BAB III
METODE PENELITIAN

Metode penelitian diperlukan untuk menjawab masalah dan menguji hipotesa. Untuk itu, dibagian ini perlu ditetapkan metode penelitian apa yang digunakan, apakah metode survei, metode eksperimen, metode kasus, metode penelitian dan pengembangan, atau kaji tindakan (*action research*) (Sudaryono, 2015: 157).

Penelitian ini dilakukan dengan cara diagnosis terhadap beberapa penyebab kerusakan *IC* di mesin *Wire Bond* IConn PT UNISEM Batam. Dari beberapa analisa yang dilakukan, maka data diolah dan dibuat sistem pakar untuk mengatasi beberapa penyebab kerusakan *IC* di mesin *Wire Bond* IConn PT UNISEM Batam. Data-data yang didapat digunakan untuk membuat *rule* sebuah aplikasi sistem pakar *forward chaining* untuk menduplikasikan kemampuan pakar, sehingga bisa dimanfaatkan teknisi di area produksi *Wire Bond* PT UNISEM Batam.

3.1 Desain Penelitian

Desain penelitian menggambarkan apa yang akan dilakukan oleh peneliti dalam terminologi teknis. Dalam hal ini, desain penelitian harus mencakup antara lain tahapan yang akan dilakukan, informasi mengenai cara penarikan *sampel* bila diperlukan survei primer, besarnya sampel, metode pengumpulan data, instrumen

penelitian, prosedur pengujian validitas, dan reliabilitas instrumen (Kuncoro, 2009 dalam Sudaryono, 2015: 157).

Berikut ini adalah gambar konsep dasar desain penelitian yang digunakan dalam penelitian ini (Gambar 3.1):

Gambar 3.1 Metode penelitian
(Sumber: Data Penelitian, 2017)

Berikut ini adalah langkah-langkah yang dilakukan dalam penelitian yaitu sebagai berikut:

1. Identifikasi Masalah

   Tahap pertama dalam melakukan penelitian adalah mengidentifikasi masalah. Pada tahapan identifikasi masalah, dilakukan diagnosa terhadap faktor-faktor penyebab kerusakan *IC* di area produksi *Wire Bond* PT UNISEM Batam. Setelah identifikasi masalah selesai, kemudian dilakukan perumusan masalah.

2. Perumusan Masalah

   Perumusan masalah bertujuan untuk menspesifikasikan masalah yang ada sehingga dapat dijawab dengan baik melalui penelitian. Setelah masalah dirumuskan, maka tahapan berikutnya adalah menentukan tujuan penelitian.

3. Menentukan Tujuan Penelitian

   Tujuan penelitian yaitu mengetahui bagaimana sistem pakar forward chaining berbasis web mendeteksi kerusakan *IC* di area produksi *Wire Bond* PT UNISEM Batam. Setelah tujuan penelitian ditentukan, maka langkah selanjutnya adalah mengumpulkan data.

4. Mengumpulkan Data

   Dalam pengunpulan data, peneliti mengambil data dengan melakukan wawancara dengan seorang pakar mesin *Wire Bond* (Moch. Ali Rohman, S.T.) dan studi literature yaitu mengumpulkan buku yang berkaitan dengan kecerdasan buatan, sistem pakar, pemrograman *web*, *UML*, *manual book* mesin *Wire Bond* IConn, dan *Asembly Spesification* (*AS*) PT UNISEM Batam tentang kerusakan *IC*. Setelah bahan-bahan yang mendukung dalam penelitian didapatkan maka tahapan selanjutnya adalah melakukan studi literatur.

5. Melakukan Studi Literatur

   Mempelajari dan mengkaji buku-buku yang berhubungan dengan penelitian sehingga menambah wawasan dan pengetahuan tentang objek yang akan diteliti. Dalam hal ini peneliti mempelajari tentang kecerdasan buatan, sistem pakar, *UML*, membangun pemrograman *web*, *manual book* mesin *Wire Bond* KNS IConn, dan *Asembly Spesification* (AS) PT UNISEM Batam tentang

kerusakan *IC*. Setelah studi literatur dilakukan, maka tahapan selajutnya adalah menganalisa data yang didapat.

6. Menganalisa Data

   Setelah data-data yang berhubungan dengan penelitian didapatkan, maka langkah selanjutnya dengan mengalisa data-data yang diperoleh. Pada tahapan mengalalisa data, ditentukan terlebih dahulu operasional variabel dan indikator yang akan digunakan supaya mempermudah dalam proses pengolahan data. Jika operasional variable dan indikator sudah ditentukan maka tahapan berikutnya adalah mengolah data.

7. Mengolah Data Menggunakan Sistem Pakar *Metode Forward Chaining*

   Data-data yang sudah dianalisa kemudian diolah menggunakan sistem pakar *forward chaining* untuk membuat *rule-rule* yang akan digunakan saat sistem pakar melakukan penelusuran berdasarkan gelaja yang ada. Setelah *rule-rule* sebagai *knowlege base* sistem pakar dibuat maka tahapan selanjutnya adalah mengimplementasikan sistem pakar kedalam pemrograman berbasis *web*.

8. Mengimplementasikan Sistem Pakar Dalam Bentuk Pemrograman Berbasis *Web*

   Setelah data yang diperoleh diolah menggunakan sistem pakar metode forward chaining, data-data tersebut disusun dan dibuat kode-kode untuk mempermudah pembuatan program. Pembuatan aplikasi sistem pakar dilakukan mulai dari perancangan sistem yaitu: desain *UML*, desain antarmuka, desain *database*, dan desain basis pengetahuan (aturan). Setelah itu proses pengodean menggunakan bahasa *HTML*, *PHP*, *CSS*, *JavaScript*, dan *jQuery* dengan *editor Notepad++*.

Langkah terakhir dalam pembuatan aplikasi sistem pakar adalah membuat *database* dengan menggunakan *phpMyAdmin* yang ada pada program *XAMPP*. Dalam pembuatan program, peneliti menggunakan referensi program dari buku karya Bunafit Nugroho yang berjudul Aplikasi Sistem Pakar dengan PHP & Editor Dreamweaver, juga arahan mengenai tampilan program oleh Ibu Anggia Dasa Putri, S.Kom., M.Kom. sebagai dosen pembimbing dan dosen pengajar mata kuliah Pemrograman web, serta masukan dan ajaran dari Mas Dedi Suhendra, S.Kom yang pernah melakukan penelitian dengan judul tesis Sistem Pakar Untuk Mendeteksi Kerusakan *Air Conditioner* (*AC*) Menggunakan Metode *Forward Chaining* Berbasis *Web*. Sistem pakar yang sudah dibuat kemudian diuji keakuratannya pada tahapan pengujian hasil penelitian. Setelah program dibuat, maka aplikasi sistem pakar tersebut diimplementasikan pada salah satu komputer yang sudah terinstal aplikasi *XAMPP* ada di area produksi *Wire Bond* PT UNISEM Batam.

9. Pegujian Hasil Penelitian

   Tahapan ini adalah pengujian aplikasi yang telah dibuat. Pengujian dilakukan oleh pakar yaitu sorang engineer proses produksi (Moch. Ali Rohman, S.T.). Pengujian dilakukan dengan 2 sistem yaitu *black box testing* dan pengujian keakuratan dengan pakar. Pengujian *black box testing* yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungisi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Setelah lolos pengujian aplikasi maka tahapan berikutnya adalah dengan

pengujian tingkat keakuratan sistem pakar. Pengujian dengan pakar dilakukan dengan merujuk pada data diagnosa gejala kerusakan unit yang telah disusun (data dari *Line Distribution Alert*) sebanyak 10 studi kasus. Kemudian hasil pengujian dihitung tingkat persentasi keakuratannya. Setelah pengujian dilakukan maka tahapan berikutnya adalah menarik kesimpulan.

10. Menarik Kesimpulan

    Tahapan terakhir dalam penelitian ini yaitu menyimpulkan hasil penelitian yang berisi jawaban singkat terhadap rumusan masalah berdasarkan data-data yang ada. Dalam tahap ini, peneliti juga memberikan saran yang penting untuk membantu dalam memecahkan permasalahan yang ada.

3.1.1 Teknik Pengumpulan Data

Teknik pengumpulan data merupakan langkah yang paling strategis dalam penelitian, karena tujuan utama dari penelitian adalah mendapatkan data (Sugiyono, 2012:224). Langkah-langkah yang dilakukan untuk pengumpulan data dalam penelitian ini adalah sebagai berikut:

1. Studi literatur

   Studi literatur yaitu mengumpulkan data dan menambah wawasan dengan belajar dari iAS1006 yang berisi cara penangana kerusakan pada mesin *Wire Bond*, iAS1007 yang berisi spesifikasi tentang kerusakan *IC* yang ada di PT UNISEM Batam, mempelajari tentang *manual book* mesin *Wire Bond* IConn, mempelajari buku tentang pembuatan aplikasi sistem pakar bebasis *web*, membaca jurnal penelitian yang berkaitan dengan topik bahasan penelitian.

2. Wawancara

   Melakukan wawancara dengan Moch. Ali Rochman S.T. yang bekerja sebagai *Process Engineer Wire Bond* dan berpengalaman lebih dari 6 tahun dalam menganalisa dan menangani kerusakan *IC* di PT UNISEM Batam. Wawancara dilakukan dengan memberikan beberapa pertanyaan yang sudah disusun sebelumnya, kemudian jawaban tersebut ditulis oleh peneliti pada *form* wawancara. Semua kegiatan wawancara dengan pakar direkam dan di foto sebagai bukti penelitian.

3.2 Operasional Variabel

Variabel penelitian adalah suatu atribut atau sifat atau nilai dari orang, objek atau kegiatan yang mempunyai variasi tertentu yang ditetapkan oleh peneliti untuk dipelajari dan kemudian ditarik kesimpulannya (Sugiyono, 2014:38).

Berikut ini adalah tabel hubungan antara variabel dan indikator dalam penelitian ini yaitu (Tabel 3.1):

Tabel 3.1 Tabel Hubungan Variabel dan Indikator

Variabel	Indikator
Kerusakan mesin *Wire Bond* Iconn	*Ball non stick*
	Lifted wedge
	Damaged wire

(Sumber: Data Penelitian, 2017)

Dalam Tabel 3.1 diatas menjelaskan hubungan anatar variabel dan indikator. Variabelnya adalah kerusakan mesin *Wire Bond* IConn, sedangkan indikatornya adalah 3 kerusakan *IC* yaitu: *Ball Non Stick, Lifted Wedge,* dan *Damaged Wire*.

Pada Tabel 3.2 menjelaskan indikator, penyebab, dan solusi kerusakan *IC* dimesin *Wire Bond* KNS IConn.

Tabel 3.2 Tabel Penyebab dan Solusi Kerusakan *IC*

Indikator	Penyebab	Solusi
Ball Non Stick	Tidak *optimize 1st bond* parameter	Gunakan maksimum *ball* parameter dan kurangi FAB parameter sesuai DC (*Data Collector*)
	Downholder kotor (terdapat *foreign material*)	Bersihkan *downholder*
	Floating die pad	Atur ketinggian *top clamp*
	Material yang *reject*	Hubungi *Process Engineer* untuk membuat *optimize parameter* dan membuat laporan ke *Customer*
	Temperatur pada *Heater Block* tidak tepat	Naikkan temperatur *offset* pada menu *heater block*
Lifted Wedge	*Lead frame* terkontaminasi	Dilakukan plasma *cleaning* pada material dengan program 3
	Floating pada permukaan *lead*	Cek kondisi *red tape* pada *top clamp*, cek kebersihan permukaan *downholder*, atur ketinggian *top clamp*
	Kurang *optimize wedge bond* parameter	Gunakan maksimum *wedge* parameter sesuai *DC* (*Data Collector*)
	Impedansi *capillary* terlalu tinggi	Kalibrasi ulang *USG impedance* pada mesin
	Temperatur pada *Heater Block* tidak tepat	Naikkan temperatur *offset* pada menu *heater block*
Damaged Wire	Tidak pas/sejajar *indexer table* dengan slot *magazine output*	*Setting magazine output*
	Die pad tidak tepat pada *cavity Down Holder*	Lakukan *teaching* ulang *OLC* pada mesin

Tabel 3.2 Lanjutan

Damaged Wire	Tidak lancar strip pada *indexer table* (*indexer tabel* terlalu sempit)	Atur lebar *indexer track* sesuai dengan lebar *lead frame*
	Tidak *optimize loop* parameter	*Optimize loop* parameter sesuai DC dan ukur ketinggian *loop*
	Tertabrak oleh *top clamp*	Atur ketinggian *top clamp* mencapai maksimal dan perlambat *step indexer*
	Tertabrak oleh capillary	Cek jarak *bondpad opening*, ganti aplikasi *capillary* lebih kecil atau ganti *bonding sequence*

(Sumber: Data Penelitian 2017)

Pada Tabel 3.2 diatas, masing-masing indikator memiliki beberapa penyebab kerusakan. Dari penyebab kerusakan tersebut, juga disertakan solusi dari setiap penyebab kerusakan.

3.3 Perancangan Sistem

A.S. dan Shalahuddin (2011: 23) perancangan sistem merupakan upaya untuk mengkonstruksi sebuah sistem yang memberikan kepuasan akan spesifikasi kebutuhan fungsional, memenuhi target, memenuhi kebutuhan secara implisit atau eksplisit dari segi performa maupun penggunaan sumber daya, kepuasan batasan pada proses desain dari segi biaya, waktu, dan perangkat.

3.3.1 Desain Basis Pengetahuan

Peneliti melakukan proses akuisisi pengetahuan dengan mengumpulkan pengetahuan dan fakta dari sumber-sumber yang tersedia. Sumber pengetahuan dan fakta diperoleh melalui wawancara dengan seorang *Process Engineer* yang berpengalaman, selain itu peneliti melakukan studi literatur tentang materi yang berkaitan mesin *Wire Bond* IConn dan penyebab-penyebab kerusakan *IC*.

Sumber pengetahuan yang telah didapatkan dari seorang pakar ditampilkan dalam table nama kerusakan *IC* (Tabel 3.3), table penyebab dan solusi kerusakan *IC* (Tabel 3.4), table gejala kerusakan *IC* (Tabel 3.5) dan table aturan (Tabel 3.6) berikut ini:

Tabel 3.3 Tabel Nama Kerusakan *IC*

Kode	Indikator
KU01	Ball non stick
KU02	Lifted wedge
KU03	Damaged wire

(Sumber: Data Penelitian 2017)

Pada Tabel 3.3 diatas, masing-masing indikator diberikan kode untuk membedakan satu dengan yang lainnya. Kode KU01 untuk kerusakan *IC* berupa *Ball Non Stick*, Kode KU02 untuk kerusakan *IC* berupa *Lifted Wedge*, dan Kode KU03 untuk kerusakan *IC* berupa *Damaged Wire*.

Tabel 3.4 Tabel Penyebab dan Solusi Kerusakan *IC*

Kode	Penyebab	Solusi
PP01	Tidak *optimize 1st bond* parameter	Gunakan maksimum *ball* parameter dan kurangi FAB parameter sesuai *DC* (*Data Collector*)

Tabel 3.4 Lanjutan

PP02	Temperatur pada *Heater Block* tidak tepat	Cek temperatur *HB*, naikkan temperatur *offset* pada menu *heater block*
PP03	Material yang *reject*	Hubungi *Process Engineer* untuk membuat o*ptimize* parameter dan membuat laporan ke *Customer*
PP04	*Downholder* kotor (terdapat *foreign material*)	Bersihkan *downholder*
PP05	*Floating die pad*	Atur ketinggian *top clamp*
PP06	*Lead frame* terkontaminasi	Dilakukan plasma *cleaning* pada material dengan program 3, *optimize wedge* parameter
PP07	*Floating* pada permukaan *lead*	Cek kondisi *red tape* pada *top clamp*, cek kebersihan permukaan *downholder*, atur ketinggian *top clamp*
PP08	Kurang *optimize wedge bond* parameter	Gunakan maksimum *wedge* parameter sesuai *DC* (*Data Collector*)
PP09	Impedansi *capillary* terlalu tinggi	Kalibrasi ulang *USG impedance* pada mesin
PP10	Tidak pas/sejajar *indexer table* dengan slot *magazine output*	*Setting magazine output*
PP11	*Die pad* tidak tepat pada *cavity Down Holder*	Lakukan *teaching* ulang *OLC* pada mesin
PP12	Tidak lancar strip pada *indexer table* (*indexer tabel* terlalu sempit).	Atur lebar *indexer track* sesuai dengan lebar *lead frame*
PP13	Tidak *optimize loop* parameter	*Optimize loop* parameter sesuai DC dan ukur ketinggian loop.
PP14	Tertabrak oleh *top clamp*	Atur ketinggian *top clamp* mencapai maksimal dan perlambat *step indexer*
PP15	Tertabrak oleh *capillary*	Cek jarak *bondpad opening*, ganti aplikasi capillary lebih kecil atau ganti *bonding sequence*

(Sumber: Data Penelitian 2017)

Pada Tabel 3.4, menunjukkan bahwa masing-masing diberikan kode untuk membedakan antara satu penyebab dengan yang lainnya. Dari masing-masing penyebab diberikan solusi untuk mengatasi penyebab kerusakan *IC*.

Tabel 3.5 Tabel Gejala Kerusakan

Kode Gejala	Nama Gejala
GG01	*Reading ball shear test* rendah *(lower control level*)
GG02	Bentuk ball bond normal dengan ukuran antara 2x - 5x diameter *wire*
GG03	ketebalan ball bond lebih dari 1x diameter *wire*
GG04	Sebagian ball bond pada unit terlihat tipis (*flat ball*)
GG05	Terdapat kelainan warna pada *bondpad*
GG06	Terdapat tanda gores pada bagian bawah *die pad*
GG07	Terdapat kelainan warna pada permukaan *lead*
GG08	Tidak terlihat imprint *capillary* di *wedge*
GG09	Terlihat *peeloff* pada *wedge flare*
GG10	Ukuran *wedge flare* kurang dari 1.3x diameter *wire*
GG11	Reading wedge pull rendah
GG12	Terlihat halfmoon imprint capillary
GG13	Ada bekas luka pada ujung *side rail* sebelah kanan atas atupun kiri atas
GG14	Ada bekas luka pada *tie bar*
GG15	Ada bekas luka pada *side rail* bagian bawah atau atas sebelah kanan
GG16	Tidak ada bekas luka pada permukaan *lead frame*
GG17	Terlihat goresan pada *wire*
GG18	Ada indikasi *vacum error*
GG19	Bentuk *wire* dan *loop* tertekan bengkok ke kiri
GG20	Bentuk *wire* bengkong di daerah necking (1st *kink*)
GG21	Terlihat *wire* melengkung kebawah (*sagging wire*)
GG22	*Strip* tidak lancar di *indexer table*
GG23	Tidak terdapat luka gores pada *strip*

(Sumber: Data Penelitian 2017)

Pada Tabel 3.5 diatas, menunjukkan pengkodean dari masing-masing gejala kerusakan *IC* supaya membedakan gejala kerusakan satu dengan yang lainnya.

Data aturan berisi relasi antara data-data nama kerusakan *IC*, penyebab kerusakan *IC* dan gejala kerusakan *IC* yang telah diberi kode sebelumnya. Data-data yang didapat kemudian dibuat relasi antar data sehingga menghasilkan *rule-rule* dalam sistem pakar yang memudahkan penyusunan basis pengetahuan. Berikut ini adalah tabel data aturan (Tabel 3.6):

Tabel 3.6 Tabel Data Aturan

Kode Indikator	Kode Penyebab	Kode Gejala
KU01	PP01	GG01, GG03
KU01	PP02	GG01,GG02
KU01	PP03	GG01, GG02, GG05
KU01	PP04	GG04, GG06
KU01	PP05	GG04
KU02	PP06	GG07, GG10, GG11
KU02	PP07	GG09, GG12
KU02	PP08	GG08, GG11
KU02	PP09	GG10, GG11
KU03	PP10	GG13, GG21
KU03	PP11	GG14, GG18
KU03	PP12	GG15, GG22
KU03	PP13	GG16, GG17
KU03	PP14	GG19, GG23
KU03	PP15	GG20, GG23

(Sumber: Data Penelitian 2017)

Pada Tabel 3.6 diatas, indikator, penyebab dan gejala dibuat kode yang berbeda-beda. Pengkoden ini dibuat untuk memudahkan dalam penyusunan kaidah produksi yang akan dibuat. Setiap penyebab mempunyai gejala yang berbeda, tetapi ada beberapa penyebab mempunyai salah satu ciri gejala sama dengan penyabab

lainnya. Urutan pengkodean penyebab disesuaikan/dikelompokkan sesuai dengan kode kerusakan *IC* (Kode Indikator).

3.3.2 Pembentukan aturan

Sutojo, dkk. (2016: 9) setiap rule terdiri dari dua bagian, yaitu bagian *IF* disebut *evidence* (fakta-fakta) dan bagian *THEN* disebut hipotesis atau kesimpulan. Representasi pengetahuan pada dasarnya berupa aturan *IF – THEN* dalam sebuah sistem pakar. Data-data yang sudah disusun dalam tabel aturan (Tabel 3.7), dirangkai menjadi suatu kaidah aturan dalam sistem pakar. Berikut ini adalah tabel aturan *inference* dalam sistem pakar ini:

Tabel 3.7 Aturan *Inference*

Aturan	Kaidah
R01	IF GG01 AND GG03 THEN PP01
R02	IF GG01 AND GG02 THEN PP02
R03	IF GG01 AND GG02 AND GG05 THEN PP03
R04	IF GG04 AND GG6 THEN PP04
R05	IF GG04 THEN PP05
R06	IF GG07 AND GG10 AND GG11 THEN PP06
R07	IF GG09 AND GG12 THEN PP07
R08	IF GG10 AND GG11 THEN PP08
R09	IF GG08 AND GG10 THEN PP09
R10	IF GG13 AND GG21 THEN PP10
R11	IF GG14 AND GG18 THEN PP11
R12	IF GG15 AND GG22 THEN PP12
R13	IF GG16 AND GG17 THEN PP13
R14	IF GG19 AND 23 THEN PP 14
R15	IF GG20 AND GG23 THEN PP15

(Sumber: Data Penelitian 2017)

Berdasarkan Tabel 3.7, dapat disimpulkan bahwa terdapat 15 aturan (*rule*). Berikut ini adalah pembahasannya:

1. *Rule* 1

   Jika gejala kerusakan adalah *reading ball shear test* rendah (*lower control level*) (GG1) dan ketebalan *ball bond* lebih dari 1x diameter *wire* (GG03) maka kerusakan disebabkan oleh tidak *optimize 1st bond parameter* (PP01).

2. *Rule* 2

   Jika gejala kerusakan adalah *reading ball shear test rendah* (*lower control level*) (GG01) dan bentuk *ball bond* normal dengan ukuran antar 2x - 5x diameter *wire* (GG02) maka kerusakan disebabkan oleh temperatur pada *heater block* tidak tepat (PP02).

3. *Rule* 3

   Jika gejala kerusakan adalah *reading ball shear test* rendah (*lower control level*) (GG01) dan bentuk *ball bond normal* dengan ukuran antara 2x - 5x (GG02) dan terdapat kelainan warna pada *bond pad* (GG05) maka kerusakan disebabkan oleh material yang *reject* (PP03).

4. *Rule* 4

   Jika gejala kerusakan adalah sebagian *ball bond* pada unit terlihat tipis (*flat ball*) (GG04) dan terdapat tanda gores pada bagian bawah *die pad* (GG06) maka kerusakan disebabkan oleh *downholder* kotor (terdapat *foreign material*) (PP04).

5. *Rule* 5

   Jika gejala kerusakan adalah sebagian *ball bond* pada unit terlihat tipis (*flat ball*)

(GG04) maka kerusakan disebabkan oleh *floating die pad* (PP05).

6. *Rule* 6

   Jika gejala kerusakan adalah terdapat kelainan warna pada permukaan *lead* (GG07) dan ukuran *wedge flare* kurang dari 1.3x diameter *wire* (GG10) dan *reading wedge pull* rendah (GG11) maka kerusakan disebabkan oleh *lead frame* terkontaminasi (PP06).

7. *Rule* 7

   Jika gejala kerusakan adalah terlihat *peeloff* pada *wedge flare* (GG09) dan terlihat *halfmoon imprint capillary* (GG12) maka kerusakan disebabkan oleh *floating* pada permukaan *lead* (PP07).

8. *Rule* 8

   Jika gejala kerusakan adalah ukuran *wedge flare* kurang dari 1.3x diameter *wire* (GG10) dan *reading wedge pull* rendah (GG11) maka kerusakan disebabkan oleh kurang *optimize wedge bond parameter* (PP08).

9. *Rule* 9

   Jika gejala kerusakan adalah tidak terlihat *imprint capillary* di *wedge* (GG08) dan ukuran *wedge flare* kurang dari 1.3x diameter *wire* (GG10) maka kerusakan disebabkan oleh impedansi *capillary* terlalu tinggi (PP09).

10. *Rule* 10

    Jika gejala kerusakan adalah ada bekas luka pada ujung *side rail* sebelah kanan atas atupun kiri atas (GG13) dan terlihat *wire* melengkung kebawah (*sagging wire*) (GG21) maka kerusakan disebabkan oleh tidak pas/sejajar *indexer table* dengan *slot magazine output* (PP10).

11. *Rule* 11

    Jika gejala kerusakan adalah ada bekas luka pada *tie bar* (GG14) dan ada indikasi *vacum error* (GG18) maka kerusakan disebabkan oleh *die pad* tidak tepat pada *cavity down holder* (PP11).

12. *Rule* 12

    Jika gejala kerusakan adalah ada bekas luka pada *side rail* bagian bawah atau atas sebelah kanan (GG15) dan strip tidak lancar di *indexer table* (GG22) maka kerusakan disebabkan oleh *indexer tabel* terlalu sempit (PP12).

13. *Rule* 13

    Jika gejala kerusakan adalah tidak ada bekas luka pada permukaan *lead frame* (GG16) dan terlihat goresan pada *wire* (GG17) maka kerusakan disebabkan oleh tidak *optimize loop parameter* (GG13).

14. *Rule* 14

    Jika gejala kerusakan adalah bentuk *wire* dan *loop* tertekan bengkok ke kiri (GG19) dan tidak terdapat luka gores pada *strip* (GG23) maka kerusakan disebabkan oleh tertabrak oleh *top clamp* (PP14).

15. *Rule* 15

    Jika gejala kerusakan adalah bentuk *wire* bengkong di daerah *necking* (*1st kink*) (GG20) dan tidak terdapat luka gores pada *strip* (GG23) maka kerusakan disebabkan oleh tertabrak oleh *capillary* (PP15).

Setelah tabel aturan *inference* (Table 3.7) disusun, maka langkah selanjutnya adalah membuat tabel keputusan. Berikut ini adalah tabel relasi gejala dan penyebab (Tabel 3.8) dari sistem pakar yang akan dibuat:

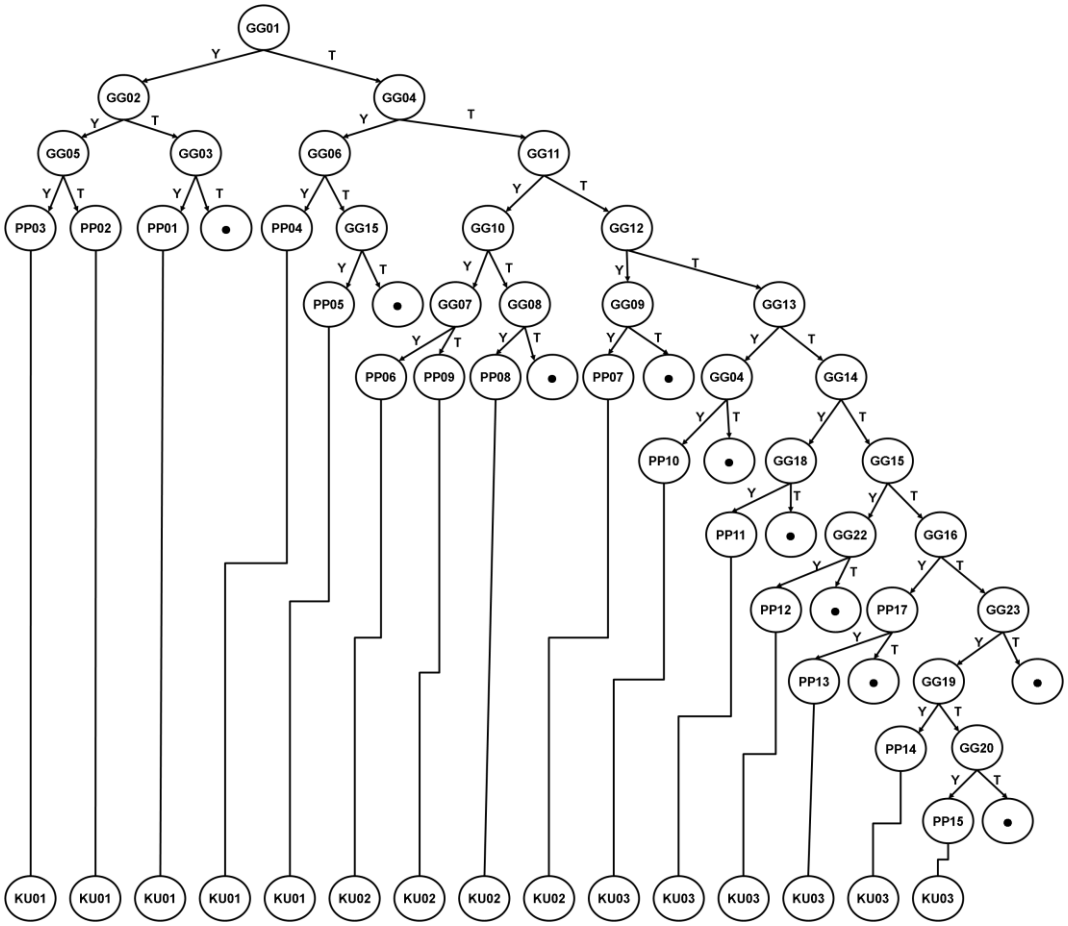
Tabel 3.8 Tabel Relasi Gejala dan Penyebab

Gejala	KU01					KU02				KU03					
	PP01	PP02	PP03	PP04	PP05	PP06	PP07	PP08	PP09	PP10	PP11	PP12	PP13	PP14	PP15
GG01	√	√	√												
GG02		√	√												
GG03	√														
GG04				√	√										
GG05			√												
GG06				√											
GG07						√									
GG08								√							
GG09							√								
GG10						√			√						
GG11						√		√	√						
GG12							√								
GG13										√					
GG14											√				
GG15												√			
GG16													√		
GG17													√		
GG18											√				
GG19														√	
GG20															√
GG21										√					
GG22												√			
GG23														√	√

(Sumber: Data Penelitian, 2017)

Pada Tabel 3.8 diatas, kolom penyebab (PP) dikelompokkan dan diurutkan berdasarkan kode kerusakan *IC* (KU), setelah itu diberi tanda centang untuk baris kode gejala (GG) yang memenuhi aturan dari masing-masing penyebab. Hal ini dibuat untuk memudahkan dalam menyusun aturan kaidah produksi sistem pakar yang akan dibuat.

Setelah disusun tabel relasi gejala dan penyebab (Tabel 3.8**)** diatas maka dapat dibuat pohon keputusan (Gambar 3.2) sebagai berikut:



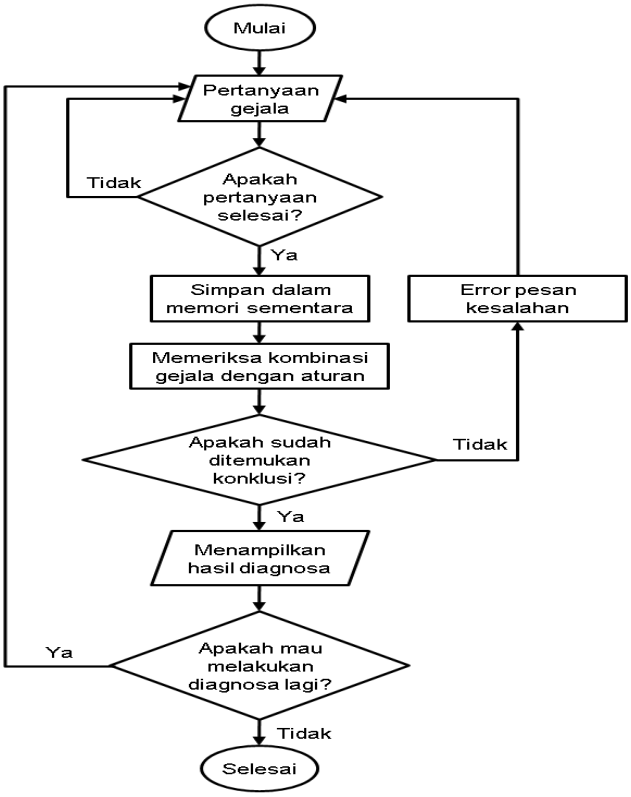
Gambar 3.2 Pohon Keputusan
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.2 menunjukkan pohon keputusan yang memperlihatkan hubungan antara gejala dan penyebab. Penelusuran menggunakan sistem *dept first search*. *Depth first search* merupakan metode pencarian yang dilakukan pada suatu simpul dalam setiap level yang dimulai dari kiri. Jika pada level yang terdalam solusi belum ditemukan, maka pencarian dilanjutkan pada simpul sebelah kanan dan simpul yang kiri dapat dihapus dari memori (Suyanto, 2014 dalam jurnal penelitian Harsono, dkk., 2015). Penelusuran dimulai dari gejala awal yaitu *reading ball shear test* rendah (*lower control level*) (GG01) . Gejala ini dipilih pertama karena untuk kerusakan *IC* yaitu *Ball Non Stick*, yang mengukur daya tempel *ball*

bond terhadap *bond pad* adalah *ball shear test*. Semakin tinggi hasil pengukuran *ball shear* maka kondisinya baik, jika hasilnya rendah maka kondisinya kurang baik (berpotensi untuk menghasilkan *reject Ball Non Stick*). Proses penelusuran dari pohon keputusan ditentukan berdasarkan jawaban yang dipilih oleh *user*. Apabila *user* menjawab "ya" (Y), maka penelusuran menuju ke simpul sebelah kiri level berikutnya yaitu bentuk *ball bond* normal dengan ukuran antara 2x - 5x diameter *wire* (GG02). *User* akan diberikan pertanyaan (GG02), jika menjawab "ya" (Y) maka pohon akan mengarah kesimpul kiri yaitu akan mengajukan pertanyaan lagi tentang bentuk ball bond normal dengan ukuran antara 2x - 5x diameter wire (GG05). Jika *user* menjawab "ya" (Y) maka ditemukan penyebab yaitu material yang *reject* (PP03) tetapi apabila menjawab "tidak" (T) maka jawaban diagnosanya adalah temperatur pada *Heater Block* tidak tepat (PP02). Jika user menjawab "tidak" (T) pada saat pertanyaan *ball bond* normal dengan ukuran antara 2x - 5x diameter *wire* (GG02), maka penelusuran akan menuju ke simpul sebelah kanan level berikutnya yaitu gejala ke 3 (GG03). Apabila pertanyaan ketebalan *ball bond* lebih dari 1x diameter *wire* (GG03) dijawab "ya" (Y) maka ditemukan penyebab yaitu tidak *optimize 1st bond* parameter, tetapi apabila menjawab "tidak" (T) maka akan menemukan simpul mati (•) yang berati tidak menghasilkan kesimpulan atau tidak menemukan kunklusi. Jika penelusuran sampai pada simpul mati (•) maka penelusuran akan kembali lagi ke kondisi awal yaitu melakukan penelusuran dari simpul pertama (GG1).

3.3.3 Struktur Kontrol (Mesin Inferensi)

Mesin inferensi dalam sistem pakar ini menggunakan metode penelusuran *forward chaining*, yaitu analisa penelusuran maju mulai dari fakta-fakta yang berupa ciri-ciri/gejala yang ditimbulkan pada kerusakan *IC* sehingga dapat meguji kebenaran hipotesis yaitu penyebab kerusakan, sehingga menghasilkan solusi dari penyebab kerusakan. Berikut ini f*lowchart* mesin inferensi sistem pakar *forward chining* digambarkan pada Gambar 3.3 berikut ini:



Gambar 3.3 Flowchart Mesin Inferensi Sistem Pakar
(Sumber: Data Penelitian, 2017)

Langkah-langkah yang digunakan dalam proses penelusuran masalah adalah sebagai berikut:

1. Mulai mengakses sistem.

2. Mengajukan pertanyaan tentang gejala-gejala kerusakan *IC* di mesin *Wire Bond* KNS IConn.
3. Setelah jawaban pertanyaan dijawab oleh user, maka jawaban akan disimpan kedalam tabel gejala kerusakan sementara.
4. Sistem akan memeriksa jawaban tersebut dan membandingkannya denga rule yang sudah dibuat sehingga menghasilkan jawaban, jika semua pertanyaan yang diajukan kepada user tetapi belum memenuhi konklusi pada sistem, maka akan keluar pesan pengulangan diagnosa.
5. Jika jawaban pertanyaan dari user sesuai dengan rule yang ada pada database, maka aplikasi akan menampilkan diagnosa berupa gejala kerusakan, penyebab kerusakan, dan solusi untuk memperbaiki kerusakan mesin Wire Bond KNS IConn.
6. Ada pertanyaan untuk melakukan pengulangan diagnosa, jika menjawab "Ya" maka akan kembali ke menu diagnosa awal dan jika menjawab "Tidak" maka diagnosa berhenti. Selesai.

3.3.4 Desain *UML* (*Unified Modeling Language*)

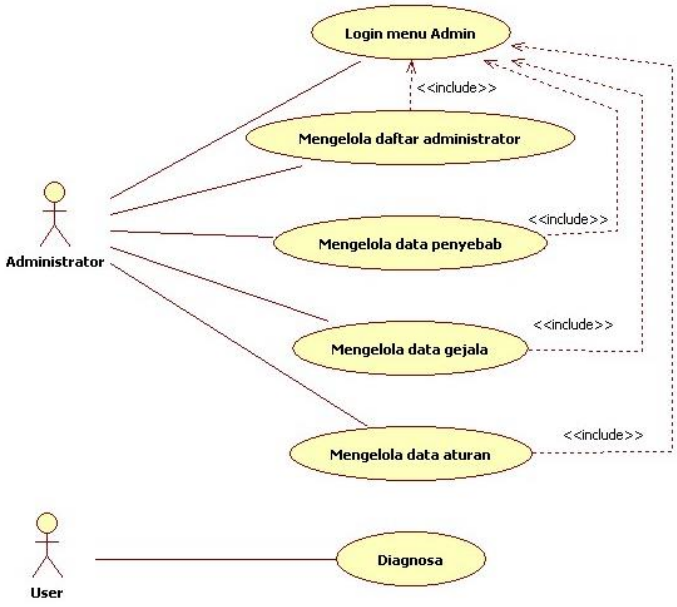
UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (A.S. dan Shalahuddin, 2011: 113).

Desain *UML* dibuat untuk memudahkan dalam pembuatan program. Pemodelan *UML* menggunakan alat bantu *software StarUML* versi 2.5.1. Berikut ini adalah diagram UML yang digunakan dalam perancangan program:

1. *Use Case Diagram*

Use case diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem informasi yang akan dibuat. Diagram ini mendeskripsikan sebuah interaksi antara satu sistem atau lebih aktor dengan sistem informasi yang akan dibuat (A.S dan Shalahuddin, 2011: 130-131).

Pada Gambar 3.4 berikut ini menggambarkan *use case diagram* yang dirancang dalam penelitian ini:



Gambar 3.4 Use Case Diagram
(Sumber: Data Penelitian, 2017)

Terdapat 2 aktor yaitu *administrator* dan u*ser*. *Administrator* melakukan interaksi dengan sistem berupa mengelola daftar *administrator*, mengelola data penyebab, mengelola data gejala, dan mengelola data aturan. Semua interaksi yang dilakukan

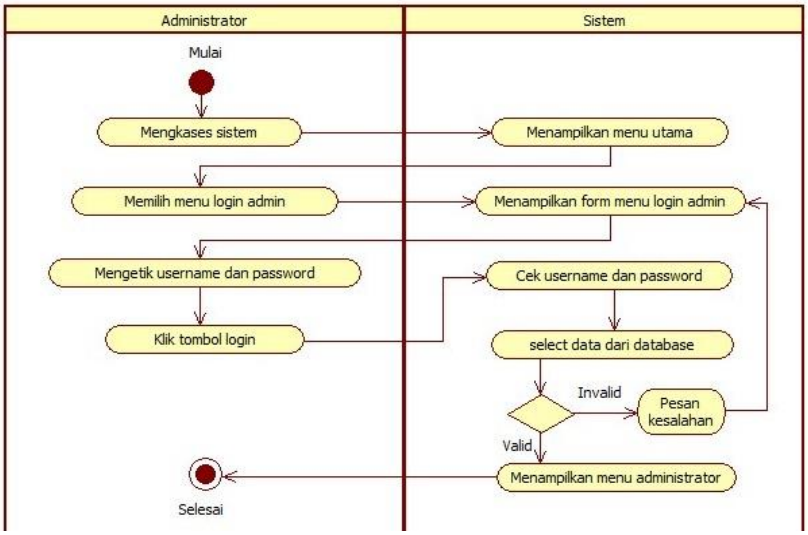
administrator dilakukan setelah *login* pada Menu Admin. Sedangkan *user* berinteraksi dengan sistem yaitu melakukan diagnosa. Sebelum diagnosa dilakukan, user dimunta untuk memasukkan nama pada form pendaftaran. Diagnosa dilakukan dengan cara menjawab pertanyaan yang diajukan oleh sistem, setelah semua jawaban sesuai *rule*, maka sistem akan menampilkan penyebab dan solusi permasalah. Kegiatan yang dilakukan *user* tanpa melalui proses *login* ke sistem.

2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis. *Activity diagram* menggambarkan aktivitas sistem bukan apa yang dilakukan aktor (A.S dan Shalahuddin: 2011: 134). Berikut ini adalah *activity diagram* yang dirancang dalam penelitian ini:

a. *Activity diagram login adminiatrator*

Activity diagram login administrator merupakan *UML* yang menggambarkan kegiatan pengguan pada halaman *login administrator*.

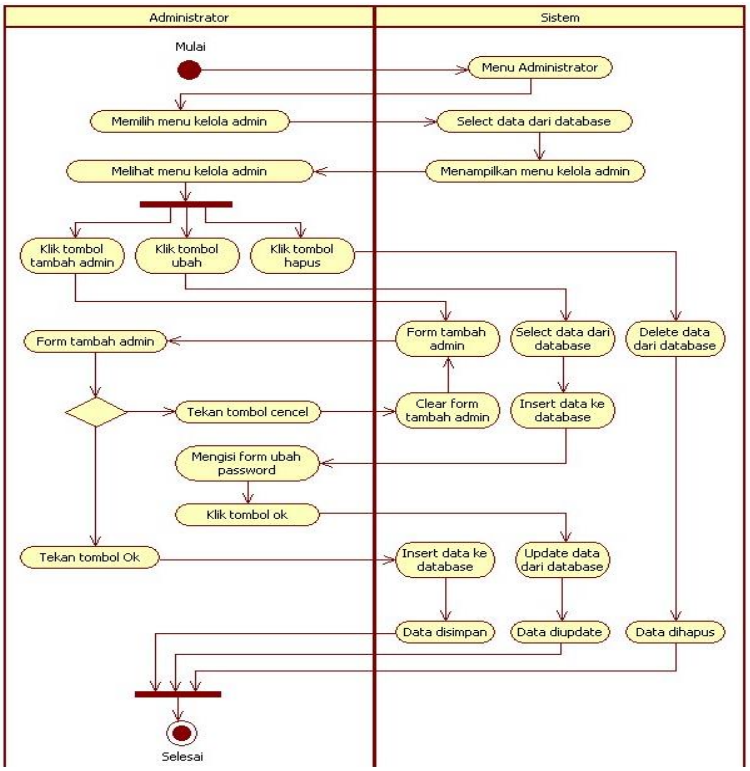


Gambar 3.5 *Activity Diagram* Login Administrator
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.5 diatas, proses *login administrator* adalah *administrator* mengases sistem, kemudian sistem akan menampilkan halaman Menu Utama. Kemudian *administrator* memilih Menu Login Admin, maka sistem akan menampilkan *form* menu *login* admin. Administrator mengetik *username* dan *password* kemudian klik tombol *login*. Maka sistem akan mengecek *username* dan *password* kemudian dicocokkan dengan data yang ada di *database*. Jika *username* dan *password* tidak sesuai dengan yang ada di *database* maka sistem akan menampilkan pesan kesalahan dan sistem kembali menampilkan *form* menu *login* admin, apabila benar maka sistem akan menampilkan menu *administrator*.

b. *Activity diagram* kelola menu admin

Activity diagram kelola admin merupakan kegiatan *administrator* dalam mengelola daftar pengguna (*administrator*). Berikut ini gambar *activity diagram* kelola menu admin (Gambar 3.6):



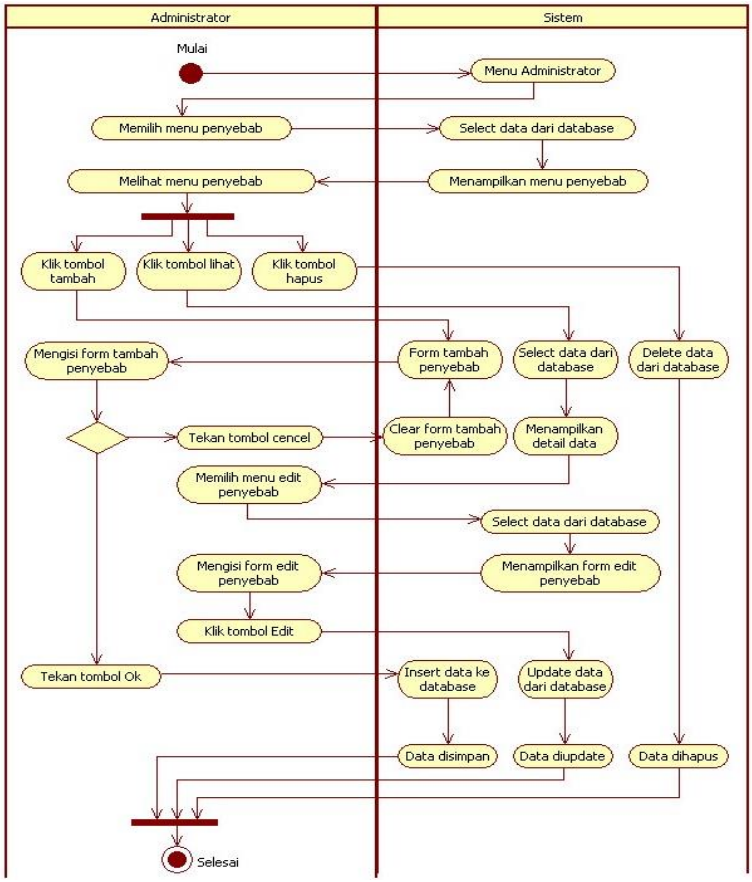
Gambar 3.6 *Activity Diagram* Kelola Menu Admin
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.6 diatas, menu administrator yang terbuka setelah pengguna melakukan *login* admin. Sistem akan menampilkan menu admin, kemudian pengguna (administrator) memilih menu kelola admin. Sistem memanggil data dari database dan menampilkan menu kelola admin. Pengguna melihat 3 pilihan yaitu tambah admin, ubah, dan hapus. Jika pengguna mengklik tombol tambah admin maka sistem akan menampilkan *form* tambah admin, kemudian pengguna mengisi *form* tambah admin. Pengguna klik tombol Ok maka data akan dimasukkan ke *database* kemudian data disimpan di *database*, jika pengguna menekan tombol *cancel* maka sistem akan membersihkan *form* tambah admin. Jika pengguna mengklik tombol ubah, maka sistem akan mengambil data dari *database*. Pengguna mengisi *form* ubah *password* kemudian klik tombol Ok. Sistem akan melakukan

update database. Jika pengguna pengguna menekan tombol *delete* maka data yang ada di *database* akan terhapus.

c. *Activity diagram* mengelola penyebab

Activity diagram mengelola penyebab merupakan diagram *UML* yang menggambarkan kegiatan administrator dalam mengelola data penyebab. Berikut ini adalah gambar *activity diagram* kelola menu admin (Gambar 3.6):



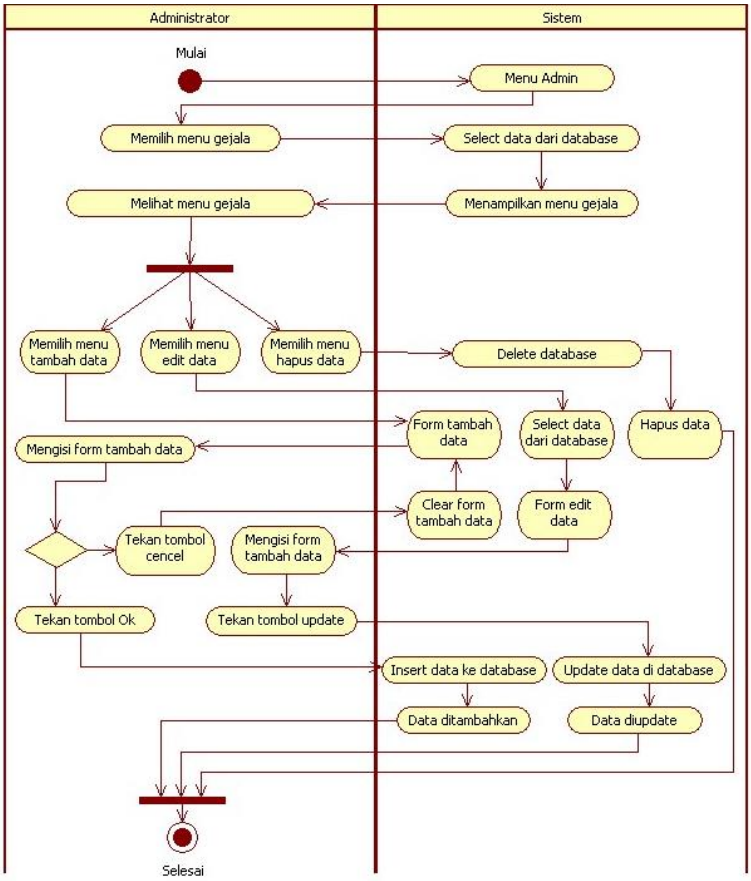
Gambar 3.7 *Activity Diagram* Mengelola Penyebab
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.7 diatas, *administrator* mulai mengakses Menu Administrator kemudian sistem menampilkan Menu Administrator. *Administrator* memilih menu penyebab sehingga sistem mengambil data dari *database* dan menampilkan Menu

Penyabab. *Administrator* melihat ada 3 pilihan yaitu tambah, lihat, dan hapus penyebab. Pertama, jika *administrator* menekan tombol tambah maka sistem akan menampilkan *form* tambah penyebab. *Administrator* akan mengisi *form* tambah penyebab, kemudian tekan tombol Ok maka sistem akan memasukkan data baru tersebut ke *database* kemudian menampilkan pesan data disimpan. Kedua, jika *administrator* menekan tombol lihat maka sistem akan mengambil data dari *database* kemudian menampilkan detail data. Administrator menekan tombol *edit* penyebab maka sistem akan mengambil data dari *database* dan menampilkan *form edit* penyebab. Kemudian administrator mengisi *form edit* penyebab setelah itu menekan tombol *edit* maka sistem akan melakukan *update* data ke *database* dan menampilkan pesan data di-*update*. Ketiga, jika *administrator* menekan tombol hapus maka sistem akan mengapus data dari *database* dan menampilkan pesan data dihapus. Proses selesai.

d. *Activity diagram* mengelola gejala

Activity diagram mengelola gejala merupakan diagram *UML* yang menggambarkan kegiatan *administrator* dalam mengelola data gejala. Berikut ini gambar *activity diagram* mengelola gejala (Gambar 3.8):



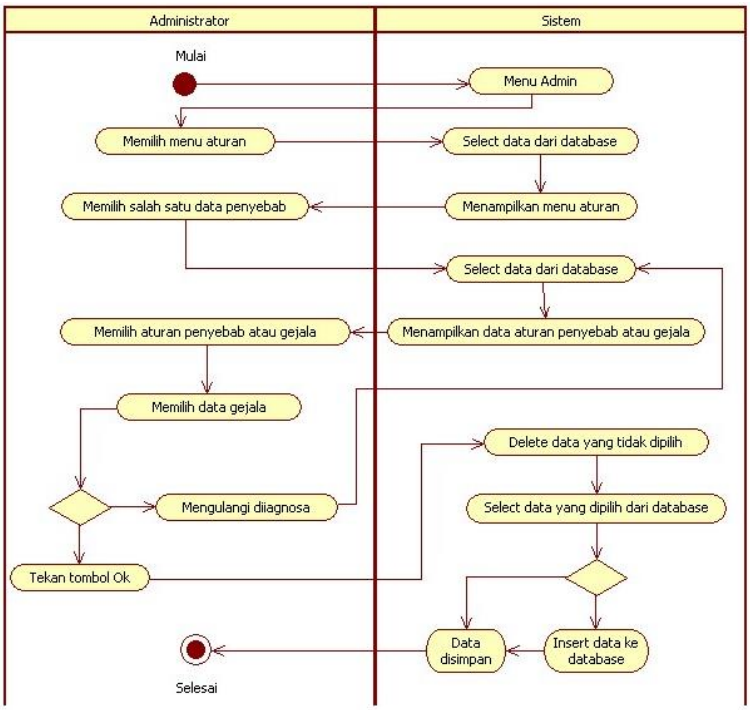
Gambar 3.8 *Activity Diagram* Mengelola Gejala
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.8 diatas, *administrator* mulai mengakses Menu Admin dan sistem menampilkan Menu Admin. *Administrator* memilih menu gejala kemudian sistem mengambil data dari *database* dan menampilkan menu gejala. *Administrator* melihat ada 3 pilihan yaitu tambah, *edit*, dan hapus gejala. Pertama, jika *administrator* menekan tombol tambah maka sistem akan menampilkan *form* tambah gejala. *Administrator* akan mengisi *form* tambah gejala, kemudian tekan tombol tambah maka sistem akan memasukkan data baru tersebut ke *database* kemudian menampilkan pesan data disimpan. Kedua, jika *administrator* menekan tombol *edit* maka sistem akan mengambil data dari *database* kemudian

menampilkan *form edit* data gejala. *Administrator* menekan tombol *update* maka sistem akan melakukan *update* data ke *database* dan menampilkan pesan data di-*update*. Ketiga, jika *administrator* menekan tombol hapus maka sistem akan mengapus data gejala dari *database*. Proses selesai.

e. *Activity diagram* aturan

Activity diagram aturan merupakan diagram *UML* yang menggambarkan kegiatan *administrator* dalam mengelola data aturan. Berikut ini gambar *activity diagram* aturan (Gambar 3.9):



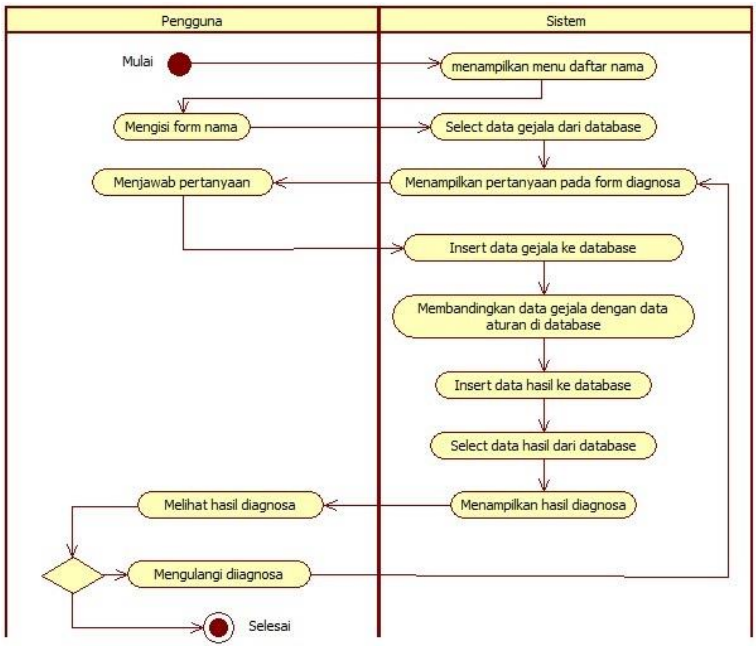
Gambar 3.9 *Activity Diagram* Aturan
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.9 diatas, *administrator* mulai dengan mengakses Menu Admin dan sistem menampilkan Menu Admin. *Administrator* memilih Menu Aturan kemudian sistem akan mengambil data dari *database* dan menampilkan data aturan penyebab atau gejala. Administrator memilih aturan penyebab atau gejala

dan mamilih salah satua atau beberapa data gejala kemudian menekan tombol ok maka sistem akan menghapus data yang tidak dipilih. Data yang sudah dipilih dari *database* akan ditambahkan jika data sebelumnya belum ada dan jika sudah ada maka data langsung disimpan.

f. *Activity diagram* diagnosa

Activity diagram diagnosa merupakan diagram *UML* yang menggambarkan kegiatan pengguna (*user*) dalam menggunakan menu diagnosa. Berikut ini gambar *activity diagram* diagnosa (Gambar 3.10):



Gambar 3.10 *Activity Diagram* Diagnosa
(Sumber: Data Penelitian, 2017)

Pada Gambar 3.10 diatas, pengguna memulai diagnosa dengan menampilkan *form* daftar nama, maka sistem mengambil data dari *database* dan menampilkan pertanyaan pada *form* diagnosa. Pengguna menjawab pertanyaan yang berupa *multiple choice* ya atau tidak. Kemudian sistem akan mengambil data dari *database* dan membandingkan jawaban dengan data aturan di *database*. Kemudian sistem

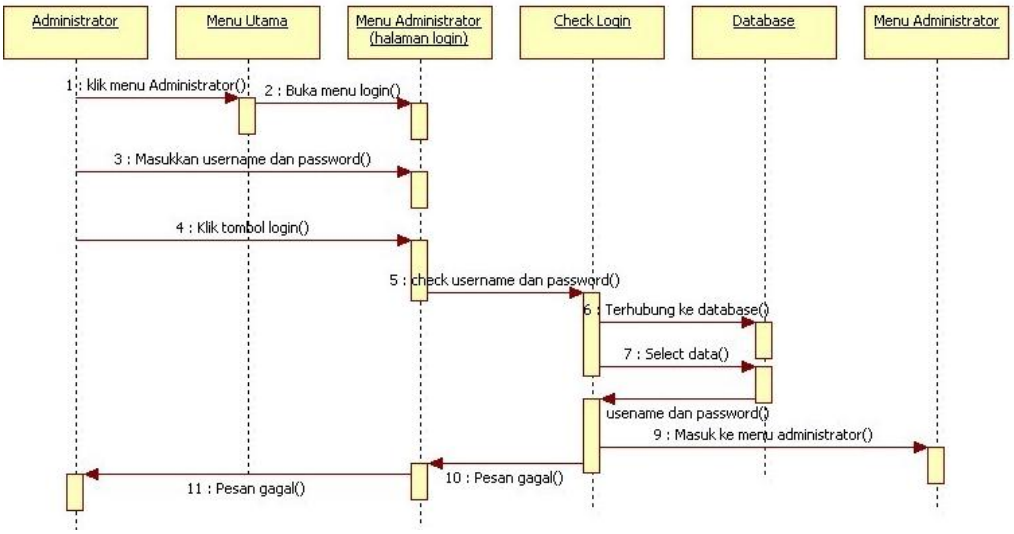
akan memasukkan data hasil ke *database* sementara dan mengambil data dari *database* untuk ditampilkan. Pengguna akan melihat hasil diagnosa. Jika pengguna menekan tombol diagnosa ulang maka sistem akan mengulangi diagnosa dan menampilkan kembali pertanyaan pada *form* diagnosa. Jika pengguna tidak mengulangi diagnosa maka proses selesai.

3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek (A.S dan Shalahuddin: 2011: 137). Berikut ini adalah *sequence diagram* yang dirancang dalam penelitian ini:

a. *Sequence diagram log in administrator*

Sequence diagram log in administrator merupakan urutan waktu kegiatan *administrator* saat melakukan *log in*. Berikut ini gambar *sequence* diagram *login administrator* (Gambar 3.11):

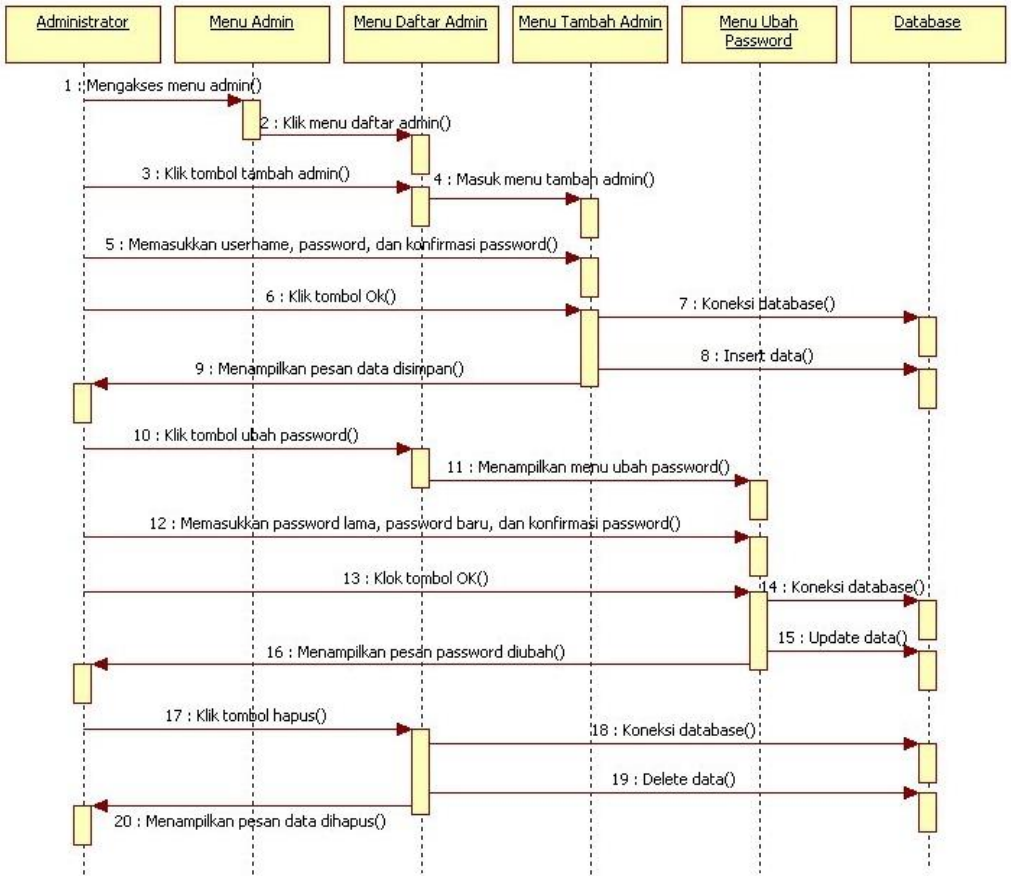


Gambar 3.11 *Sequence Diagram Login Administrator*
(Sumber: Data Penelitian, 2017)

Administrator melakukan klik Menu Administrator, kemudian Menu Utama Administrator menampilkan/membuka *form login*. *Administrator* memasukkan *username* dan *password* ke menu Halaman Login kemudian *administrator* melakukan klik tombol Login. Dari Halaman Login Administrator, sistem akan mengecek *username* dan *password* yang sudah dimasukkan kemudian sistem akan terhubung dengan *database*. Sistem Check Login akan mengambil data dari *database*, setelah itu *username* dan *password* akan dicocokan oleh sistem. Jika *username* dan *password* benar maka Menu Halaman Administrator akan ditampilkan. Jika *username* dan *password* salah (tidak sesuai dengan *database*) maka sistem akan menampilkan pesan gagal di Halaman Login, dan *adminitrstor* akan mengetahui pesan gagal tersebut.

b. *Sequence diagram* kelola *administrator*

   Sequence diagram kelola *administartor* merupakan urutan waktu kegiatan *administrator* saat mengelola daftar pengguna (*administartor*). Berikut ini *sequence diagram* kelola *administartor* digambarkan pada Gambar 3.12.

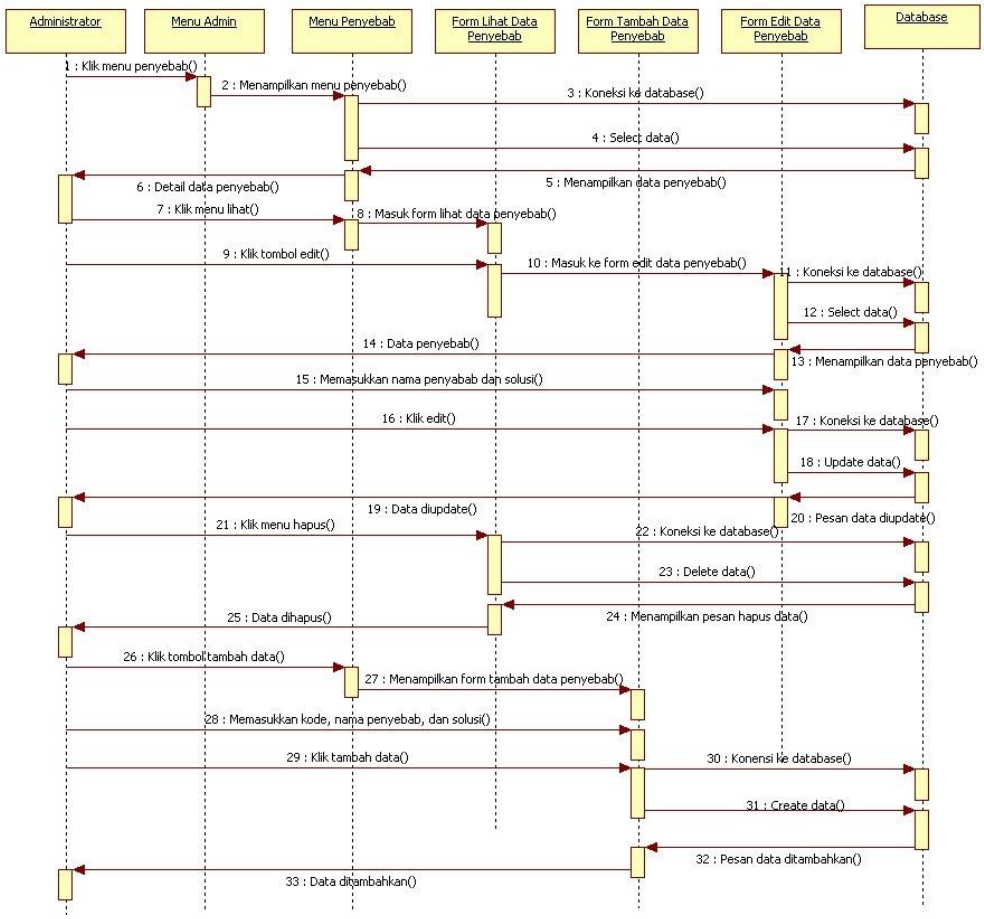


Gambar 3.12 *Sequence Diagram* Kelola Menu Admin
(Sumber: Data Penelitian, 2017)

Administrator mengakses Menu Admin, kemudian menekan Menu Daftar Admin. Setelah Menu Daftar Admin terbuka, *administrator* menekan tombol Tambah Admin. Dari menu Daftar Admin, sistem akan mengarahkan ke Menu Tambah Admin. Kemudian administrator akan memasukkan *username*, *password* dan konfirmasi *password* pada Menu Tambah Admin dan administrator menekan tombol Ok. Menu Tambah Admin melakukan koneksi ke *database* dan memasukkan data ke dalam *database*. Menu Tambah Admin akan menampilkan pesan data disimpan pada *administrator*.

c. *Sequence diagram* mengelola penyebab

Sequence diagram mengelola penyebab merupakan urutan waktu kegiatan *administrator* saat melakukan pengelolaan penyebab kerusakan. Berikut ini gambar s*equence diagram* mengelola penyebab (Gambar 3.13):



Gambar 3.13 *Sequence Diagram* Mengelola Penyebab
(Sumber: Data Penelitian, 2017)

Administrator mengakses Menu Admin, kemudian pilih Menu Penyebab maka terhubung ke *database* dan mengambil data. Pada Menu Admin akan menampilkan seluruh data penyebab. *Administrator* menekan tombol lihat pada Menu Penyebab, kemudian sistem akan menampilkan *Form* Lihat Data Penyebab. *Administrator* menekan tombol e*dit* pada *Form* Lihat Data Penyebab maka sistem akan menampilkan *Form Edit* Data Penyebab. Dari *Form Edit* Data Penyebab akan

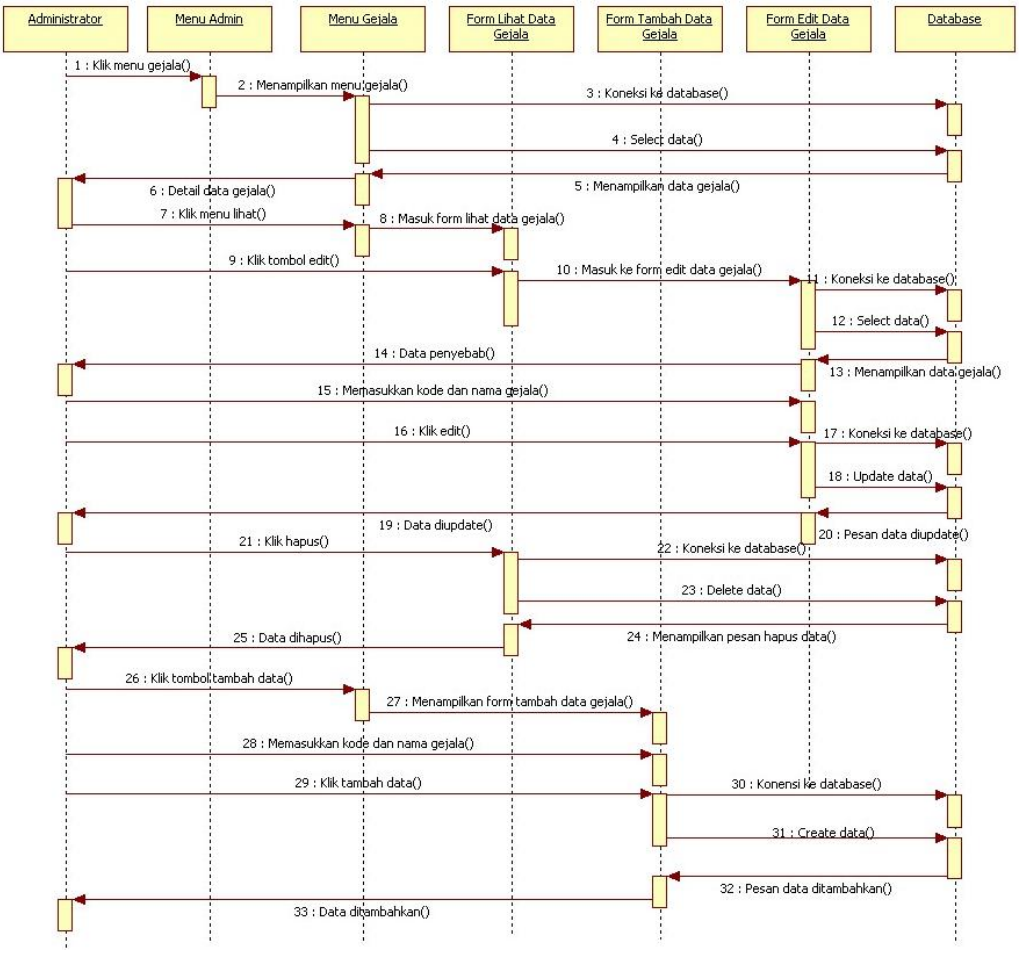
terkoneksi ke *database* dan mengambil data. *Form Edit* Data Penyebab menampilkan data penyebab yang dipilih *administrator* untuk diubah. *Administrator* memasukkan data penyebab dan solusi yang baru kemudian menekan tombol e*dit*, maka sistem akan melakukan *update* data ke *database*. Setelah *database* diperbarui maka akan keluar pesan bahwa data diperbarui.

Administrator menekan tombol Hapus pada Form Lihat Data Penyebab, maka akan terhubung ke *database* dan menghapus data. Jika data sudah terhapus dari *database* maka sistem akan menampilkan pesan data sudah dihapus.

Administrator menekan tombol Tambah Data pada Form Lihat Data Penyebab. Menu Penyebab menampilkan Form Tambah Data Penyebab. Kemudian *administrator* memasukkan nama penyebab dan solusi kedalam Form Tambah Data Penyebab, tetapi untuk kode sudah secara otomatis dari sistem. Setelah semua *form* terisi maka *administrator* menekan tombol Tambah Data, maka sistem akan terkoneksi ke *database* dan membuat data baru. Kemudian sistem akan menampilkan pesan data ditambahkan ke pada *administrator*.

d. *Sequence diagram* mengelola gejala

Sequence diagram mengelola gejala merupakan urutan waktu kegiatan *administrator* saat mengelola gejala kerusakan. Berikut ini *sequence diagram* mengelola gejala digambarkan pada Gambar 3.14.



Gambar 3.14 *Sequence Diagram* Mengelola Gejala
(Sumber: Data Penelitian, 2017)

Administrator menekan tombol Menu Gejala pada Menu Admin, kemudian sistem menampilkan Menu Gejala. Menu Gejala terkoneksi dengan *database* dan mengambil data, kemudian menampilkan data gejala ke Menu Gejala. Menu Gejala akan terkoneksi dengan *database* dan mengambil data kemudian menampilkan detail data gejala ke dalam Menu Gejala. *Administrator* menekan tombol Lihat, maka sistem akan menampilkan Form Lihat Data Gejala. Kemudian *administrator* menekan tombol Edit, maka akan masuk ke Form Edit Data Gejala. Sistem akan terhubungi dengan *database* dan mengambil data, kemudian menampilkan data

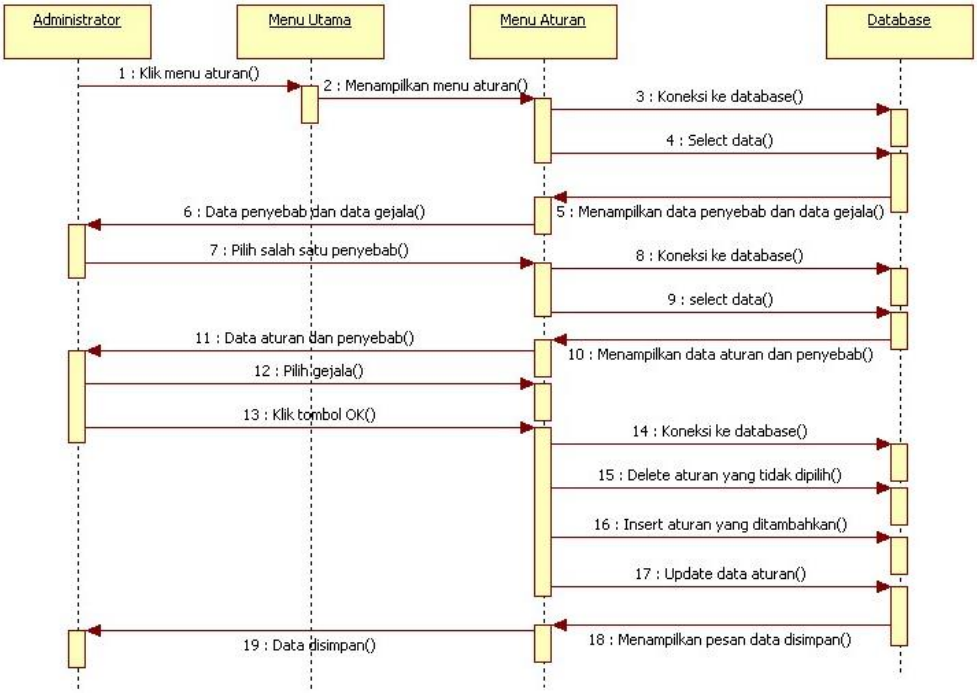
yang akan diubah. Administrator memasukkan data gejala yang baru dan menekan tombol Edit. Sistem akan terhubung dengan *database* dan memperbarui data, kemudian menampilkan pesan bahwa data sudah diperbarui kapada *administrator*.

Administrator menekan tombol Hapus pada Form Lihat Data Gejala, maka sistem akan terhubung dengan *database* dan menghapus data gejala dari *database*. Sistem akan menampilkan pesan bahwa data telah dihapus kepada *administrator*.

Administrator menekan tombol Tambah Data maka sistem menampilkan Form Tambah Data. *Administrator* memasukkan nama gejala pada Form Edit Data Gejala. Sistem akan terhubung dengan *database* dan akan membuat data baru, setelah data ditambahkan maka sistem akan menampilkan pesan data ditambahkan.

e. *Sequence diagram* mengelola aturan

Sequence diagram mengelola aturan merupakan urutan waktu kegiatan administrator saat melakukan pengelolaan aturan. Berikut ini *sequence diagram* mengelola aturan digambarkan pada Gambar 3.15.



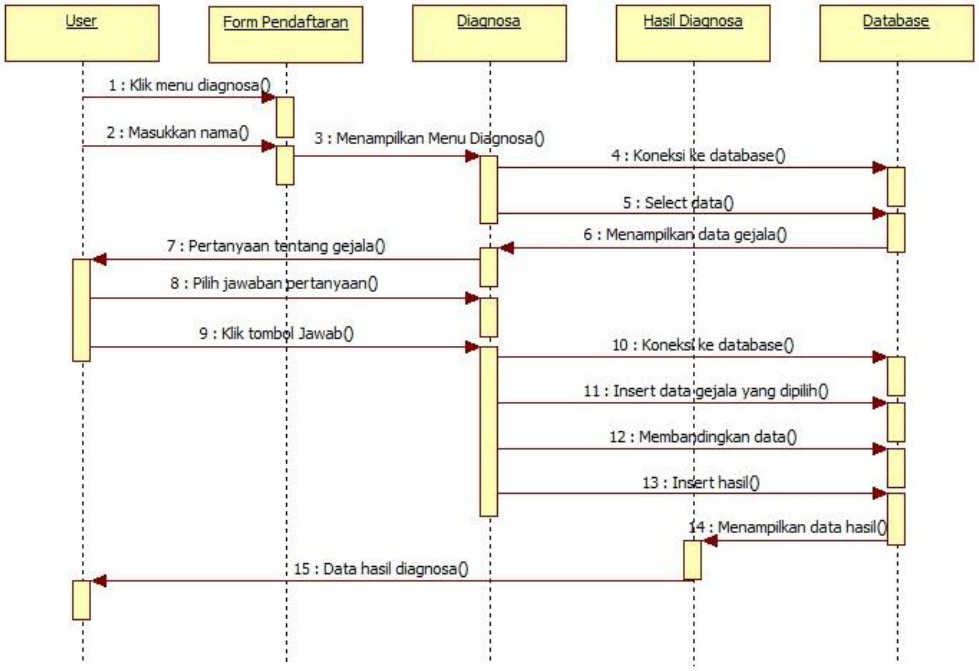
Gambar 3.15 *Sequence Diagram* Aturan
(Sumber: Data Penelitian, 2017)

Administrator menekan tombol Menu Aturan dari Menu Utama Administrator, maka sistem akan menampilkan Menu Aturan. Kemudian terhubung ke *database* dan mengambil data. Sistem menampilkan data penyebab dan data gejala. *Administrator* memilih salah satu kode kerusakan *IC* dan juga penyebab, kemudian mencentang gejala yang sesuai. Sistem akan terhubung dengan *database*. Jika gejala yang dicentang merupakan pembaruan data maka sistem menghapus gejala yang lama dan akan memasukkan data yang baru. Jika gejala yang dicentang merupakan gejala baru maka sistem akan menambahkan data baru ke *database*. Setelah itu administrator menekan tombol Ok untuk menyimpan data. Setelah data disimpan dalam *database* maka sistem akan menampilkan pesan data disimpan.

f. *Sequence diagram* diagnosa

Sequence diagram diagnosa merupakan urutan waktu kegiatan pengguna

(*user*) saat melakukan diagnosa kerusakan *IC*. Berikut ini gambar *sequence diagram* diagnosa (Gambar 3.16):



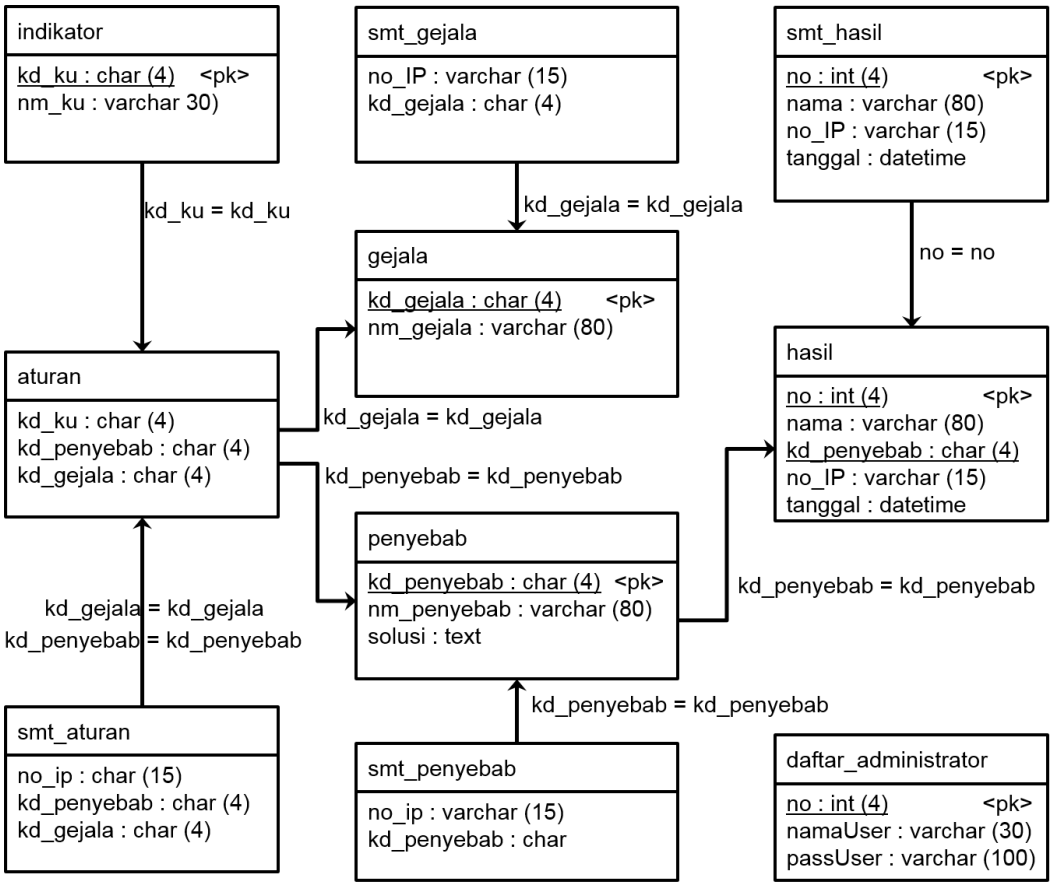
Gambar 3.16 *Sequence Diagram* Diagnosa
(Sumber: Data Penelitian, 2017)

User menekan tombol Menu Diagnosa kemudian tampil Form Pendaftaran. *User* memasukkan nama di Form Pendaftaran kemudian sistem akan menampilkan Menu Diagnosa. Selanjutnya terhubung dengan *database* dan mengambil data dari *database*. Sistem menampilkan data gejala, kemudian mengajukan pertanyaan tentang gejala kerusakan *IC*. *User* memilih salah satu jawaban kemudian tekan tombol Jawab. Maka akan terhubung dengan *database* dan memasukkan data gejala yang diperoleh ke *database* sementara, sistem akan membandingkan data sementara dengan data aturan. Jika sudah menemukan konklusi maka hasil diagnosa disimpan ke *database* dan memampilkan data hasil. Data hasil diagnosa

berupa nama kerusakan *IC*, gejala kerusakan, penyebab kerusakan, dan solusi kerusakan.

3.3.5 Desain *Database*

A.S. dan Shalahuddin (2011: 48) *PDM* (*Physical Data Model*) adalah model yang menggunakan sejumlah table untuk menggambarkan data serta hubungan antar data-data tersebut. PDM merupakan konsep yang menerangkan detail bagaimana data disimpan di dalam *database*. Berikut ini adalah konsep database yang dibuat dalam penelitian:



Gambar 3.17 Desain *Phisical Data Model*
(Sumber: Data Penelitian, 2017)

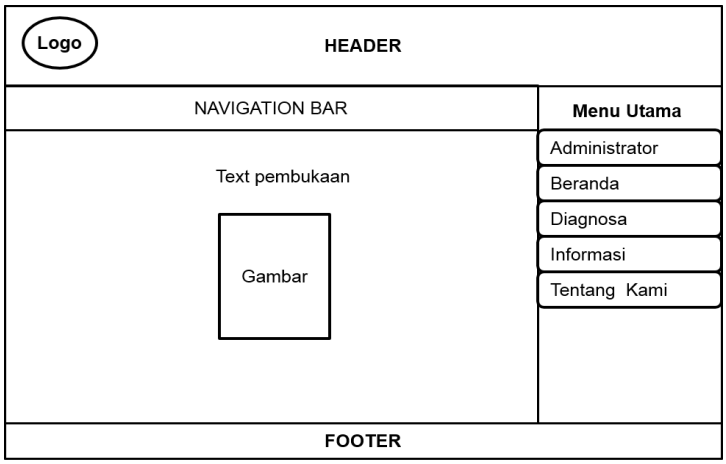
Pada Gambar 3.17 diatas, terdapat 10 tabel *database* yang terdiri dari: tabel indikator, tabel aturan, tabel smt_aturan, tabel gejala, tabel smt_gejala, tabel penyebab, tabel smt_penyebab, tabel hasil, tabel smt_hasil, dan tabel daftar_administrator. Dalam tabel indikator mempunyai kd_ku yang akan digunakan pada tabel aturan yaitu memiliki relasi yang sama pada kd_ku tabel aturan. Tabel tabel smt_aturan merupakan penyimpanan sementara untuk menyimpan data penyebab (kd_penyebab) dan gejala (kd_gejala) yang akan digunakan pada tabel aturan. Tabel smt_gejala berfungsi untuk menyimpan data sementara dari tabel gejala yang berupa gejala (kd_gejala) yang dijawab pengguna saat diagnosa. Begitu juga tabel smt_penyebab yang menyimpan data sementara penyebab yang akan terhubung dengan tabel penyebab dengan kesamaan relasi kode penyebab (kd_penyebab). Dari data gejala yang didapatkan saat diagnosa akan dicocokan denga tabel aturan yang sudah dibuat, maka akan diketahui penyebab kerusakan (kd_penyebab) yang akan dicocokkan dengan data pada tabel aturan. Sedangkan tabel smt_hasil digunakan menyimpan sementara data pengunjung. Tabel hasil akan memanggil isi dari tabel pengunjung (smt_hasil) dan tabel penyebab (kd_penyebab) berupa nama penyebab dan solusi. Tabel daftar_administrator berisi nama dan *password* pengguna.

3.3.6 Desain Antarmuka

Berikut ini adalah desain antarmuka yang akan dibuat pada aplikasi sistem pakar kerusakan *IC* di mesin *Wire Bond* IConn:

1. Rancangan Halaman Beranda

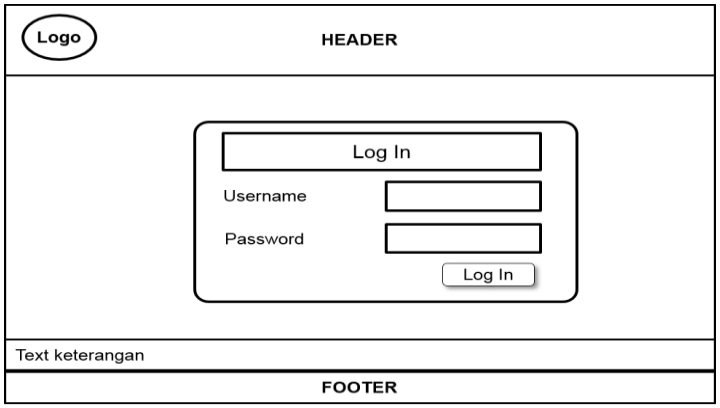
Halaman Beranda menampilkan informasi tentang aplikasi sistem pakar deteksi kerusakan *IC* di mesin *Wire Bond* IConn. Berisi tentang pengertian sistem pakar dan acara menggunakan aplikasi.



Gambar 3.18 Rancangan Halaman Beranda
(Sumber: Data Penelitian, 2017)

2. Rancangan Halaman *Log In Administrator*

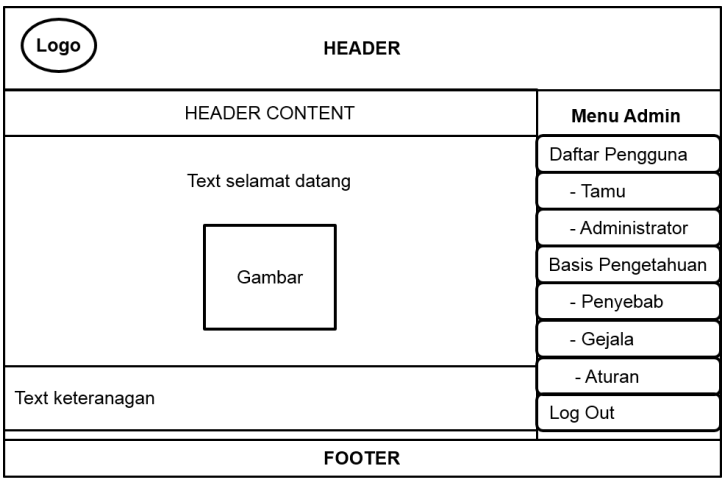
Halaman *Log In Administrator* akan tampil saat *user* menekan Menu Administrator. Pada halaman ini *user* dihadapkan *form* untuk mengisi *username* dan *password* sebagai *administrator*. Jika *username* dan *password* benar maka akan diarahkan ke halaman Administrator, jika salah maka akan keluar pesan kesalahan.



Gambar 3.19 Rancangan Halaman *Login Administrator*
(Sumber: Data Penelitian, 2017)

3. Rancangan Halaman Utama Administrator

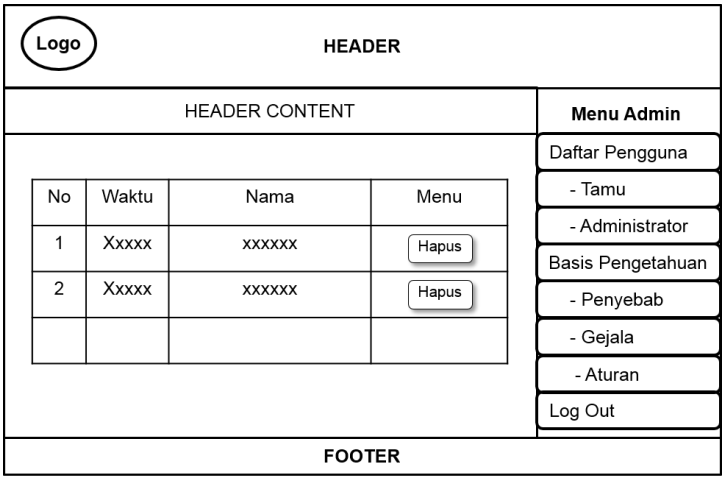
Halaman Utama Administrator berbeda dengan halaman pengguna, yaitu pengguna akan diijinkan masuk jika sudah benar memasukkan *username* dan *password* di menu *Log In Administrator*. Halaman ini terdapat tambahan beberapa menu yaitu menu Penyebab, menu Gejala, menu Aturan, menu Daftar Pengguna tamu dan administrator. Jika ingin keluar dari halaman *Administrator*, ada tombol *Logout* yang akan mengarahkan ke halaman *Log In Administrator*.



Gambar 3.20 Rancangan Halaman Utama Administrasi
(Sumber: Data Penelitian, 2017)

4. Rancangan Halaman Daftar Pengguna (Tamu)

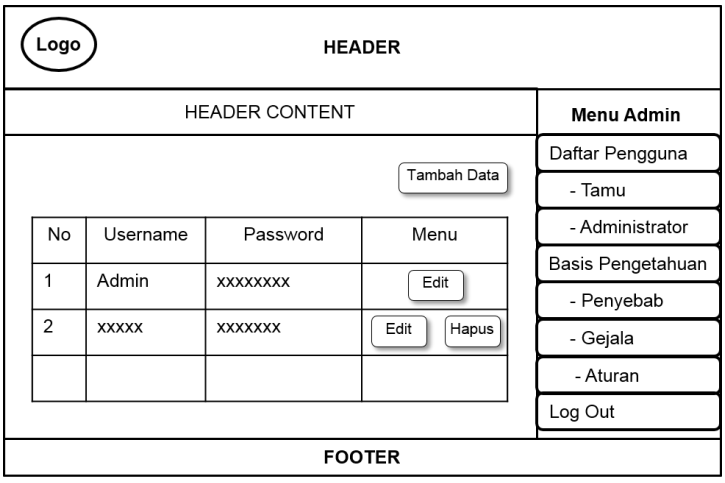
Halaman Daftar Pengguna (Tamu) menampilkan daftar tabel pengunjung yang sudah pernah melakukan diagnosa di sistem pakar ini. Pada halaman ini terdapat tabel dan tombol hapus yang befungsi untuk menghapus pengguna dari *database*.



Gambar 3.21 Rancangan Halaman Administrasi
(Sumber: Data Penelitian, 2017)

5. Rancangan Halaman Daftar Pengguna (*Administrator*)

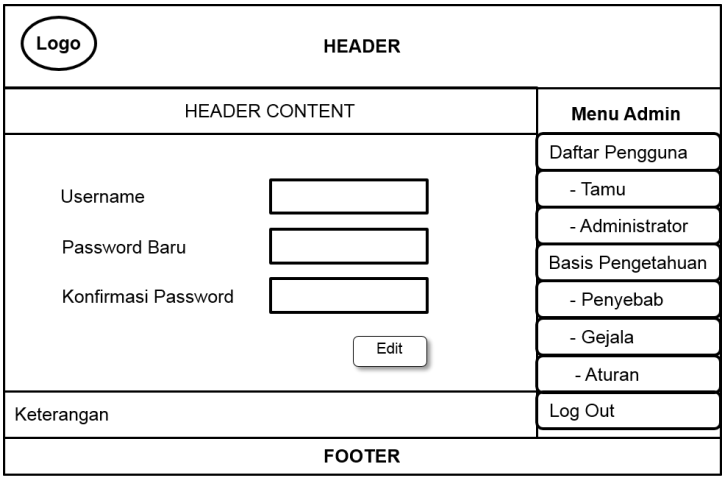
Halaman Daftar Pengguna (*Administrator*) memuat tampilan daftar nama-nama *administrator* yang ada pada program sistem pakar. Pada halaman ini terdapat 3 tombol, yaitu: tombol Tambah Admin berfungsi untuk mengarahkan *user* ke halaman Tambah Administrator, tombol Ubah berfungsi untuk mengubah *password administrator*, dan tombol Hapus berfungsi untuk menghapus daftar *administrator* yang akan dihapus.



Gambar 3.22 Rancangan Halaman Daftar Pengguna (*Administrator*)
(Sumber: Data Penelitian, 2017)

6. Rancangan Halaman *Edit Password Administrator*

Halaman *Edit Password* dapat diakses dengan cara mengklik tombol Ubah Password pada halaman *Administrator* ataupun dengan menekan tombol Ubah pada halaman Daftar Pengguna (*Administrator*). Halaman ini berfungsi untuk mengubah *pasword administrator*. Pengguna (*administrator*) akan dihadapkan dengan *form* yang berisi *username*, *password* lama, *password* baru, dan konformasi *password*. Setelah mengisinya maka tekan tombol Ok, jika *password* benar maka akan ada pesan *password* berhasil di-*update* dan jika salah maka akan tampil pesan kesalahan.

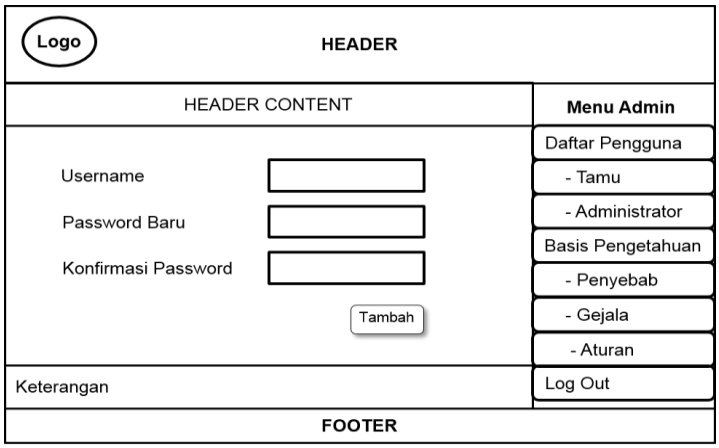


Gambar 3. Rancangan halaman *Edit Password Administrator*
(Sumber: Data Penelitian, 2017)

7. Rancangan Halaman Tambah *Administrator*

Halaman Tambah *Administrator* berfungsi untuk menambah daftar *administrator* pada *database*. Pada halaman ini pengguna (*administrator*) akan dihadapkan dengan *form* yang berisi u*sername*, *password*, dan konformasi

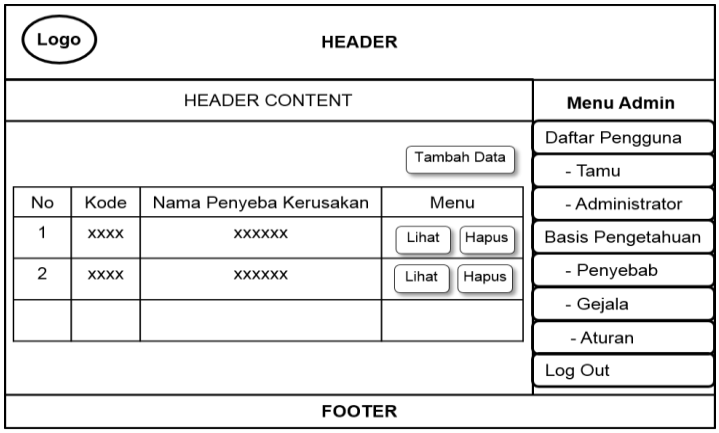
password. Setelah mengisi *form*, maka klik tombol Ok untuk menambahkan data ke *database*.



Gambar 3.23 Rancangan Halaman Tambah *Administrator*
(Sumber: Data Penelitian, 2017)

8. Rancangan Halaman Basis Pengetahuan (Penyebab)

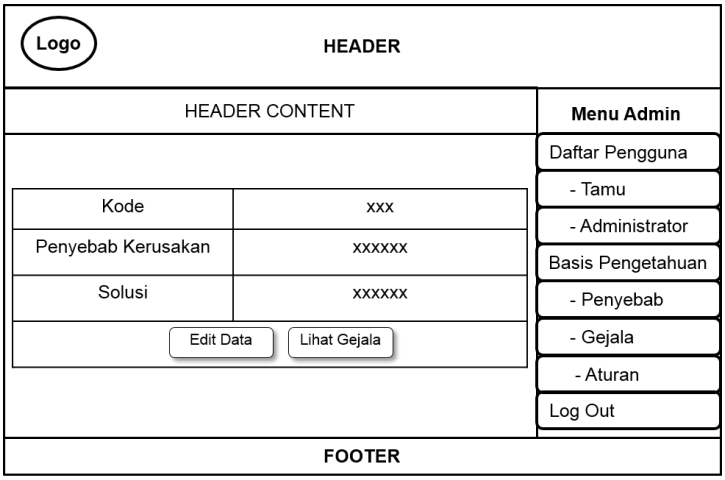
Halaman Basis Pengetahuan (Penyebab) menampilkan kode kerusakan, gejala kerusakan dan disediakan menu untuk melihat dan menghapusnya. Pada halaman ini juga disediakan tombol Tambah Data yang akan menghubungkan ke halaman Tambah Penyebab. Menu Lihat akan menghubungkan ke halaman Lihat Data Penyebab.



Gambar 3.24 Rancangan Halaman Utama Penyebab
(Sumber: Data Penelitian, 2017)

9. Rancangan Halaman Lihat Data Penyebab

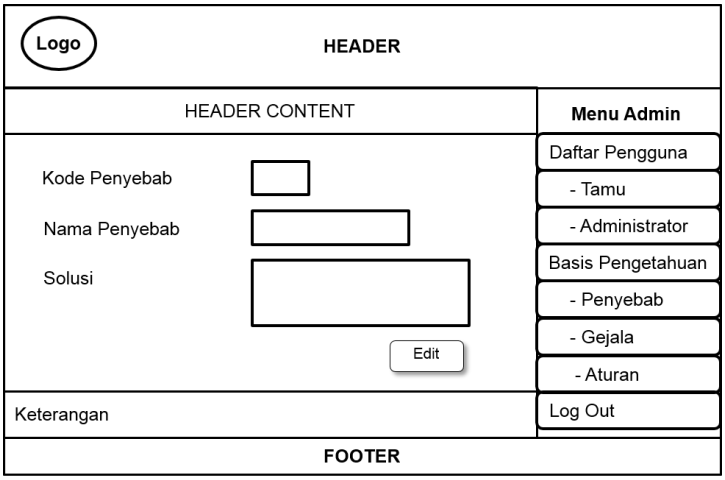
Halaman Lihat Data Penyebab akan menampilkan kode kerusakan, penyebab kerusakan, dan solusi. Jika pengguna (*administrator*) akan mengubah data penyebab yang sudah dipilih maka klik tombol Edit Data. Dengan menekan tombol Edit Data maka *administrator* akan ditampilkan halaman Edit Penyebab.



Gambar 3.25 Rancangan Halaman Lihat Data Penyebab
(Sumber: Data Penelitian, 2017)

10. Rancangan Halaman *Edit* Penyebab

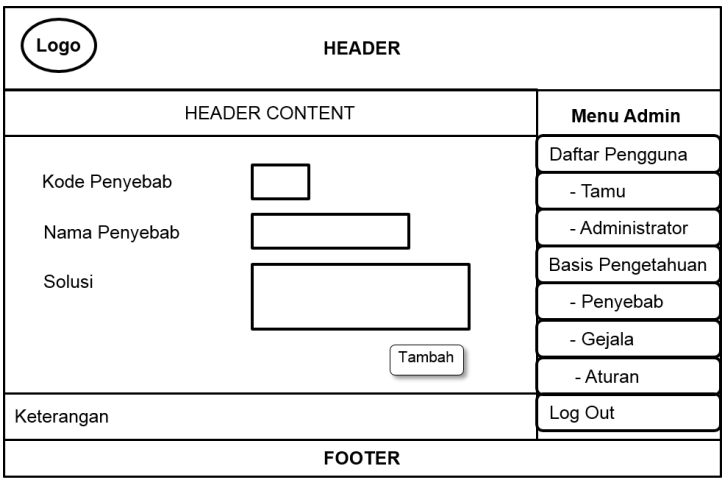
Halaman *Edit* Penyebab menampilkan kode kerusakan, penyebab kerusakan, dan solusi. *Administrator* bisa langsung mengubah data penyebab dengan mengisi *form* edit penyebab. Setelah mengisi/mengubahnya maka klik tombol *Edit* supaya data dapat diperbarui di *database*.



Gambar 3.26 Rancangan Halaman Edit Penyebab
(Sumber: Data Penelitian, 2017)

11. Rancangan Halaman Tambah Penyebab

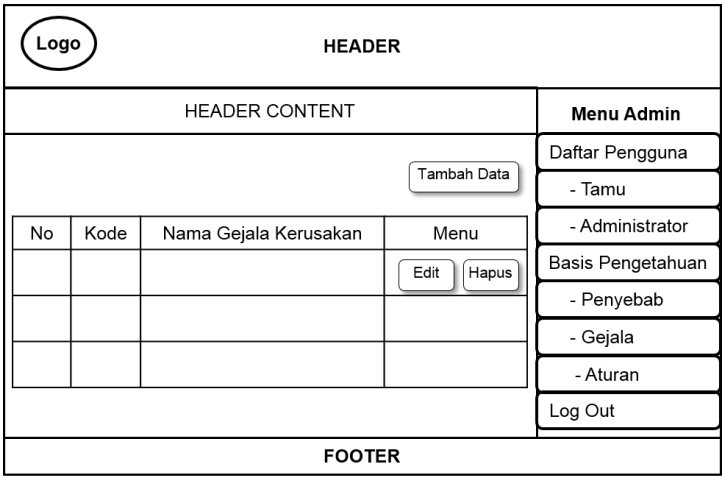
Halaman Tambah Penyebab menampilkan *form* yang berfungsi untuk menambah data penyebab baru di *database*. Halaman ini berisi *form* kode kerusakan, penyebab kerusakan, dan solusi. Tombol Tambah berfungsi untuk memperbarui data ke *database* jika semua *form* sudah diisi.



Gambar 3.27 Rancangan Halaman Tambah Penyebab
(Sumber: Data Penelitian, 2017)

12. Rancangan Halaman Basis Pengetahuan (Gejala)

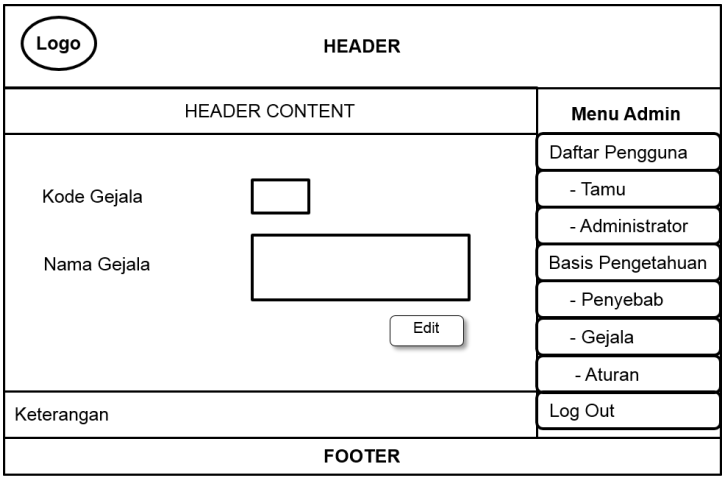
Halaman Basis Pengetahuan (Gejala) menampilkan tabel daftar gejala yang berupa kode gejala, nama gejala kerusakan dan juga menu untuk melihat dan menghapus. Pada halaman ini terdapat tombol Tambah Data yang berfungsi untuk menampilkan halaman Tambah Gejala. Menu Lihat berfungsi untuk menampilkan halaman Lihat Gejala dan Menu Hapus untuk menghapus data gejala yang dipilih untuk dihapus.



Gambar 3.28 Rancangan Halaman Basis Pengetahuan (Gejala)
(Sumber: Data Penelitian, 2017)

13. Rancangan Halaman *Edit* Gejala

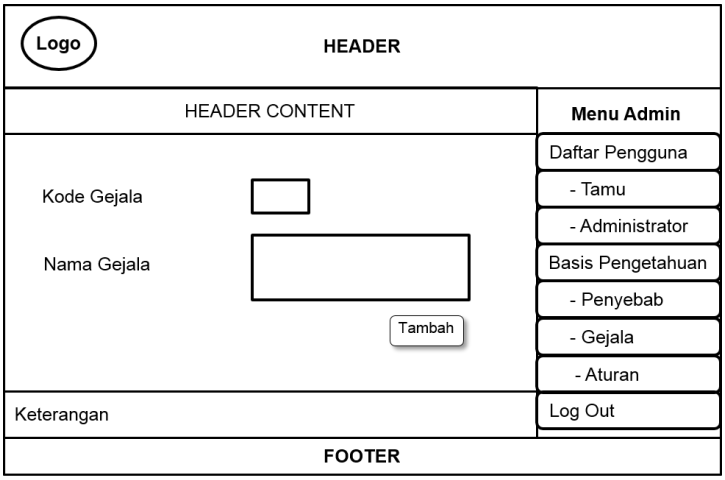
Halaman *Edit* Gejala menampilkan *form* untuk mengubah gejala kerusakan. Berisi tampilan form kode gejala, gejala kerusakan, dan tombol *Edit*. Jika tombol *Edit* ditekan maka data yang ada di *form* akan di-*update* di *database*.



Gambar 3.29 Rancangan Halaman *Edit* Gejala
(Sumber: Data Penelitian, 2017)

14. Rancangan Halaman Tambah Gejala

Halaman Tambah Gejala menampilkan *form* berisi kode gejala dan gejala kerusakan. Tombol Tambah berfungsi untuk menambahkan isi *form* Tambah Gejala kedalam *database*.

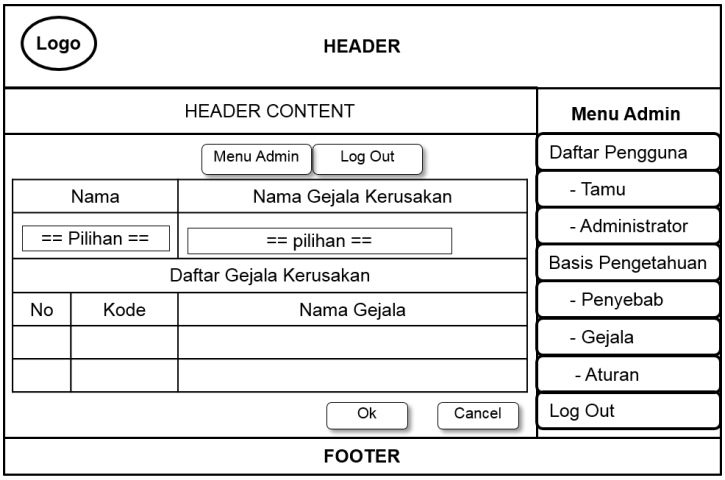


Gambar 3.30 Rancangan Halaman Tambah Gejala
(Sumber: Data Penelitian, 2017)

15. Rancangan Halaman Aturan

Halaman Aturan berisi daftar aturan penyebab kerusakan *IC* yaitu: kode dan nama gejala. Untuk memudahkan *administrator* dalam mencari daftar gejala, maka

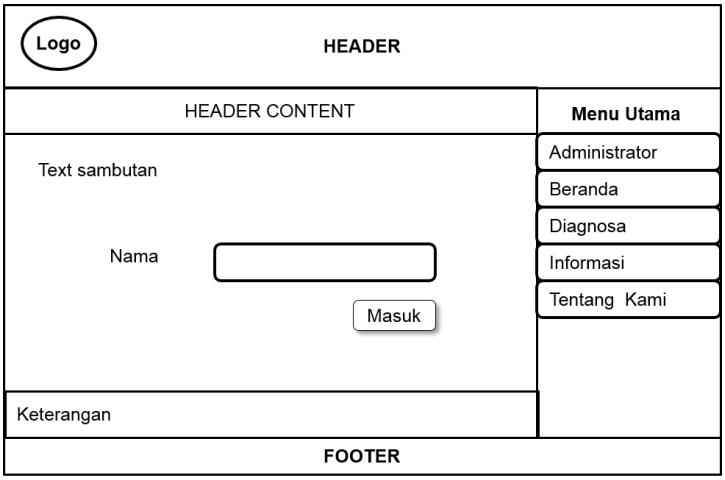
disediakan *filter list* yaitu tombol nama kerusakan *IC* dan penyebab kerusakan. Pada halaman ini juga disediakan tombol untuk mengubah data aturan. Radio button yang ada pada daftar digunakan untuk memilih data aturan yang akan diubah, setelah itu klik tombol Ok mengubah ataupun menambahkan data aturan. Sedangkan tombol *Cancel* untuk membatalkan perintah mengubah data aturan.



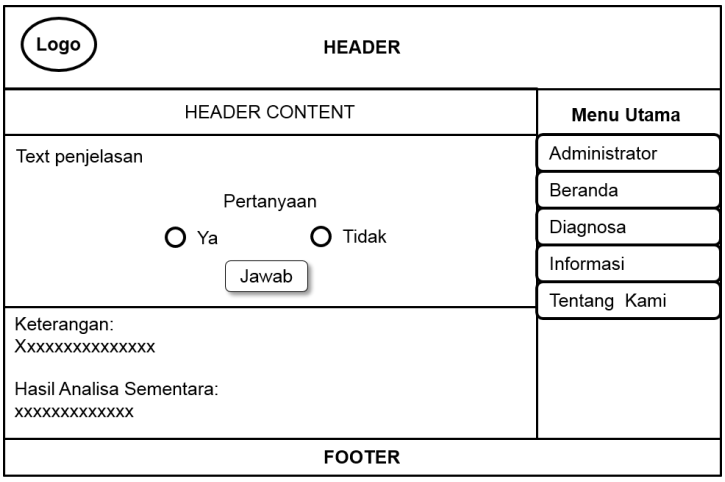
Gambar 3.31 Rancangan Halaman Aturan
(Sumber: Data Penelitian, 2017)

16. Rancangan Halaman Diagnosa

Halaman Diagnosa akan menampilkan beberapa pertanyaan kepada *user*. *User* akan memilih jawaban dengan mengklik *radiobuttom* pilihan “Ya” atau “Tidak” sesuai gejala yang dialami pada mesin, setelah itu baru klik tombol JAWAB. Pertanyaan akan berulang lagi sampai akhir gejala.



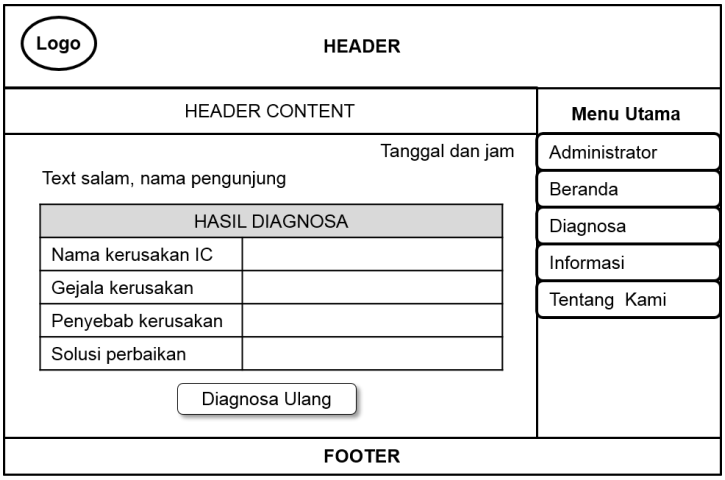
Gambar 3.32 Halaman Daftar Tamu
(Sumber: Data Penelitian, 2017)



Gambar 3.33 Rancangan Halaman Diagnosa
(Sumber: Data Penelitian, 2017)

17. Rancangan Halaman Hasil Diagnosa

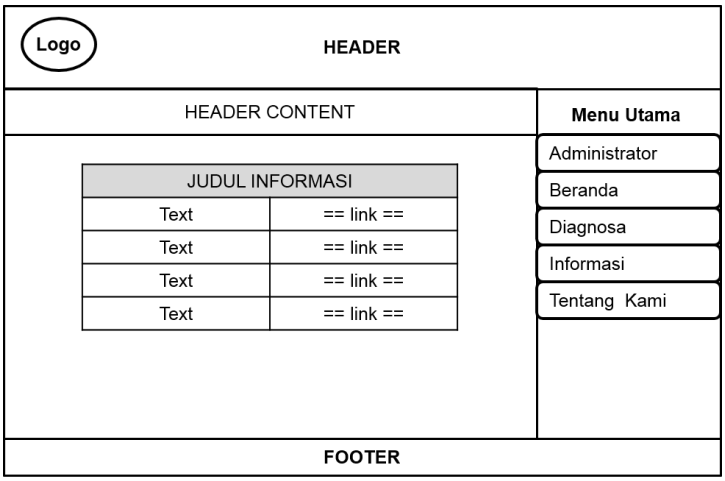
Setelah beberapa pertanyaan dijawab di halaman Diagnosa, maka pada saat jawaban terakhir dijawab akan menampilkan halaman Hasil Diagnosa. Halaman ini akan menampilkan hasil diagnosa berupa: nama *reject IC*, gejala kerusakan, penyebab kerusakan, dan solusi. Jika *user* akan mengulangi diagnosa, maka klik tombol Diagnosa Ulang. Tombol Diagnosa Ulang akan menampilkan halaman Diagnosa kembali.



Gambar 3.34 Rancangan Halaman Hasil Diagnosa
(Sumber: Data Penelitian, 2017)

18. Rancangan Halaman Informasi

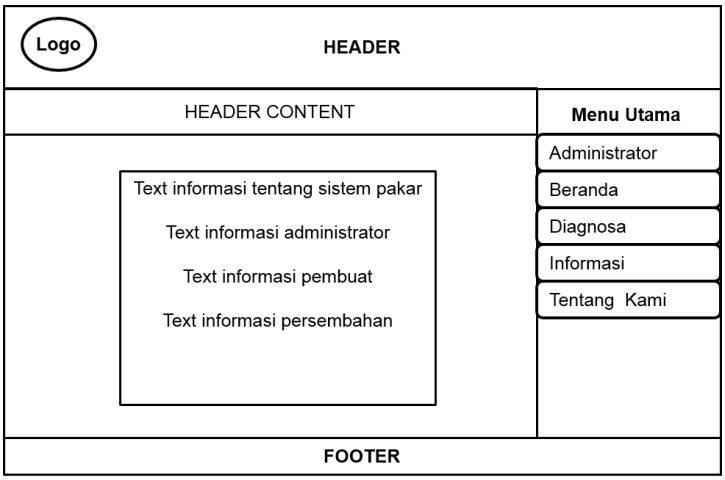
Halaman Informasi akan menampilkan informasi yang berhubungan tentang kerja teknisi *Wire Bond* PT UNISEM Batam yaitu iAS1006 (A*ssembly* S*pesification* tentang proses *Wire Bond*), iAS1007 (A*ssembly* S*pesification* tentang kriteria *reject Wire Bond* dan *Die Attach*), Handbook Machine IConn (penduan tentang mesin KNS IConn). Masing-masing item tersebut disediakan *link* yang tinggal klik untuk membukanya.



Gambar 3.35 Rancangan Halaman Informasi
(Sumber: Data Penelitian, 2017)

19. Rancangan Halaman Tentang Kami

Halaman Tentang Kami memuat informasi mengenai sistem pakar Wire Bond, informasi alamat email administrator, informasi tentang pembuat program, dan informasi tentang persembahan program ini.



Gambar 3.36 Rancangan Halaman Tentang Kami
(Sumber: Data Penelitian, 2017)

3.4 Lokasi dan Jadwal Penelitian

3.4.1 Lokasi Penelitian

Penelitian ini dilakukan di PT UNISEM Batam Jl. S. Parman Kav 201, Batamindo Industrial Park, Mukakuning, Batam (29433). Alasan peneliti memilih perusahaan ini sebagai lokasi penelitian adalah peneliti bekerja di PT UNISEM Batam sebagai *Process Engineer* area *Wire Bond* sehingga peneliti mudah mendapatkan data dan efisiensi waktu dalam melakukan penelitian. Selain itu juga,

peneliti sudah bekerja selama 10 tahun lebih perusahaan tersebut dengan bidang yang sama, jadi peneliti juga bisa sebagai narasumber penelitian.

3.4.2 Jadwal Penelitian

Jadwal penelitian perlu dibuat untuk menggambarkan kapan dan berapa lama waktu yang diperlukan untuk melakukan setiap langkah dalam penelitian. Selain itu, jadwal penelitian juga merupakan tengat (*deadline*) bagi peneliti yang bersangkutan untuk dapat melaksanakan dan menyelesaikan penelitian. Berikut ini adalah tabel jadwal kegiatan yang dilakukan selama penelitian berlangsung:

Tabel 3.9 Tabel Jadwal Penelitian

No	Kegiatan	Jadwal																					
		September 2016				Oktober 2016				November 2016				Desember 2016				Januari 2017				Fabruari 2017	
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2
1	Pengajuan Judul																						
2	Penyusunan Bab I																						
3	Penyusunan Bab II																						
4	Penyusunan Bab III																						
5	Penyusunan Bab IV																						
6	Penyusunan Bab V, Daftar Pustaka, Lampiran																						

(Sumber: Data Penelitian 2017)