilkan pada DataGrid.

Keluar:

1. Pakar menekan tombol Keluar.
2. Sistem akan menampilkan halaman menu utama pakar.

6.7 Form Pengelolaan Penyebab dan Solusi

Gambar 7. PengelolaanPenyebabSolusi.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk mengelola sebab dan solusi kerusakan.

1) *Textbox* txtIdSebab untuk menerima masukan id sebab.
2) *Textbox* txtSebab untuk menerima masukan sebab kerusakan.
3) *Combobox* cbIdJenis untuk menerima masukan id jenis.

4) *Textbox* txtSolusi untuk menerima masukan solusi kerusakan.
5) DataGrid DGSebab untuk menampilkan seluruh data sebab dan solusi kerusakan yang ada dalam basis data.
6) DataGrid DGJenis untuk menampilkan seluruh data jenis kerusakan yang ada dalam basis data.
7) Tombol Simpan untuk menyimpan data sebab dan solusi kerusakan.
8) Tombol Ubah untuk mengubah data sebab dan solusi kerusakan.
9) Tombol Hapus untuk menghapus data sebab dan solusi kerusakan.
10) Tombol Reset untuk mengosongkan semua TextBox, ComboBox serta merefresh DataGrid.
11) Tombol Keluar untuk mengembalikan ke halaman menu utama pakar.

Event

Simpan:

1. Pakar memasukkan id sebab, sebab kerusakan, id jenis, dan solusi kerusakan.
2. Pakar menekan tombol Simpan.
3. Sistem akan membuat sebuah *object* dari *class* SebabSolusi, kemudian memasukkan data sebab dan solusi kerusakan. Kemudian mengecek apakah data sudah ada dalam basis data. Jika belum sistem akan memasukkan data baru ke dalam basis data.

Ubah:

1. Pakar memilih data sebab dan solusi kerusakan yang akan diubah.

2. Pakar menekan tombol Ubah.
3. Sistem akan membuat sebuah *object* dari *class* SebabSolusi, kemudian memasukkan data sebab dan solusi kerusakan. Kemudian mengecek apakah data sudah ada dalam basis data, jika belum maka sistem akan mengubah data sebab dan solusi kerusakan yang bersangkutan dalam basis data.

Hapus:

1. Pakar memilih data sebab dan solusi kerusakan yang akan dihapus.
2. Pakar menekan tombol Hapus.
3. Sistem akan membuat sebuah *object* dari *class* SebabSolusi. Jika data tidak sedang digunakan, maka sistem akan menghapus data sebab dan solusi kerusakan dari dalam basis data berdasarkan id sebab.

Reset:

1. Pakar menekan tombol Reset.
2. Sistem akan mengosongkan semua TextBox, ComboBox dan merefresh data yang ditampilkan pada DataGrid.

Keluar:

1. Pakar menekan tombol Keluar.
2. Sistem akan menampilkan halaman menu utama pakar.

6.8 Form Penelusuran Kerusakan

Gambar 8. PenelusuranKerusakan.cs

Deskripsi

Rancangan antarmuka ini digunakan untuk menampilkan informasi penelusuran kerusakan.

1) *ComboBox* cbMacam untuk menerima masukan macam kerusakan *hardware*.
2) *TextBox* txtPertanyaan untuk menampilkan pertanyaan yang disediakan oleh sistem.

3) *RadioButton* rbYa dan rbTidak merupakan pilihan untuk menjawab pertanyaan yang diberikan oleh sistem.
4) *TextBox* txtSebab untuk menampilkan penyebab terjadinya kerusakan.
5) *TextBox* txtSolusi untuk menampilkan solusi untuk kerusakan yang terjadi.
6) Tombol Reset untuk mengosongkan semua TextBox serta ComboBox.
7) Tombol Lanjut untuk melakukan proses penelusuran.
8) Tombol Keluar untuk mengembalikan ke halaman menu utama pakar.

Event

Lanjut:

1. Pengguna (Pakar-User) memilih macam kerusakan yang akan ditampilkan.
2. Pengguna menjawab pertanyaan yang diberikan oleh sistem.
3. Pengguna menekan tombol Lanjut.
4. Sistem akan membuat sebuah *object* dari *class* MacamKerusakan, JenisKerusakan dan SebabSolusi, kemudian memasukkan nama macam kerusakan. Kemudian mengambil data-data yang memenuhi kriteria penelusuran dari basis data dan melakukan penelusuran kerusakan.
5. Sistem menampilkan informasi hasil dari penelusuran yang berupa sebab dan solusi kerusakan.

Reset:

1. Pakar menekan tombol Reset.
2. Sistem akan mengosongkan semua TextBox, dan ComboBox.

Keluar:

1. Pakar menekan tombol Keluar.
2. Sistem akan menampilkan halaman menu utama pakar.

SKPL

SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK

Sistem Pakar Untuk *Troubleshooting* Perangkat Keras Komputer

(SiPaTrou)

Disusun oleh:

Rika Novenawati

03 07 03695

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
Ditulis oleh							
Diperiksa oleh							
Disetujui oleh							

DAFTAR HALAMAN PERUBAHAN

Halaman	Revisi	Halaman	Revisi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

1 Pendahuluan

1.1 Tujuan

Dokumen Spesifikasi Kebutuhan Perangkat Lunak (SKPL) *SiPaTrou* (Sistem Pakar Untuk *Troubleshooting* Perangkat Keras Komputer) ini tersebut merupakan dokumen spesifikasi kebutuhan perangkat lunak Sistem Pakar Untuk *Troubleshooting* Perangkat Keras Komputer yang diberi nama *SiPaTrou*. SKPL ini digunakan untuk mendefinisikan fungsi perangkat lunak, yang meliputi antarmuka eksternal, dan atribut, serta mendefinisikan fungsi perangkat lunak, juga mendefinisikan batasan perancangan perangkat lunak yang akan dibangun. Dokumen ini digunakan oleh pengembang perangkat lunak sebagai acuan teknis untuk pembangunan perangkat lunak *SiPaTrou*.

1.2 Lingkup Masalah

Perangkat Lunak *SiPaTrou* dikembangkan dengan tujuan untuk:

1. Menangani ***login,*** **ubah password, pengelolaan pakar, pengelolaan data macam kerusakan, data jenis kerusakan, dan data penyebab kerusakan beserta solusi** oleh pakar **(SKPL_01)**.
2. Menangani proses **penelusuran kerusakan hardware** oleh pakar dan user biasa (**SKPL_02**).

Perangkat lunak *SiPaTrou* berjalan pada komputer *stand alone* dan *DBMS* yang digunakan adalah *Microsoft SQL Server 2000*.

1.3 Definisi Istilah dan Singkatan

Daftar definisi dan akronim yang digunakan:

Tabel 1. Tabel Definisi Akronim dan Singkatan

Keyword/Phrase	Definisi
SKPL	Merupakan dokumen spesifikasi kebutuhan dari perangkat lunak yang akan dikembangkan.
SKPL_XX	Kode yang merepresentasikan kebutuhan perangkat lunak dimana XX merupakan nomor fungsi produk.
SiPaTrou	Perangkat lunak yang berguna untuk memberikan informasi tentang *troubleshooting* perangkat keras komputer.
Hak Akses	Hak yang dimiliki oleh user untuk menggunakan sistem.
DBMS	*DataBase Management System* atau pengelola manajemen data base.
Basis data	Kumpulan data yang terkait yang diorganisasikan dalam struktur tertentu dan dapat diakses dengan cepat.
User	Orang yang berinteraksi dengan sistem.

1.4 Referensi

Dokumen yang digunakan sebagai acuan dalam rencana pembangunan perangkat lunak ini adalah:

- Dewi, Findra Kartika Sari. **GL-FINGERS-03**, *Spesifikasi Kebutuhan Perangkat Lunak*, Jurusan Teknik Informatika – UAJY.

- Nurdini, Ratna, **SKPL-Chosen**, Program Studi Teknik Informatika Universitas Atma Jaya Yogyakarta, 2007.
- http://ilmukomputer.com:81/umum/yanti-uml.php

1.5 Deskripsi Umum (Overview)

Modul fungsional yang akan dikembangkan dalam perangkat lunak *SiPaTrou* adalah berbasis *windows form (desktop)*, yaitu aplikasi *windows* yang menangani informasi antarmuka maupun pengelolaan bagi pemakai *software SiPaTrou*. Modul ini meliputi penyediaan informasi tentang solusi untuk *troubleshooting* perangkat keras komputer, dan pengelolaan data.

Cara kerja sistem modul ini secara global adalah:

1. Pertama kali, sistem akan menampilkan halaman dimana *user* diminta untuk memilih pilihan menu.
2. Bila *user* memilih menu pilihan Pakar, maka akan muncul halaman *login* dan akan diminta untuk memasukkan *username* dan *password*.
3. Sistem akan memeriksa dalam *basis data username* dan *password* tersebut valid atau tidak.
4. Jika *username* dan *password* tersebut *valid*, sistem akan menampilkan menu Pakar, meliputi menu ubah password, menu pengelolaan dan menu penelusuran kerusakan.

5. Sedangkan jika *user* memilih menu pilihan User, maka sistem akan menampilkan menu untuk User biasa, yaitu menu penelusuran kerusakan.

2 Deskripsi Keseluruhan

2.1 Perspektif Produk

Perangakat lunak *SiPaTrou* merupakan perangkat lunak yang berbasis aplikasi *windows* yang berfungsi untuk membantu pengguna yang ingin mencari informasi mengenai *troubleshooting* atau kerusakan-kerusakan pada perangkat keras komputer beserta cara penanggulangannya berdasarkan informasi gejala-gejala yang telah diberikan.

2.2 Fungsi Produk

Fungsi produk perangkat lunak *SiPaTrou* adalah:

1. Pakar
   a. ***Login* (SKPL_01_01)**, yaitu fungsi yang digunakan untuk memperoleh akses ke sistem. Login didasarkan pada nama pakar dan password yang berupa rangkaian karakter.
   b. **Ubah Password (SKPL_01_02)**, yaitu fungsi yang digunakan untuk mengubah password pakar.
   c. **Pengelolaan Pakar (SKPL_01_03)**, yaitu fungsi yang mengelola data pakar.
      Terdiri: **Tambah Pakar (SKPL_01_03_01)**
   d. **Pengelolaan Macam Kerusakan (SKPL_01_04)**, yaitu fungsi yang mengelola data macam kerusakan pada perangkat keras komputer.

Terdiri: **Tambah Macam (SKPL_01_04_01)**
Ubah Macam (SKPL_01_04_02)
Hapus Macam (SKPL_01_04_03)

e. **Pengelolaan Jenis Kerusakan (SKPL_01_05)**, yaitu fungsi yang mengelola data jenis kerusakan pada perangkat keras komputer.
Terdiri: **Tambah Jenis (SKPL_01_05_01)**
Ubah Jenis (SKPL_01_05_02)
Hapus Jenis (SKPL_01_05_03)

f. **Pengelolaan Penyebab dan Solusi (SKPL_01_06)**, yaitu fungsi yang mengelola data penyebeb kerusakan pada perangkat keras komputer beserta solusinya.
Terdiri: **Tambah Sebab dan Solusi (SKPL_01_06_01)**
Ubah Sebab dan Solusi (SKPL_01_06_02)
Hapus Sebab dan Solusi (SKPL_01_06_03)

2. All User (Pakar dan User/Pengguna)
   a. **Penelusuran Kerusakan Hardware (SKPL_02_01)**, yaitu fungsi untuk menampilkan informasi/solusi mengenai *troubleshooting* (kerusakan-kerusakan) pada perangkat keras komputer yang diperoleh dari gejala-gejala yang telah diberikan.

2.3 Karakteristik Pengguna

Karakteristik pengguna yang menggunakan perangkat lunak *SiPaTrou* yaitu:

1. Pakar
   a. Memahami pengoperasian komputer.

b. Memahami pengoperasian perangkat lunak yang digunakan.

c. Memiliki kemampuan pemrograman.

d. Mengerti serta dapat menjalankan fungsi-fungsi yang terdapat dalam perangkat lunak (*SiPaTrou*) sesuai hak akses yang dimiliki.

e. Memahami sistem komputer tempat perangkat lunak dijalankan.

2. User/Pengguna

a. Tidak diperlukan pengalaman dan keahlian teknis tertentu.

b. Memahami pengoperasian komputer.

c. Tingkat kebutuhan bervariasi.

2.4 Batasan-batasan

Batasan dalam pembangunan perangkat lunak *SiPaTrou* yaitu:

1. Kebijaksanaan umum

   Mengacu pada tujuan pembangunan perangkat lunak *SiPaTrou*.

2. Keterbatasan perangkat keras

   Ditentukan kemudian setelah pengembang mengetahui ketersediaan perangkat keras pada pengguna.

2.5 Asumsi dan Ketergantungan

Asumsi yang digunakan dalam pengembangan perangkat lunak *SiPaTrou* yaitu:

1. Tersedia perangkat lunak yang sesuai dengan kebutuhan untuk mengoperasikan produk perangkat lunak *SiPaTrou*.
2. Tersedia pengendali *ODBC* atau *ODBC Driver* untuk antarmuka *DBMS Microsoft SQL Server 2000*.

3 Kebutuhan Khusus

3.1 Kebutuhan Antarmuka Eksternal

3.1.1 Antarmuka Pemakai

Pemakai nerinteraksi dengan antarmuka yang ditampilkan dalam bentuk form-form.

Interaksi sistem dengan pengguna memungkinkan pengguna untuk melakukan proses penelusuran kerusakan hardware.

Interaksi sistem dengan pakar memungkinkan pakar untuk melakukan pengelolaan pakar, ubah password, pengelolaan macam kerusakan, pengelolaan jenis kerusakan, pengelolaan penyebab dan solusi, serta melakukan proses penelusuran kerusakan hardware.

3.1.2 Antarmuka Perangkat Keras

Kebutuhan antarmuka perangkat keras yang dikembangkan *SiPaTrou* berkenaan dengan antarmuka perangkat keras pada bagian 2.1 perspektif produk, yaitu:

1. *Monitor,* digunakan sebagai media interaksi antara pengguna dengan sistem.

2. *Mouse*, digunakan untuk mengenali input data yang dilakukan pengguna berkaitan dengan *even click, drag and drop* dan *on focus*.
3. *Keyboard,* digunakan melakukan input data berupa karakter atau text atau menu *pull down* yang harus diinputkan oleh pengguna.

3.1.3 Antarmuka Perangkat Lunak

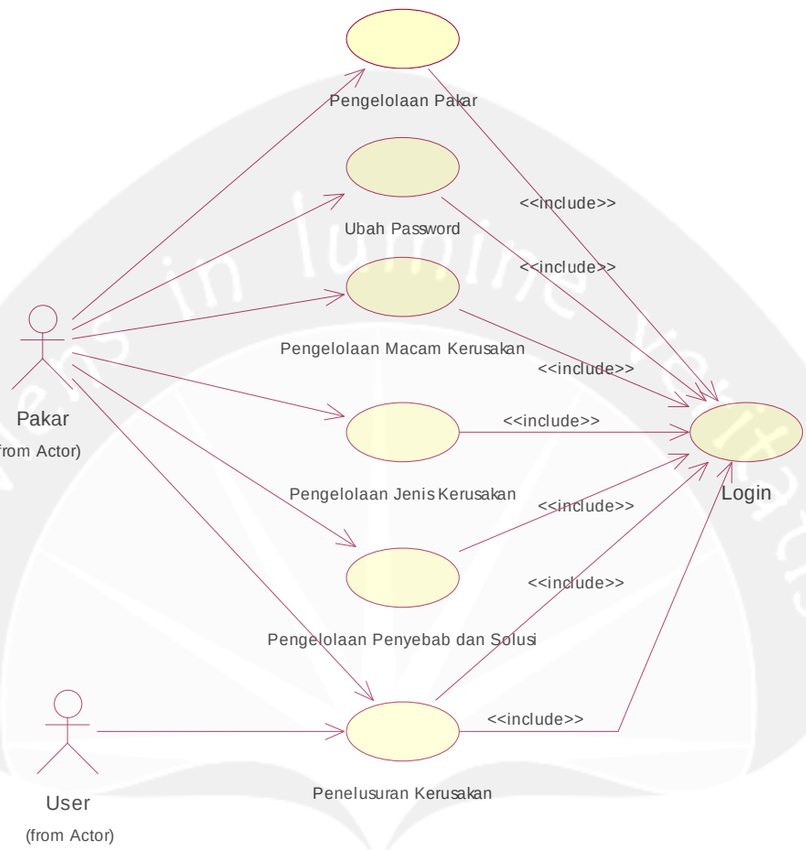
Perangkat lunak yang digunakan dalam mengoperasikan perangkat lunak *SiPaTrou* adalah sebagai berikut:

1. Nama : *Windows Me/NT/2000/XP*
   Sumber : *Microsoft*
   Sebagai sistem operasi dimana perangkat lunak *SiPaTrou* dijalankan.
2. Nama : *SQL Server 2000*
   Sumber : *Microsoft*
   Sebagai *DBMS* yang dibutuhkan dalam mengoperasikan perangkat lunak *SiPaTrou*.
3. Nama : *C#.NET*
   Sumber : *Microsoft*
   Sebagai *tool* perancang antarmuka aplikasi *SiPaTrou*.

Definisi antarmuka dalam bentuk isi pesan dan format mengacu pada dokumen panduan pengguna masing-masing perangkat lunak.

3.2 Kebutuhan Fungsionalitas Perangkat Lunak

3.2.1 *Use Case* Diagram



Gambar 1. *Use Case* Diagram

3.2.2 Spesifikasi Kebutuhan Fungsionalitas

3.2.2.1 *Use Case* : *Login*

Tabel 1. Spesifikasi Kebutuhan Fungsionalitas : Login

Use Case ID	SKPL_01_01
Use Case Name	*Login*
Brief Description	*Use case* ini digunakan untuk memperoleh akses ke sistem. Login didasarkan pada *username* dan *password* yang berupa rangkaian karakter.
Primary Actor	Pakar
Supporting Actor	-
Basic Flow	1. *Use Case* dimulai ketika aktor memilih menu pilihan Pakar. 2. Sistem menampilkan antarmuka untuk *login*. 3. Aktor memasukkan nama pakar dan *password*. 4. Sistem memeriksa nama pengguna dan *password* aktor. 5. Sistem memberikan akses ke aktor sesuai hak aksesnya. E. Nama pengguna password atau tidak sesuai. 6. *Use Case* selesai.
Alternative Flow	-
Error Flow	E. Nama pengguna atau *password* tidak sesuai (masukan masih kosong atau tidak sesuai dengan tipe data). • Sistem menampilkan peringatan bahwa nama pengguna atau *password* tidak

	sesuai. • Kembali ke *Basic Flow* langkah ke-3.
PreConditions	-
PostConditions	Aktor memasuki sistem sesuai hak aksesnya (*Login*).

3.2.2.2 *Use Case* : Ubah Password

Tabel 2. Spesifikasi Kebutuhan Fungsionalitas : Ubah Password

Use Case ID	SKPL_01_02
Use Case Name	Ubah Password
Brief Description	*Use case* ini digunakan oleh aktor untuk mengubah *password* milik aktor.
Primary Actor	Pakar
Supporting Actor	-
Basic Flow	1. *Use Case* dimulai ketika aktor memilih untuk mengubah *password*. 2. Sistem menampilkan antarmuka ubah password. 3. Aktor mengubah *password*. 4. Aktor meminta sistem untuk mengubah password. 5. Sistem mengubah password dalam basis data. E. Masukan tidak sesuai. 6. *Use Case* selesai.
Alternative Flow	-
Error Flow	E. Masukan tidak sesuai (masukan *password* baru atau konfirmasi *password* baru tidak sesuai).

	• Sistem menampilkan peringatan bahwa *password* lama atau *password* baru atau konfirmasi *password* baru tidak sesuai. • Kembali ke *Basic Flow* langkah ke-2.
PreConditions	Aktor telah memasuki sistem melalui *Use Case* login sebagai pakar.
PostConditions	Data *password* pakar di basis data berubah (ubah).

3.2.2.3 *Use Case* : Pengelolaan Pakar

Tabel 3. Spesifikasi Kebutuhan Fungsionalitas : Pengelolaan Pakar

Use Case ID	SKPL_01_03
Use Case Name	Pengelolaan Pakar
Brief Description	*Use case* ini digunakan oleh untuk mengelola data pakar.
Primary Actor	Pakar
Supporting Actor	-
Basic Flow	1. *Use Case* dimulai ketika aktor memilih untuk mengelola data pakar. 2. Sistem menampilkan antarmuka pengelolaan data pakar dan menampilkan seluruh data pakar yang ada di dalam basis data. 3. Aktor memilih untuk menambah data pakar dengan memasukkan data pakar tersebut. 4. Aktor meminta sistem untuk menyimpan data.

	5. Data tersimpan ke dalam basis data. E.1 Masukan tidak sesuai. 6. *Use Case* selesai.
Alternative Flow	
Error Flow	E.1 Masukan tidak sesuai (masukan masih kosong, tidak sesuai dengan tipe data atau sudah ada dalam basis data). • Sistem menampilkan peringatan bahwa masukan tidak sesuai. • Kembali ke *Basic Flow* langkah ke-3. E.2 Data masih digunakan. • Sistem menampilkan peringatan bahwa data masih digunakan. • Kembali ke *Basic Flow* langkah ke-2.
PreConditions	Aktor telah memasuki sistem melalui *Use Case* login sebagai pakar.
PostConditions	Data pakar di basis data berubah (tambah) atau tetap (tampil).

3.2.2.4 *Use Case* : Pengelolaan Macam Kerusakan

Tabel 4. Spesifikasi Kebutuhan Fungsionalitas : Pengelolaan Macam Kerusakan

Use Case ID	SKPL_01_04
Use Case Name	Pengelolaan Macam Kerusakan
Brief Description	*Use case* ini digunakan oleh untuk mengelola data macam kerusakan.
Primary Actor	Pakar
Supporting Actor	-

Basic Flow	1. *Use Case* dimulai ketika aktor memilih untuk mengelola data macam kerusakan. 2. Sistem menampilkan antarmuka pengelolaan data macam kerusakan dan menampilkan seluruh data macam kerusakan yang ada di dalam basis data. 3. Aktor memilih untuk menambah data macam kerusakan dengan memasukkan data macam kerusakan tersebut. A.1 Aktor memilih untuk mengubah data macam kerusakan. A.2 Aktor memilih untuk menghapus data macam kerusakan. 4. Aktor meminta sistem untuk menyimpan data. 5. Data tersimpan ke dalam basis data. E.1 Masukan tidak sesuai. 6. *Use Case* selesai.
Alternative Flow	A.1 Aktor memilih untuk mengubah data macam kerusakan. 1. Aktor memilih data macam kerusakan yang akan diubah. 2. Sistem menampilkan data yang akan diubah. 3. Aktor mengubah detail macam kerusakan. 4. *Flow* berlanjut ke *Basic Flow* langkah ke-5.

	A.2 Aktor memilih untuk menghapus data macam kerusakan. 1. Aktor memilih data macam kerusakan yang akan dihapus. 2. Aktor meminta sistem untuk menghapus data. 3. Sistem menghapus detail macam kerusakan tersebut dari basis data. E.2 Data masih digunakan. 4. Kembali ke Basic Flow langkah ke-6.
Error Flow	E.1 Masukan tidak sesuai (masukan masih kosong, tidak sesuai dengan tipe data atau sudah ada dalam basis data). • Sistem menampilkan peringatan bahwa masukan tidak sesuai. • Kembali ke *Basic Flow* langkah ke-3. E.2 Data masih digunakan. • Sistem menampilkan peringatan bahwa data masih digunakan. • Kembali ke *Basic Flow* langkah ke-2.
PreConditions	Aktor telah memasuki sistem melalui *Use Case* login sebagai pakar.
PostConditions	Data macam kerusakan di basis data berubah (tambah, ubah, hapus) atau tetap (tampil).

3.2.2.5 *Use Case* : Pengelolaan Jenis Kerusakan

Tabel 5. Spesifikasi Kebutuhan Fungsionalitas : Pengelolaan Jenis Kerusakan

Use Case ID	SKPL_01_05
Use Case Name	Pengelolaan Jenis Kerusakan
Brief Description	*Use case* ini digunakan oleh untuk mengelola data jenis kerusakan.
Primary Actor	Pakar
Supporting Actor	-
Basic Flow	1. *Use Case* dimulai ketika aktor memilih untuk mengelola data jenis kerusakan. 2. Sistem menampilkan antarmuka pengelolaan data jenis kerusakan dan menampilkan seluruh data jenis kerusakan yang ada di dalam basis data. 3. Aktor memilih untuk menambah data jenis kerusakan dengan memasukkan data jenis kerusakan tersebut. A.1 Aktor memilih untuk mengubah data jenis kerusakan. A.2 Aktor memilih untuk menghapus data jenis kerusakan. 4. Aktor meminta sistem untuk menyimpan data. 5. Data tersimpan ke dalam basis data. E.1 Masukan tidak sesuai. 6. *Use Case* selesai.
Alternative Flow	A.1 Aktor memilih untuk mengubah data jenis kerusakan.

	1. Aktor memilih data jenis kerusakan yang akan diubah. 2. Sistem menampilkan data yang akan diubah. 3. Aktor mengubah detail jenis kerusakan. 4. *Flow* berlanjut ke *Basic Flow* langkah ke-5. A.2 Aktor memilih untuk menghapus data jenis kerusakan. 1. Aktor memilih data jenis kerusakan yang akan dihapus. 2. Aktor meminta sistem untuk menghapus data. 3. Sistem menghapus detail jenis kerusakan tersebut dari basis data. E.2 Data masih digunakan. 4. Kembali ke Basic Flow langkah ke-6.
Error Flow	E.1 Masukan tidak sesuai (masukan masih kosong, tidak sesuai dengan tipe data atau sudah ada dalam basis data). • Sistem menampilkan peringatan bahwa masukan tidak sesuai. • Kembali ke *Basic Flow* langkah ke-3. E.2 Data masih digunakan. • Sistem menampilkan peringatan bahwa data masih digunakan. • Kembali ke *Basic Flow* langkah ke-2.

PreConditions	Aktor telah memasuki sistem melalui *Use Case* login sebagai pakar.
PostConditions	Data jenis kerusakan di basis data berubah (tambah, ubah, hapus) atau tetap (tampil).

3.2.2.6 *Use Case* : Pengelolaan Penyebab dan Solusi

Tabel 6. Spesifikasi Kebutuhan Fungsionalitas : Pengelolaan Penyebab dan Solusi

Use Case ID	SKPL_01_06
Use Case Name	Pengelolaan Penyebab dan Solusi
Brief Description	*Use case* ini digunakan oleh untuk mengelola data sebab dan solusi kerusakan.
Primary Actor	Pakar
Supporting Actor	-
Basic Flow	1. *Use Case* dimulai ketika aktor memilih untuk mengelola data sebab dan solusi kerusakan. 2. Sistem menampilkan antarmuka pengelolaan data sebab dan solusi kerusakan dan menampilkan seluruh data sebab dan solusi kerusakan yang ada di dalam basis data. 3. Aktor memilih untuk menambah data sebab dan solusi kerusakan dengan memasukkan data sebab dan solusi kerusakan tersebut. A.1 Aktor memilih untuk mengubah data sebab dan solusi kerusakan.

	A.2 Aktor memilih untuk menghapus data sebab dan solusi kerusakan. 4. Aktor meminta sistem untuk menyimpan data. 5. Data tersimpan ke dalam basis data. E.1 Masukan tidak sesuai. 6. *Use Case* selesai.
Alternative Flow	A.1 Aktor memilih untuk mengubah data sebab dan solusi kerusakan. 1. Aktor memilih data sebab dan solusi kerusakan yang akan diubah. 2. Sistem menampilkan data yang akan diubah. 3. Aktor mengubah detail sebab dan solusi kerusakan. 4. *Flow* berlanjut ke *Basic Flow* langkah ke-5. A.2 Aktor memilih untuk menghapus data sebab dan solusi kerusakan. 1. Aktor memilih data sebab dan solusi kerusakan yang akan dihapus. 2. Aktor meminta sistem untuk menghapus data. 3. Sistem menghapus detail sebab dan solusi kerusakan tersebut dari basis data. E.2 Data masih digunakan. 4. Kembali ke Basic Flow langkah ke-6.
Error Flow	E.1 Masukan tidak sesuai (masukan masih

	kosong, tidak sesuai dengan tipe data atau sudah ada dalam basis data). • Sistem menampilkan peringatan bahwa masukan tidak sesuai. • Kembali ke *Basic Flow* langkah ke-3. E.2 Data masih digunakan. • Sistem menampilkan peringatan bahwa data masih digunakan. • Kembali ke *Basic Flow* langkah ke-2.
PreConditions	Aktor telah memasuki sistem melalui *Use Case* login sebagai pakar.
PostConditions	Data sebab dan solusi kerusakan di basis data berubah (tambah, ubah, hapus) atau tetap (tampil).

3.2.2.7 *Use Case* : Penelusuran Kerusakan

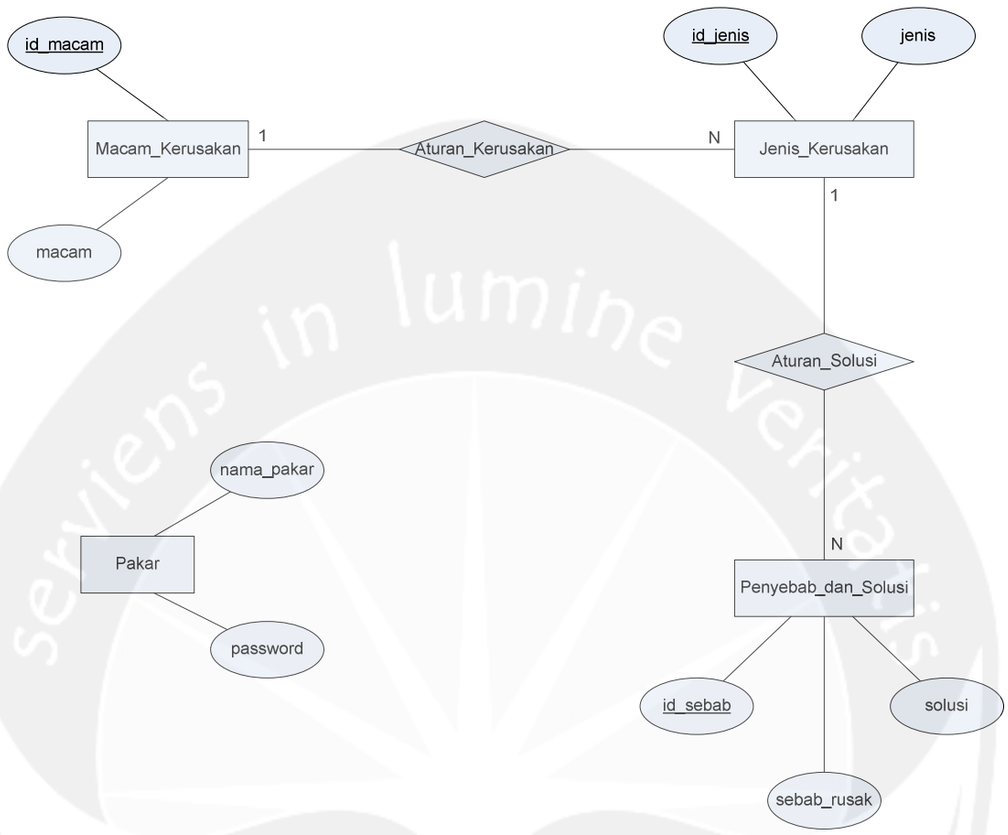
Tabel 7. Spesifikasi Kebutuhan Fungsionalitas : Penelusuran Kerusakan

Use Case ID	SKPL_02_01
Use Case Name	Penelusuran Kerusakan
Brief Description	*Use case* ini digunakan untuk menampilkan informasi mengenai penelusuran kerusakan *hardware* komputer.
Primary Actor	Pakar dan User/Pengguna
Supporting Actor	-
Basic Flow	1. *Use Case* dimulai ketika aktor memilih untuk menampilkan informasi penelusuran kerusakan *hardware*

	komputer.. 2. Sistem menampilkan antarmuka penelusuran kerusakan. 3. Aktor memasukkan macam kerusakan *hardware*. 4. Aktor menjawab semua pertanyaan dari sistem. 5. Sistem menampilkan informasi penyebab dan solusi kerusakan. E. Solusi kerusakan tidak terdapat dalam basis data. 6. *Use Case* selesai
Alternative Flow	-
Error Flow	E. Solusi kerusakan tidak terdapat dalam basis data. • Sistem menampilkan peringatan bahwa solusi kerusakan tidak terdapat dalam basis data. • Kembali ke *Basic Flow* langkah ke-2.
PreConditions	-
PostConditions	Data-data di basis data tetap.

4 Entity Relationship Diagram (ERD)

4.1 Entity Relationship Diagram



Gambar 1. ***Entity Relationship Diagram***