CASE BASED REASONING MENGGUNAKAN *ALGORITMA K-NEAREST NEIGHBOR* (*K-NN*) UNTUK DIAGNOSA PENYAKIT LAMBUNG

SKRIPSI

Oleh

Andre Bhaskoro Suprayogi

NIM 112410101045

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2017

CASE BASED REASONING MENGGUNAKAN *ALGORITMA K-NEAREST NEIGHBOR* (*K-NN*) UNTUK DIAGNOSA PENYAKIT LAMBUNG

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat

untuk menyelesaikan Program Studi Sistem Informasi (S1)

dan mencapai gelar Sarjana Komputer

Oleh

Andre Bhaskoro Suprayogi

NIM 112410101045

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2017

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Allah SWT, yang telah memberikan kelancaran dan kemudahan dalam menyelesaikan skripsi ini;
2. Ayahanda Edy Suprayogi dan Ibu Rumini;
3. Kakak Angga Kurniawan Suprayogi dan Adik Adhinda Rachma Amini;
4. Saudara-saudaraku berserta seluruh keluarga besar;
5. Sahabatku bersama doa dan bantuannya;
6. Guru dan teman sejak taman kanak-kanak hingga perguruan tinggi;
7. Almamater Program Studi Sistem Informasi Universitas Jember.

MOTTO

“Hidup ini seperti sepeda. Agar tetap seimbang, kau harus terus bergerak”

~ Albert Einstein

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Andre Bhaskoro Suprayogi

NIM : 112410101045

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "*Case Based Reasoning* Menggunakan *Algoritma K-Nearest Neighbor* (*K-NN*) untuk Diagnosa Penyakit Lambung" adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, Oktober 2017

Yang menyatakan,

Andre Bhaskoro Suprayogi

NIM 112410101045

SKRIPSI

***CASE BASED REASONING* MENGGUNAKAN *ALGORITMA K-NEAREST NEIGHBOR* (*K-NN*) UNTUK DIAGNOSA PENYAKIT LAMBUNG**

oleh :

Andre Bhaskoro Suprayogi

NIM 112410101045

Pembimbing :

Dosen Pembimbing Utama : Prof. Drs. Slamin, M.Comp.Sc., Ph.D

NIP. 196704201992011001

Dosen Pembimbing Pendamping : Diah ayu Retnani W, ST ., M.Eng

NIP. 198603052014042001

PENGESAHAN PEMBIMBING

Skripsi berjudul *Case Based Reasoning* Menggunakan *Algoritma K-Nearest Neighbor* (*K-NN*) untuk Diagnosa Penyakit Lambung, telah diuji dan disahkan pada :

hari, tanggal : Jumat, 27 Oktober 2017

tempat : Program Studi Sistem Informasi Universitas Jember.

Disetujui oleh :

Pembimbing I,	Pembimbing II,
Prof. Drs. Slamin, M.Comp.Sc., Ph.D	Diah Ayu Retnani W, ST ., M.Eng
NIP. 196704201992011001	NIP. 198603052014042001

PENGESAHAN PENGUJI

Skripsi *Case Based Reasoning* Menggunakan *Algoritma K-Nearest Neighbor* (*K-NN*) untuk Diagnosa Penyakit Lambung telah diuji dan disahkan pada :

hari, tanggal : Jumat, 27 Oktober 2017

tempat : Program Studi Sistem Informasi Universitas Jember.

Tim Penguji :

Penguji I,	Penguji II,
Prof. Dr. Saiful Bukhori, ST., M.Kom	Oktalia Juwita, S.Kom., M.MT
NIP 196811131994121001	NIP 198110202014042001

Mengesahkan

Ketua Program Studi,

Prof. Drs. Slamin, M.Comp.Sc., Ph.D

NIP 196704201992011001

RINGKASAN

***Case Based Reasoning* Menggunakan *Algoritma K-Nearest Neighbor* (*K-NN*) untuk Diagnosa Penyakit Lambung;** Andre Bhaskoro Suprayogi; 112410101045; 167 halaman; Program Studi Sistem Informasi Universitas Jember.

Gangguan kesehatan pada lambung memiliki berbagai jenis, antara lain adalah sakit maag (Gastritis), Dispepsia, Kanker Lambung, Tukak Lambung, Gastroparesis, Gastroenteritis dan Gastroesophageal Reflux Disease (GERD). Kanker lambung merupakan salah satu jenis penyakit pada lambung yang sangat berbahaya. Angka prevalensi kanker lambung mencapai peringkat keempat terbanyak dari semua jenis kanker. Kanker lambung juga menempati nomor urut kedua terbanyak penyebab kematian akibat kanker di dunia. Sekitar 880.000 orang terdiagnosa sebagai kanker lambung dan 700.000 orang diantaranya meninggal dunia akibat penyakit kanker lambung.

Keluhan maupun gejala pada penyakit lambung bermacam- macam dan tidak menutup kemungkinan sulit untuk mengetahui dan menentukan jenis penyakit lambung yang diderita. Seseorang yang memiliki gejala-gejala tertentu memungkinkan mengidap penyakit pada lambung yang paling berbahaya yaitu kanker lambung. Dokter spesialis sebagai seorang pakar dapat dikolaborasikan dengan perangkat lunak untuk memudahkan penderita dalam mendiagnosa penyakit lambung. Hasil diagnosa yang cepat dan praktis pada perangkat lunak dapat diwujudkan dengan menerapkan sistem berbasis pengetahuan.

Salah satu elemen utama sistem berbasis pengetahuan yaitu kemampuan penalaran berbasis kasus (Case Base Reasoning). Proses *similarity* pada tahap *retrive* CBR dapat di terapkan *Algoritma k-Nearest Neighbor* (*k-NN*). Penerapan *Algoritma K-NN* pada sistem berbasis pengetahuan CBR dapat memberikan hasil diagnosa yang cepat dan praktis serta memberikan saran yang tepat kepada penderita untuk mendapatkan informasi alternatif pengobatan yang sesuai dengan jenis penyakitnya.

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "*Case Based Reasoning* Menggunakan *Algoritma K-Nearest Neighbor* (*K-NN*) untuk Diagnosa Penyakit Lambung". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember. Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Prof. Drs. Slamin, M.Comp.Sc., Ph.D., selaku Dosen Pembimbing Utama dan Ketua Program Studi Sistem Informasi Universitas Jember;
2. Diah Ayu Retnani W, ST ., M.Eng selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi;
3. Drs. Antonius Cahya P, M.App., Sc., Ph.D sebagai dosen pembimbing akademik, yang telah mendampingi penulis sebagai mahasiswa;
4. Seluruh Bapak dan Ibu dosen beserta staf karyawan di Program Studi Sistem Informasi Universitas Jember;
5. Bapak Edy Suprayogi dan Ibu Rumini tercinta yang selalu mendukung dan mendoakan;
6. Kakak Angga Kurniawan Suprayogi dan Adik Adhinda Rachma Amini serta keluarga;
7. Wardhatul Jannah serta keluarga;
8. Dokter Devi Chintya Kumalasari dari RS. Citra Husada Jember yang telah meluangkan waktu untuk membantu dalam melancarkan skripsi ini;
9. Keluarga besar NEOTION angkatan 2011 yang telah menjadi keluarga selama menempuh pendidikan S1;
10. Kelompok KKN PPM Dusun Calok, Kecamatan Arjasa Kabupaten Jember Tahun 2015;

11. Keluarga Basar SISMADAPALA XXVIII yang telah memberi semangat dan doa;
12. Sahabat seperjuangan yang selalu menemani dan memberikan semangat selama proses penulisan skripsi, Dhani, Egit, Riska, Chadek, Arie, Fidia, Indra, Dery, Iqbal Dkk;
13. Teman-Teman Program Studi Sistem Informasi di semua angkatan:
14. Semua pihak yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, oleh sebab itu penulis mengharapkan adanya masukan yang bersifat membangun dari semua pihak. Penulis berharap skripsi ini dapat bermanfaat bagi semua pihak.

Jember, 8 November 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1. PENDAHULUAN

Bab ini merupakan bab awal dari laporan tugas akhir. Pada bab ini akan dibahas tentang latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah, dan sistematika penulisan.

1.1 Latar Belakang

Lambung memiliki peranan besar dalam angggota tubuh yang berfungsi untuk mencerna makanan serta mengubah makanan menjadi sumber energi yang dibutuhkan oleh tubuh. Lambung merupakan salah satu organ yang cukup rentan terhadap cidera atau terluka. Salah satu faktor yang dapat memicu gangguan kesehatan pada lambung diantaranya asam lambung meningkat. Peningkatan asam lambung dalam tubuh bisa mengakibatkan gangguan kesehatan lambung.

Gangguan kesehatan pada lambung memiliki berbagai jenis, antara lain adalah sakit maag (Gastritis), Dispepsia, Kanker Lambung, Tukak Lambung, Gastroparesis, Gastroenteritis dan Gastroesophageal Reflux Disease (GERD). Kanker lambung merupakan salah satu jenis penyakit pada lambung yang sangat berbahaya. Angka prevalensi kanker lambung mencapai peringkat keempat terbanyak dari semua jenis kanker. Kanker lambung juga menempati nomor urut kedua terbanyak penyebab kematian akibat kanker di dunia. Sekitar 880.000 orang terdiagnosa sebagai kanker lambung dan 700.000 orang diantaranya meninggal dunia akibat penyakit kanker lambung. Tingkat kejadianya bervariasi dan berhubungan dengan letak geografi. Enam puluh persen kanker lambung berada pada Negara yang sedang berkembang (Ilawati Pristiani, 2013).

Keluhan maupun gejala pada penyakit lambung bermacam- macam dan tidak menutup kemungkinan sulit untuk mengetahui dan menentukan jenis penyakit lambung yang diderita. Peran dokter diperlukan untuk mendiagnosa gejala-gejala maupun keluhan dari penderita agar penderita penyakit lambung segera mengetahui jenis penyakit apa yang diderita. Penderita penyakit lambung pada umumnya membutuhkan

informasi yang cepat dan praktis dari seorang dokter, namun terkendala masalah biaya pemeriksaan. Selain biaya, umumnya penderita meremehkan penyakitnya, jika penyakitnya dirasa parah baru memeriksa ke dokter.

Dokter sebagai seorang pakar dapat dikolaborasikan dengan teknologi informasi (perangkat lunak) untuk memudahkan penderita dalam mendiagnosa sendiri penyakit lambung. Hasil diagnosa yang cepat dan praktis menggunakan teknologi informasi (perangkat lunak) dapat mengurangi angka kematian penderita penyakit lambung akibat terlambat diketahui jenis penyakitnya sehingga terlambat dalam melakukan penanganan. Hasil diagnosa yang cepat dan praktis pada perangkat lunak dapat diwujudkan dengan menerapkan sistem berbasis pengetahuan. Sistem berbasis pengetahuan mampu mengakuisisi pengetahuan seorang pakar pada perangkat lunak. Salah satu elemen utama sistem berbasis pengetahuan yaitu kemampuan penalaran berbasis kasus (Case Base Reasoning). Kemampuan penalaran berbasis kasus (Case Base Reasoning) CBR pada sistem berbasis pengetahuan mampu mengidentifikasi case berdasarkan kemiripan dengan sangat baik (Mukhlason, Prakoso, & Anggraeni 2016).

Proses similarity pada tahap retrive CBR dapat di terapkan Algoritma k-Nearest Neighbor (k-NN). Sistem Case Based Reasoning dan Algoritma k-NN dapat melakukan perhitungan similarity antara kasus baru dengan kasus training dengan keakuratan validasi 100% (Putri, Andreswari, & Efendi 2016). Penerapan Algoritma K-NN pada sistem berbasis pengetahuan CBR dapat memberikan hasil diagnosa yang cepat dan praktis serta memberikan saran yang tepat kepada penderita untuk mendapatkan informasi alternatif pengobatan yang sesuai dengan jenis penyakitnya.

1.2 Rumusan Masalah

Berdasarkan uraian diatas dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana megembangkan sistem berbasis pengetahuan *Case Based Reasoning* (CBR) untuk diagnosa penyakit lambung.

2. Bagaimana menerapkan *Algoritma K-Nearest Neighbor* (*K-NN*) pada sistem berbasis pengetahuan CBR untuk diagnosa penyakit lambung.

1.3 Tujuan dan Manfaat

Berikut merupakan tujuan yang ingin dicapai dan manfaat yang ingin diperoleh dalam penelitian ini.

1.3.1 Tujuan

Tujuan dari penelitian adalah untuk menganalisa:

1. Mengembangkan sistem berbasis pengetahuan CBR untuk diagnosa penyakit lambung.
2. Menerapkan *Algoritma K-Nearest Neighbor* (*K-NN*) pada sistem berbasis pengetahuan CBR untuk diagnosa penyakit lambung.

1.3.2 Manfaat

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1.3.2.1 Bagi Akademis

Penelitian ini diharapkan dapat memberi masukan informasi kepada pembaca pada umumnya, khususnya kepada Program Studi Sistem Informasi Universitas Jember terkait dengan pengembangan perangkat lunak berbasis pengetahuan (Case Based Reasoning) yang di implementasikan di dunia kesehatan khususnya penyakit lambung bahwa sistem berbasis pengetahuan (Case Based Reasoning) dapat mendiagnosa sebuah *case* dengan akurat.

1.3.2.2 Bagi Peneliti

Penelitian ini merupakan suatu pengalaman untuk pembuktian teori atau materi yang didapat dari perkuliahan dengan implementasi nyata.

1.3.2.3 Bagi Pihak Lain

Penelitian ini dapat dijadikan bahan pertimbangan dalam proses diagnosa penyakit lambung dan dapat dijadikan bahan referensi bagi peneliti lain yang ingin mengembangkan hasil penelitian ini di kemudian hari.

1.4 Batasan Masalah

Agar tidak terjadi penyimpangan dalam proses penelitian dan pengembangan sistem, maka ditetapkan beberapa batasan permasalahan. Adapun batasan masalah dalam penelitian ini adalah :

1. Sistem khusus untuk diagnosa penyakit lambung.
2. Proses diagnosa penyakit lambung menggunakan sistem berbasis pengetahuan *case based reasoning* (CBR) dengan penerapan *Algoritma k-Nearest Neighbor* (*k-NN*) yang disesuaikan dengan *objek* penelitian .
3. Jenis penyakit yang didiagnosa berjumlah 7 jenis penyakit lambung, yaitu maag *(Gastritis)*, *dispepsia*, kanker lambung, tumor lambung, *gastroparesis*, *gastroenteritis* dan *gastroesophageal reflux disease* (GERD).
4. Hasil diagnosa sistem berupa kemungkinan jenis penyakit lambung beserta informasi alternatif pengobatanya.

1.5 Sistematika Penulisan

Adapun sistematika penulisan skripsi ini adalah sebagai berikut:

1. Pendahuluan

   Bab ini memuat uraian tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika penulisan skripsi yang masing-masing tertuang secara eksplisit dalam sub bab tersendiri.

2. Tinjauan Pustaka

   Bab ini memaparkan tinjauan terhadap hasil-hasil penelitian terdahulu berkaitan dengan rumusan masalah penelitian, landasan materi dan konsep penyakit lambung, *Case Based Reasoning* (CBR), dan *Algoritma K-Nearest Neighbor* (*K-NN*).

3. Metodologi Penelitian

   Bab ini menguraikan tentang tempat dan waktu penelitian, jenis penelitian, alur penelitian, teknik pengumpulan data, pengembangan sistem, dan Gambaran umum sistem.

4. Pengembangan Sistem

   Bab ini berisi uraian tentang langkah-langkah yang ditempuh dalam proses menganalisis dan merancang sistem yang hendak dibangun meliputi analisis kebutuhan sistem, deskripsi umum sistem, desain sistem, pengkodean sistem, dan pengujian sistem.

5. Hasil dan Pembahasan

   Bab ini memaparkan secara rinci pemecahan masalah melalui analisis yang disajikan dalam bentuk deskripsi dibantu dengan ilustrasi berupa tabel dan Gambar untuk memperjelas hasil penelitian.

6. Penutup

   Bab ini terdiri atas kesimpulan atas penelitian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Dalam bab ini dijelaskan teori-teori serta pustaka yang digunakan untuk penelitian. Teori-teori ini diambil dari berbagai buku literatur, jurnal dan internet. Teori-teori dan pustaka yang digunakan dan dibahas dalam penelitian ini, yaitu penyakit lambung, sistem pakar, *Case Based Reasoning (CBR)*, *Algoritma K-Nearest Neighbor* (*K-NN*).

2.1 Penyakit Lambung

Lambung merupakan salah satu organ pada sistem pencernaan manusia yang memiliki fungsi untuk mencerna makanan dan menyerap sari-sari makanan sebagai sumber nutrisi. Lambung memiliki beberapa enzim, diantaranya enzim renin, pepsin, dan asam klorida. Organ lambung bekerja dengan cara melumatkan makanan hingga benar-benar hancur seperti bubur. Zat asam pada lambung dapat menyebabkan gangguan pada lambung jika Zat tersebut dikeluarkan secara berlebihan. Menurut (Ariani & Findawati, 2015) berikut ini macam-macam gangguan (penyakit) pada lambung beserta gejalanya yang sudah dirangkum pada Tabel 2.1 dan Tabel 2.2.

Tabel 2.1 Jenis penyakit lambung

Kode	Nama Penyakit
P1	Grastitis (maag)
P2	Dispepsia
P3	Kanker lambung
P4	GERD
P5	Gastroenteritis
P6	Gastroparesis
P7	Tukak lambung

Tabel 2.1 berisi daftar penyakit lambung yang disertai dengan kode penyakit seperti contohnya kode P1 dengan nama penyakit Gastritis (maag).

Tabel 2.2 Gejala pada jenis penyakit lambung

Kode	Gejala	P1	P2	P3	P4	P5	P6	P7
G1	Rasa nyeri dan rasa tidak nyaman pada perut	√				√	√	
G2	Rasa penuh didaerah lambung / daerah perut	√						
G3	Perut kembung	√	√	√	√	√	√	
G4	Rasa mual	√	√			√	√	√
G5	Muntah	√		√		√	√	√
G6	Rasa nyeri pada uluhati	√	√					√
G7	Selera makan berkurang	√	√	√			√	√
G8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas		√	√				
G9	Sering sendawa		√					
G10	Sering merdengar suara usus yang keras (borborigmi)		√					
G11	Sembelit		√					
G12	Diare		√			√		
G13	Kehilangan berat badan secara mendadak			√			√	
G14	Demam atau meriang			√	√	√		
G15	Kejang perut			√	√	√	√	
G16	Keluarnya BAB warna hitam pekat			√				
G17	Pendarahan (BAB darah atau Muntah darah)			√				
G18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher				√			
G19	Rasa ada makanan/minuman balik ke mulut				√			
G20	Mulut terasa asam dan pahit				√			
G21	Batuk menahun				√			
G22	Serak dan tenggorakan sakit				√			
G23	Lesu (mudah lelah)					√		
G24	Perasaan kenyang berlebihan walaupunhanya makan sedikit						√	
G25	Dehidrasi berat					√		√
G26	Rasa nyeri pada tukak deodenum							√
G27	Sakit pada tukak lambung							√

Tabel 2.2 berisi daftar 27 nama gejala beserta relasinya terhadap 7 penyakit lambung dan diberikan kode pada setiap gejalanya. Contoh pada kolom P1 pada Tabel 2.2, kode P1 bisa dilihat nama penyakitnya di Tabel 2.1, yaitu Gastritis (maag). Kolom P1 memiliki 7 gejala yaitu G1, G2, G3, G4, G5, G6, G7, bisa dilihat pada baris gejala yang dicentang.

Penelitian tentang diagnosa penyakit lambung pernah dilakukan oleh (Ritonga, 2013) dengan judul *"Sistem Pakar Mendiagnosa Penyakit Lambung Menggunakan Teori Certainty Factor"*. Jenis penyakit lambung yang digunakan pada penelitian ini berjumlah 3 yaitu Maag (Gastritis), Dispepsia dan Gastroesophageal Reflux Disease (GERD). Dalam penelitianya, penulis menyarankan teori yang digunakan tidak harus menggunakan teori *certainty factor*, namun dapat dikembangkan dengan teori-teori yang lainnya.

2.2 Sistem Pakar

Sistem pakar adalah sebuah perangkat lunak yang memiliki pengetahuan spesifik yang meggunakan penalaran seperti halnya pakar dalam memecahkan suatu masalah. Sistem pakar adalah salah satu jalan untuk mendapatkan pemecahan masalah secara lebih cepat dan mudah. Sistem pakar membantu seseorang yang memiliki sedikit pengetahuan dapat menyelesaikan masalah yang cukup rumit. Sistem pakar juga dapat membantu aktifitas pakar yang fungsinya sebagai asisten yang berpengalaman yang memiliki pengetahuan yang dibutuhkan (Arifin, 2011).

Menurut (Bukhori, 2012), sistem pakar disusun berdasarkan tiga modul utama, yaitu:

1. Modul yang berfungsi menerima pengetahuan. Pada modul ini sistem pakar menerima pengetahuan dari pakar atau ahli (*human expert*).
2. Modul Konsultasi. Pada modul ini, sistem memberikan jawaban kepada user mengenai permasalahan yang dimasukan kedalam sistem.
3. Modul penjelasan. Pada modul ini sistem pakar memberikan penjelasan bagaimana sebuah solusi diperoleh.

Untuk memenuhi modul-modul tersebut sistem pakar memerlukan komponen-komponen yang berfungsi untuk menyelesaikan permasalahan pada setiap modulnya. Secara umum komponen penyusun arsitektur sistem pakar adalah tatap muka (*user interface*), mesin inferensi (*inference engine*), basis pengetahuan (*knowledge base*), dan basisdata atau memory kerja (*working memory*).

Keterangan:

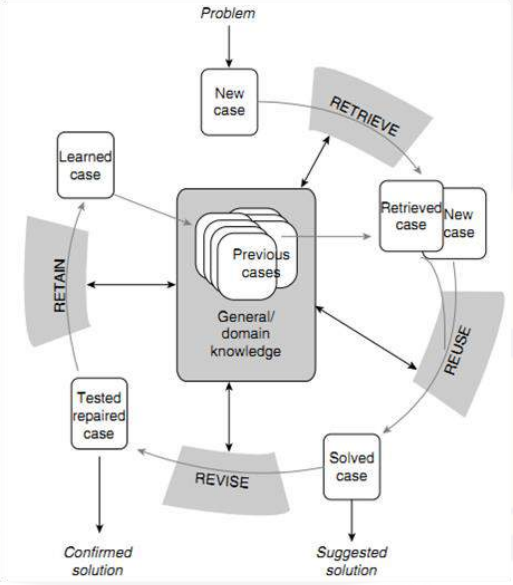
1. *User interface* adalah media yang digunakan untuk berkomunikasi antara sistem pakar dengan pengguna. Menerima informasi dari pengguna dan mengkonversikan kedalam bahasa yang dapat diolah oleh sistem, serta menyajikan hasil yang telah diolah sistem kepada pengguna adalah fungsi dari *user interface.*
2. *Inference enginer* adalah komponen yang mengandung prosedur penalaran yang digunakan oleh pakar dalam menyelesaikan permasalahan. *Inference enginer* berisi metodologi penalaran tentang informasi yang ada pada *knowledge base* dan dalam *workplace* untuk memformulasikan solusi.
3. *Knowledge base* adalah komponen yang mempresentasikan pengetahuan dari seseorang atau beberapa orang ahli / pakar yang dibutuhkan untuk memahami, menformulasikan dan menyelsaikan permaslahan.
4. *Workplace* adalah sekumpulan memori (*working memory*) yang digunakan untuk perubahan data termasuk keputusan sementara.
5. Perbaikan pengetahuan, seorang pakar memiliki kemampuan untuk menganailisis dan meningkatkan kinerjanya serta kemampuan belajar dari kinerjanya. Kemampuan tersebut penting dalam pembelajaran terkomputerisasi, sehingga program mampu menganalisis penyebab keberhasilan dan kegagalan dalam mengambil keputusan, serta mengeveluasi apakah pengetahuan yang ada masih sesuai digunakan untuk proses pengambilan keputusan di masa sekarang dan masa mendatang.

Penelitian tentang sistem pakar pernah dilakukan oleh (Akmal & Winiarti, 2014). Pada penelitian yang berjudul *"Sistem Pakar Untuk Mendiagnosa Penyakit Lambung dengan Implementasi Teori CBR (Case-Based Reasoning) Berbasis WEB"*, penulis menyimpulkan perangkat lunak sistem pakar yang dihasilkan mampu mendiagnosa penyakit lambung dengan perhitungan nilai kepastian menggunakan Teori *Certainty Factor* dan Teori *Case Based Reasoning* (CBR).

2.3 *Case Based Reasoning* (CBR)

Case Based Reasoning (CBR) merupakan salah satu metode pemecahan masalah yang diadopsi dari solusi masalah-masalah sebelumnya yang memiliki kemiripan dengan masalah baru yang dihadapi untuk mendapatkan solusinya (Nurdiansyah, 2010). CBR memiliki kemampuan untuk diagnosa berbasis kasus data yang memberikan informasi secara otomatis berdasarkan pengetahuan terdahulu yang dapat direvisi untuk menyesuaikan dengan permasalahan baru, sehingga pengetahuan CBR akan terus berkembang. Pemecahan masalah baru pada CBR dilakukan dengan cara mencari permsalahan sejenis pada masa lampau dan memberikan solusi berdasarkan permasalahan yang paling mirip yang ada pada *case memory*, dan permasalahan yang dapat digunakan untuk memecahkan masalah disimpan pada *case memory* untuk memecahkan permasalahan di masa datang (Anggraeni, Mukhlason, & Prakoso, 2012).

Menurut (Putri, Andreswari & Efendi, 2016) terdapat empat proses yang terjadi pada metode CBR dalam menyelesaikan masalah seperti Gambar 2.1.



Gambar 2.1 Proses *Case Based Reasoning*

1. *Retrive*

   Mendapatkan/memperoleh kembali kasus yang paling menyerupai/relevan (similar) dengan kasus yang baru. Tahap retrieval ini dimulai dengan menggambarkan/

menguraikan sebagian masalah, dan diakhiri jika ditemukannya kecocokan terhadap masalah sebelumnya yang tingkat kecocokannya paling tinggi.

2. *Reuse*

   Pada tahapan ini sistem akan menggunakan informasi permasalahan sebelumnya yang memiliki kesamaan permasalahan untuk meyelesaikan permasalahan yang baru dan mengunakan kembali informasi dan pengetahuan sebelumya untuk menyelesaikan masalah yang baru. Pada tahapan *reuse* yang akan dilakukan adalah menyalin, menyeleksi dan melengkapi informasi yang akan digunakan selanjutnya. Kriterianya adalah kasus sebelumya yang memiliki kemiripan paling tinggi dengan kasus baru yang nantinya akan disarankan sebagai solusinya. Tidak semua permasalahan yang sebelumnya memiliki kemiripan masalah, oleh karena itu diperlukan adanya nilai kemiripan untuk mempermudah dalam menentukan data yang memiliki kemiripan tertinggi. Kriteria tingkat kemiripan dapat dilihat pada Tabel 2.3.

Tabel 2.3 Kriteria kemiripan / similarity

Nilai desimal kemiripan	Kriteria kemiripan
0 – 0,39	Rendah
0,4 – 0,79	Sedang
0,8 – 1	Tinggi

Sumber: (Kusuma & Chairani, 2014)

3. *Revise*

   Pada proses ini informasi mengenai solusi yang diberikan akan dikalkulasi, dievalusasi, dan diperbaiki kembali untuk meminimalisir kesalahan-kesalahan yang terjadi pada permasalahan baru.

4. *Retain*

   Pada proses ini solusi akan diindekskan, diintegrasikan, dan mengekstrak solusi yang baru dan selanjutnya disimpan dalam *knowledge base* untuk menyelesaikan permaslahan selanjutnya. Proses ini tidak *mereplace knowledge* lama dengan yang

baru, namun menambahkannya untuk diseleksi kembali ketika terjadi permasalahan baru dan yang pasti adalah persamaan permasalahan yang diseleksi.

Penelitian tentang *case based reasoning* telah di lakukan oleh (Nurdiansyah, 2010). Dalam penelitiannya yang berjudul *"Case Based Reasoning Untuk Pendukung Diagnosa Gangguan Pada Anak Autis"* peneliti menyimpulkan bahhwa sistem yang diteliti berhasil membantu Psikolog maupun orang tua dalam mendiagnosa awal gejala penyakit autis yang diderita oleh anak.

2.4 *Algoritma k-Nearest Neighbor* (*k-NN*)

Algoritma k-Nearest Neighbor (*k-NN*) adalah sebuah metode yang digunankan untuk klasifikasi objek berdasarkan data pembelajaran yang berjarak paling dekat dengan objek tersebut. *k-NN* termasuk kelompok *instance-based learning*. *Algoritma k-NN* mencari kelompok k objek dalam data training yang paling dekat atau mirip dengan objek pada data baru atau data testing (Putri, Andreswari & Efendi, 2016). Untuk menghitung kemiripan kasus, digunakan Persamaan 1.

$$Similarity\ (p,q) = \frac{(S_1 \times W_1) + (S_2 \times W_2) + \cdots + (S_n \times W_n)}{W_1 + W_2 + \cdots + W_n} \quad \text{..... (Persamaan 1)}$$

Keterangan:

P : Kasus baru (Diagnosa baru)

q : Kasus yang ada dalam penyimpanan (*case*)

w : weight (bobot yang diberikan pada atribut ke-i)

s : *Similarity* (nilai kemiripan)

Menurut Penelitian (Putri, Andreswari, & Efendi 2016) yang berjudul *"Implementasi Metode CBR (Case Based Reasoning) dalam Pemilihan Pestisida Terhadap Hama Padi Sawah Menggunakan Algoritma k-Nearest Neighbor (k-NN) (Studi Kasus Kabupaten Seluma)"* menyimpulkan bahwa penerapan *Algoritma k-NN* pada sistem *case based reasoning* dapat melakukan perhitungan similarity antara kasus baru dengan kasus training dengan keakuratan validasi 100%. Metode CBR yang

digunakan pada penelitian (Putri, Andreswari & Efendi 2016) menggunakan *objek* pestisida yang memiliki sub kriteria dalam proses perhitungan *k-NN*, sedangkan pada penelitian ini proses perhitungan *k-NN* tidak memerlukan sub kriteria karena objek penyakit lambung hanya memiliki kriteria gejala tanpa sub kriteria gejala.

Pada penelitian ini dibangun sebuah sistem berbasis pengetahuan (CBR) karena CBR mampu megidentifikasi suatu *case* dengan tepat, selain itu juga menerapkan *algoritma k-NN* untuk perhitungan nilai kemiripan antara suatu *case* dengan *case memory*. Proses penentuan nilai kemiripan (similarias) ini terdapat pada tahap *retrive* pada sistem CBR. Pengetahuan terkait jenis penyakit dan gejala penyakit lambung pada penelitian ini diambil dari penelitian (Ariani & Findawati, 2015) yang memberikan informasi data 7 jenis penyakit lambung dan dat 27 gejala yang berhubungan dengan jenis penyakitnya. Tambahan informasi mengenai jenis penyakit lambung juga didapatkan dari studi literatur.

BAB 3. METODOLOGI PENELITIAN

Pada bab metodologi penelitian dipaparkan beberapa hal yang meliputi jenis penelitian, alur penelitian, teknik pengumpulan data, dan model pengembangan sistem.

3.1 Tempat Dan Waktu Penelitian

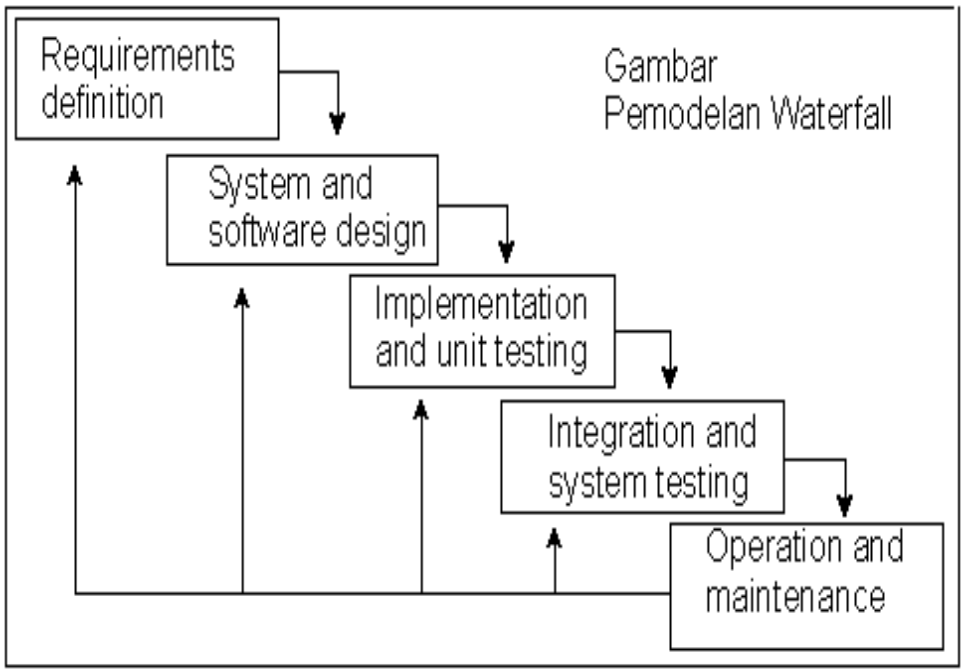
Penelitian dilakukan di Rumah Sakit Citra Husada Jember. Waktu yang dibutuhkan untuk penelitian selama dua bulan yaitu pada bulan Juni 2017 hingga September 2017.

3.2 Jenis Penelitian

Penelitian ini menggunakan jenis penelitian pengembangan sistem. Penelitian jenis pengembangan sistem dipilih karena yang dilakukan pada penelitian ini adalah pengembangan sistem yaitu sistem berbasis pengetahuan dengan judul "*Case Based Reasoning* Menggunakan *Algoritma K-Nearest Neighbor* (*K-NN*) untuk Diagnosa Penyakit Lambung".

3.3 Tahap Pengembangan Sistem

Penelitian ini akan dilakukan dengan menerapkan Teori *Software Defelopment Life Cycle* (SDLC). SDLC yang digunakan dalam pembangunan perangkat lunak dalam penelitian ini adalah model *waterfall*. Pada model *waterfall* ini perangkat lunak dibangun dari identifikasi semua kebutuhan seperti data jenis penyakit lambung dan data gejala penyakit lambung hingga perangkat lunak tersebut di uji. Model ini dimulai dengan tahap analisis, desain, kode, test dan pemeliharaan sistem (Pressman, 2009). Model ini dipilih karena pembangunan sistem ini masih dalam skala kecil, sehingga dokumentasi pengembangan dapat terorganisir. Tahapan Model *Waterfall* dijelaskan pada Gambar 3.1.



Gambar 3.1 Model *Waterfall*

Sumber: (Pressman, 2009).

3.3.1 Tahapan Analisis Kebutuhan (*requirement analysis*)

Tahap *requirement analysis* adalah tahap menganalisa apa yang dibutuhkan oleh sistem. Data kebutuhan dibagi menjadi kebutuhan fungsional dan non-fungsional. Pada sistem ini dibutuhkan juga pengumpulan data gejala dan data penyakit serta alternatif pengobatan yang akan digunakan untuk diagnosa penyakit lambung. Data yang sudah terkumpul akan menentukan bagaimana fitur yang akan dibangun pada sistem.

3.3.1.1 Teknik Pengumpulan Data

Pada tahap ini menganalisa kebutuhan yang dilakukan dengan mengumpulkan data yang diperlukan untuk proses diagnosa penyakit lambung. Teknik pengumpulan data dilakukan 2 jenis data yaitu data primer dan data sekunder.

Data primer digunakan sebagai data yang diperoleh secara langsung dari subjek atau objek penelitian tujuanya mendapatkan informasi penyakit lambung dan data pembobotan gejala langsung dari pakar dengan cara wawancara dan kuesioner. Wawancara dalam penelitian ini menggunkan teknik wawancara tidak terstruktur dengan tujuan untuk memperoleh informasi yang ada relevansinya dengan pokok persoalan penelitian mengenai latar belakang objek penelitian dan data berupa informasi yang diperlukan untuk mengembangkan sistem diagnosa penyakit lambung. Teknik pengumpulan data menggunakan kuesioner dilakukan untuk mendapaktan data bobot gejala penyakit lambung dari subjek penelitian (Pakar) secara langsung.

Data sekunder diperoleh dengan cara studi pustaka. Data sekunder digunakan untuk menunjang data primer tujuanya mendapatkan informasi tambahan mengenai objek penelitian yaitu penyakit lambung terutama informasi mengenai jenis penyakit lambung berserta gejala-gejalanya dan informasi mengenai teori-teori yang digunakan pada penelitian ini.

3.3.2 Tahapan Desain Sistem (*System Design*)

Tahap sistem *design* adalah tahap yang dilakukan setelah data analisis telah terkumpul. Pada tahap desain ini menggunakan bahasa *Unified Modeling Language* (*UML*). Bahasa ini digunakan karena mendukung konsep *Object Oriented Design* (*OOP*) sesuai dengan kode program yang akan digunakan. Beberapa diagram pada *UML* adalah sebagai berikut:

1. *Business Process*

   Business process adalah diagram yang memperlihatkan inputan yang digunakan sistem, *output* yang dihasilkan sistem, dan tujuan pembuatan sistem.
2. *Use Case Diagram*

   Use case diagram adalah Gambaran fitur dari sistem yang dijalankan oleh aktor. Pada diagram ini dapat dilihat juga hak akses dari aktor.

3. *Squance Diagram*

   Squance diagram adalah diagram yang menggambarkan interaksi antara objek satu dengan yang lain di dalam sistem yang dibangun pada urutan waktu. Diagram juga menggambarkan interaksi antara aktor, fitur, serta data yang berjalan.

4. *Activity Diagram*

   Activity diagram adalah pengGambaran alir sistem yang akan dibangun, bagaimana sistem dari awal hingga sistem ditutup, serta bagaimana alir sistem ketika diimplementasikan dengan Teori yang digunakan.

5. *Class Diagram*

   Class diagram adalah diagram yang menggambarkan kelas-kelas dalam sebuah sistem dan hubungannya antara satu kelas dengan yang lain. Dalam kelas ini juga ditampilkan atribut dan operasi yang ada pada sistem.

6. *Entity Relation Diagram* (*ERD*)

   Entity Relation Diagram (*ERD*) adalah diagram yang menggambarkan relasi objek-objek dasar data dalam sebuah basis data.

3.3.3 Tahapan Pengkodean Sistem

Desain yang sudah dibuat pada tahap sebelumnya digunakan sebagai acuan dalam pembangunan sistem. Sistem dibangun menggunakan bahasa pemrograman Page Hypertext Pre-Processor (PHP) dengan tool yang digunakan Visual Studio Code, dan database yang digunakan adalah DBMS MySql menggunakan tool XAMPP. Proses pengkodean menggunakan framework CodeIgniter (CI) karena mendukung penggunaan konsep Object Oriented Programming (OOP). Semua proses dalam Sistem CBR untuk Diagnosa Penyakit lambung menggunakan bahasa pemrograman Page Hypertext Pre-Processor (PHP).

3.3.4 Tahapan Pengujian Sistem (*Testing*)

Tahap *testing* harus dilakukan sebelum sistem digunakan oleh publik. Tahap ini dilakukan agar dapat mengetahui apakah sistem yang dibangun sesuai dengan kebutuhan yang telah dianalisis diawal. Serta agar mengetahui apakah terdapat kesalahan pada sistem yang dibangun. Tahap testing dilakukan guna menyempurnakan sistem. Pada tahap testing ini dilakukan pengujian dengan dengan Teori *white-box* dan Teori *black-box*.

3.3.4.1 White-box Testing

White-box testing merupakan Teknik pengujian yang dilakukan dengan cara melihat modul yang sudah dibuat dengan program-program yang ada dan menganalisa apakah terjadi kesalahan atau tidak pada penulisan kode program. Pengujian *white-box* merupakan teknik pengujian jalur dasar yang digunakan untuk menentukan kompleksitas logis dengan menentukan rangkaian dasar jalur eksekusinya. Tahapan teknik pengujian jalur dasar meliputi *listing program*, grafik alir, kompleksitas siklomatik, jalur program independen, dan pengujian *basis set*. Pengujian *White-box Testing* akan dilakukan pada alur proses diagnosa penyakit lambung, karena pada proses ini memiliki logika program yang cukup rumit dan panjang dibandingkan pada proses-proses lain yang terdapat pada sistem diagnose penyakit lambung. Berikut ini penjelasan dari setiap tahap pengujian degan teknik *White-box testing*:

1. *Listing Program*

   Listing Program merupakan tahap untuk mengumpulkan baris-baris kode yang nantinya akan diuji. Setiap langkah dari kode-kode yang ada diberi nomor baik menjalankan statement biasa atau penggunaan kondisi dalam program. Contoh penerapan tahapan ini dapat dilihat pada Gambar 3.2.

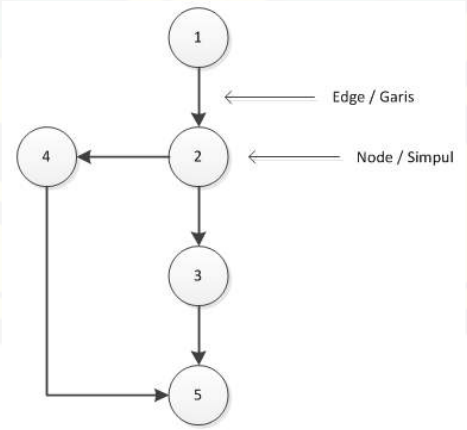
```
$panjang  = $_POST['p'];                    1
$lebar    = $_POST['l'];
    if($panjang == $lebar)                  2
    {
       $jenisBangun = 'Persegi';            3
    }else
                                {
       $jenisBangun = 'Persegi Panjang';    4
    }
                                                5
    $luas = $panjang * $lebar;
    echo 'Luas bangun '.$jenisBangun.' adalah '.$luas;
```

Gambar 3.2 *Listing program*

Sumber (Pressman, 2009)

2. Grafik Alir

Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah grafik alir yang menggunakan notasi lingkaran (simpul atau node) dan anak panah (link atau edge). Notasi ini menggambarkan aliran kontrol logika yang digunakan dalam suatu bahasa pemrograman. Grafik alir merupakan sebuah notasi sederhana yang digunakan untuk mempresentasikan aliran kontrol (Pressman, 2009). Aliran kontrol yang digambarkan merupakan hasil penomoran dari *listing program*. Grafik alir digambarkan dengan node-node (simpul) yang dihubungkan dengan edge-edge (garis) yang menggambarkan alur jalannya program. Contoh penggambaran diagram alir dapat dilihat pada Gambar 3.3.



Gambar 3.3 Contoh Diagram Alir

Sumber (Pressman, 2009)

3. *Cyclomatic Complexity*

   Cyclomatic complexity adalah alat pengukuran untuk mengidentifikasi kompleksitas dari suatu program dengan cara menelusuri nomor dari jalur independen melalui source code-nya. *Cyclomatic complexity* merupakan metrik perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logis suatu program (Pressman, 2009). Rumus yang digunakan untuk menghitung kompleksitas siklomatik, yaitu:

   V(G) = E – N + 2 ……..(Persamaan 2)

   Keterangan:

   V(G) : Kompleksitas siklomatik

   E : Jumlah edge

   N : Jumlah node

4. Jalur Program Independen

   Jalur program independen atau *independent path* adalah alur dari manapun dalam program yang memperkenalkan sedikitnya satu kumpulan perintah pemrosesan atau kondisi baru (Pressman, 2009). Bila dinyatakan dalam grafik alir, jalur independen harus bergerak setidaknya sepanjang satu edge yang belum dilintasi sebelum jalur tersebut didefinisi (Pressman, 2009).

5. Pengujian Basis Set

   Pada bagian ini diberikan contoh data yang akan memaksa pelaksanaan jalur di basis set. Data yang dieksekusi dimasukkan ke dalam grafik alir apakah sudah melewati basis set yang tersedia. Sistem telah memenuhi syarat kelayakan software jika salah satu jalur yang dieksekusi setidaknya satu kali.

3.3.4.2 Black-box Testing

Black-box testing adalah teknik menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian dilakukan dengan membuat kasus uji

yang bersifat mencoba semua fungsi apakah sesuai dengan spesifikasi yang dibutuhkan (Rosa, 2011). Contoh tabel pengujian *black-box* seperti pada Tabel 3.1.

Tabel 3.1 Uji *Black Box*

Kelas Uji	Skenario Uji	Hal yang Diharapkan	Kesimpulan

3.3.5 Tahapan Pemeliharaan Sistem

Pemeliharaan Sistem diperlukan ketika aplikasi telah digunakan oleh *user*. Ketika aplikasi dijalankan mungkin saja masih terjadi kesalahan atau *error* yang tidak ditemukan sebelumnya. Sehingga diperlukan perbaikan pada aplikasi tersebut.

BAB 4. PENGEMBANGAN SISTEM

Bab ini menguraikan mengenai analisis kebutuhan dan perancangan hingga tahap pengkodean dan pengujian aplikasi yang digunakan dalam proses pengembangan atau pembangunan sistem diagnosa penyakit lambung.

4.1 Analisis Kebutuhan Sistem

Berdasarkan metode pengembangan sistem model *waterfall*, tahapan awal yang dilakukan adalah tahapan analisis. Tahapan analisis ini dilakukan terhadap objek penelitian untuk memperoleh kebutuhan-kebutuhan dari sistem yang dibangun, baik berupa kebutuhan fungsional maupun kebutuhan nonfungsional. Dimana hasil analisa tersebut sangat mempengaruhi fungsionalitas sistem yang dibangun untuk dapat digunakan sesuai dengan fungsi dan kebutuhan pengguna. Kebutuhan fungsional dan nonfungsianal diperoleh dari pengumpulan data berupa wawancara terhadap pakar, kuesioner pembobotan gejala, dan studi literatur dengan tujuan mendapatkan semua informasi untuk membangun sistem CBR diagnosa penyakit lambung menggunakan *algoritma k-NN*.

4.1.1 Kebutuhan Fungsional

Kebutuhan fungsional sistem berisi fitur-fitur inti yang harus dipenuhi dalam sistem agar sistem mampu difungsikan sesuai dengan tujuan dan kebutuhan pengguna terhadap sistem itu sendiri. Kebutuhan fungsional dari sistem diagnosa penyakit lambung, yaitu:

1. Sistem mampu mengelola data *user* meliputi (*insert*, *update*, *delete* dan *detail*).
2. Sistem mampu mengedit data *user* meliputi (*update*, dan ubah *password*).
3. Sistem mampu mengelola data basis kasus meliputi (*update*, *detail*, dan *delete*).
4. Sistem mampu mengelola data diagnosa meliputi (*new case*, *detail*, dan *delete*).
5. Sistem mampu mengelola data gejala meliputi (*insert*, *update*,, dan *delete*).
6. Sistem mampu mengelola data pennyakit meliputi (*insert*, *update*, dan *delete*).

7. Sistem mampu menampilkan hasil diagnosa penyakit lambung menggunakan *Algoritma K-NN*.
8. Sistem *login* dengan *username* dan *password*.

4.1.2 Kebutuhan Non-Fungsional

Kebutuhan *non-fungsional* merupakan fitur-fitur yang dimiliki untuk mendukung sistem dalam memenuhi fungsionalitasnya untuk dapat memenuhi kebutuhan dari pengguna. Kebutuhan *non-fungsional* dari sistem ini, yaitu :

1. Sistem dijalankan menggunakan *browser* pada laptop.
2. Sistem memiliki tampilan *user friendly*.

4.2 Deskripsi Umum Sistem

Sistem *Case Based Reasoning* (CBR) Menggunakan *Algoritma K-NN* Untuk Diagnosa Penyakit Lambung adalah sebuah sistem yang dapat digunakan untuk mendiagnosa gejala-gejala maupun keluhan penyakit lambung. Sistem memiliki 3 hak akses yaitu, pengguna (user), pakar, dan admin. Peran pakar pada sistem ini sebagai penentu nilai bobot pada setiap gejala. Pakar juga berperan pada tahap *retain* CBR yaitu meninjau dan menyimpan kasus baru yang berhasil mendapatkan hasil diagnosa agar dapat digunakan oleh kasus-kasus selanjutnya yang mirip dengan kasus tersebut. Pengguna (user) akan di berikan beberapa pertanyaan yang di tampilkan sitem. Masing masing pertanyaan memiliki nilai/bobot yang nantinya akan digunakan untuk menentukan nilai *similarity* antara kasus lama dengan kasus yang baru diinputkan pada tahap *retrive* CBR, pada tahap ini juga di terapkan *Algoritma K-NN* .

4.2.1 Statement Of Perpose (SOP)

CBR Diagnosa Penyakit Lambung diperlukan untuk diagnosa penyakit lambung sebagai upaya pemberian saran alternatif pengobatan yang sesuai dengan jenis penyakit lambung yang didiagnosa. Sistem yang dibangun berbasis pengetahuan dengan domain

Case Based Reasoning. *Algoritma K-NN* diterapkan pada tahap retrive CBR untuk mendapatkan nilai *similaritas* antara kasus lama dengan kasus yang baru diinputkan (diagnosa baru). Nilai yang terdekat dengan kasus lama akan menjadi hasil diagnosa.

4.2.2 Fungsi Sistem

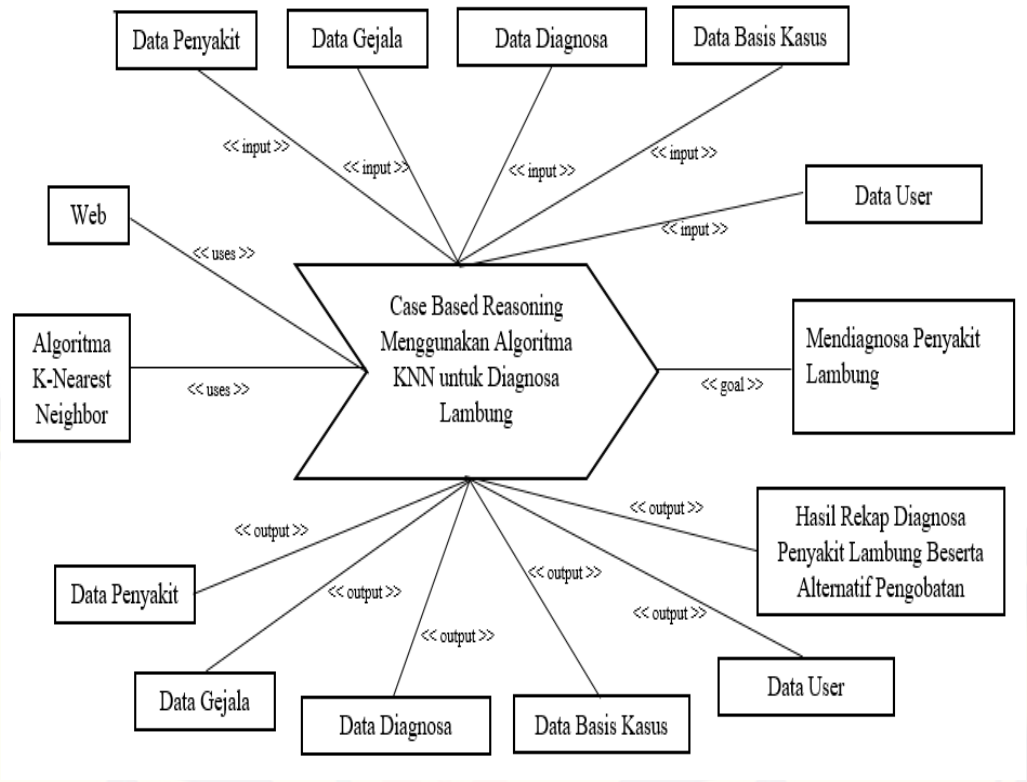
Sistem yang akan dibuat berupa sistem berbasis pengetahuan dengan domain *Case Based Reasoning* yang didukung dengan *Algoritma K-NN*. Sistem ini merupakan sistem yang dapat memberikan pemecahan masalah yang terjadi pada penderita penyakit lambung untuk mengetahui jenis penyakit lambung yang diderita serta alternatif pengobatan.

4.3 Desain Sistem

Tahapan yang dilakukan setelah melakukan analisis kebutuhan sistem yaitu tahap melakukan perencanaan pengembangan sistem yang dapat digambarkan dengan desain sistem. Desain sistem CBR diagnosa penyakit lambung ini, meliputi *bussiness process*, *use case diagram*, *skenario*, *activity diagram*, *sequence diagram*, dan *class diagram*.

4.3.1 Business Process

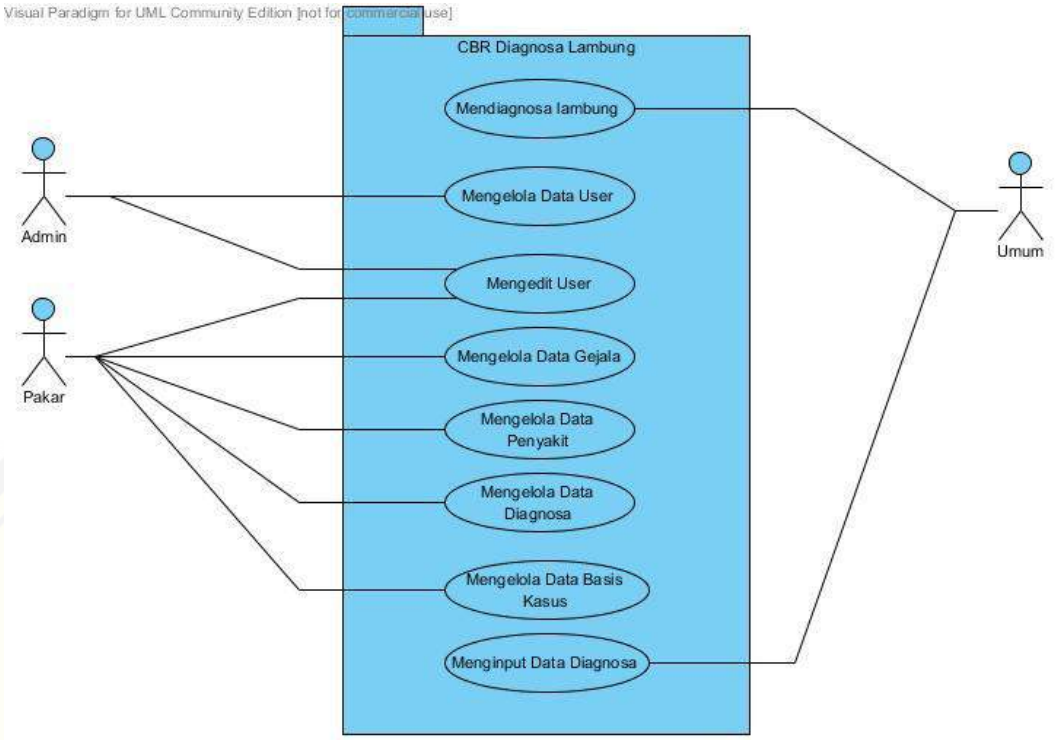
Bussiness process digunakan untuk menggambarkan sebuah proses suatu sistem secara keseluruhan, lengkap dengan *resources* dan informasi yang dibutuhkan seperti *input* dan *output* aplikasi, sehingga mendorong terjadinya proses dan tujuan yang telah ditentukan, seperti pada Gambar 4.1.



Gambar 4.1 *Bussines Process* Sistem CBR Diagnosa Lambung

4.3.2 *Use Case* Diagram

Use case diagram merupakan diagram yang menggambarkan tugas yang dilakukan *user*, baik manusia maupun mesin/komputer. *Use case* digambarkan dari beberapa aktor, *use-case*, dan interaksi diantara komponen tersebut yang dapat memberikan informasi dari sistem yang akan dikembangkan. Fitur-fitur pada sistem ini terdapat 8 fitur yang digambarkan dengan *elips* dan terdapat 3 tipe aktor yaitu admin, pakar, umum (*user*) . Pada Gambar 4.2 digambarkan *use case* diagram sistem CBR Diagnosa Penyakit Lambung yang akan dikembangkan.



Gambar 4.2 *Use Case* Diagram

4.3.2.1 Definisi *Use Case Diagram*

Definisi *use case* merupakan penjelasan dari setiap *use case* yang merupakan fitur dari sistem. Penjelasan definisi dari *use case* sistem CBR diagnosa penyakit lambung dilihat pada Tabel 4.1.

Tabel 4.1 Penjelasan *Use Case*

No		Deskripsi
1	**Mendiagnosa Penyakit Lambung**	
	Diagnosa	*Use case* yang menggambarkan proses menginputkan nama, usia, latar belakang dan memilih gejala yang diderita.

No		Deskripsi
2	**Mengelola Data User**	
	Tambah *User*	*Use case* yang menggambarkan proses menambahkan *user* ke sistem.
	Update User	*Use case* yang menggambarkan proses mengubah *user* yang disimpan dalam sistem.
	Detail User	*Use case* yang menggambarkan proses melihat *user* dalam sistem.
	Ubah *Password*	*Use case* yang menggambarkan proses mengubah password *user* yang disimpan dalam sistem.
3	**Mengedit User**	
	Update User	*Use case* yang menggambarkan proses mengubah data *user* yang disimpan dalam sistem.
	Ubah *Password*	*Use case* yang menggambarkan proses mengubah password *user* yang disimpan dalam sistem.
4	**Mengelola Data Gejala**	
	Tambah Gejala	*Use case* yang menggambarkan proses menginputkan gejala beserta bobot
	Update Gejala	*Use case* yang menggambarkan proses mengubah gejala yang disimpan dalam sistem.
	Delete Gejala	*Use case* yang menggambarkan proses menghapus gejala dari sistem.

No		Deskripsi
5	**Mengelola Data Penyakit**	
	Tambah Nama Penyakit	*Use case* yang menggambarkan proses menambahkan nama penyakit ke sistem.
	Update Nama Penyakit	*Use case* yang menggambarkan proses mengubah nama penyakit yang disimpan dalam sistem.
	Delete Nama Penyakit	*Use case* yang menggambarkan proses menghapus nama penyakit dari sistem.
6	**Mengelola Data Diagnosa**	
	Revise Diagnosa	*Use case* yang menggambarkan proses mengubah data diagnosa dan menyimpan data hasil revisi di data basis kasus.
	Detail Diagnosa	*Use case* yang menggambarkan proses melihat detail diagnosa dalam sistem.
	Delete Diagnosa	*Use case* yang menggambarkan proses menghapus data diagnosa dari sistem.
7	**Mengelola Data Basis Kasus**	
	Update Basis Kasus	*Use case* yang menggambarkan proses mengubah data basis kasus yang disimpan dalam sistem.
	Delete Basis Kasus	*Use case* yang menggambarkan proses menghapus data basis kasus yang disimpan dalam sistem.
8	**Menginput Data Diagnosa**	
	Menambah Data Diagnosa	*Use case* yang menggambarkan proses menambah data diagnosa.

4.3.3 Skenario

Skenario diagram digunakan untuk menjelaskan fitur yang ada di *use case* diagram. Skenario menjelaskan alur sistem dan keadaan yang akan terjadi ketika terjadi suatu *event* tertentu.

Skenario pada sistem CBR diagnosa penyakit lambung dapat disajikan pada Tabel 4.2.

Tabel 4.2 Skenario Mendiagnosa Penyakit Lambung

Nomor *Use Case*	UC-01
Nama	Mendiagnosa Penyakit Lambung
Aktor	Umum
Pre Condition	Umum harus sudah membuka halaman awal CBR Diagnosa Lambung.
Post Condition	Umum berhasil mendiagnosa penyakit lambung.
SKENARIO NORMAL MENDIAGNOSA PENYAKIT LAMBUNG	
TAMBAH DATA	
Aktor	Sistem
1. Membuka halaman awal CBR Diagnosa Lambung.	
	2. Menampilkan halaman awal CBR Diagnosa Lambung.
3. Klik menu diagnosa.	
	4. Menampilkan halaman Diagnosa Penyakit Lambung yang berisi tombol Diagnosa, tombol cancel, dan form diagnosa yang berisi Nama,

	Usia, Latar Belakang dan Pilihan Gejala
5. Klik tombol Diagnosa.	
	6. Menampilkan halaman hasil diagnosa penyakit lambung yang berisi informasi hasil diagnosa, tombol detail, tombol simpan, tombol diagnosa ulang
7. Klik tombol *detail*	
	8. Menampilkan modal detail hasil diagnosa yang berisi informasi alternatif pengobatan, penyebab penyakit lambung, tombol close.
9. Klik tombol *close*	
	10. Menutup tampilan modal detail hasil diagnosa
11. Klik tombol Simpan	
	12. Menampilkan halam awal dengan pesan “Data Anda Berhasil Tersimpan di Sistem Kami, Terima Kasih sudah berkontribusi untuk pengembangan CBR Diagnosa Lambung”
SKENARIO ALTERNATIF MENDIAGNOSA PENYAKIT LAMBUNG	
DATA DIAGNOSA TIDAK LENGKAP	
4a. Klik tombol Diagnosa	

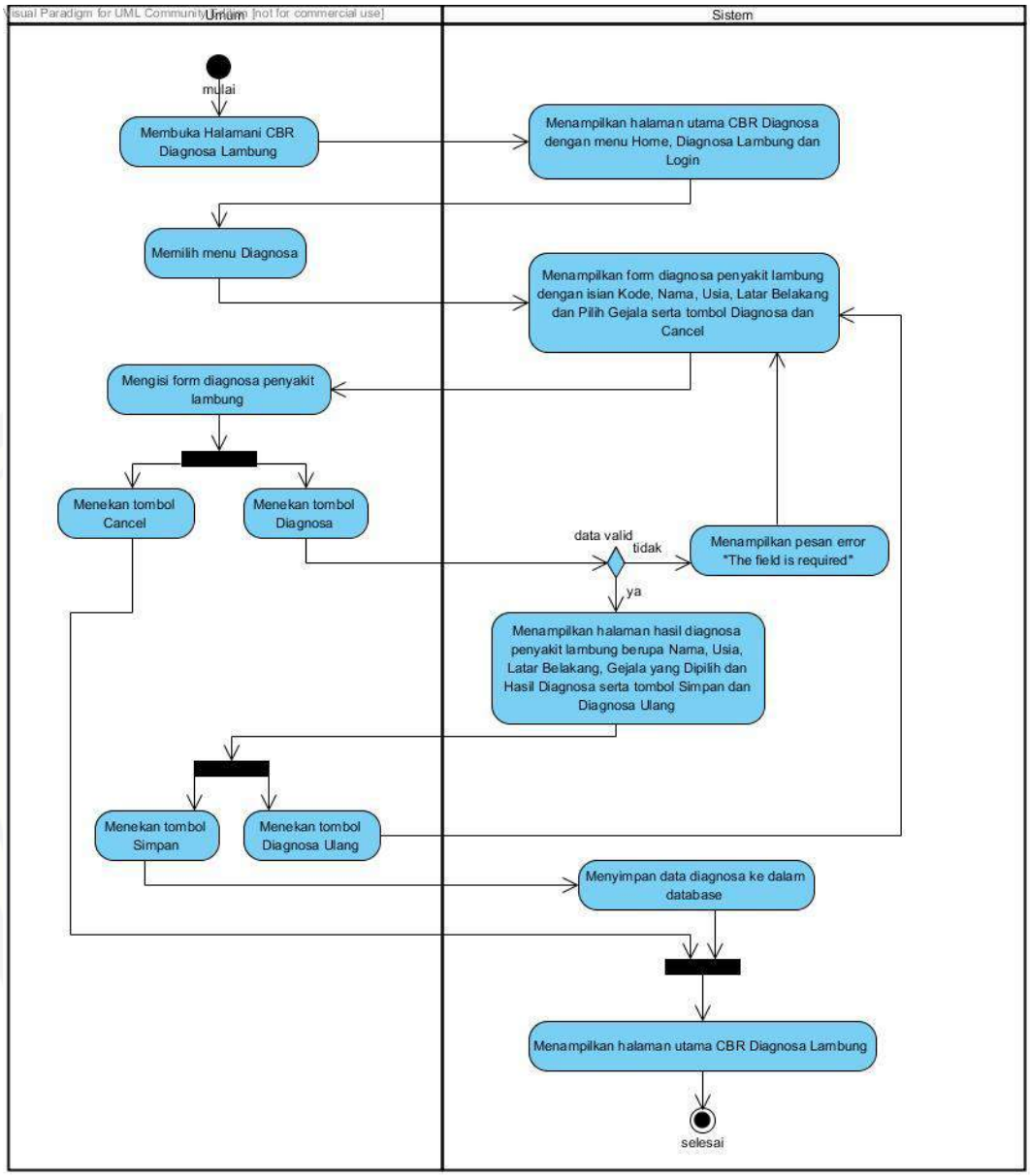
	5a. Menampilkan peringatan "*This field is required*".
SKENARIO ALTERNATIF MENDIAGNOSA PENYAKIT LAMBUNG TOMBOL *CANCEL*	
11a. Klik tombol *Cancel.*	
	12a. Menampilkan halaman awal CBR Diagnosa Lambung
SKENARIO ALTERNATIF MENDIAGNOSA PENYAKIT LAMBUNG TOMBOL DIAGNOSA ULANG	
13a. Klik tombol Diagnosa Ulang.	
	14a. Menampilkan halaman awal CBR Diagnosa Lambung

Skenario mengelola data user, skenario mengedit user, skenario mengelola data gejala, skenario mengelola data penyakit, dan skenario mengelola data diagnosa dapat dilihat pada Lampiran A.

4.3.4 Activity Diagram

Activity Diagram menggambarkan aktivitas aktor dan sistem yang saling berhubungan dalam suatu aktivitas atau *event*. *Activity Diagram* menggambarkan berbagai alur aktivitas dalam sistem yang dirancang berawal *decision*. *Activity diagram* harus sesuai dengan skenario sistem yang telah dirancang. Sistem memberikan respon pada aktivitas yang dilakukan aktor. Berikut adalah penjelasan *activity diagram* dari sistem yang dirancang.

Activity Diagram mendiagnosa penyakit lambung menggambarkan alur aktivitas pada fitur mendiagnosa penyakit lambung. Fitur mendiagnosa penyakit lambung memiliki aktivitas utama, yaitu mendiagnosa penyakit lambung. Detail urutan aktifitas dalam mendiagnosa penyakit lambung ditunjukan pada Gambar 4.3.



Gambar 4.3 *Activity Diagram* Mendiagnosa Penyakit Lambung

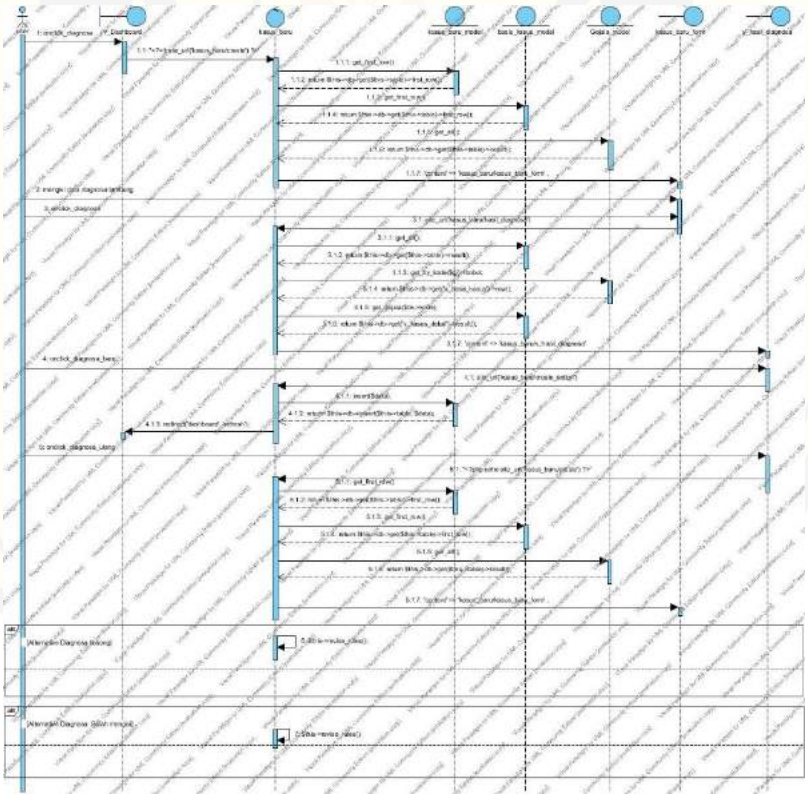
Activity diagram mengelola data user, *activity diagram* mengedit user, *activity diagram* mengelola data gejala, *activity diagram* mengelola data penyakit, *activity*

diagram mengelola data diagnosa, dan mengelola data basis kasus dapat dilihat pada Lampiran B.

4.3.5 Sequence Diagram

Sequence diagram digunakan untuk menggambarkan prilaku pada sebuah skenario. Berikut adalah salah satu penjelasan *sequence diagram* dari sistem yang dirancang.

Sequence diagram mendiagnosa penyakit lambung digunakan untuk menunjukkan interaksi antar objek pada fitur mendiagnosa penyakit lambung. Objek yang terlibat pada fitur mendiagnosa penyakit lambung dengan aktor umum, antara lain *view (*v_dashboard; kasus_baru_form; v_hasil_diagnosa;), *Controller (*Kasus_baru), *model* (Kasus_baru_model). Detail *sequence diagram* mendiagnosa penyakit lambung ditunjukan pada Gambar 4.4.

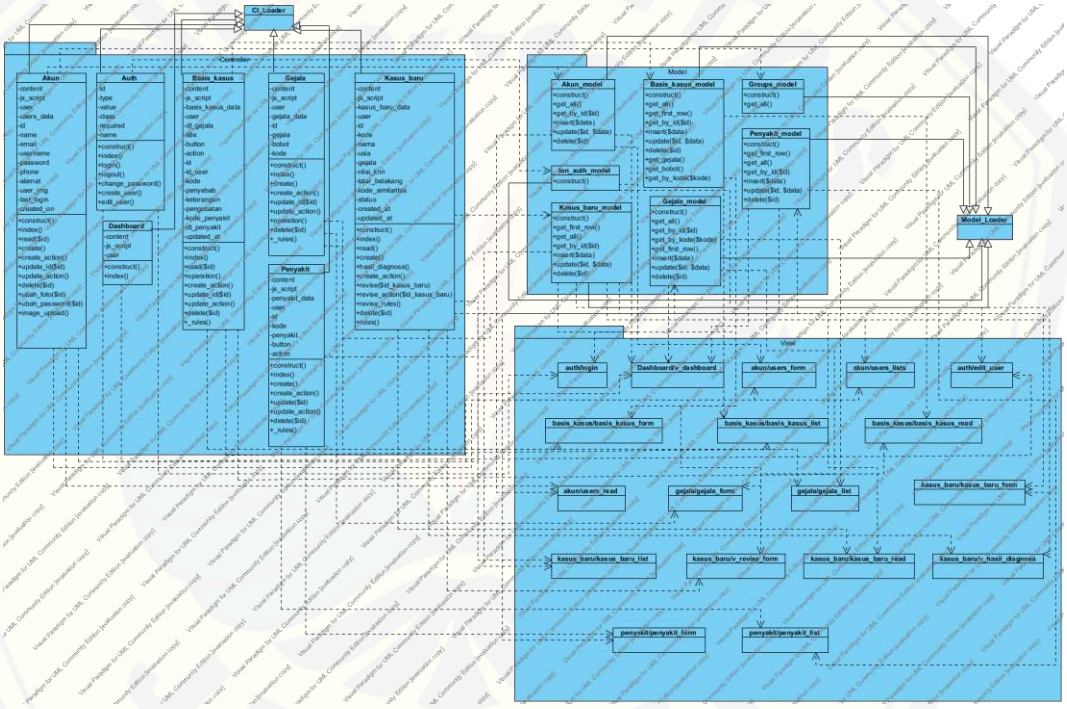


Gambar 4.4 *Sequence Diagram* Mendiagnosa Penyakit Lambung

Keseluruhan *sequence diagram* fitur yang terdapat pada sistem CBR diagnosa penyakit lambung dapat dilihat pada lampiran C.

4.3.6 Class Diagram

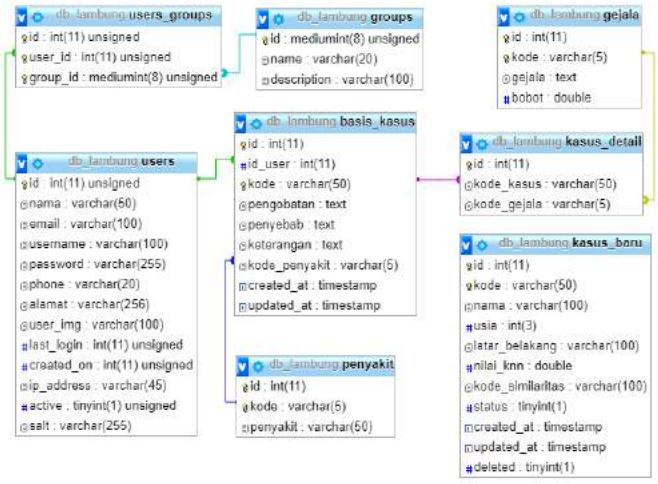
Setelah melalui tahap pembuatan desain dengan *sequence diagram*, tahap selanjutnya membuat desain perancangan *class diagram*. *Class diagram* berisi method dan data yang berbeda namun memiliki hubungan dengan yang lainnya. *Class Diagram* lebih lengkapnya dapat dilihat pada Gambar 4.5.



Gambar 4.5 *Class Diagram* Sistem CBR Diagnosa Penyakit Lambung

4.3.7 *Entity Relationship Diagram* (ERD)

Setelah pembuatan *class diagram*, tahap perancangan selanjutnya adalah membuat desain *database*. Desain ini berisi basis data yang akan digunakan oleh sistem yang akan dibangun. *Entity Relationship Diagram* lebih lengkapnya dapat dilihat pada Gambar 4.6.



Gambar 4.6 ERD Sistem CBR Diagnosa Penyakit Lambung

4.4 Pengkodean Sistem

Setelah tahap desain perancangan selesai, tahap selanjutnya dalam penelitian ini yaitu tahap pengimplementasian desain perancangan ke dalam bahasa pemrograman. Bahasa pemrograman yang dipakai adalah bahasa pemrograman PHP (*Hypertext Preprocessor*) dan menggunakan *database MySql*. Dalam perancangan sistem CBR diagnosa penyakit lambung ini menggunakan *framework Code Igniter 3* untuk memudahkan di dalam pengembangan dan penulisan *coding*. Pada tahap implementasi perancangan ini menjelaskan tentang fitur – fitur yang terdapat pada sistem CBR Diagnosa Penyakit Lambung. Dokumentasi pengkodean sistem CBR diagnosa penyakit lambung dapat dilihat pada Lampiran D. Pada pengkodean sistem ini di terapkan *algoritma k-NN* dalam sistem CBR diagnosa penyakit lambung untuk menhitung nilai similaritas antara data kasus diagnosa baru dengan *case memory.*

4.5 Pengujian Sistem

4.5.1 Metode *White-box*

Pengujian sistem dengan metode *white box* dilakukan untuk menguji sistem dari segi kode program. Pengujian ini bertujuan sebagai evaluasi apakah sistem berhasil menjalankan fungsi-fungsi, inputan, dan keluaran yang sesuai dengan spesifikasi dari

kebutuhan sistem itu sendiri. Pengujian dengan metode *white box* dilakukan oleh penulis dengan cara menghitung *independent path* yaitu dengan menggunakan suatu pengukuran kuantitatif yang bertahap mulai dari *listing program*, pembuatan diagram alir, *cyclomatic complexity*. Tahapan-tahapan pengujian dengan metode *white box* ini akan diterapkan pada fitur yang dinilai dapat mewakili sistem ini yaitu fitur mendiagnosa penyakit lambung. Tahapan pengujian jalur dasar meliputi:

a. *Listing Program*

Listing program merupakan baris-baris kode yang nantinya akan diuji. Setiap langkah dari kode-kode yang ada diberi nomor baik menjalankan *statement* biasa atau penggunaan kondisi dalam program. *Listing program* sistem CBR Diagnosa Lambung.

```
114     public function hasil_diagnosa()
115     {
116         $this->_rules();
117         if ($this->form_validation->run() == FALSE) {
118             $this->create();
119         } else {
120             $kode = $this->input->post('kode',TRUE);
121             $gejala = $this->input->post('gejala',TRUE);
122             $basis_kasus = $this->Basis_kasus_model->get_all();
123             foreach ($gejala as $gj) {
124                 $nilai_input[] = $this->Gejala_model->get_by_kode($gj)->bobot;
125             }
126             $nilai_inputan = array_sum($nilai_input);
127             $total = array();
128             foreach ($basis_kasus as $bk) {
129                 $kasus_detail = $this->Basis_kasus_model->get_gejala($bk->kode);
130                 foreach ($kasus_detail as $kd) {
131                     foreach ($gejala as $g) {
132                         if($kd->kode_gejala == $g){
133                             $bobot[$kd->kode_kasus][$g] = $kd->bobot;
134                         } else {
135                             $bobot[$kd->kode_kasus]['G0'] = 0;
136                         }
137                     }
138                 }
139             }
140             foreach ($basis_kasus as $bk) {
141                 if($bobot[$bk->kode]){
142                     $total[$bk->kode] = array_sum($bobot[$bk->kode]);
143                 } else {
144                     $total[$bk->kode] = 0;
145                 }
146             }
```

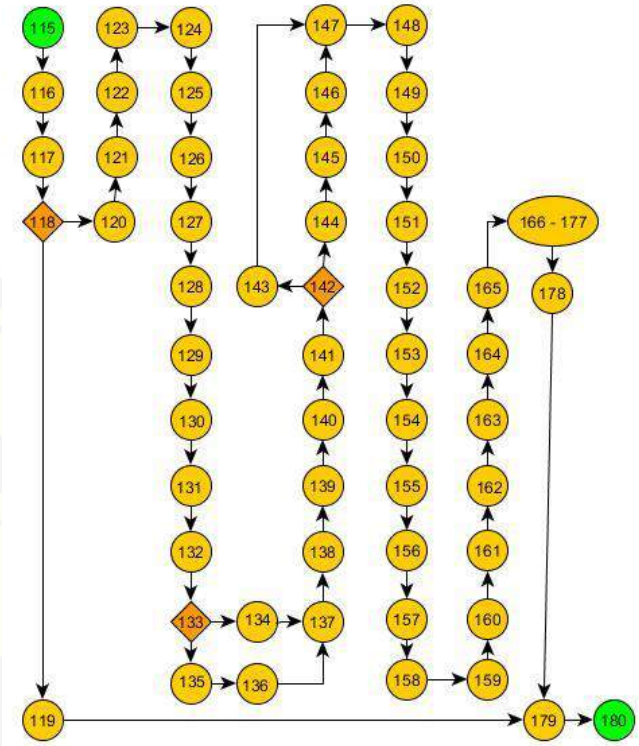
```
147             foreach ($bobot as $key => $value) {
148                 $hasil_bagi = $total[$key] / $nilai_inputan ;
149                 $persen_hasil_bagi = $hasil_bagi * 100;
150                 $arr_hasil[] = $persen_hasil_bagi."-".$key;
151             }
152             $count = count($arr_hasil);
153             $temp_kode_nilai = array();
154             for ($i=0; $i < $count; $i++) {
155                 $hasil = explode('-',$arr_hasil[$i]);
156                 $temp_kode_nilai[$hasil[1]] = $hasil[0];
157             }
158             $maxs = max($temp_kode_nilai);
159             $kode_same_val = array_keys($temp_kode_nilai, max($temp_kode_nilai));
160             for ($i=0; $i < count($kode_same_val) ; $i++) {
161                 $temp_nilai[] = $maxs;
162             }
163             $hasil_filnal_diagnosa = array_combine($kode_same_val, $temp_nilai);
164             $user = $this->ion_auth->user()->row();
165             $data = array(
166                 'user' => $user ,
167                 'content' => 'kasus_baru/v_hasil_diagnosa' ,
168                 'js_script' => 'kasus_baru/kasus_baru_js_script',
169                 'action' => site_url('kasus_baru/create_action'),
170                 'kode' => $kode,
171                 'nama' => $this->input->post('nama',TRUE),
172                 'usia' => $this->input->post('usia',TRUE),
173                 'latar_belakang' => $this->input->post('latar_belakang',TRUE),
174                 'gejala' => $gejala ,
175                 'hasil_diagnosa' =>  $hasil_filnal_diagnosa,
176             );
177             $this->load->view('layout/template', $data);
178         }
179     }
```

Gambar 4.7 *Listing program* Fitur Mendiagnosa penyakit lambung

Dari gambar 4.7 dapat dilihat fitur diagnosa penyakit lambunng yang dituliskan dengan bahasa pemrograman php dari baris nomor 114 sampai baris nomor 179. Kode pada baris ini digunakan untuk proses menghitung nilai similaritas antara diagnosa baru dengan data basis kasus.

b. Diagram Alir

Diagram alir merupakan notasi sederhana yang digunakan untuk merepresentasikan aliran kontrol. Aliran kontrol yang digambarkan merupakan hasil penomoran dari *listing* program. Diagram alir digambarkan dengan *node-node* (simpul) yang dihubungkan dengan *edge-edge* (garis) yang menggambarkan alur jalannya program. Diagram alir diagnosa penyakit lambung dapat dilihat pada Gambar 4.8.



Gambar 4.8 Diagram alir fitur Mendiagnosa penyakit lambung

Pada Gambar 4.8 dinotasikan alur proses fitur mendiagnosa penyakit lambung. Nomor 117 sampai 179 diambil dari tahap listing program. Notasi node elips untuk proses normal seperti nomor 117 sedangkan notasi node belah ketupat untuk proses percabangan (*if* atau perulangan) seperti nomor 118. Setiap notasi node memiliki *edge* (garis) masuk ataupun keluar yang sesuai dengan alur program. Diagaram alir ini akan digunakan untuk tahap brikutnya, yaitu tahap menghitung nilai kompleksitas siklomatik dan tahap menghitung jumlah jalur independen.

.

c. Kompleksitas Siklomatik (*cyclomatic complexity*)

Kompleksitas siklomatik merupakan metrik perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logis suatu program. Bila digunakan dalam konteks teknik pengujian jalur dasar, nilai yang dihitung untuk kompleksitas

siklomatik mendefinisikan jumlah jumlah jalur independen dalam basis set suatu program. Kompleksitas siklomatik diagnosa penyakit lambung berdasarkan diagram alir sebagai berikut:

CC = EDGE – NODE + 2

CC = 57 - 54 + 2

CC = 5

d. Jalur Program Independen

Jalur independen adalah setiap jalur yang melalui program yang memperkenalkan setidaknya satu kumpulan pernyataan-pernyataan pemrosesan atau kondisi baru. Bila dinyatakan dalam grafik alir, jalur independen harus bergerak setidaknya sepanjang satu *edge* yang belum dilintasi sebelum jalur tersebut didefinisi. Jalur program independen diagnosa penyakit lambung sebagai berikut:

Jalur 1 = 115-118-119-179-180

Jalur 2 = 115-118-120-133-134-137-142-144-147-179-180

Jalur 3 = 115-118-120-133-134-137-142-143-147-179-180

Jalur 4 = 115-118-120-133-135-137-142-144-147-179-180

Jalur 5 = 115-118-120-133-135-137-142-143-147-179-180

e. Pengujian Basis Set

Pada bagian ini diberikan contoh data yang akan memaksa pelaksanaan jalur di basis set. Data yang dieksekusi dimasukkan ke dalam grafik alir apakah sudah melewati basis set yang tersedia. Sistem telah memenuhi syarat kelayakan perangkat lunak jika salah satu jalur yang dieksekusi setidaknya satu kali. Pengujian basis set diagnosa penyakit lambung dapat dilihat pada Tabel 4.3.

Tabel 4.3 *Test Case* Diagnosa Penyakit Lambung

Jalur 1	
Test Case	Jika validasi data gagal maka akan kembali ke *form* diagnosa
Target yang diharapkan	Menampilkan pesan validasi data pada form diagnose
HasilPengujian	Benar
Path/Jalur	115-118-119-179-180
Jalur 2	
Test Case	Jika validasi data berhasil, mengambil data inputan gejala. Jika inputan data gejala ada di database, semua data gejala inputan disimpan ke dalam array
Target yang diharapkan	Menyimpan data inputan gejala kedalam array.
HasilPengujian	Benar
Path/Jalur	115-118-120-133-134-137-142-144-147-179-180
Jalur 3	
Test Case	Jika validasi data berhasil, mengambil data inputan gejala. Jika inputan data gejala tiadak ada di database, set array bernilai 0
Target yang diharapkan	Array data gejala bernila 0
HasilPengujian	Benar
Path/Jalur	115-118-120-133-134-137-142-143-147-179-180
Jalur 4	
Test Case	Jika data bobot gejala ada di database, ambil semua data bobot gejala sesui data inputan gejala
Target yang diharapkan	Menampilkan hasil diagnose
HasilPengujian	Benar
Path/Jalur	115-118-120-133-135-137-142-144-147-179-180

Jalur 4	
Test Case	Jika data bobot gejala tidak ada di database, membuat pesan error
Target yang diharapkan	Menampilkan pesan error
HasilPengujian	Benar
Path/Jalur	115-118-120-133-135-137-142-143-147-179-180

4.5.2 Metode *Black-box*

Pengujian *black box* berfungsi untuk menguji sistem CBR diagnosa penyakit lambung dari segi spesifikasi fungsional sistem dengan tujuan mengetahui apakah fungsi-fungsi, inputan, dan keluaran sistem sesuai dengan spesifikasi yang dibutuhkan oleh pengguna. Hasil pengujian dengan metode *black box* dapat dilihat pada Lampiran E1.

BAB 6. PENUTUP

6.1 Kesimpulan

Kesimpulan yang dapat ditarik dari penelitian yang telah dilakukan sebagai berikut:

1. Sistem dikembangkan dengan menggunakan model *waterfall.* Tahap pengembangan sistem meliputi, Analisis data yang dilakukan melalui wawancara, kuesioner dan studi literatur, Desain sistem, Penulisan kode program dengan menerapkan *algoritma k-nn* untuk proses menghitung nilai similaritas antara diagnosa baru dengan *case memory*. Pengujian *white box* yang menghasilkan nilai kompleksitas sebesar 5 sehingga baris kode metode pada sistem ini terbilang memiliki tingkat kompleksitas yang tidak terlalu tinggi dan dapat dilakukan proses *maintenance* dengan mudah jika terjadi kesalahan. Sistem CBR Diagnosa Lambung memiliki tiga hak akses yaitu *user* (masyarakat umum), admin dan pakar dengan fitur utamanya adalah diagnosa peyakit lambung. Pada pengembangan model *waterfall* untuk sistem ini ditemui beberapa kendala karena pada tahap implementasi kode program, jika terjadi perubahan kebutuhan yang mengakibatkan berubahnya alur proses sistem sehingga membutuhkan perbaikan pada sistem. Pada model waterfall hal tersebut harus memulai kembali dari proses awal (requirement), namun pada pengembangan pada penelitian ini hanya memperbaiki dari tahap desain dan implementasi kode program saja. Sehingga pengembangan sistem pada penelitian ini kurang sesuai dengan tahapan model *waterfall.*
2. Penerapan *Case Based Reasoning* pada sistem CBR Diagnosa Lambung memerlukan dua hak akses yaitu *user* (masyarakat umum) dan pakar. Gejala yang dipilih oleh *user* akan disamakan dengan data pada *case memory* dan dihitung kemiripan datanya antara data diagnosa dengan *case memory*, apabila nilai kemiripan data melebihi atau sama dengan 80% maka diagnosa tersebut dapat digunakan sebagai acuan diagnosa, apabila data kurang dari 80% maka data akan

masuk kedalam hak akses pakar yang akan dilihat kembali apakah data tersebut masuk pada tahap *revise*. Tahap *revise* akan dilakukan apabila data termasuk data kasus baru (penyakit baru) dan akan disimpan sebagai data kasus baru dengan data yang telah diperbarui. Data diagnosa baru akan diindekskan kedalam *case memory* untuk digunakan pada diagnosa selanjutnya, proses ini dinamakan tahap *retain*.

3. Implementasi *algoritma knn* pada sistem CBR Diagnosa Lambung membutuhkan data pembobotan gejala. Data bobot dejala diperoleh dengan cara membuat kuesioner dan mengolah data bobot gejala tersebut kemudian menguji data bobot gejala tersebut pada setiap jenis penyakit lambung. *Algoritma k-nn* diterapkan pada tahap retrive CBR yang digunakan untuk menghitung nilai similaritas antara diagnosa baru dengan *case memory*. Sistem diagnosa penyakit lambung ini dapat digunakan karena telah diuji hasil perhitungan menggukan *algoritma k-NN* yang ada dalam sistem sama dengan hasil perhitungan manual *algoritma k-NN* yang telah dilakukan.
4. Dari uji akurasi menggunakan 10 sampel data diagnosa, diperoleh persentase sebanyak 90% Sitem *Case Based Reasoning* menggunakan *Algoritma k-NN* sesuai dengan diagnosa pakar (dokter). Sehingga Sistem *Case Based Reasoning* menggunakan *Algoritma k-NN* pada penelitian ini dapat digunakan untuk diagnosa penyakit lambung karena sesuai dengan diagnosa dokter.

6.2 Saran

Beberapa saran dan masukan berikut diharapkan dapat memberikan perbaikan dalam penelitian selanjutnya, yaitu :

1. Metode Pengembangan SDLC pada penelitian berikutnya dapat menggunakan model lain selain model *waterfall*.
2. Pegembangan lebih lanjut pada penelitian ini diharapkan sistem dapat menyaring data diagnosa baru agar tidak langsung masuk kedalam hak akses pakar sehingga hak akses pakar lebih efsien.

3. Sistem CBR Diagnosa Lambung dapat dikembangkan dengan perrangkat lain seperti versi *mobile* atau versi desktop.
4. Proses perhitungan nilai kemiripan tidak harus menggunakan *Algoritma K-NN*, dapat dikembangkan atau dibandingkan dengan metode lainya.

DAFTAR PUSTAKA

Bukhori, S. (2012). *Kecerdasan Buatan (Teori, Pemodelan dan Simulasi).* Jember: UPT Penerbitan UNEJ.

Akmal, F., & Winiarti, S. (2014). SISTEM PAKAR UNTUK MENDIAGNOSA PENYAKIT LAMBUNG DENGAN IMPLEMENTASI METODE CBR (CASE-BASED REASONING) BERBASIS WEB, *1*, 1–13.

Ariani, D. T., & Findawati, Y. (2015). SISTEM PAKAR PENYAKIT LAMBUNG DENGAN METODE DEMPSTER SHAFER BERBASIS WEB, 1–13.

Arifin, J., Magister, P., Informasi, T., Tinggi, S., Surabaya, T., & Pakar, S. (2011). Sistem Pakar Untuk Mengidentifikasi Jenis Penyakit Pada Tanaman Jeruk Berbasis WAP. *Prosiding Konferensi Nasional "Inovasi Dalam Desain Dan Teknologi" - IDeaTech 2011 ISSN: 2089-1121 SISTEM*, 152–163.

Ilawati Pristiani. (2013). Penyakit Kanker Lambung dan Tingkat Kejadiannya. Retrieved March 29, 2017, from http://herbalkankerterbaik.blogspot.co.id/2013/07/penyakit-kanker-lambung.html

Kusuma, D. A., & Chairani. (2014). Rancang Bangun Sistem Pakar Pendiagnosa Penyakit Paru-Paru Menggunakan Metode Case Based Reasoning. *Infotel*, *6*(2), 57–62.

Mukhlason, A., Prakoso, I. M., & Anggraeni, W. (2016). Penerapan Case-Based Reasoning pada Sistem Cerdas untuk Pendeteksian dan Penanganan Dini.

Nurdiansyah, Y. (2010). CASE-BASED REASONING UNTUK PENDUKUNG DIAGNOSA GANGGUAN PADA ANAK AUTIS.

Pressman, R. S. (2009). *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman*. *Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman*. https://doi.org/10.1017/CBO9781107415324.004

Putri, T. E., Andreswari, D., & Efendi, R. (2016). IMPLEMENTASI METODE CBR (CASE BASED REASONING) DALAM PEMILIHAN PESTISIDA TERHADAP HAMA PADI SAWAH MENGGUNAKAN ALGORITMA K-NEAREST NEIGHBOR (KNN) (Studi Kasus Kabupaten Seluma), *4*(1), 80–92.

Ritonga, N. A. (2013). SISTEM PAKAR MENDIAGNOSA PENYAKIT LAMBUNG MENGGUNAKAN METODE CERTAINTY FACTOR, *1*(1), 157–161.

Rosa A.S, and M. S. (2011). Modul Pembelajaran Rekayasa Perangkat Lunak (Terstruktur dan Berorientasi Obyek). *Bandung: Modula*, *53*(1), 160. https://doi.org/10.1017/CBO9781107415324.004

LAMPIRAN

Lampiran A. Skenario

A.1 Skenario Mengelola Data User

Tabel A.1 Skenario Mengelola Data User

Nomor *Use Case*	UC-02
Nama	Mengelola Data User
Aktor	Admin
Pre Condition	Admin harus sudah *login* sistem.
Post Condition	Admin berhasil mengelola data user.

SKENARIO NORMAL MENGELOLA DATA USER	
TAMBAH USER	
Aktor	Sistem
1. Memilih menu *Kelola User*.	
	2. Menampilkan halaman kelola user yang berisi tabel user dengan kolom User Image, Nama, Email, Username, Phone, Alamat, Last Login, tombol Create New, tombol Edit dan tombol Delete.
3. Klik tombol Create New.	
	4. Menampilkan halaman Form Tambah User yang berisi form tambah user dengan kolom Nama, No Telepon, Alamat, User image, User level, Email, Username, Password, Konfirmasi Password, tombol Tambah dan tombol Cancel.

5. Mengisi Form Tambah User.	
6. Klik tombol Submit.	
	7. Menampilkan halaman kelola user yang berisi tabel user dengan kolom User Image, Nama, Email, Username, Phone, Alamat, Last Login, tombol Create New, tombol Edit dan tombol Delete.
SKENARIO ALTERNATIF MENGELOLA DATA USER	
TOMBOL CANCEL PADA TAMBAH USER	
6a. Klik tombol Cancel.	
	7a. Menampilkan halaman kelola user yang berisi tabel user dengan kolom User Image, Nama, Email, Username, Phone, Alamat, Last Login, tombol Create New, tombol Edit dan tombol Delete.
SKENARIO ALTERNATIF MENGELOLA DATA USER	
DATA TIDAK LENGKAP	
6b. Klik tombol *Tambah*.	
	7b. Menampilkan *warning* "*This field is required*".
SKENARIO NORMAL MENGELOLA DATA USER	
TOMBOL EDIT	
Aktor	Sistem
1. Memilih menu *Kelola User*.	
	2. Menampilkan halaman kelola user yang berisi tabel user dengan kolom

	User Image, Nama, Email, Username, Phone, Alamat, Last Login, tombol Create New, tombol Edit dan tombol Delete.
3. Klik tombol *Edit*.	
	4. Menampilkan form update user yang berisi tabel user dengan kolom Nomor, Nama, Email, Username, Phone, Alamat, User img, tombol Submit, tombol Change Password dan tombol Cancel.
5. Mengubah isi data form user.	
6. Klik tombol *Submit*.	
	7. Menampilkan halaman kelola user yang berisi tabel user dengan kolom User Image, Nama, Email, Username, Phone, Alamat, Last Login, tombol Create New, tombol Edit dan tombol Delete.
SKENARIO ALTERNATIF MENGELOLA DATA USER	
TOMBOL CANCEL	
6a. Klik tombol *Cancel*.	
	7a. Menampilkan halaman kelola user yang berisi tabel user dengan kolom Nomer, Nama, Email, Username, Phone, Alamat, User img, tombol

	Tambah, tombol Detail dan tombol Update.

A.2 Skenario Mengedit User

Tabel A.2 Skenario Mengelola Data Gejala

Nomor Use Case	UC-03
Nama	Mengedit Data User
Aktor	Admin dan Pakar
Pre Condition	Admin dan Pakar harus sudah login sistem.
Post Condition	Admin dan Pakar berhasil mengubah data user.
SKENARIO NORMAL MENGEDIT DATA USER	
EDIT USER	
1. Klik tombol *Profil*.	
	2. Menampilkan form update user yang berisi tabel user dengan kolom Nomor, Nama, Email, Username, Phone, Alamat, User img, tombol Submit, tombol Change Password dan tombol Cancel.
3. Mengubah isi data form user.	
4. Klik tombol *Submit*.	
	5. Menampilkan halaman home
SKENARIO ALTERNATIF MENGELOLA DATA USER	
TOMBOL CANCEL	
6a. Klik tombol *Cancel*.	

	7a. Menampilkan halaman home

A.3 Skenario Mengelola Data Gejala

Tabel A.3 Skenario Mengelola Data Gejala

Nomor *Use Case*	UC-04
Nama	Mengelola Data Gejala
Aktor	Pakar
Pre Condition	Pakar harus sudah *login* sistem.
Post Condition	Pakar berhasil mengelola data gejala.

SKENARIO NORMAL MENGELOLA DATA GEJALA	
TAMBAH GEJALA	
Aktor	Sistem
1. Memilih menu Data Gejala.	
	2. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Create New*.	
	4. Menampilkan halaman *Form Create* Gejala yang berisi *form* tambah gejala dengan kolom Kode, Gejala, Bobot, tombol *Submit* dan tombol *Cancel*.
5. Mengisi *Form Create* Gejala.	
6. Klik tombol *Submit*.	
	7. Menyimpan data baru ke database

	8. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA GEJALA TOMBOL CANCEL PADA TAMBAH USER	
6a. Klik tombol *Cancel*.	
	7a. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA GEJALA DATA TIDAK LENGKAP	
6b. Klik tombol *Submit*.	
	7b. Menampilkan *warning* "*This field is required*".
SKENARIO NORMAL MENGELOLA DATA GEJALA TOMBOL EDIT	
Aktor	Sistem
1. Memilih menu Data Gejala.	
	2. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Edit*.	

	4. Menampilkan halaman *Form Update* Gejala yang berisi *form update* gejala dengan kolom Kode, Gejala, Bobot, tombol *Submit* dan tombol *Cancel*.
5. Mengubah isi data form *update* gejala.	
6. Klik tombol *Submit*.	
	7. Menyimpan data baru ke database 8. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA GEJALA TOMBOL CANCEL	
6a. Klik tombol *Cancel*.	
	7a. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA GEJALA DATA TIDAK LENGKAP	
6b. Klik tombol *Submit*.	
	7b. Menampilkan *warning* "*This field is required*".
SKENARIO NORMAL MENGELOLA DATA GEJALA TOMBOL HAPUS	

Aktor	Sistem
1. Memilih menu Data Gejala.	
	2. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Delete*.	
	4. Menampilkan *pop up* konfirmasi "Are you sure?" juga tombol OK dan *Cancel*.
5. Klik tombol OK.	
	6. Menghapus data terpilih di database 7. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA GEJALA	
TOMBOL CANCEL	
5a. Klik tombol *Cancel*.	
	6a. Menampilkan halaman data gejala yang berisi tabel gejala *list* dengan kolom No, Kode, Gejala, Bobot, tombol *Create New*, tombol *Edit* dan tombol *Delete*.

A.4 Skenario Mengelola Data Penyakit

Tabel A.4 Tabel Skenario Mengelola Data Penyakit

Nomor *Use Case*	UC-05
Nama	Mengelola Data Penyakit
Aktor	Pakar
Pre Condition	Pakar harus sudah *login* sistem.
Post Condition	Pakar berhasil mengelola data penyakit.

SKENARIO NORMAL MENGELOLA DATA PENYAKIT	
TAMBAH PENYAKIT	
Aktor	Sistem
1. Memilih menu Data Penyakit.	
	2. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Create New*.	
	4. Menampilkan halaman *Form Create* Penyakit yang berisi *form* tambah penyakit dengan kolom Kode, Penyakit, tombol *Submit* dan tombol *Cancel*.
5. Mengisi *Form Create* Penyakit.	
6. Klik tombol *Submit*.	
	7. Menyimpan data baru ke database 8. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode,

	Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA PENYAKIT	
TOMBOL CANCEL PADA TAMBAH PENYAKIT	
6a. Klik tombol *Cancel*.	
	7a. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA PENYAKIT	
DATA TIDAK LENGKAP	
6b. Klik tombol *Submit*.	
	7b. Menampilkan *warning* "*This field is required*".
SKENARIO NORMAL MENGELOLA DATA PENYAKIT	
TOMBOL EDIT	
Aktor	Sistem
1. Memilih menu Data Penyakit.	
	2. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Edit*.	
	4. Menampilkan halaman *Form Update* Penyakit yang berisi *form update* penyakit dengan kolom Kode,

	Penyakit, tombol *Submit* dan tombol *Cancel*.
5. Mengubah isi data form *update* penyakit.	
6. Klik tombol *Submit*.	
	7. Menyimpan data baru ke database 8. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA PENYAKIT	
TOMBOL CANCEL	
6a. Klik tombol *Cancel*.	
	7a. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA PENYAKIT	
DATA TIDAK LENGKAP	
6b. Klik tombol *Submit*.	
	7b. Menampilkan *warning* "*This field is required*".
SKENARIO NORMAL MENGELOLA DATA PENYAKIT	
TOMBOL HAPUS	
Aktor	Sistem
1. Memilih menu Data Penyakit.	

	2. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Delete*.	
	4. Menampilkan *pop up* konfirmasi "Are you sure?" juga tombol OK dan *Cancel*.
5. Klik tombol OK.	
	6. Menghapus data terpilih di database 7. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA PENYAKIT TOMBOL CANCEL	
5a. Klik tombol *Cancel*.	
	6a. Menampilkan halaman data penyakit yang berisi tabel penyakit *list* dengan kolom No, Kode, Penyakit, tombol *Create New*, tombol *Edit* dan tombol *Delete*.

A.5 Skenario Mengelola Data Diagnosa

Tabel A.5 Skenario Mengelola Data Diagnosa

Nomor *Use Case*	UC-06

Nama	Mengelola Data Diagnosa
Aktor	Pakar
Pre Condition	Pakar harus sudah *login* sistem.
Post Condition	Pakar berhasil mengelola data diagnosa.

SKENARIO NORMAL MENGELOLA DATA DIAGNOSA	
TOMBOL REVISE DIAGNOSA	
Aktor	Sistem
1. Memilih menu Data Diagnosa.	
	2. Menampilkan halaman data diagnosa yang berisi tabel kasus baru diagnosa lambung dengan kolom No, Kode, Nama, Usia, Latar Belakang, Hasil Diagnosa, tombol *Revise*, tombol *Detail* dan tombol *Delete*.
3. Klik tombol *Revise*.	
	4. Menampilkan halaman *Form Revise* Basis Kasus yang berisi *form revise* basis kasus dengan kolom Kode, Jenis Penyakit Lambung, Pilih Gejala, Penyebab, Pengobatan, Keterangan, tombol *Submit* dan tombol *Cancel*.
5. Mengisi *Form Revise Basis Kasus*.	
6. Klik tombol *Submit*.	
	7. Menyimpan data baru ke database 8. Menampilkan halaman data basis kasus diagnosa lambung yang berisi

	tabel basis kasus dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan, Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA DIAGNOSA DATA TIDAK LENGKAP	
6a. Klik tombol *Submit*.	
	7a. Menampilkan *warning* "*This field is required*".
SKENARIO ALTERNATIF MENGELOLA DATA DIAGNOSA TOMBOL CANCEL PADA REVISE	
6b. Klik tombol *Cancel*.	
	7b. Menampilkan halaman data diagnosa yang berisi tabel kasus baru diagnosa lambung dengan kolom No, Kode, Nama, Usia, Latar Belakang, Hasil Diagnosa, tombol *Revise*, tombol *Detail* dan tombol *Delete*.
SKENARIO NORMAL MENGELOLA DATA DIAGNOSA TOMBOL DETAIL	
Aktor	Sistem
1. Memilih menu Data Diagnosa.	
	2. Menampilkan halaman data diagnosa yang berisi tabel kasus baru diagnosa lambung dengan kolom No, Kode, Nama, Usia, Latar

	Belakang, Hasil Diagnosa, tombol *Revise*, tombol *Detail* dan tombol *Delete*.
3. Klik tombol *Detail*.	
	4. Menampilkan halaman Hasil Diagnosa yang berisi *form* hasil diagnosa dengan kolom Nama, Usia, Gejala yang Dipilih, Latar Belakang, Hasil Diagnosa, Status, *Created at*, *Updated at*, tombol *Revise* dan tombol *Back*.
SKENARIO ALTERNATIF MENGELOLA DATA DIAGNOSA TOMBOL *REVISE* PADA *DETAIL*	
5a. Klik tombol *Revise*.	
	6a. Menampilkan halaman *Form Revise* Basis Kasus yang berisi *form revise* basis kasus dengan kolom Kode, Jenis Penyakit Lambung, Pilih Gejala, Penyebab, Pengobatan, Keterangan, tombol *Submit* dan tombol *Cancel*.
SKENARIO ALTERNATIF MENGELOLA DATA DIAGNOSA TOMBOL *BACK* PADA *DETAIL*	
5a. Klik tombol *Back*.	
	6a. Menampilkan halaman data diagnosa yang berisi tabel kasus baru diagnosa lambung dengan kolom No, Kode, Nama, Usia, Latar

	Belakang, Hasil Diagnosa, tombol *Revise*, tombol *Detail* dan tombol *Delete*.
SKENARIO NORMAL MENGELOLA DATA DIAGNOSA	
TOMBOL HAPUS	
Aktor	Sistem
1. Memilih menu Data Diagnosa.	
	2. Menampilkan halaman data diagnosa yang berisi tabel kasus baru diagnosa lambung dengan kolom No, Kode, Nama, Usia, Latar Belakang, Hasil Diagnosa, tombol *Revise*, tombol *Detail* dan tombol *Delete*.
3. Klik tombol *Delete*.	
	4. Menampilkan *pop up* konfirmasi "Are you sure?" juga tombol OK dan *Cancel*.
5. Klik tombol OK.	
	6. Menghapus data terpilih di database 7. Menampilkan halaman data diagnosa yang berisi tabel kasus baru diagnosa lambung dengan kolom No, Kode, Nama, Usia, Latar Belakang, Hasil Diagnosa, tombol *Revise*, tombol *Detail* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA DIAGNOSA	

TOMBOL CANCEL	
5a. Klik tombol *Cancel*.	
	6a. Menampilkan halaman data diagnosa yang berisi tabel kasus baru diagnosa lambung dengan kolom No, Kode, Nama, Usia, Latar Belakang, Hasil Diagnosa, tombol *Revise*, tombol *Detail* dan tombol *Delete*.

A.6 Skenario Mengelola Data Basis Kasus

Tabel A.6 Skenario Mengedit Data Basis Kasus

Nomor *Use Case*	UC-07
Nama	Mengelola Data Basis Kasus
Aktor	Pakar
Pre Condition	Pakar harus sudah *login* sistem.
Post Condition	Pakar berhasil mengelola data basis kasus.
SKENARIO NORMAL MENGELOLA DATA BASIS KASUS	
TOMBOL *DETAIL* DIAGNOSA	
Aktor	Sistem
1. Memilih menu Basis Kasus.	
	2. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan,

	Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Detail*.	
	4. Menampilkan halaman *detail* kasus yang berisi *detail* kasus dengan kolom User Input, Kode Kasus, Jenis Penyakit, Gejala, Pengobatan, Penyebab, Keterangan, Created at, Updated at dan tombol *Back*.
SKENARIO ALTERNATIF MENGELOLA DATA BASIS KASUS	
TOMBOL BACK	
5a. Klik tombol *Back*.	
	6a. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan, Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.
SKENARIO NORMAL MENGELOLA DATA BASIS KASUS	
TOMBOL EDIT	
Aktor	Sistem
1. Memilih menu Basis Kasus.	
	2. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan,

	Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Edit*.	
	4. Menampilkan halaman *form update* basis kasus yang berisi *form* basis kasus dengan kolom Kode, Jenis Penyakit Lambung, Pilih Gejala, Penyebab, Pengobatan, Keterangan, tombol *Submit* dan tombol *Cancel*.
5. Mengubah isi data *form update* basis kasus 6. Klik tombol *Submit*.	
	7. Menyimpan data baru pada database 8. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan, Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGELOLA DATA BASIS KASUS DATA TIDAK LENGKAP	
6a. Klik tombol *Submit*.	
	6a. Menampilkan *warning* "*This field is required*".
SKENARIO ALTERNATIF MENGELOLA DATA BASIS KASUS TOMBOL *CANCEL* PADA *EDIT*	
6b. Klik tombol *Cancel*.	

	6a. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan, Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.

A.7 Skenario Menghapus Data Basis Kasus

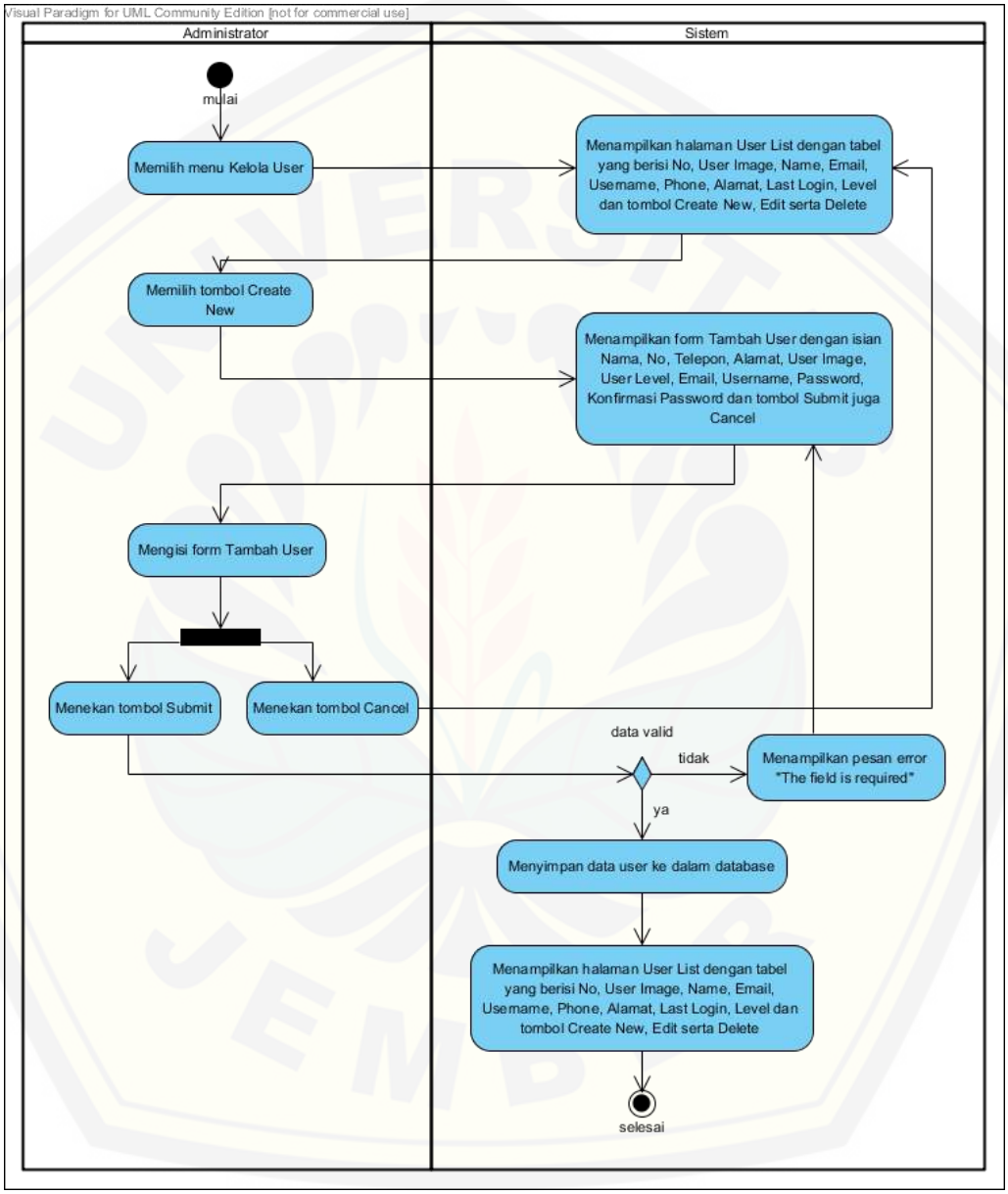
Tabel A.7 Skenario Menghapus Data Basis Kasus

Nomor *Use Case*	UC-07
Nama	Menghapus Data Basis Kasus
Aktor	Pakar
Pre Condition	Pakar harus sudah *login* sistem.
Post Condition	Pakar berhasil menghapus data basis kasus.
SKENARIO NORMAL MENGHAPUS DATA BASIS KASUS	
TOMBOL HAPUS	
Aktor	Sistem
1. Memilih menu Basis Kasus.	
	2. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan, Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.
3. Klik tombol *Delete*.	

	4. Menampilkan *pop up* konfirmasi "Are you sure?" juga tombol OK dan *Cancel*.
5. Klik tombol OK.	
	6. Menghapus data terpilih di database 7. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan, Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.
SKENARIO ALTERNATIF MENGHAPUS DATA BASIS KASUS TOMBOL CANCEL	
5a. Klik tombol *Cancel*.	
	6a. Menampilkan halaman data basis kasus diagnosa lambung yang berisi tabel basis kasus diagnosa lambung dengan kolom No, Kode, Pengobatan, Penyebab, Keterangan, Penyakit, tombol *Detail*, tombol *Edit* dan tombol *Delete*.

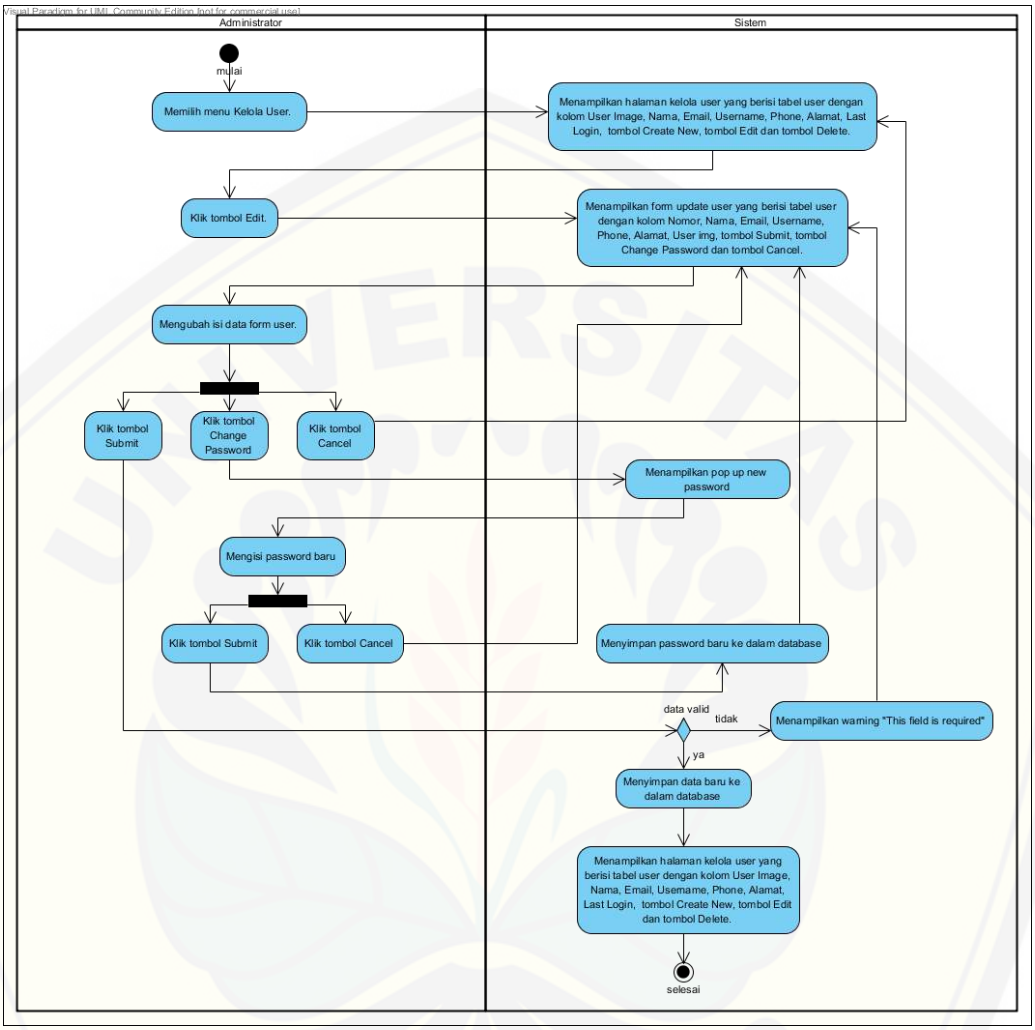
Lampiran B. ***Activity Diagram***

B.1 *Activity diagram* menambah data user



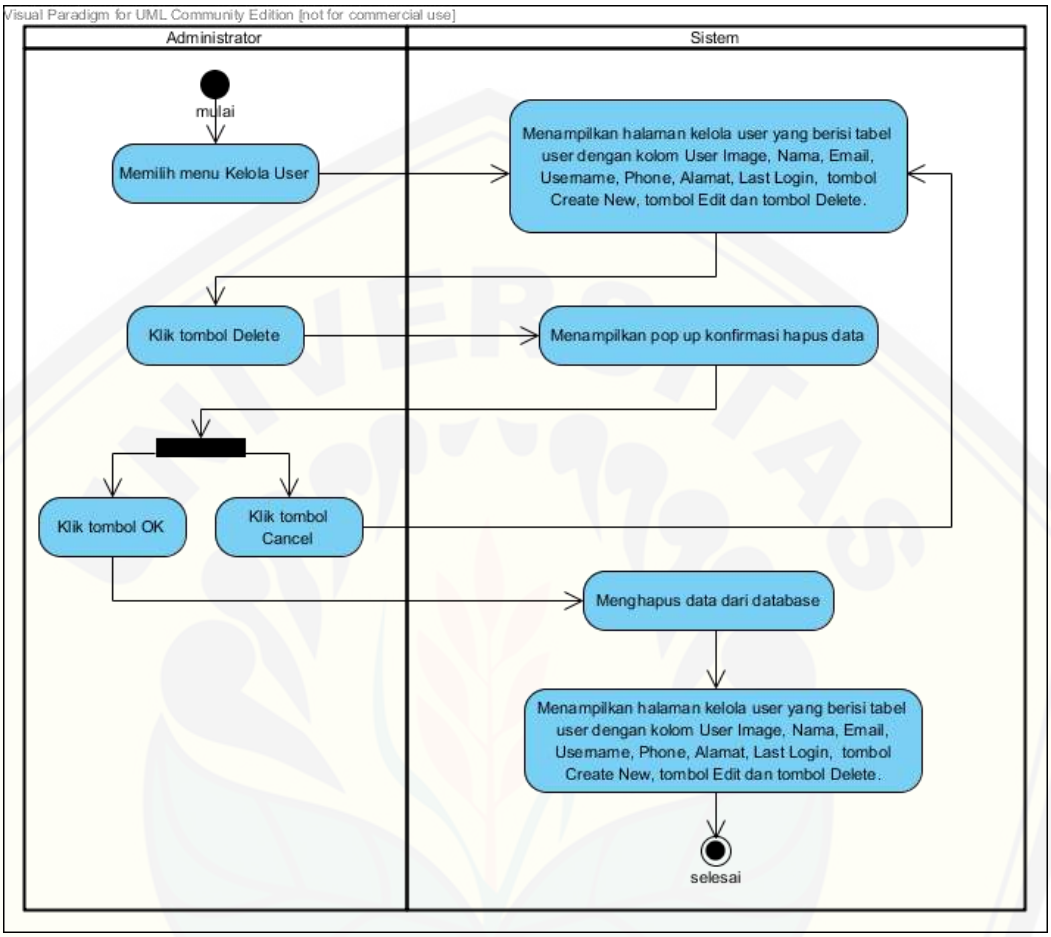
Gambar B.1 *Activity diagram* menambah data *user*

B.2 *Activity diagram* mengedit data user



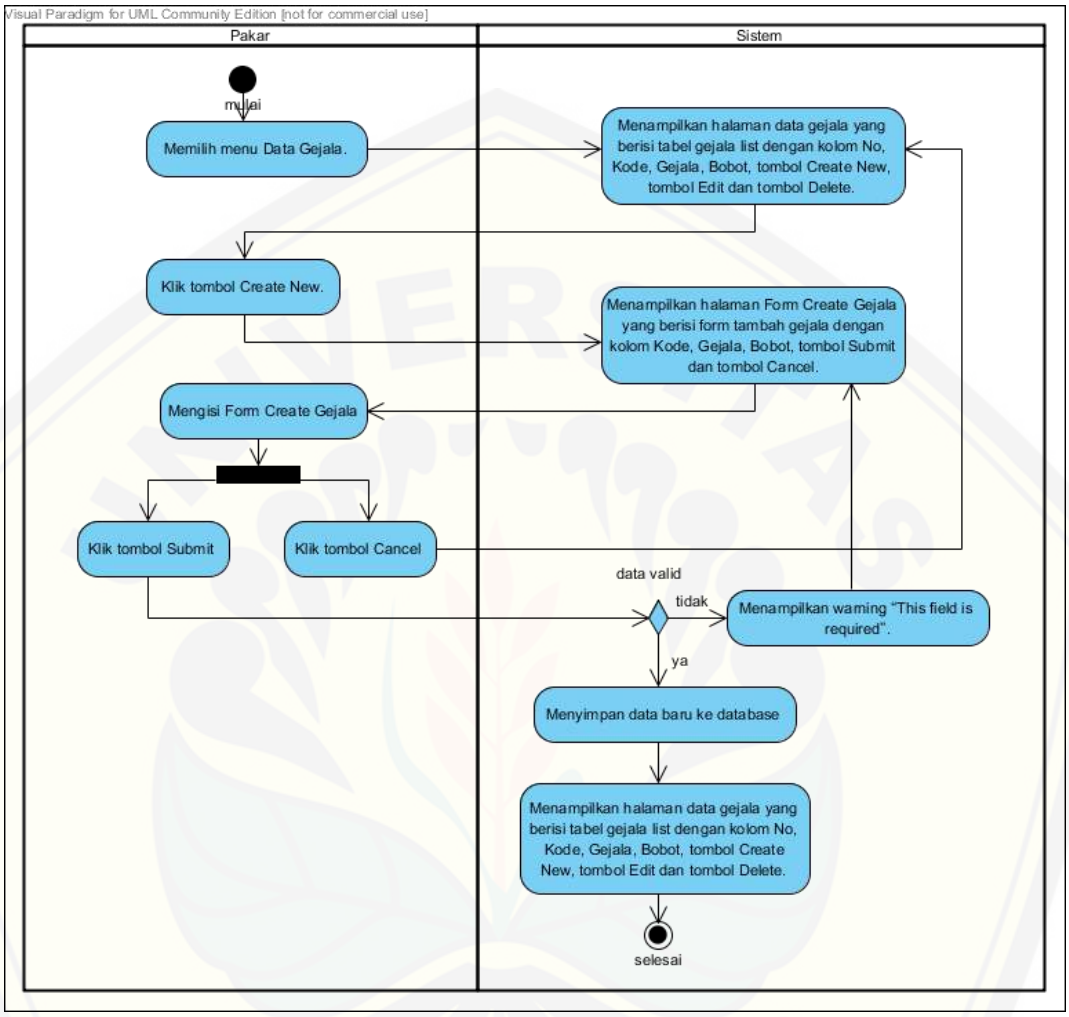
Gambar B.2 *Activity diagram* mengedit data user

B.3 *Activity diagram* menhapus data user



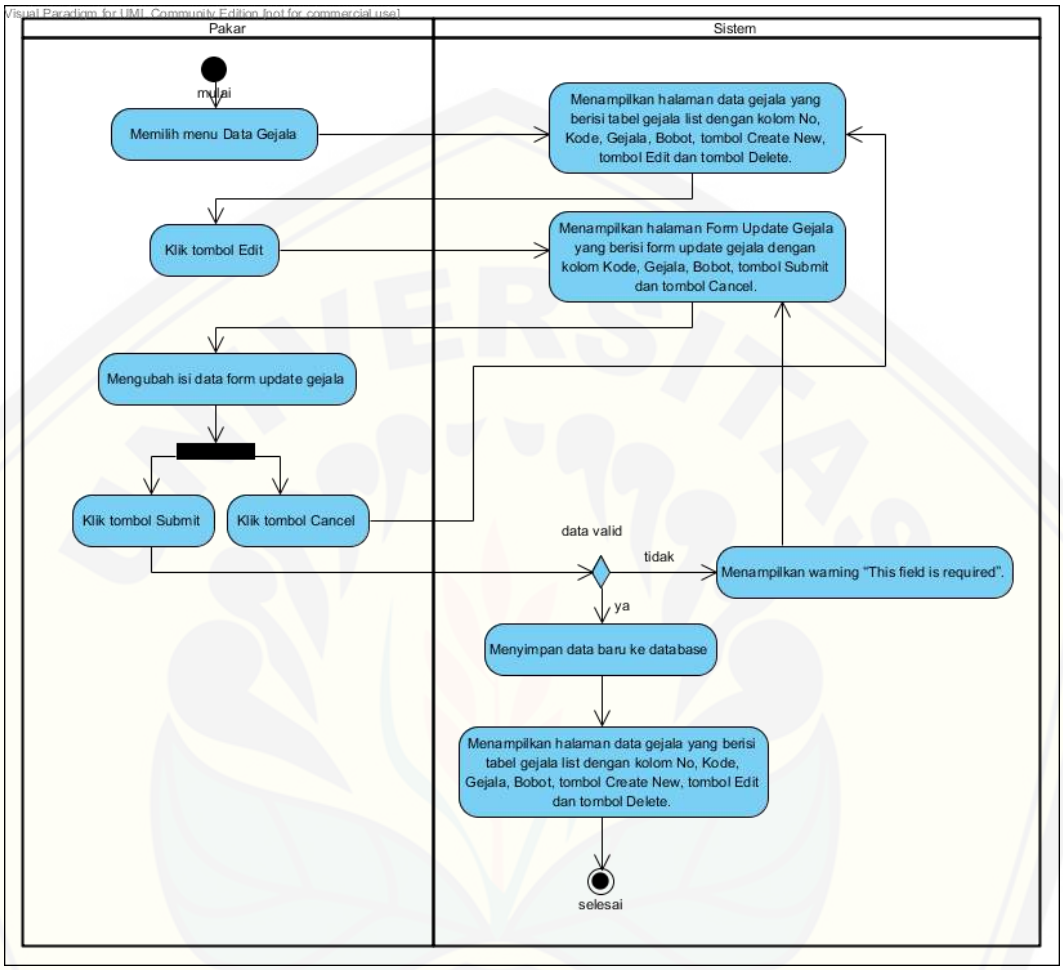
Gambar B.3 *Activity diagram* menhapus data user

B.4 *Activity diagram* menambah data gejala



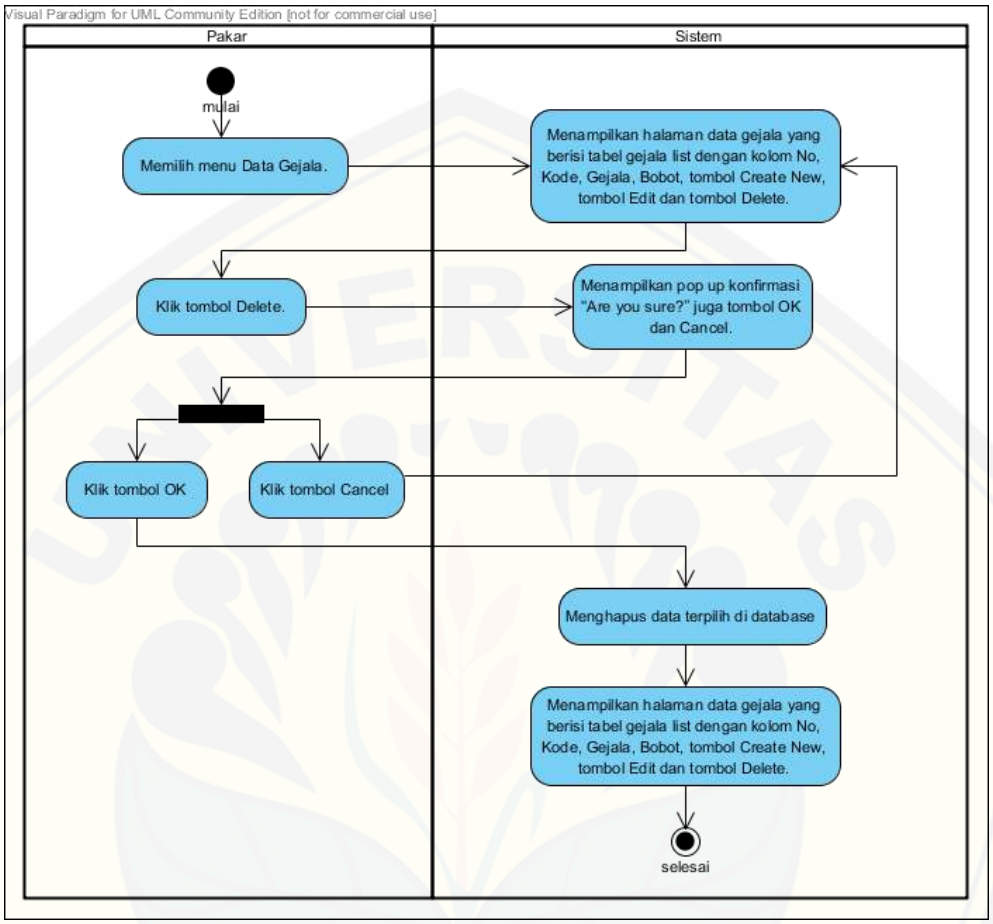
Gambar B.4 *Activity diagram* menambah data gejala

B.5 *Activity diagram* mengedit data gejala



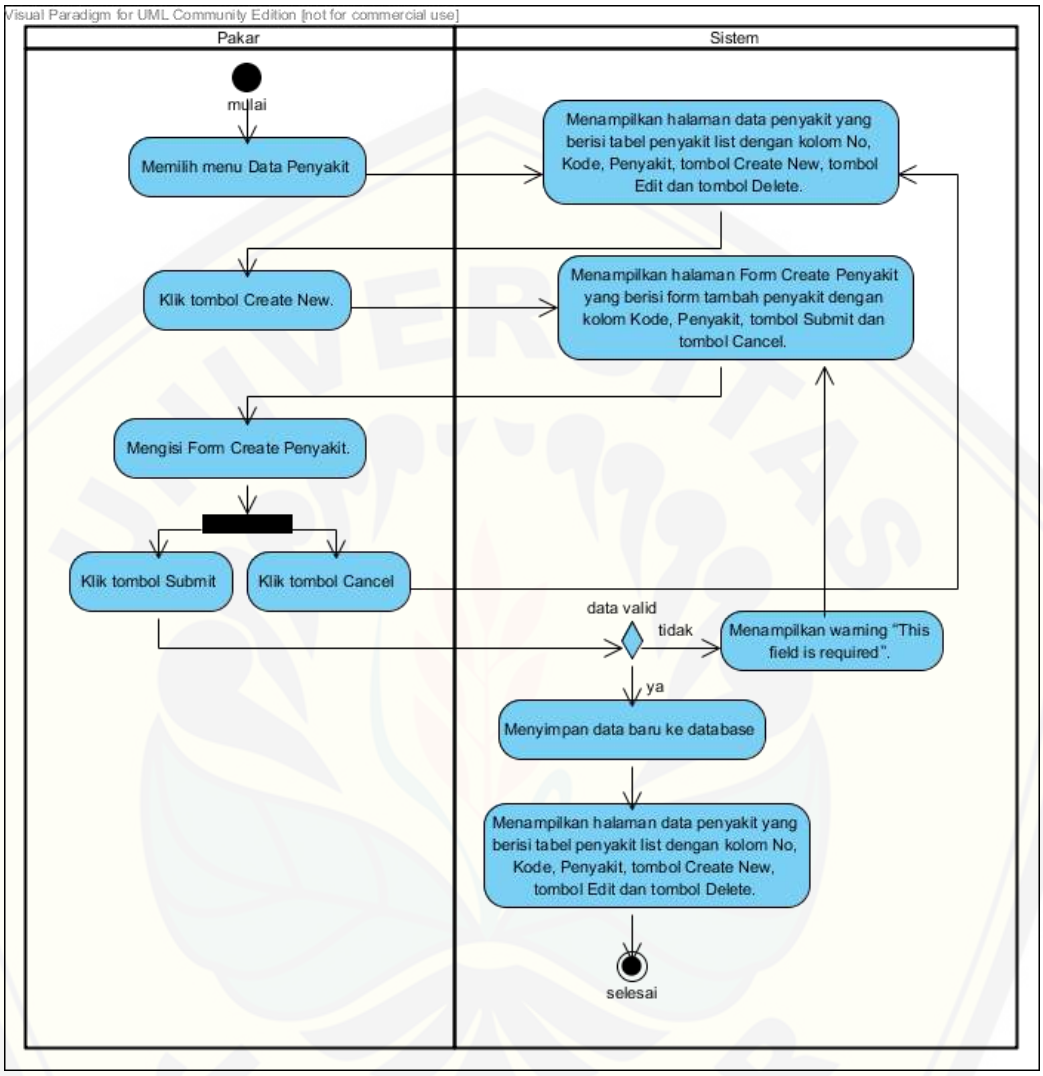
Gambar B.5 *Activity diagram* mengedit data gejala

B.6 *Activity diagram* menghapus data gejala



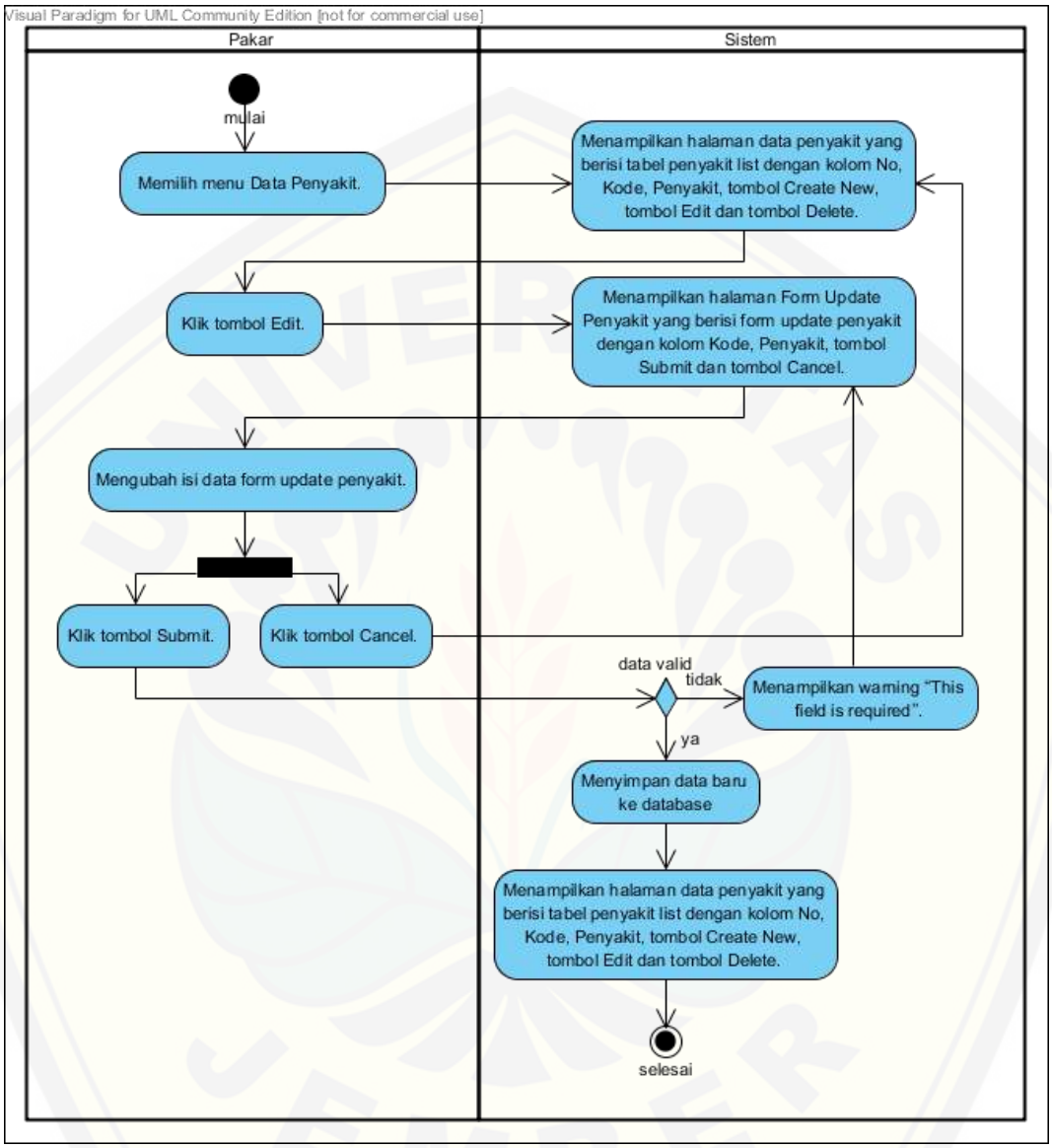
Gambar B.6 *Activity diagram* menghapus data gejala

B.7 *Activity diagram* menambah data penyakit



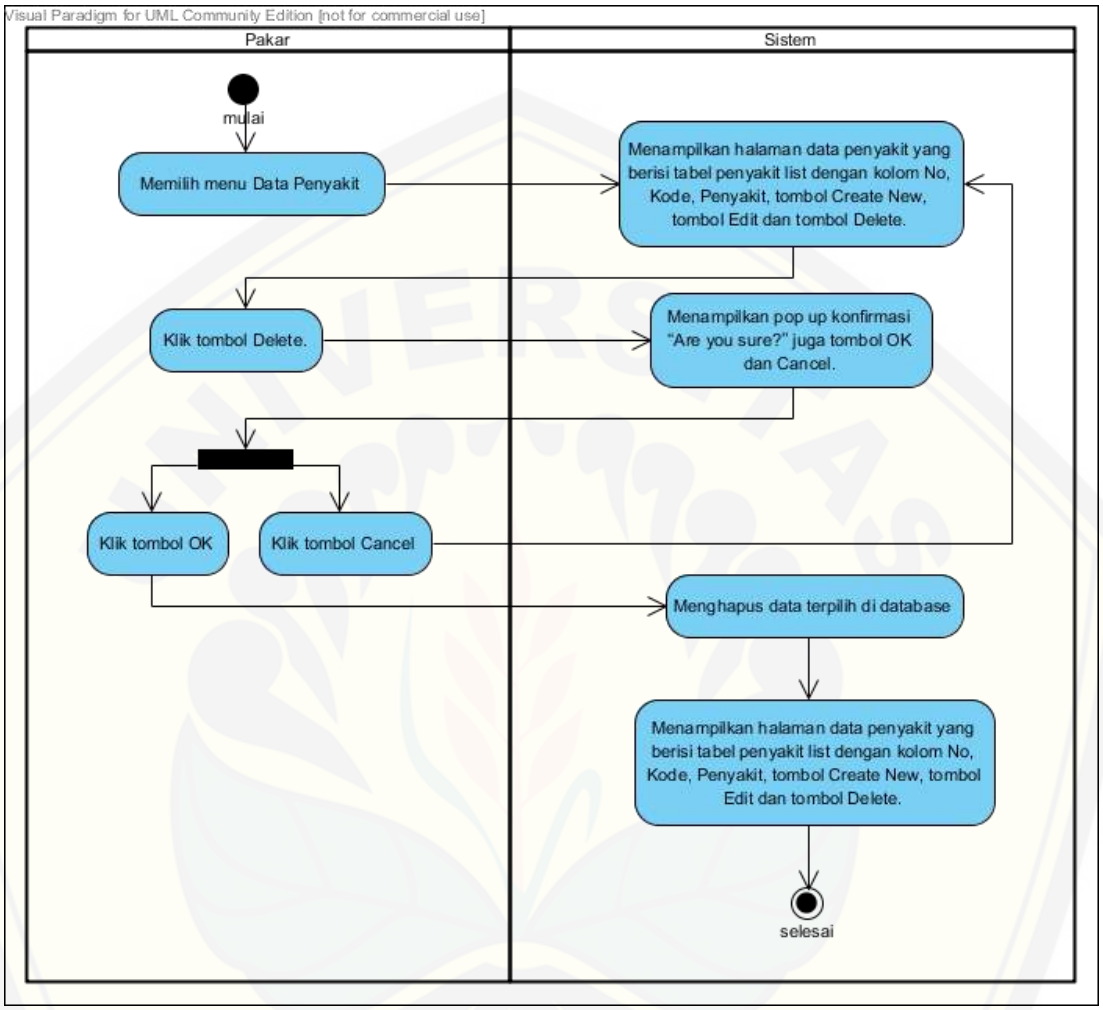
Gambar B.7 *Activity diagram* menambah data penyakit

B.8 *Activity diagram* mengedit data penyakit



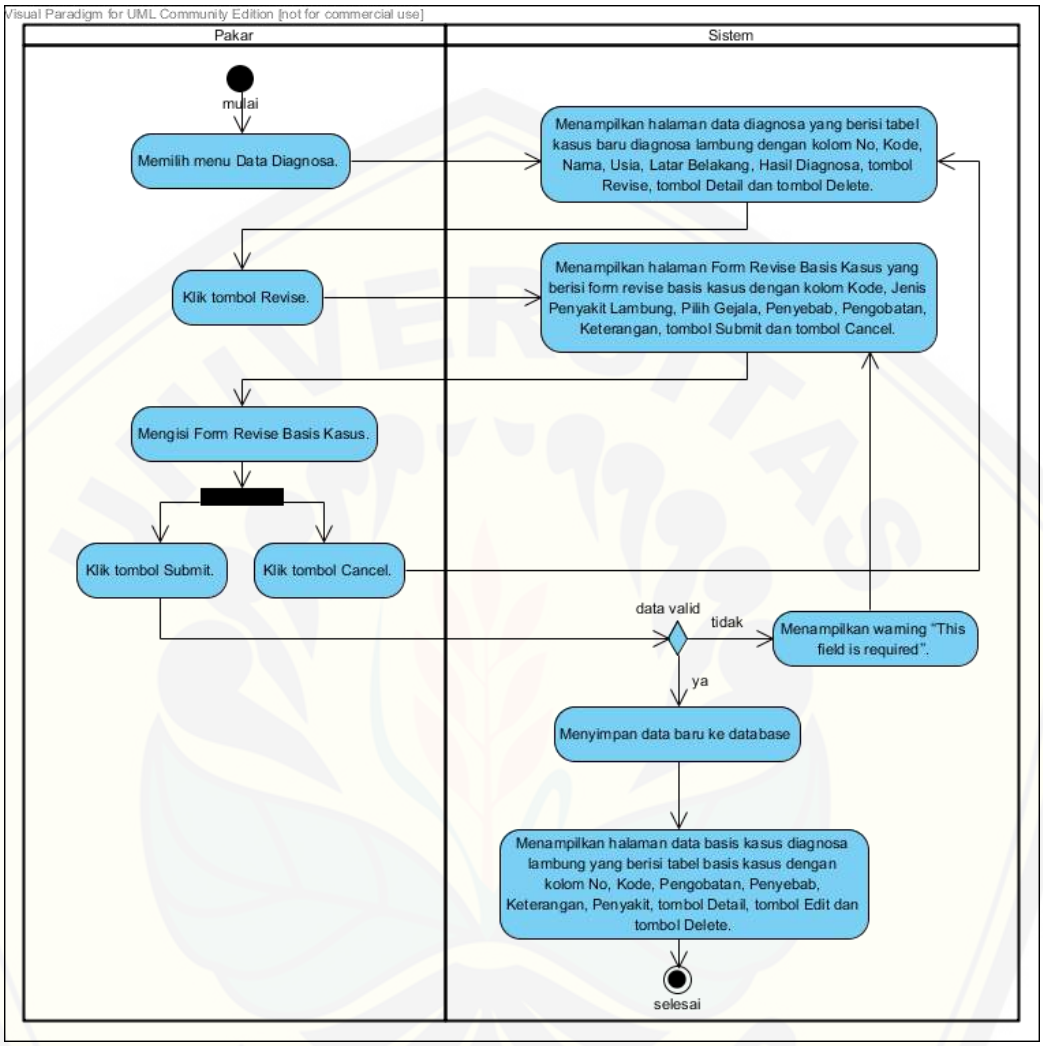
Gambar B.8 *Activity diagram* mengedit data penyakit

B.9 *Activity diagram* menghapus data penyakit



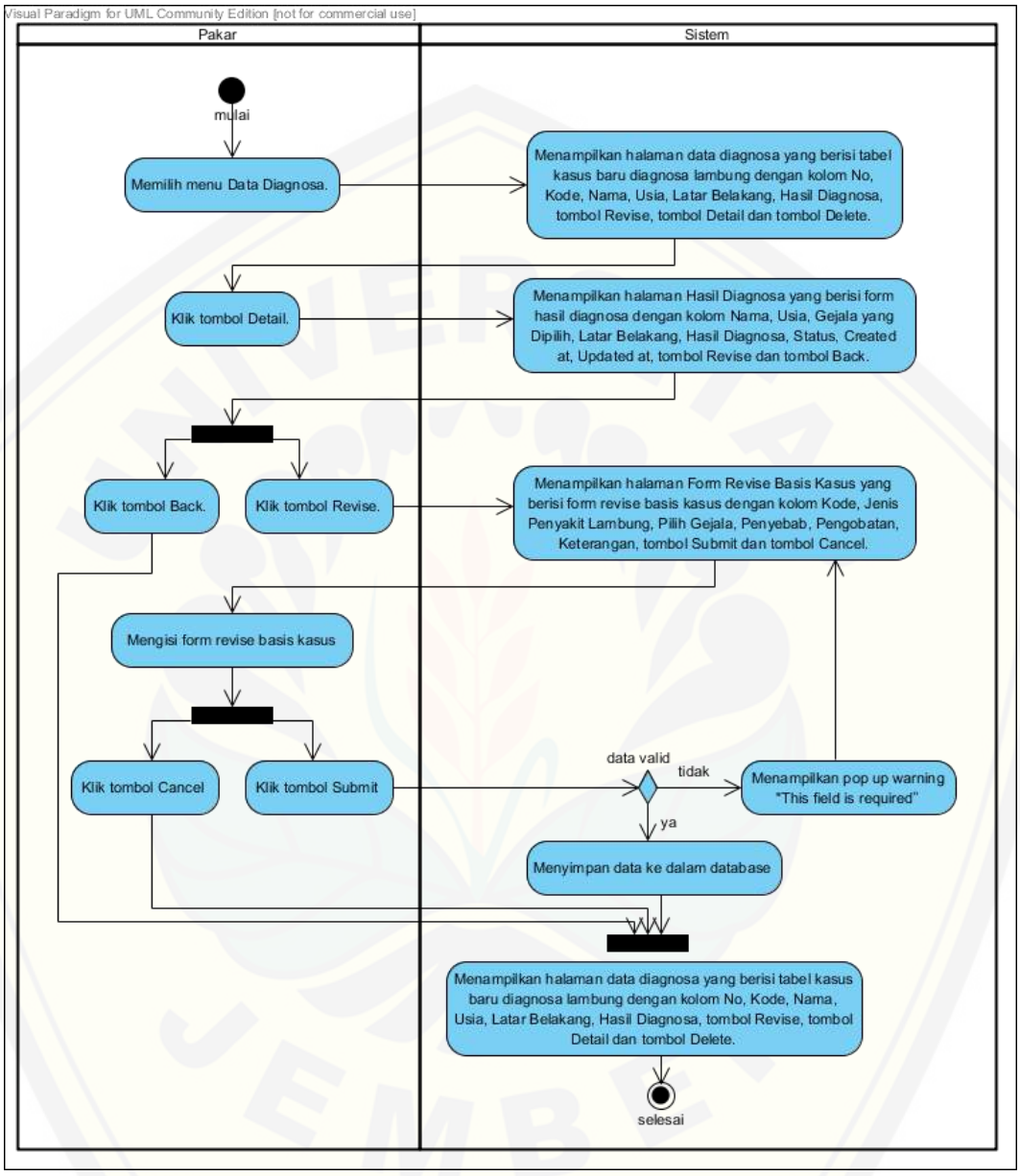
Gambar B.9 *Activity diagram* menghapus data penyakit

B.10 *Activity diagram* melakukan *revise* data diagnosa



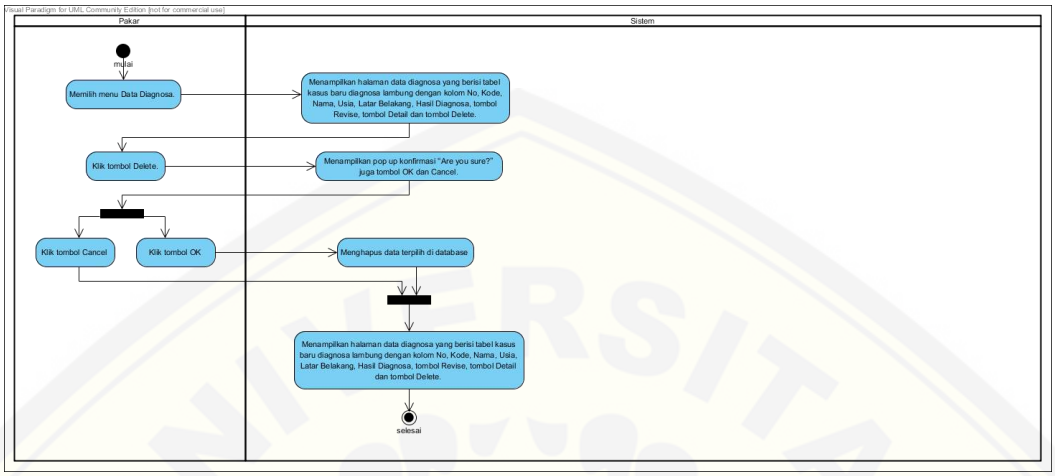
Gambar B.10 *tivity diagram* melakukan *revise* data diagnosa

B.11 *Activity diagram* melihat detail data diagnosa



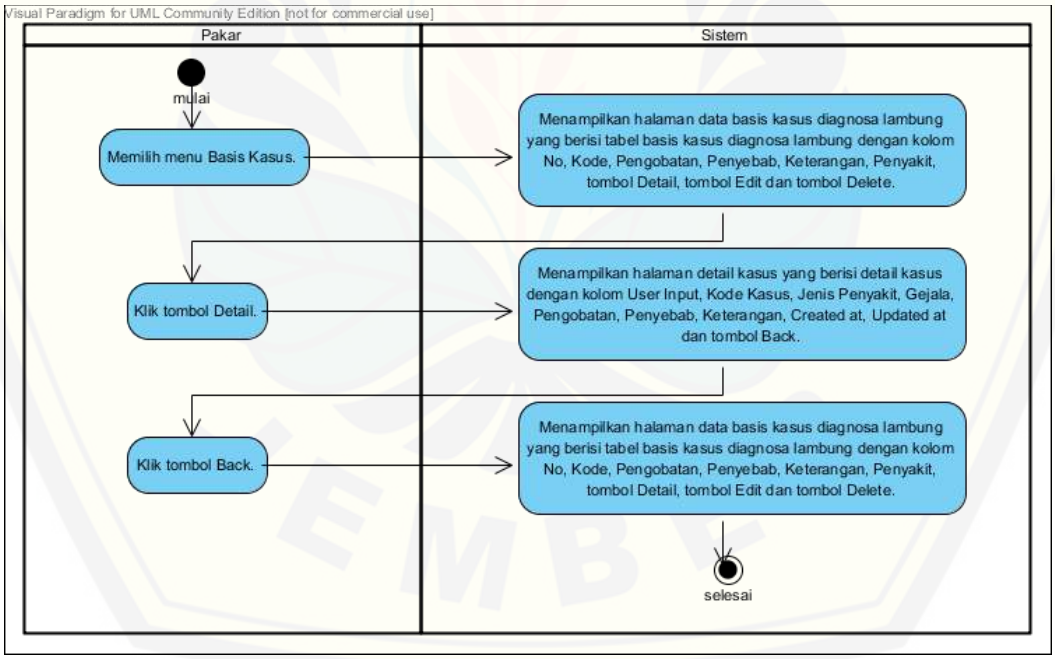
Gambar B.11 *Activity diagram* melihat detail data diagnosa

B.12 *Activity diagram* menghapus data diagnosa



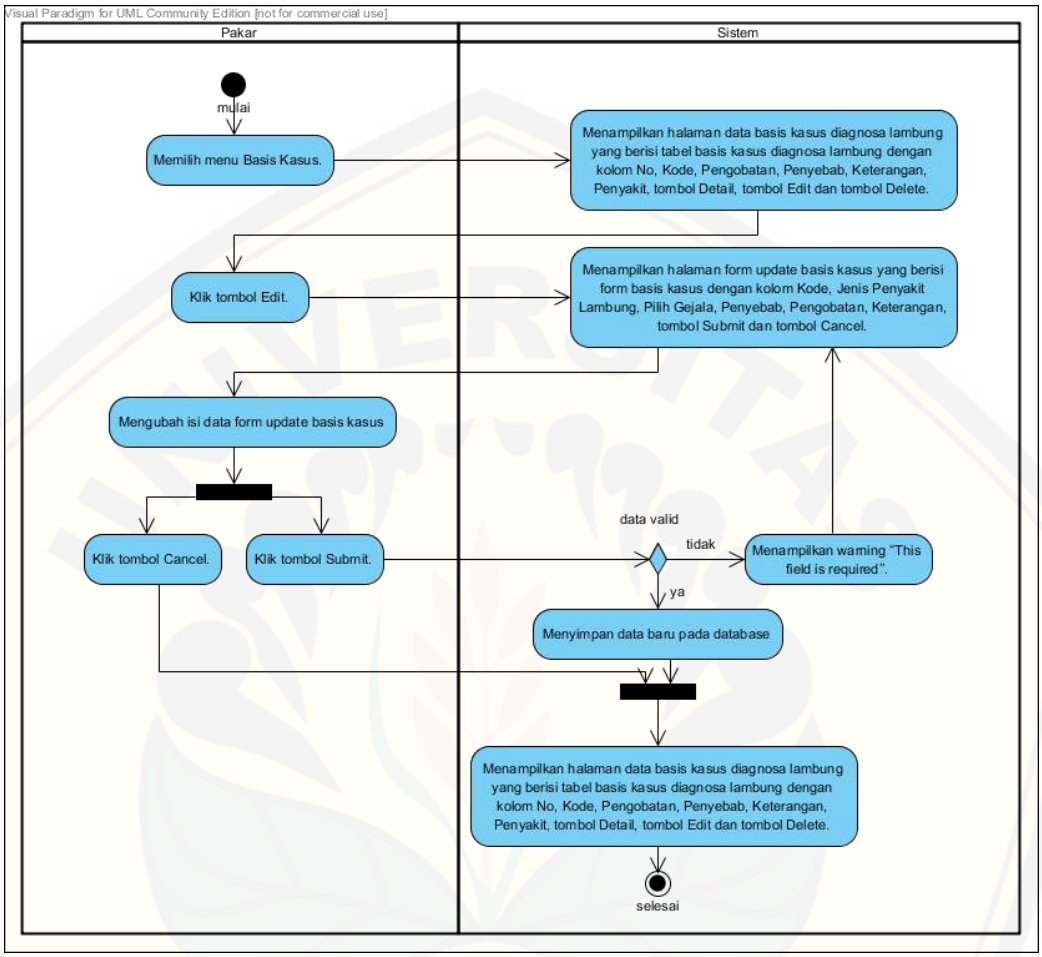
Gambar B.12 *Activity diagram* menghapus data diagnosa

B.13 *Activity diagram* melihat detail data basis kasus



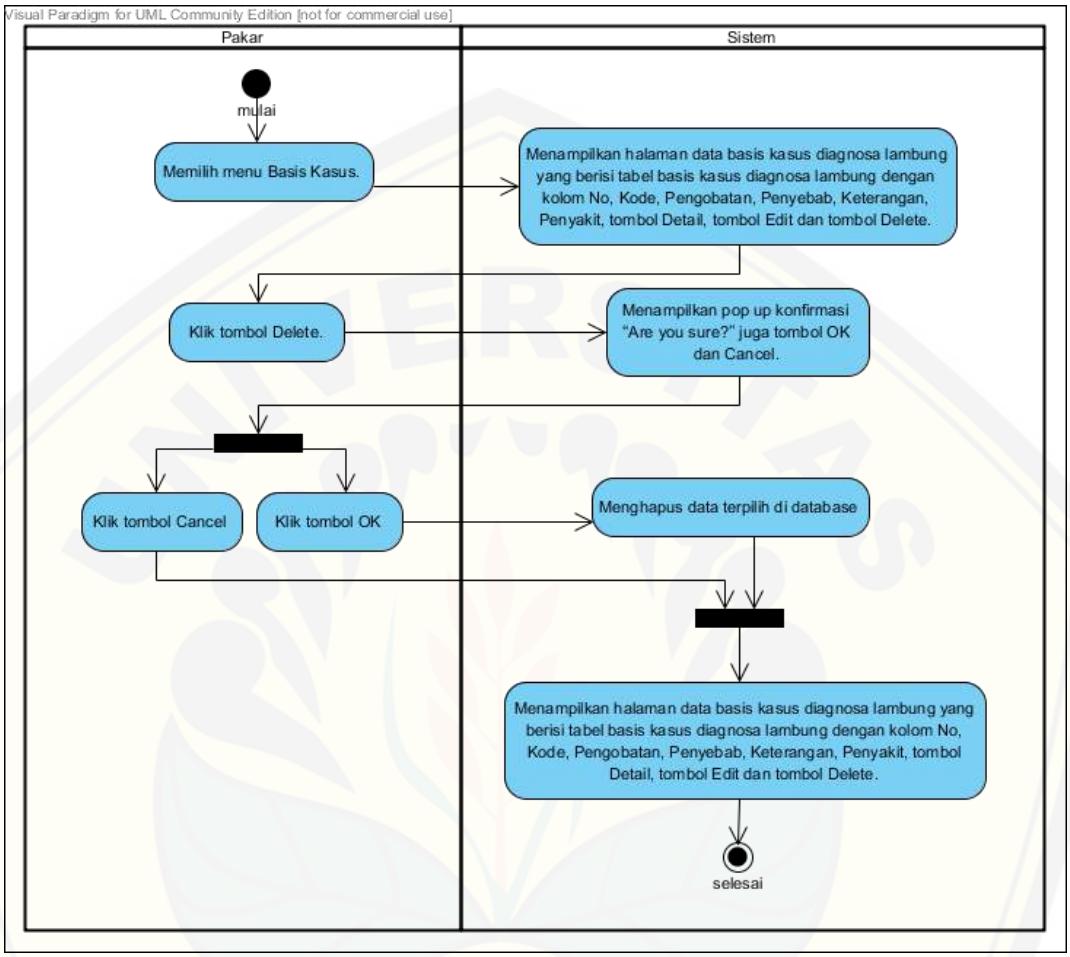
Gambar B.13 *Activity diagram* melihat detail data basis kasus

B.14 *Activity diagram* mengedit data basis kasus



Gambar B.14 *Activity diagram* mengedit data basis kasus

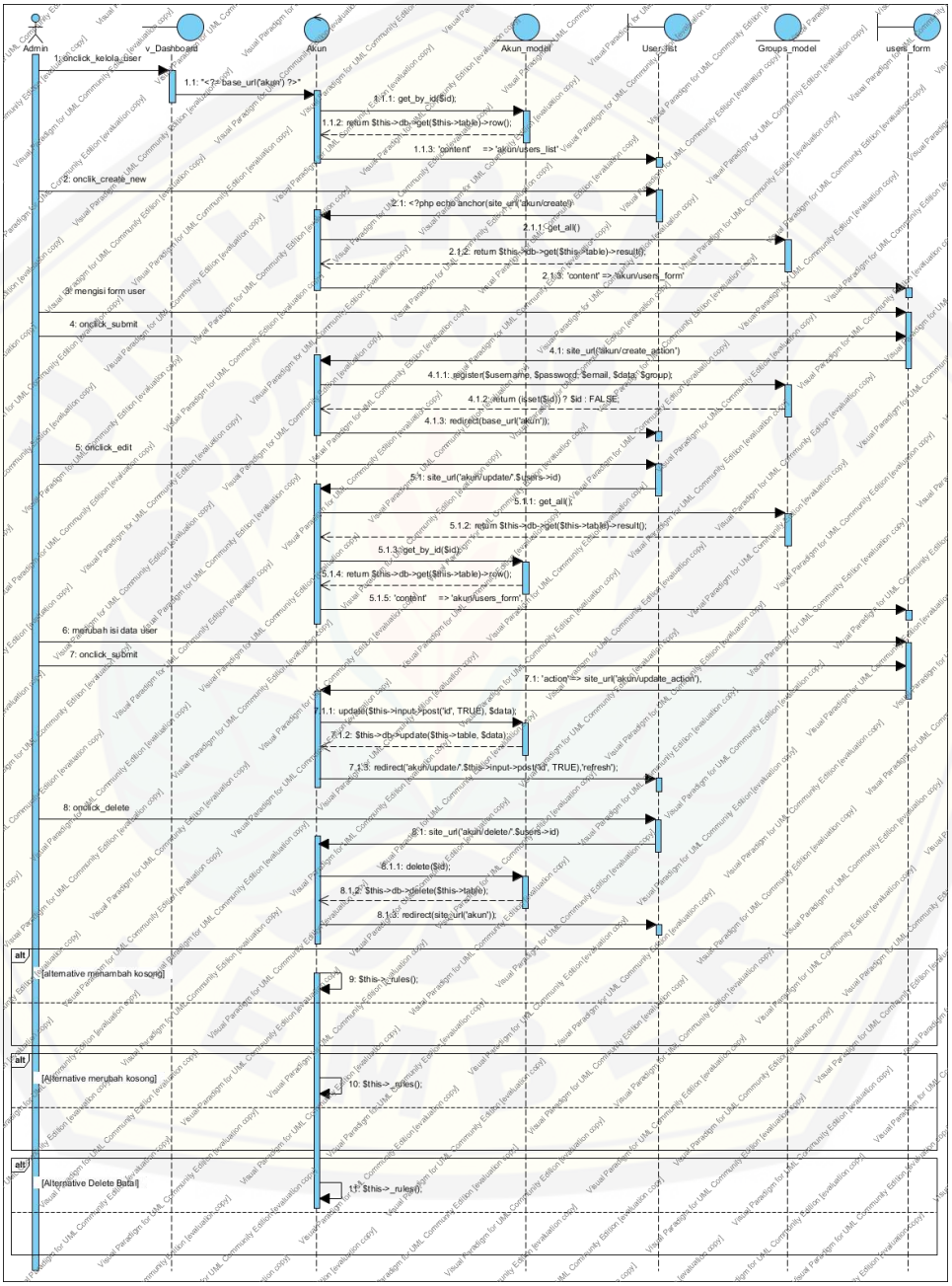
B.15 *Activity diagram* menghapus data basis kasus



Gambar B.15 *Activity diagram* menghapus data basis kasus

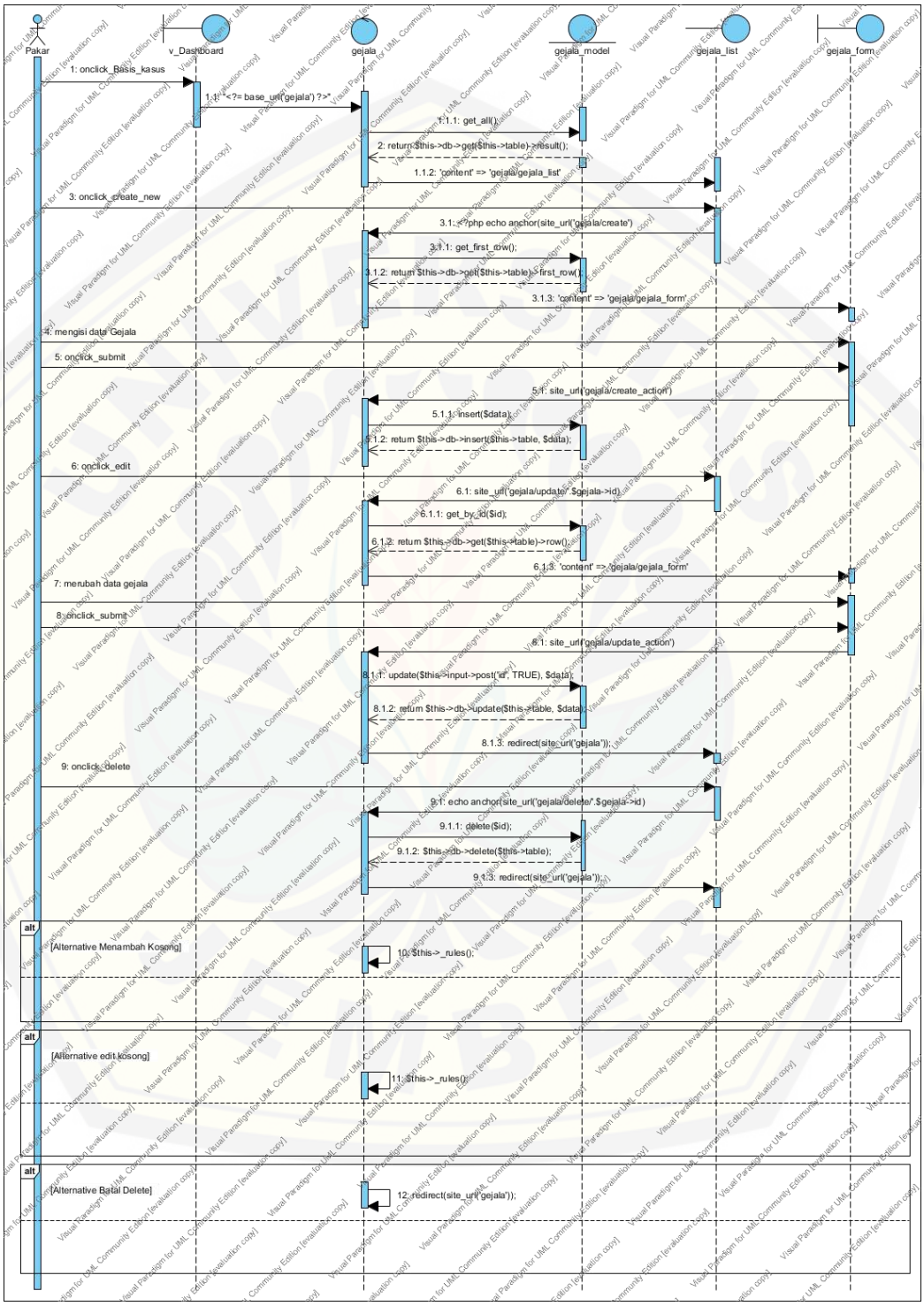
Lampiran C. *Sequence Diagram*

C.1 *Sequence diagram* mengelola data *user*



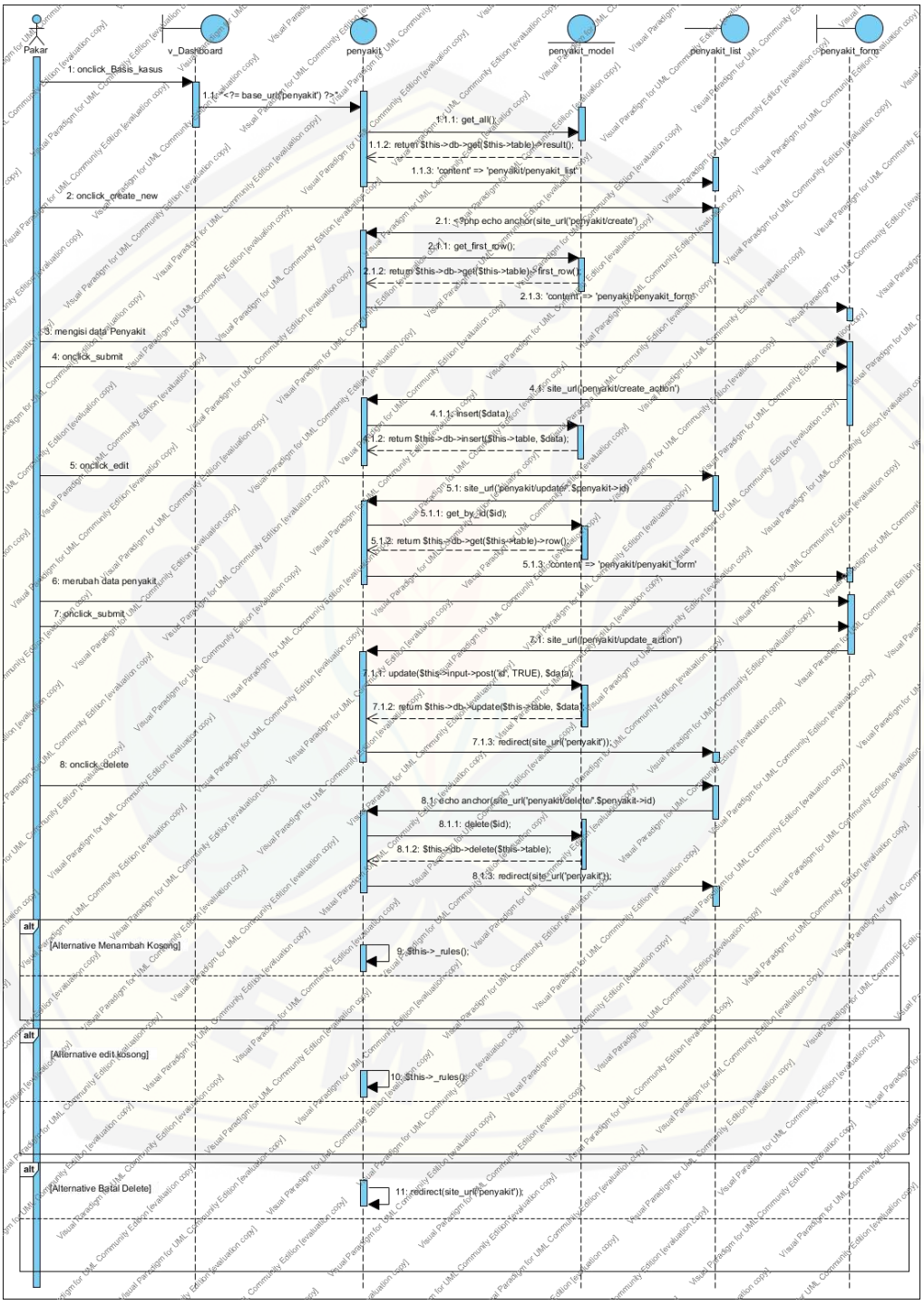
Gambar C.1 *Sequence diagram* mengelola data *user*

C.2 *Sequence diagram* mengelola data gejala



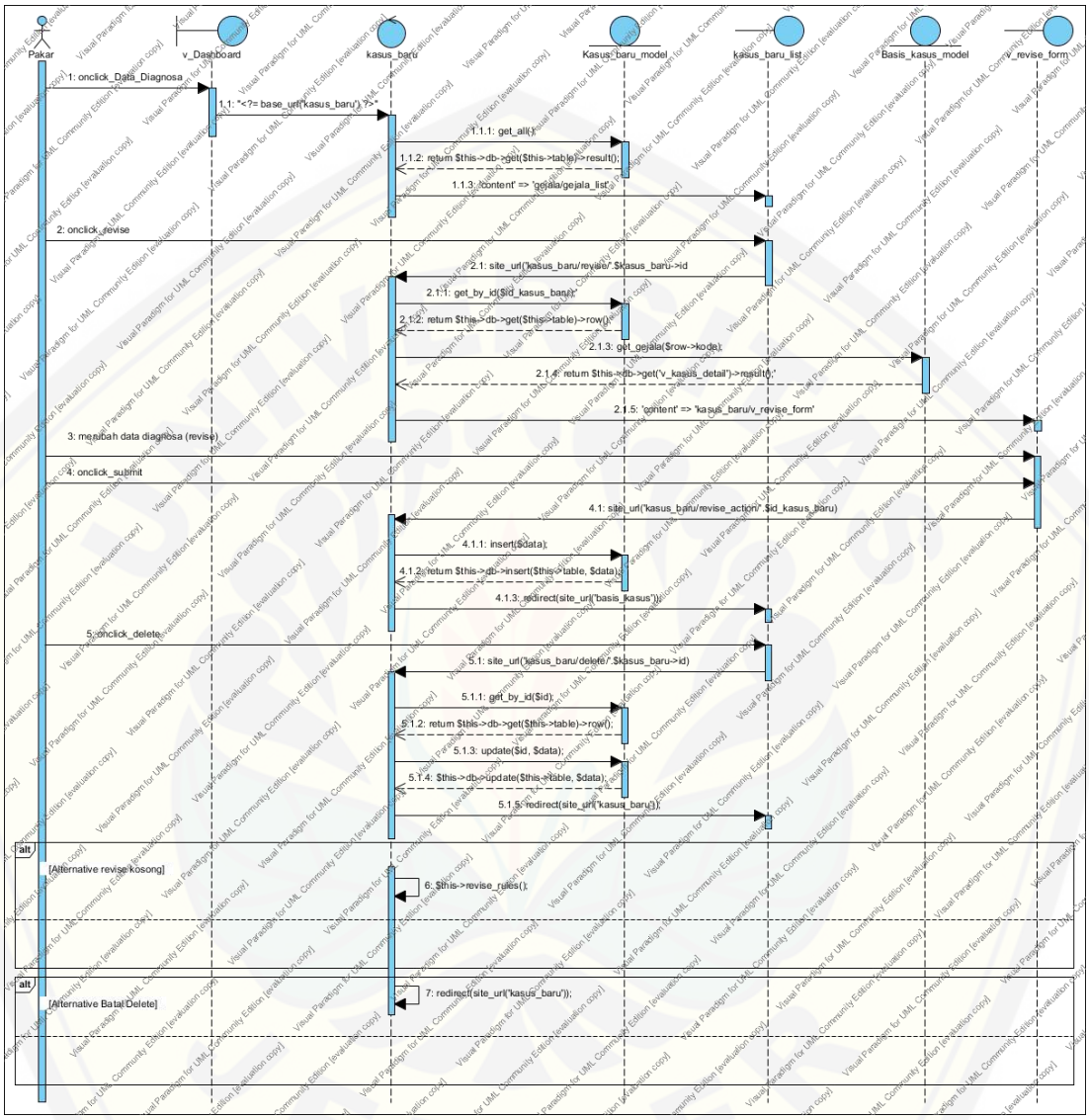
Gambar C.2 *Sequence diagram* mengelola data gejala

C.3 *Sequence diagram* mengelola data penyakit



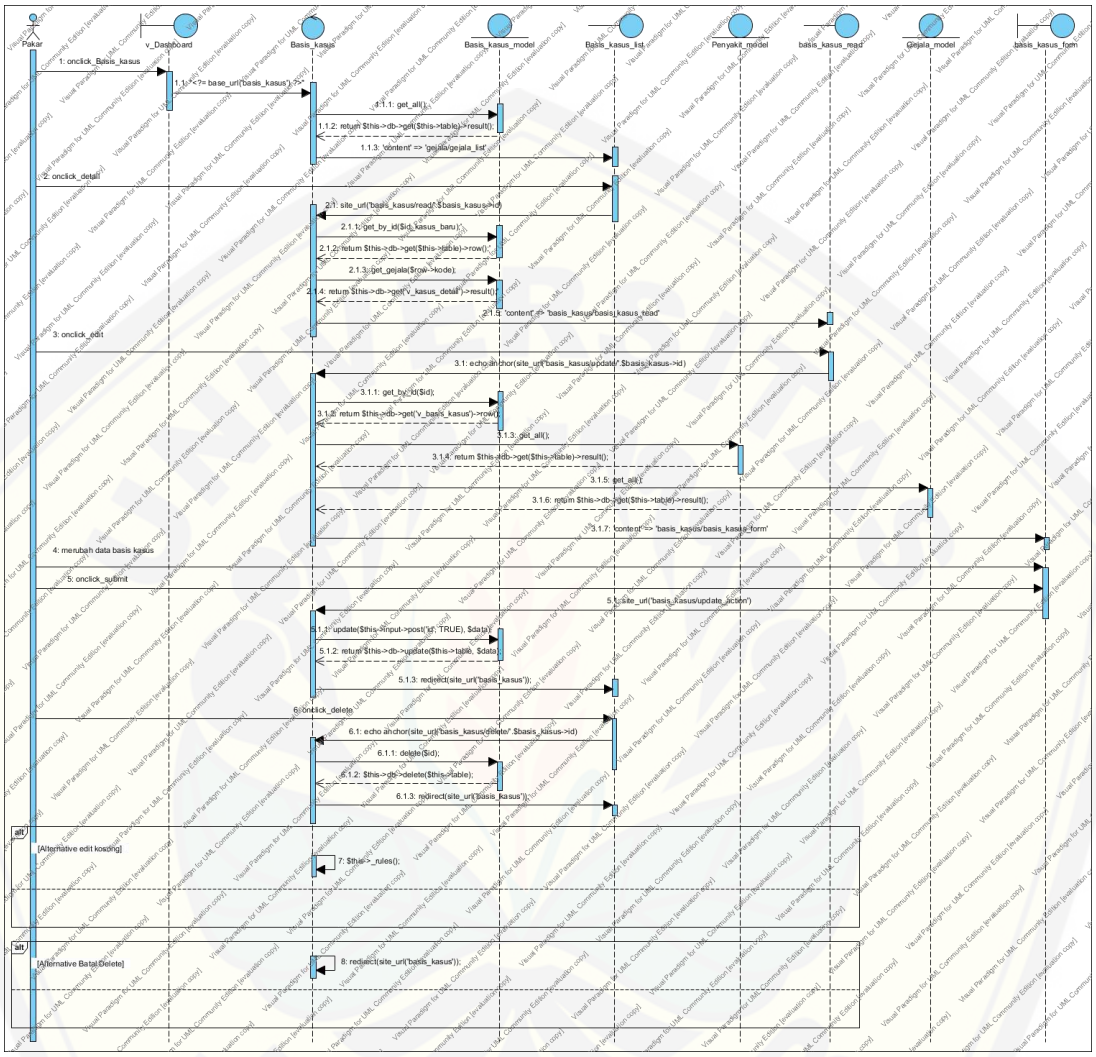
Gambar C.3 *Sequence diagram* mengelola data penyakit

C.3 *Sequence diagram* mengelola data diagnosa



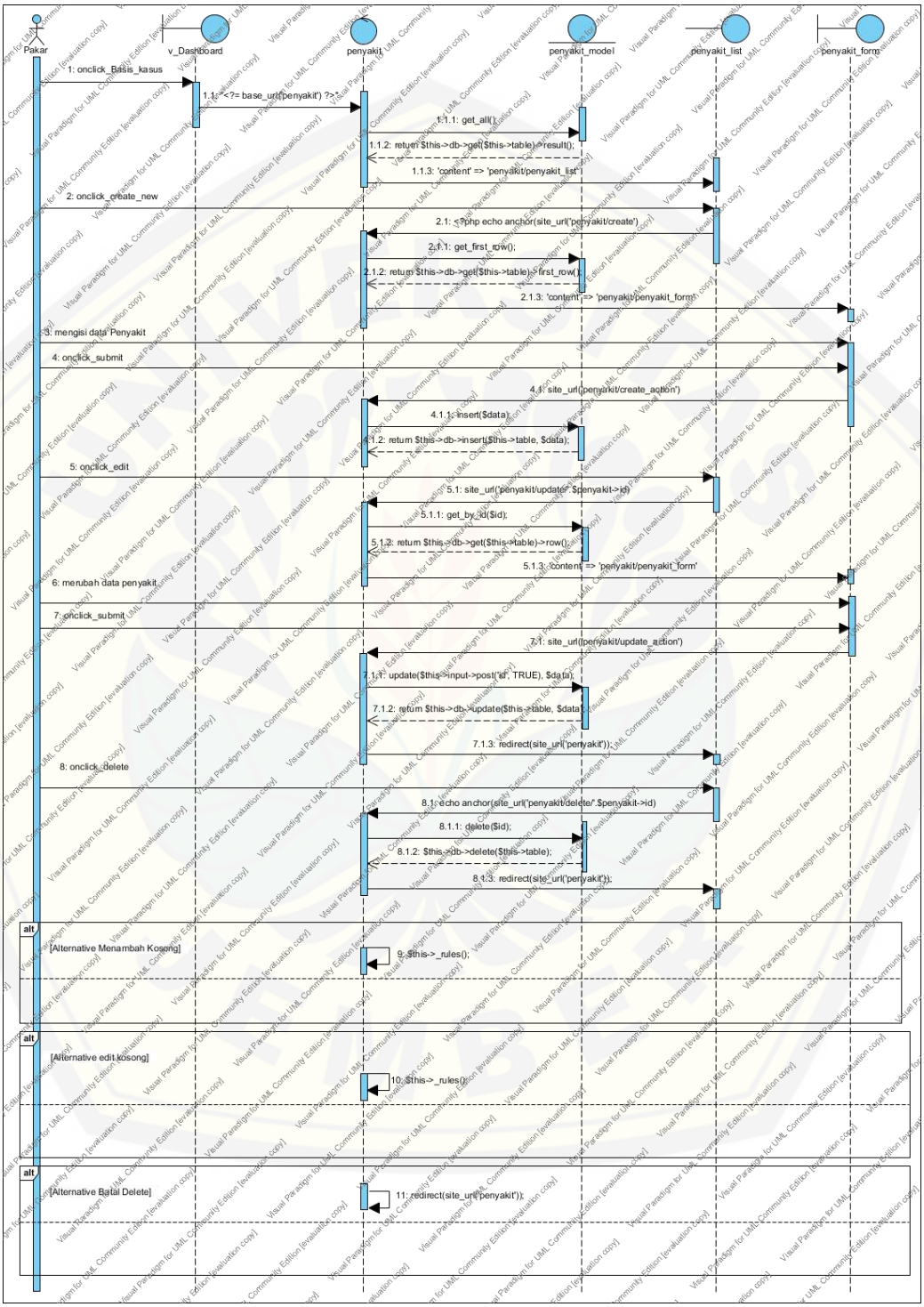
Gambar C.4 *Sequence diagram* mengelola data diagnosa

C.4 *Sequence diagram* mengelola data basis kasus



Gambar C.5 *Sequence diagram* mengelola data basis kasus

C.5 *Sequence diagram* mengelola data penyakit



Gambar C.6 *Sequence diagram* mengelola data penyakit

Lampiran D. Implementasi *coding*

D.1 Mengelola data user

D.1.1 *Controller* Akun (Akun.php)

```php
<?php

if (!defined('BASEPATH')) exit('No direct script access allowed');

class Akun extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
        $this->load->model('Akun_model');
        $this->load->model('Groups_model');
        if (!$this->ion_auth->logged_in())
        {
            redirect('auth/login', 'refresh');
        }
    }

    public function index()
    {
        $users = $this->ion_auth->users()->result();
        $user = $this->ion_auth->user()->row();

        $data = array(
            'content'     => 'akun/users_list',
            'js_script' => 'akun/users_js_script',
            'user'        => $user ,
            'users_data' => $users
        );

        $this->load->view('layout/template', $data);
    }

    public function read($id)
    {
        $user = $this->ion_auth->user()->row();
        $this->breadcrumbs->push('Users', '/users');
        $row = $this->Akun_model->get_by_id($id);
        if ($row) {
            $data = array(
                'content'     => 'akun/users_read',
                'user'        => $user ,
                'js_script' => 'akun/_js_script',
                'id' => $row->id,
                'name' => $row->name,
                'email' => $row->email,
                'username' => $row->username,
                'password' => $row->password,
                'phone' => $row->phone,
                'alamat' => $row->alamat,
                'user_img' => $row->user_img,
                'ip_address' => $row->ip_address,
                'last_login' => $row->last_login,
                'salt' => $row->salt,
                'activation_code' => $row->activation_code,
                'forgotten_password_code' => $row->forgotten_password_code,
                'forgotten_password_time' => $row->forgotten_password_time,
                'remember_code' => $row->remember_code,
                'active' => $row->active,
                'created_on' => $row->created_on,
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('akun'));
        }
    }

    public function create()
    {
        $grup = $this->Groups_model->get_all();
        $user = $this->ion_auth->user()->row();
        $data = array(
            'content'     => 'akun/users_form',
            'js_script' => 'akun/users_js_script',
            'user'        => $user ,
            'grup'        => $grup ,
            'groupss'    => set_value('groupss') ,
            'button' => 'Tambah',
            'action' => site_url('akun/create_action'),
            'id' => set_value('id'),
```

```
            'name' => set_value('name'),
            'email' => set_value('email'),
            'username' => set_value('username'),
            'password' => set_value('password'),
            'phone' => set_value('phone'),
            'alamat' => set_value('alamat'),
            'user_img' => set_value('user_img'),
        );
        $this->load->view('layout/template', $data);
    }

    public function create_action()
    {
        $this->form_validation->set_rules('name', 'name', 'trim|required');
        $this->form_validation->set_rules('email', 'email', 'trim|required|valid_emails|valid_email|is_unique[users.email]');
        $this->form_validation->set_rules('username', 'username', 'trim|required|is_unique[users.username]');
        $this->form_validation->set_rules('password', 'password', 'trim|required');
        $this->form_validation->set_rules('password2', 'Konfirmasi Password', 'trim|matches[password]');
        $this->form_validation->set_rules('phone', 'phone', 'trim|required');
        $this->form_validation->set_rules('alamat', 'alamat', 'trim|required');
        $this->form_validation->set_rules('user_img', 'User Image', 'callback_image_upload');
        $this->form_validation->set_rules('group_id', 'Level User', 'trim|required');
        $this->form_validation->set_rules('id', 'id', 'trim');
        $this->form_validation->set_error_delimiters('<span class="text-danger">', '</span>');

        if ($this->form_validation->run() == FALSE) {
            $this->create();
        } else {
            $CI =& get_instance();
            $upload_data = $CI->upload->data();
            $img = $upload_data['file_name'];
            $data = array(
                'nama' => $this->input->post('name',TRUE),
                'phone' => $this->input->post('phone',TRUE),
                'alamat' => $this->input->post('alamat',TRUE),
                'user_img' => $img,
            );

            $username = $this->input->post('username',TRUE);
            $password = $this->input->post('password',TRUE);
            $email = $this->input->post('email',TRUE);
            $group = array($this->input->post('group_id'));

            $this->ion_auth->register($username, $password, $email, $data, $group);

            $this->session->set_flashdata('message', 'Create Record Success');
            redirect(base_url('akun'));
        }
    }

    public function update($id)
    {
        $grup = $this->Groups_model->get_all();
        $user = $this->ion_auth->user()->row();
        $row = $this->Akun_model->get_by_id($id);
        if ($row) {
            $data = array(
                'content'     => 'akun/users_form',
                'js_script' => 'akun/users_js_script',
                'user'        => $user ,
                'grup'        => $grup ,

                'button' => 'Update',
                'action' => site_url('akun/update_action'),
                'id' => set_value('id', $row->user_id),
                'name' => set_value('name', $row->nama),
                'email' => set_value('email', $row->email),
                'username' => set_value('username', $row->username),
                'password' => set_value('password', $row->password),
                'phone' => set_value('phone', $row->phone),
                'alamat' => set_value('alamat', $row->alamat),
                'groupss'   => set_value('groupss', $row->name) ,
                'user_img' => set_value('user_img', $row->user_img),
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('akun'));
        }
    }
```

```php
public function update_action()
{
    $this->form_validation->set_rules('name', 'name', 'trim|required');
    $this->form_validation->set_rules('email', 'email', 'trim|required');
    $this->form_validation->set_rules('username', 'username', 'trim|required');
    $this->form_validation->set_rules('phone', 'phone', 'trim|required');
    $this->form_validation->set_rules('alamat', 'alamat', 'trim|required');
    $this->form_validation->set_error_delimiters('<span class="text-danger">', '</span>');

    if ($this->form_validation->run() == FALSE) {
        $this->update($this->input->post('id', TRUE));
    } else {
        $data = array(
            'nama' => $this->input->post('name',TRUE),
            'email' => $this->input->post('email',TRUE),
            'username' => $this->input->post('username',TRUE),
            'phone' => $this->input->post('phone',TRUE),
            'alamat' => $this->input->post('alamat',TRUE),
        );

        $this->Akun_model->update($this->input->post('id', TRUE), $data);
        $this->session->set_flashdata('message', 'Update Record Success');
        if ($this->ion_auth->is_admin()) {
            redirect(site_url('akun'),'refresh');
        } else {
            redirect('akun/update/'.$this->input->post('id', TRUE),'refresh');
        }
    }
}

public function delete($id)
{
    $row = $this->Akun_model->get_by_id($id);
    if ($row) {
        unlink( FCPATH . "images/users/" . $row->user_img );
        $this->Akun_model->delete($id);
        $this->session->set_flashdata('message', 'Delete Record Success');
        redirect(site_url('akun'));
    } else {
        $this->session->set_flashdata('message', 'Record Not Found');
        redirect(site_url('akun'));
    }
}

public function ubah_foto($id)
{
    $this->form_validation->set_rules('user_img', 'User Image', 'callback_image_upload');
    if ($this->form_validation->run() == FALSE) {
        $this->update($id);
    } else {
        $CI =& get_instance();
        $upload_data = $CI->upload->data();
        $gambar = $upload_data['file_name'];
        $data = array(
            'user_img' => $gambar,
        );
        $row = $this->Akun_model->get_by_id($id);
        unlink( FCPATH . "images/users/" . $row->user_img );
        $this->Akun_model->update($id,$data);

        if ($this->ion_auth->is_admin()) {
            redirect(site_url('akun/update/'.$id);
        }
        // else {
        //     redirect('home/update_akun/'.$id,'refresh');
        // }

    }

}

public function ubah_password($id)
{
    $password = $this->input->post('password',TRUE);
    $this->form_validation->set_rules('password', 'password', 'trim|required');
    if ($this->form_validation->run() == FALSE) {
        $this->update($id);
    } else {
        $data = array(
            'password' => $password,
        );
        $this->ion_auth->update($id,$data);
        $this->session->set_flashdata('message', 'Password Berhasil diubah');
        if ($this->ion_auth->is_admin()) {
            redirect(site_url('akun/update/'.$id);
        } else {
            redirect('home/update_akun/'.$id,'refresh');
        }
    }
}
public function image_upload(){
    if($_FILES['user_img']['size'] != 0){
        $upload_dir = './images/users/';
        if (!is_dir($upload_dir)) {
            mkdir($upload_dir);
        }
        $config['upload_path']   = $upload_dir;
        $config['allowed_types'] = 'gif|jpg|png|jpeg';
        $config['file_name']     = 'usr_img_'.substr(md5(rand()),0,7);
        $config['overwrite']     = true;
```

```
            $config['max_size']     = '51200';

            $this->load->library('upload', $config);
            if (!$this->upload->do_upload('user_img')){
                $this->form_validation->set_message('image_upload', $this->upload->display_errors());
                return false;
            }
            else{
                $this->upload_data['file'] =  $this->upload->data();
                return true;
            }
        }
        else{
            $this->form_validation->set_message('image_upload', "No file selected");
            return false;
        }
    }
}
```

Gambar D.1 *Controller* Akun.php

D.1.1 *Model* Akun (Akun_model.php)

```php
<?php

if (!defined('BASEPATH'))
    exit('No direct script access allowed');

class Akun_model extends CI_Model
{
    public $table = 'users';
    public $id = 'id';
    public $order = 'DESC';

    function __construct()
    {
        parent::__construct();
    }

    // get all
    function get_all()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->result();
    }

    // get data by id
    function get_by_id($id)
    {
        $this->db->join('users_groups', 'users.id = users_groups.user_id', 'left');
        $this->db->join('groups', 'users_groups.group_id = groups.id', 'left');
        $this->db->where('users.id', $id);
        return $this->db->get($this->table)->row();
    }

    // insert data
    function insert($data)
    {
        $this->db->insert($this->table, $data);
    }

    // update data
    function update($id, $data)
    {
        $this->db->where($this->id, $id);
        $this->db->update($this->table, $data);
    }

    // delete data
    function delete($id)
    {
        $this->db->where($this->id, $id);
        $this->db->delete($this->table);
    }

}
```

Gambar D.2 *Model* Akun (Akun_model.php)

D.2 Mengelola data gejala

D.2.1 *Controller* Gejala (Gejala.php)

```php
<?php

if (!defined('BASEPATH'))
    exit('No direct script access allowed');

class Gejala extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
        $this->load->model('Gejala_model');
        if (!$this->ion_auth->logged_in())
        {
            redirect('auth/login', 'refresh');
        }
    }

    public function index()
    {
        $gejala = $this->Gejala_model->get_all();
        $user = $this->ion_auth->user()->row();
        $data = array(
            'content' => 'gejala/gejala_list' ,
            'js_script' => 'gejala/gejala_js_script',
            'gejala_data' => $gejala,
            'user' => $user ,
        );
        $this->load->view('layout/template', $data);
    }

    public function json() {
        header('Content-Type: application/json');
        echo $this->Gejala_model->json();
    }

    public function read($id)
    {
        $user = $this->ion_auth->user()->row();
        $row = $this->Gejala_model->get_by_id($id);
        if ($row) {
            $data = array(
                'js_script' => 'gejala/gejala_js_script',
                'id' => $row->id,
                'kode' => $row->kode,
                'gejala' => $row->gejala,
                'bobot' => $row->bobot,
                'content' => 'gejala/gejala_read' ,
                'user' => $user ,
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('gejala'));
        }
    }

    public function create()
    {
        $user = $this->ion_auth->user()->row();
        $first_row = $this->Gejala_model->get_first_row();
        if (!$first_row ) {
            $kode = "G1";
        } else {
            $kodenya = $first_row->kode;
            $pisah = explode('G',$kodenya);
            $num = $pisah[1] + 1;
            $kode = "G".$num;
        }
        $data = array(
            'content' => 'gejala/gejala_form' ,
```

```
            'js_script' => 'gejala/gejala_js_script',
            'user' => $user ,
            'button' => 'Create',
            'action' => site_url('gejala/create_action'),
            'id' => set_value('id'),
            'kode' => $kode,
            'gejala' => set_value('gejala'),
            'bobot' => set_value('bobot'),
        );
        $this->load->view('layout/template', $data);
    }

    public function create_action()
    {
        $this->_rules();
        if ($this->form_validation->run() == FALSE) {
            $this->create();
        } else {
            $data = array(
                'kode' => $this->input->post('kode',TRUE),
                'gejala' => $this->input->post('gejala',TRUE),
                'bobot' => $this->input->post('bobot',TRUE),
            );
            $this->Gejala_model->insert($data);
            $this->session->set_flashdata('message', 'Create Record Success');
            redirect(site_url('gejala'));
        }
    }

    public function update($id)
    {
        $user = $this->ion_auth->user()->row();
        $row = $this->Gejala_model->get_by_id($id);
        if ($row) {
            $data = array(
                'js_script' => 'gejala/gejala_js_script',
                'content' => 'gejala/gejala_form' ,
                'user' => $user,
                'button' => 'Update',
                'action' => site_url('gejala/update_action'),
                'id' => set_value('id', $row->id),
                'kode' => set_value('kode', $row->kode),
                'gejala' => set_value('gejala', $row->gejala),
                'bobot' => set_value('bobot', $row->bobot),
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('gejala'));
        }
    }

    public function update_action()
    {
        $this->_rules();
        if ($this->form_validation->run() == FALSE) {
            $this->update($this->input->post('id', TRUE));
        } else {
```

```
            $data = array(
                'kode' => $this->input->post('kode',TRUE),
                'gejala' => $this->input->post('gejala',TRUE),
                'bobot' => $this->input->post('bobot',TRUE),
            );
            $this->Gejala_model->update($this->input->post('id', TRUE), $data);
            $this->session->set_flashdata('message', 'Update Record Success');
            redirect(site_url('gejala'));
        }
    }

    public function delete($id)
    {
        $row = $this->Gejala_model->get_by_id($id);
        if ($row) {
            $this->Gejala_model->delete($id);
            $this->session->set_flashdata('message', 'Delete Record Success');
            redirect(site_url('gejala'));
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('gejala'));
        }
    }

    public function _rules()
    {
        $this->form_validation->set_rules('kode', 'kode', 'trim|required');
        $this->form_validation->set_rules('gejala', 'gejala', 'trim|required');
        $this->form_validation->set_rules('bobot', 'bobot', 'trim|required|numeric');

        $this->form_validation->set_rules('id', 'id', 'trim');
        $this->form_validation->set_error_delimiters('<span class="text-danger">', '</span>');
    }

}
```

Gambar D.3 *Controller* Gejala (Gejala.php)

D.2.2 Model Gejala (Gejala_model.php)

```
<?php

if (!defined('BASEPATH'))
    exit('No direct script access allowed');

class Gejala_model extends CI_Model
{
    public $table = 'gejala';
    public $id = 'id';
    public $order = 'DESC';

    function __construct()
    {
        parent::__construct();
    }

    // datatables
    function json() {
        $this->datatables->select('id,kode,gejala,bobot');
        $this->datatables->from('gejala');
        //add this line for join
        //$this->datatables->join('table2', 'gejala.field = table2.field');
        $this->datatables->add_column('action', anchor(site_url('gejala/read/$1'),'Read')." | ".anchor(site_url('gejala/update/$1'),
        'Update')." | ".anchor(site_url('gejala/delete/$1'),'Delete','onclick="javasciprt: return confirm(\'Are You Sure ?\')"'), 'id');
        return $this->datatables->generate();
    }

    // get all
    function get_all()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->result();
    }

    // get data by id
    function get_by_id($id)
    {
        $this->db->where($this->id, $id);
        return $this->db->get($this->table)->row();
    }
```

```php
    function get_by_kode($kode)
    {
        $this->db->where('kode', $kode);
        return $this->db->get($this->table)->row();
    }

    function get_first_row()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->first_row();
    }

    // get total rows
    function total_rows($q = NULL) {
        $this->db->like('id', $q);
        $this->db->or_like('kode', $q);
        $this->db->or_like('gejala', $q);
        $this->db->or_like('bobot', $q);
        $this->db->from($this->table);
        return $this->db->count_all_results();
    }
    // get data with limit and search
    function get_limit_data($limit, $start = 0, $q = NULL) {
        $this->db->order_by($this->id, $this->order);
        $this->db->like('id', $q);
        $this->db->or_like('kode', $q);
        $this->db->or_like('gejala', $q);
        $this->db->or_like('bobot', $q);
        $this->db->limit($limit, $start);
        return $this->db->get($this->table)->result();
    }
    // insert data
    function insert($data)
    {
        $this->db->insert($this->table, $data);
    }
    // update data
    function update($id, $data)
    {
        $this->db->where($this->id, $id);
        $this->db->update($this->table, $data);
    }
    // delete data
    function delete($id)
    {
        $this->db->where($this->id, $id);
        $this->db->delete($this->table);
    }
}
```

Gambar D.4 Model Gejala (Gejala_model.php)

D.3 Mengelola data penyakit

D.3.1 *Controller* Penyakit (Penyakit.php)

```php
<?php
if (!defined('BASEPATH'))
    exit('No direct script access allowed');
class Penyakit extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
        $this->load->model('Penyakit_model');
        if (!$this->ion_auth->logged_in())
        {
            redirect('auth/login', 'refresh');
        }
    }
    public function index()
    {
        $penyakit = $this->Penyakit_model->get_all();
        $user = $this->ion_auth->user()->row();
        $data = array(
            'content' => 'penyakit/penyakit_list' ,
            'js_script' => 'penyakit/penyakit_js_script',
            'penyakit_data' => $penyakit,
            'user' => $user ,
        );
        $this->load->view('layout/template', $data);
    }
    public function json() {
        header('Content-Type: application/json');
        echo $this->Penyakit_model->json();
    }
    public function read($id)
    {
        $user = $this->ion_auth->user()->row();
        $row = $this->Penyakit_model->get_by_id($id);
        if ($row) {
            $data = array(
                'js_script' => 'penyakit/penyakit_js_script',
                'id' => $row->id,
                'kode' => $row->kode,
                'penyakit' => $row->penyakit,
                'content' => 'penyakit/penyakit_read' ,
                'user' => $user ,
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('penyakit'));
        }
    }
    public function create()
    {
        $user = $this->ion_auth->user()->row();
        $first_row = $this->Penyakit_model->get_first_row();
        if (!$first_row ) {
            $kode = "P1";
        } else {
            $kodenya = $first_row->kode;
            $pisah = explode('P',$kodenya);
            $num = $pisah[1] + 1;
            $kode = "P".$num;
        }

        $data = array(
            'content' => 'penyakit/penyakit_form' ,
            'js_script' => 'penyakit/penyakit_js_script',
            'user' => $user ,
            'button' => 'Create',
            'action' => site_url('penyakit/create_action'),
            'id' => set_value('id'),
            'kode' => $kode,
            'penyakit' => set_value('penyakit'),
        );
        $this->load->view('layout/template', $data);
    }
    public function create_action()
    {
        $this->_rules();
        if ($this->form_validation->run() == FALSE) {
            $this->create();
        } else {
```

```php
            $data = array(
                'kode' => $this->input->post('kode',TRUE),
                'penyakit' => $this->input->post('penyakit',TRUE),
            );
            $this->Penyakit_model->insert($data);
            $this->session->set_flashdata('message', 'Create Record Success');
            redirect(site_url('penyakit'));
        }
    }
    public function update($id)
    {
        $user = $this->ion_auth->user()->row();
        $row = $this->Penyakit_model->get_by_id($id);
        if ($row) {
            $data = array(
                'js_script' => 'penyakit/penyakit_js_script',
                'content' => 'penyakit/penyakit_form' ,
                'user' => $user,
                'button' => 'Update',
                'action' => site_url('penyakit/update_action'),
                'id' => set_value('id', $row->id),
                'kode' => set_value('kode', $row->kode),
                'penyakit' => set_value('penyakit', $row->penyakit),
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('penyakit'));
        }
    }
    public function update_action()
    {
        $this->_rules();
        if ($this->form_validation->run() == FALSE) {
            $this->update($this->input->post('id', TRUE));
        } else {
            $data = array(
                'kode' => $this->input->post('kode',TRUE),
                'penyakit' => $this->input->post('penyakit',TRUE),
            );
            $this->Penyakit_model->update($this->input->post('id', TRUE), $data);
            $this->session->set_flashdata('message', 'Update Record Success');
            redirect(site_url('penyakit'));
        }
    }
    public function delete($id)
    {
        $row = $this->Penyakit_model->get_by_id($id);
        if ($row) {
            $this->Penyakit_model->delete($id);
            $this->session->set_flashdata('message', 'Delete Record Success');
            redirect(site_url('penyakit'));
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('penyakit'));
        }
    }
    public function _rules()
    {
        $this->form_validation->set_rules('kode', 'kode', 'trim|required');
        $this->form_validation->set_rules('penyakit', 'penyakit', 'trim|required');

        $this->form_validation->set_rules('id', 'id', 'trim');
        $this->form_validation->set_error_delimiters('<span class="text-danger">', '</span>');
    }
}
```

Gambar D.5 *Controller* Penyakit (Penyakit.php)

D.3.1 Model (Penyakit_model.php)

```php
<?php
if (!defined('BASEPATH'))
    exit('No direct script access allowed');

class Penyakit_model extends CI_Model
{
    public $table = 'penyakit';
    public $id = 'id';
    public $order = 'DESC';

    function __construct()
    {
        parent::__construct();
    }
    // datatables
    function json() {
        $this->datatables->select('id,kode,penyakit');
        $this->datatables->from('penyakit');
        //add this line for join
        //$this->datatables->join('table2', 'penyakit.field = table2.field');
        $this->datatables->add_column('action', anchor(site_url('penyakit/read/$1'),'Read')." | ".anchor(site_url('penyakit/update/$1'),
        'Update')." | ".anchor(site_url('penyakit/delete/$1'),'Delete','onclick="javascript: return confirm(\'Are You Sure ?\')"'), 'id');
        return $this->datatables->generate();
    }
    // get all
    function get_all()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->result();
    }
    // get data by id
    function get_by_id($id)
    {
        $this->db->where($this->id, $id);
        return $this->db->get($this->table)->row();
    }

    function get_by_kode($kode)
    {
        $this->db->where('kode', $kode);
        return $this->db->get($this->table)->row();
    }
    function get_first_row()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->first_row();
    }
    // get total rows
    function total_rows($q = NULL) {
        $this->db->like('id', $q);
        $this->db->or_like('kode', $q);
        $this->db->or_like('penyakit', $q);
        $this->db->from($this->table);
        return $this->db->count_all_results();
    }
    // get data with limit and search
    function get_limit_data($limit, $start = 0, $q = NULL) {
        $this->db->order_by($this->id, $this->order);
        $this->db->like('id', $q);
        $this->db->or_like('kode', $q);
        $this->db->or_like('penyakit', $q);
        $this->db->limit($limit, $start);
        return $this->db->get($this->table)->result();
    }
    // insert data
    function insert($data)
    {
        $this->db->insert($this->table, $data);
    }
    // update data
    function update($id, $data)
    {
        $this->db->where($this->id, $id);
        $this->db->update($this->table, $data);
    }
    // delete data
    function delete($id)
    {
        $this->db->where($this->id, $id);
        $this->db->delete($this->table);
    }
}
```

Gambar D.6 Model (Penyakit_model.php)

D.4 Mengelola data penyakit

D.4.1 *Controller* Diagnosa (Kasus_baru.php)

```php
<?php
if (!defined('BASEPATH'))
    exit('No direct script access allowed');
class Kasus_baru extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
        $this->load->model('Kasus_baru_model');
        $this->load->model('Basis_kasus_model');
        $this->load->model('Gejala_model');
        $this->load->model('Penyakit_model');
    }
    public function index()
    {
        if (!$this->ion_auth->logged_in())
        {
            redirect('auth/login', 'refresh');
        }
        $kasus_baru = $this->Kasus_baru_model->get_all();
        $user = $this->ion_auth->user()->row();
        $data = array(
            'content' => 'kasus_baru/kasus_baru_list' ,
            'js_script' => 'kasus_baru/kasus_baru_js_script',
            'kasus_baru_data' => $kasus_baru,
            'user' => $user ,
        );
        $this->load->view('layout/template', $data);
    }
    public function json() {
        header('Content-Type: application/json');
        echo $this->Kasus_baru_model->json();
    }
    public function read($id)
    {
        $user = $this->ion_auth->user()->row();
        $row = $this->Kasus_baru_model->get_by_id($id);
        if ($row) {
            $gejala = $this->Basis_kasus_model->get_gejala($row->kode);
            $data = array(
                'js_script' => 'kasus_baru/kasus_baru_js_script',
                'id' => $row->id,
                'kode' => $row->kode,
                'nama' => $row->nama,
                'usia' => $row->usia,
                'latar_belakang' => $row->latar_belakang,
                'nilai_knn' => $row->nilai_knn,
                'kode_similaritas' => $row->kode_similaritas,
                'status' => $row->status,
                'created_at' => $row->created_at,
                'updated_at' => $row->updated_at,
                'content' => 'kasus_baru/kasus_baru_read' ,
                'user' => $user ,
                'gejala' => $gejala ,
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('kasus_baru'));
        }
    }
    public function json_get_gejala($kode)
    {
        $this->db->where('kode_kasus', $kode);
        $query = $this->db->get('v_kasus_detail');
        echo json_encode($query->result());
    }
    public function create() {
        $user = $this->ion_auth->user()->row();
        $date = new DateTime();
        //$timestamp = $date->format('U');
        $first_row_kasus_baru = $this->Kasus_baru_model->get_first_row();
        $first_row_basis_kasus = $this->Basis_kasus_model->get_first_row();
        if(!$first_row_kasus_baru){
            if (!$first_row_basis_kasus ) {
                $kode = "K1";
            } else {
                $kodenya = $first_row_basis_kasus->kode;
                $pisah = explode('K',$kodenya);
```

```
            $num = $pisah[1] + 1;
            $kode = "K".$num;
        }
    } else {
        $kodenya = $first_row_kasus_baru->kode;
        $pisah = explode('K',$kodenya);
        $num = $pisah[1] + 1;
        $kode = "K".$num;
    }
    //$kode = "K".$timestamp;
    $dt_gejala = $this->Gejala_model->get_all();

    $data = array(
        'content' => 'kasus_baru/kasus_baru_form' ,
        'js_script' => 'kasus_baru/kasus_baru_js_script',
        'user' => $user ,
        'button' => 'Create',
        'action' => site_url('kasus_baru/hasil_diagnosa'),
        'id' => set_value('id'),
        'kode' => $kode,
        'nama' => set_value('nama'),
        'usia' => set_value('usia'),
        'latar_belakang' => set_value('latar_belakang'),
        'dt_gejala' => $dt_gejala ,
    );
    $this->load->view('layout/template', $data);
}
public function hasil_diagnosa()
{
    $this->_rules();
    if ($this->form_validation->run() == FALSE) {
        $this->create();
    } else {
        $kode = $this->input->post('kode',TRUE);
        $gejala = $this->input->post('gejala',TRUE);
        $basis_kasus = $this->Basis_kasus_model->get_all();
        foreach ($gejala as $gj) {
            $nilai_input[] = $this->Gejala_model->get_by_kode($gj)->bobot;
        }
        $nilai_inputan = array_sum($nilai_input);
        $total = array();
        foreach ($basis_kasus as $bk) {
            $kasus_detail = $this->Basis_kasus_model->get_gejala($bk->kode);
            foreach ($kasus_detail as $kd) {
                foreach ($gejala as $g) {
                    if($kd->kode_gejala == $g){
                        $bobot[$kd->kode_kasus][$g] = $kd->bobot;
                    } else {
                        $bobot[$kd->kode_kasus]['G0'] = 0;
                    }
                }
            }
        }
        foreach ($basis_kasus as $bk) {
            if($bobot[$bk->kode]){
                $total[$bk->kode] = array_sum($bobot[$bk->kode]);
            } else {
                $total[$bk->kode] = 0;
            }
        }
        foreach ($bobot as $key => $value) {
            $hasil_bagi = $total[$key] / $nilai_inputan ;
            $persen_hasil_bagi = $hasil_bagi * 100;
            $arr_hasil[] = $persen_hasil_bagi."-".$key;
        }
        $count = count($arr_hasil);
        $temp_kode_nilai = array();
        for ($i=0; $i < $count; $i++) {
            $hasil = explode('-',$arr_hasil[$i]);
            $temp_kode_nilai[$hasil[1]] = $hasil[0];
        }
        $maxs = max($temp_kode_nilai);
        $kode_same_val = array_keys($temp_kode_nilai, max($temp_kode_nilai));
        for ($i=0; $i < count($kode_same_val) ; $i++) {
            $temp_nilai[] = $maxs;
        }
        $hasil_filnal_diagnosa = array_combine($kode_same_val, $temp_nilai);
        $user = $this->ion_auth->user()->row();
        $data = array(
            'user' => $user ,
            'content' => 'kasus_baru/v_hasil_diagnosa' ,
            'js_script' => 'kasus_baru/kasus_baru_js_script',
            'action' => site_url('kasus_baru/create_action'),
            'kode' => $kode,
            'nama' => $this->input->post('nama',TRUE),
            'usia' => $this->input->post('usia',TRUE),
            'latar_belakang' => $this->input->post('latar_belakang',TRUE),
            'gejala' => $gejala ,
            'hasil_diagnosa' =>  $hasil_filnal_diagnosa,
        );
        $this->load->view('layout/template', $data);
```

```
    }
}
public function create_action()
{
    $this->_rules();
    if ($this->form_validation->run() == FALSE) {
        $this->create();
    } else {
        $kode = $this->input->post('kode',TRUE);
        $gejala = $this->input->post('gejala',TRUE);
        $data = array(
            'kode' => $kode,
            'nama' => $this->input->post('nama',TRUE),
            'usia' => $this->input->post('usia',TRUE),
            'latar_belakang' => $this->input->post('latar_belakang',TRUE),
            'nilai_knn' => $this->input->post('nilai_knn',TRUE),
            'kode_similaritas' => $this->input->post('kode_similaritas',TRUE) ,
        );
        $this->Kasus_baru_model->insert($data);
        foreach ($gejala as $g) {
            $this->db->insert('kasus_detail', array('kode_kasus' => $kode , 'kode_gejala' => $g ));
        }
        if (!$this->ion_auth->logged_in())
        {
            $this->session->set_flashdata('message', 'Create Record Success');
            redirect('dashboard', 'refresh');
        } else {
            $this->session->set_flashdata('message', 'Create Record Success');
            redirect(site_url('kasus_baru'));
        }
    }
}
public function revise($id_kasus_baru)
{
    if (!$this->ion_auth->logged_in())
    {
        redirect('auth/login', 'refresh');
    }
    $user = $this->ion_auth->user()->row();
    $row = $this->Kasus_baru_model->get_by_id($id_kasus_baru);
    if($row){
        $kode_kasus = $row->kode;
        $kode_penyakit = $row->kode_similaritas;
        $gejala = $this->Basis_kasus_model->get_gejala($row->kode);
        $data = array(
            'js_script' => 'kasus_baru/kasus_baru_js_script',
            'content' => 'kasus_baru/v_revise_form' ,
            'action' => site_url('kasus_baru/revise_action/'.$id_kasus_baru),
            'user' => $user,
            'id_user' => $user->id ,
            'kode_kasus' => $kode_kasus ,
            'kode_penyakit' => $kode_penyakit,
            'gejala' => $gejala,
            'id_user' => $user->id,
            'pengobatan' => set_value('pengobatan'),
            'penyebab' => set_value('penyebab'),
            'keterangan' => set_value('keterangan'),
            'id_kasus_baru' => $id_kasus_baru,
        );
        $this->load->view('layout/template', $data);
    } else {
        echo "data tidak ditemukan";
    }
}
public function revise_action($id_kasus_baru)
{
    $this->revise_rules();
    if ($this->form_validation->run() == FALSE) {
        $this->revise($id_kasus_baru);
    } else {
        $kode = $this->input->post('kode',TRUE);
        $gejala = $this->input->post('gejala',TRUE);
        $data = array(
            'id_user' => $this->input->post('id_user',TRUE),
            'kode' => $kode,
            'pengobatan' => $this->input->post('pengobatan',TRUE),
            'penyebab' => $this->input->post('penyebab',TRUE),
            'keterangan' => $this->input->post('keterangan',TRUE),
            'kode_penyakit' => $this->input->post('kode_penyakit',TRUE),
        );
        $this->Basis_kasus_model->insert($data);
        $this->session->set_flashdata('message', 'Create Record Success');
        redirect(site_url('basis_kasus'));
    }
}
public function revise_rules()
{
    $this->form_validation->set_rules('id_user', 'id user', 'trim|required');
    $this->form_validation->set_rules('kode', 'kode', 'trim|required');
    $this->form_validation->set_rules('pengobatan', 'pengobatan', 'trim|required');
    $this->form_validation->set_rules('penyebab', 'penyebab', 'trim|required');
    $this->form_validation->set_rules('keterangan', 'keterangan', 'trim|required');
    $this->form_validation->set_rules('kode_penyakit', 'kode penyakit', 'trim|required');
    $this->form_validation->set_rules('id', 'id', 'trim');
    $this->form_validation->set_error_delimiters('<span class="text-danger">', '</span>');
}
public function delete($id)
{
    if (!$this->ion_auth->logged_in())
    {
        redirect('auth/login', 'refresh');
```

```
        }
        $row = $this->Kasus_baru_model->get_by_id($id);
        if ($row) {
            //$this->Kasus_baru_model->delete($id);
            $data = array(
                'deleted' => 1,
            );
            $this->Kasus_baru_model->update($id, $data);
            $this->session->set_flashdata('message', 'Delete Record Success');
            redirect(site_url('kasus_baru'));
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('kasus_baru'));
        }
    }
    }
    public function tambah_penyakit_action($id_kasus_baru)
    {
        $data = array(
            'kode' => $this->input->post('kode',TRUE),
            'penyakit' => $this->input->post('penyakit',TRUE),
        );
        $this->Penyakit_model->insert($data);
        redirect(site_url('kasus_baru/revise/'.$id_kasus_baru));
    }
    public function _rules()
    {
        $this->form_validation->set_rules('kode', 'kode', 'trim|required');
        $this->form_validation->set_rules('nama', 'nama', 'trim|required');
        $this->form_validation->set_rules('usia', 'usia', 'trim|required|numeric');
        $this->form_validation->set_rules('latar_belakang', 'latar belakang', 'trim|required');
        $this->form_validation->set_rules('gejala[]','gejala','trim|required');

        $this->form_validation->set_rules('id', 'id', 'trim');
        $this->form_validation->set_error_delimiters('<span class="text-danger">', '</span>');
    }
}
```

Gambar D.7 *Controller* Diagnosa (Kasus_baru.php)

D.4.1 Model Diagnosa (Kasus_baru_model.php)

```
<?php
if (!defined('BASEPATH'))
    exit('No direct script access allowed');
class Kasus_baru_model extends CI_Model
{
    public $table = 'kasus_baru';
    public $id = 'id';
    public $order = 'DESC';
    function __construct()
    {
        parent::__construct();
    }
    // datatables
    function json() {
        $this->datatables->select('id,kode,nama,usia,latar_belakang,nilai_knn,status,created_at,updated_at');
        $this->datatables->from('kasus_baru');
        //add this line for join
        //$this->datatables->join('table2', 'kasus_baru.field = table2.field');
        $this->datatables->add_column('action', anchor(site_url('kasus_baru/read/$1'),'Read')." | ".anchor(site_url('kasus_baru/update/$1'),
        'Update')." | ".anchor(site_url('kasus_baru/delete/$1'),'Delete','onclick="javasciprt: return confirm(\'Are You Sure ?\')"'), 'id');
        return $this->datatables->generate();
    }
    function get_first_row()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->first_row();
    }
    // get all
    function get_all()
    {
        $this->db->order_by($this->id, $this->order);
        $this->db->where('deleted',0);
        return $this->db->get($this->table)->result();
    }
    // get data by id
    function get_by_id($id)
    {
        $this->db->where($this->id, $id);
        return $this->db->get($this->table)->row();
    }
```

```php
    function get_by_kode($kode)
    {
        $this->db->where('kode', $kode);
        return $this->db->get($this->table)->row();
    }
    // get total rows
    function total_rows($q = NULL) {
        $this->db->from($this->table);
        $this->db->where('deleted',0);
        return $this->db->count_all_results();
    }
    // get data with limit and search
    function get_limit_data($limit, $start = 0, $q = NULL) {
        $this->db->order_by($this->id, $this->order);
        $this->db->like('id', $q);
        $this->db->or_like('kode', $q);
        $this->db->or_like('nama', $q);
        $this->db->or_like('usia', $q);
        $this->db->or_like('latar_belakang', $q);
        $this->db->or_like('nilai_knn', $q);
        $this->db->or_like('status', $q);
        $this->db->or_like('created_at', $q);
        $this->db->or_like('updated_at', $q);
        $this->db->limit($limit, $start);
        return $this->db->get($this->table)->result();
    }
    // insert data
    function insert($data)
    {
        $this->db->insert($this->table, $data);
    }
    // update data
    function update($id, $data)
    {
        $this->db->where($this->id, $id);
        $this->db->update($this->table, $data);
    }
    // delete data
    function delete($id)
    {
        $this->db->where($this->id, $id);
        $this->db->delete($this->table);
    }

}
```

Gambar D.8 Model Diagnosa (Kasus_baru_model.php)

D.5 Mengelola data basis kasus

D.5.1 *Controller* Basis Kasus (Basis_kasus.php)

```php
<?php
if (!defined('BASEPATH'))
    exit('No direct script access allowed');
class Basis_kasus extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
        $this->load->model('Basis_kasus_model');
        $this->load->model('Penyakit_model');
        $this->load->model('Gejala_model');
        if (!$this->ion_auth->logged_in())
        {
            redirect('auth/login', 'refresh');
        }
    }
    public function index()
    {
        $basis_kasus = $this->Basis_kasus_model->get_all();
        $user = $this->ion_auth->user()->row();
        $data = array(
            'content' => 'basis_kasus/basis_kasus_list' ,
            'js_script' => 'basis_kasus/basis_kasus_js_script',
            'basis_kasus_data' => $basis_kasus,
            'user' => $user ,
        );
        $this->load->view('layout/template', $data);
    }
    public function json() {
        header('Content-Type: application/json');
        echo $this->Basis_kasus_model->json();
    }
    public function read($id)
    {
        $user = $this->ion_auth->user()->row();
        $row = $this->Basis_kasus_model->get_by_id($id);
        if ($row) {
            $gejala = $this->Basis_kasus_model->get_gejala($row->kode);
            $data = array(
                'js_script' => 'basis_kasus/basis_kasus_js_script',
                'row' => $row ,
                'content' => 'basis_kasus/basis_kasus_read' ,
                'user' => $user ,
                'dt_gejala' => $gejala ,
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('basis_kasus'));
        }
    }
    public function create()
    {
        $user = $this->ion_auth->user()->row();
        $date = new DateTime();
        $timestamp = $date->format('U');
        $kode = "K".$timestamp;
        $dt_penyakit = $this->Penyakit_model->get_all();
        $dt_gejala = $this->Gejala_model->get_all();
        $data = array(
```

```
            'content' => 'basis_kasus/basis_kasus_form' ,
            'js_script' => 'basis_kasus/basis_kasus_js_script',
            'title' => 'Create Basis Kasus',
            'user' => $user ,
            'button' => 'Create',
            'action' => site_url('basis_kasus/create_action'),
            'id' => set_value('id'),
            'id_user' => $user->id,
            'kode' => $kode,
            'pengobatan' => set_value('pengobatan'),
            'penyebab' => set_value('penyebab'),
            'keterangan' => set_value('keterangan'),
            'kode_penyakit' => set_value('kode_penyakit'),
            'dt_penyakit' => $dt_penyakit ,
            'dt_gejala' => $dt_gejala ,
        );
        $this->load->view('layout/template', $data);
    }
    public function create_action()
    {
        $this->_rules();
        if ($this->form_validation->run() == FALSE) {
            $this->create();
        } else {
            $kode = $this->input->post('kode',TRUE);
            $gejala = $this->input->post('gejala',TRUE);
            $data = array(
                'id_user' => $this->input->post('id_user',TRUE),
                'kode' => $kode,
                'pengobatan' => $this->input->post('pengobatan',TRUE),
                'penyebab' => $this->input->post('penyebab',TRUE),
                'keterangan' => $this->input->post('keterangan',TRUE),
                'kode_penyakit' => $this->input->post('kode_penyakit',TRUE),
            );
            $this->Basis_kasus_model->insert($data);
            foreach ($gejala as $g) {
                $this->db->insert('kasus_detail', array('kode_kasus' => $kode , 'kode_gejala' => $g ));
            }
            $this->session->set_flashdata('message', 'Create Record Success');
            redirect(site_url('basis_kasus'));
        }
    }
    public function update($id)
    {
        $user = $this->ion_auth->user()->row();
        $row = $this->Basis_kasus_model->get_by_id($id);
        if ($row) {
            $dt_penyakit = $this->Penyakit_model->get_all();
            $dt_gejala = $this->Gejala_model->get_all();
            $data = array(
                'js_script' => 'basis_kasus/basis_kasus_js_script',
                'content' => 'basis_kasus/basis_kasus_form' ,
                'title' => 'Update Basis Kasus',
                'user' => $user,
                'button' => 'Update',
                'action' => site_url('basis_kasus/update_action'),
                'id' => set_value('id', $row->id),
                'id_user' => set_value('id_user', $row->id_user),
                'kode' => set_value('kode', $row->kode),
                'pengobatan' => set_value('pengobatan', $row->pengobatan),
                'penyebab' => set_value('penyebab', $row->penyebab),
                'keterangan' => set_value('keterangan', $row->keterangan),
                'kode_penyakit' => set_value('kode_penyakit', $row->kode_penyakit),
                'dt_penyakit' => $dt_penyakit ,
                'dt_gejala' => $dt_gejala ,
            );
            $this->load->view('layout/template', $data);
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('basis_kasus'));
        }
    }
    public function update_action()
    {
        $this->_rules();
        if ($this->form_validation->run() == FALSE) {
            $this->update($this->input->post('id', TRUE));
        } else {
            $date = new DateTime();
            $timestamp = $date->format('Y-m-d H:i:s');
            $kode = $this->input->post('kode',TRUE);
```

```
            $gejala = $this->input->post('gejala',TRUE);
            $data = array(
                'id_user' => $this->input->post('id_user',TRUE),
                'kode' => $kode,
                'pengobatan' => $this->input->post('pengobatan',TRUE),
                'penyebab' => $this->input->post('penyebab',TRUE),
                'keterangan' => $this->input->post('keterangan',TRUE),
                'kode_penyakit' => $this->input->post('kode_penyakit',TRUE),
                'updated_at' => $timestamp,
            );
            $this->db->where('kode_kasus', $kode);
            $this->db->delete('kasus_detail');
            $this->Basis_kasus_model->update($this->input->post('id', TRUE), $data);
            foreach ($gejala as $g) {
                $this->db->insert('kasus_detail', array('kode_kasus' => $kode , 'kode_gejala' => $g ));
            }
            $this->session->set_flashdata('message', 'Update Record Success');
            redirect(site_url('basis_kasus'));
        }
    }

    public function delete($id)
    {
        $row = $this->Basis_kasus_model->get_by_id($id);
        if ($row) {
            $this->Basis_kasus_model->delete($id);
            $this->db->where('kode_kasus', $row->kode);
            $this->db->delete('kasus_detail');
            $this->session->set_flashdata('message', 'Delete Record Success');
            redirect(site_url('basis_kasus'));
        } else {
            $this->session->set_flashdata('message', 'Record Not Found');
            redirect(site_url('basis_kasus'));
        }
    }
    public function _rules()
    {
        $this->form_validation->set_rules('id_user', 'id user', 'trim|required');
        $this->form_validation->set_rules('kode', 'kode', 'trim|required');
        $this->form_validation->set_rules('pengobatan', 'pengobatan', 'trim|required');
        $this->form_validation->set_rules('penyebab', 'penyebab', 'trim|required');
        $this->form_validation->set_rules('keterangan', 'keterangan', 'trim|required');
        $this->form_validation->set_rules('kode_penyakit', 'kode penyakit', 'trim|required');
        $this->form_validation->set_rules('gejala[]','gejala','trim|required');
        $this->form_validation->set_rules('id', 'id', 'trim');
        $this->form_validation->set_error_delimiters('<span class="text-danger">', '</span>');
    }
}
```

Gambar D.9 *Controller* Basis Kasus (Basis_kasus.php)

D.5.1 Model Basis Kasus (Basis_kasus _model.php)

```
<?php
if (!defined('BASEPATH'))
    exit('No direct script access allowed');
class Basis_kasus_model extends CI_Model
{
    public $table = 'basis_kasus';
    public $id = 'id';
    public $order = 'DESC';

    function __construct()
    {
        parent::__construct();
    }
    // datatables
    function json() {
        $this->datatables->select('id,id_user,kode,pengobatan,penyebab,keterangan,kode_penyakit,created_at,updated_at');
        $this->datatables->from('basis_kasus');
        $this->datatables->add_column('action', anchor(site_url('basis_kasus/read/$1'),'Read'." | ".anchor(site_url
        ('basis_kasus/update/$1'),'Update')." | ".anchor(site_url('basis_kasus/delete/$1'),'Delete','onclick="javasciprt: return confirm
        (\'Are You Sure ?\')"'), 'id');
        return $this->datatables->generate();
    }
    // get all
    function get_all()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->result();
    }
    function get_first_row()
    {
        $this->db->order_by($this->id, $this->order);
        return $this->db->get($this->table)->first_row();
    }
    function get_gejala($kode_kasus)
    {
        $this->db->where('kode_kasus', $kode_kasus);
        return $this->db->get('v_kasus_detail')->result();
    }
    function get_bobot($kode_kasus,$kode_gejala)
    {
```

```
        $this->db->where('kode_kasus', $kode_kasus);
        $this->db->where('kode_gejala', $kode_gejala);
        return $this->db->get('v_kasus_detail')->row();
    }
    // get data by id
    function get_by_id($id)
    {
        $this->db->where($this->id, $id);
        return $this->db->get('v_basis_kasus')->row();
    }
    function get_by_kode($kode)
    {
        $this->db->where('kode', $kode);
        return $this->db->get('v_basis_kasus')->row();
    }
    // insert data
    function insert($data)
    {
        $this->db->insert($this->table, $data);
    }
    // update data
    function update($id, $data)
    {
        $this->db->where($this->id, $id);
        $this->db->update($this->table, $data);
    }

    // delete data
    function delete($id)
    {
        $this->db->where($this->id, $id);
        $this->db->delete($this->table);
    }
}
```

Gambar D.10 Model Basis Kasus (Basis_kasus _model.php)

Lampiran E. *Testing*

E.1 Pengujian *Black Box*

Tabel E.1 Pengujian *Black Box*

No.	Menu	Fungsi	Kasus	Hasil	Keterangan
1.	Login	Menu ini berfungsi sebagai keamanan sistem. memilah hak akses *user* yang dapat menggunakan sistem ini.	Ketika *user* memasukan *username* dan password dengan benar	Menampilkan halaman sesuai level *user*	Berhasil
2	Home	Menampilkan halaman home	Ketika *user* telah melakukan login, sistem akan menampilkan halaman home	Menampilkan halaman home	Berhasil
3.	Kelola *user*	Untuk melihat data *user*, menambah *user*,	Ketika *admin* klik menu kelola *user* menampilkan halaman kelola data *user*	Menampilkan halaman kelola data *user*	Berhasil

No.	Menu	Fungsi	Kasus	Hasil	Keterangan
		ubah data *user*, dan menghapus data user	Ketika *admin* klik tombol *"Create New"*	Menampilkan form tambah data *user*	Berhasil
			Ketika *admin* klik tombol *"Delete"*	Menghapus data user	Berhasil
			Ketika *admin* klik tombol *update*	Menampilkan halaman *update* data *user*	Berhasil
4.	Diagnosa	Melakukan diagnosa penyakit lambung	Ketika *user* klik menu Diagnosa	Menampilkan halaman form diagnosa	Berhasil
			Ketika *user* klik tombol Diagnosa	Menampilkan halaman hasil diagnosa	Berhasil
			Ketika *user* klik tombol *Submit* pada halaman form hasil Diagnosa	Menyimpan hasil diagnosa	Berhasil
5.	Mengedit *User*	Merubah data *user*	Ketika pakar atau admin klik *"profil"*	Menampilkan halaman form *update user*	Berhasil

No.	Menu	Fungsi	Kasus	Hasil	Keterangan
			Ketika pakar atau admin klik tombol *Submit* pada form *update user*	Merubah data *user*	Berhasil
			Ketika pakar atau admin klik *change password* pada form *update user*	Merubah *password user*	Berhasil
6.	Data Diagnosa	Mengelola data diagnosa	Ketika pakar klik menu Data Diagnosa	Menampilkan halaman Data Diagnosa	Berhasil
			Ketika pakar klik tombol *delete*	Menghapus Data Diagnosa	Berhasil
			Ketika pakar klik tombol *detail*	Menampilkan halaman detail diagnosa	Berhasil
7.	Gejala	Mengelola Data Gejala	Ketika pakar klik menu Gejala	Menampilkan halaman Data Gejala	Berhasil
			Ketika pakar klikt tombol *Create New*	Menampilkan halaman form data gejala	Berhasil
			Ketika pakar klik tombol *Delete*	Menghapus Data Gejala	Berhasil

No.	Menu	Fungsi	Kasus	Hasil	Keterangan
			Ketika pakar klik tombol *Edit*	Menampilkan halaman form update gejala	Berhasil
8.	Penyakit	Mengelola Data Penyakit	Ketika pakar klik menu Penyakit	Menampilkan halaman Data Penyakit	Berhasil
			Ketika pakar klikt tombol *Create New*	Menampilkan halaman form data penyakit	Berhasil
			Ketika pakar klik tombol *Delete*	Menghapus Data penyakit	Berhasil
			Ketika pakar klik tombol *Edit*	Menampilkan halaman form update penyakit	Berhasil
9.	Basis Kasus	Mengelola data basis kasus	Ketika pakar klik tombol *detail*	Menampilkan halaman detail basis kasus	Berhasil
			Ketika pakar klik tombol *Edit*	Menampilkan halaman update basis kasus	Berhasil
			Ketika pakar klik tombol *Delete*	Menghapus Data basis kasus	Berhasil

Lampiran F. Kuesioner tingkat kepentingan gejala pada setiap jenis penyakit lambung

KUESIONER

Tingkat Kepentingan Gejala Pada Jenis Penyakit Lambung

Hari/Tanggal Pengisian : 25 September 2017 / Senin

Nama Narasumber : dr. Devi Chintya Kumalasari

Pekerjaan : Dokter Manajemen Rumah Sakit Citra Husada Jember

Alamat : Jln. Soekarno hatta no. 36 Jember

Tanda Tangan :

Hasil pengisian kuesioner akan digunakan untuk keperluan penelitian dalam rangka penyusunan Skripsi dengan judul ***CASE BASED REASONING MENGGUNAKAN ALGORITMA K-NEAREST NEIGHBOR (K-NN) UNTUK DIAGNOSA PENYAKIT LAMBUNG.*** *Penelitian ini dilaksanakan oleh Andre Bhaskoro Suprayogi (Mahasiswa Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Jember), dibawah bimbingan* Prof. Drs. Slamin, M.Comp.Sc., Ph.D, Diah ayu Retnani W, ST ., M.Eng.

PENGISIAN TINGKAT KEPENTINGAN GEJALA JENIS PENYAKIT LAMBUNG
Grastitis (maag)

No	Gejala Grastitis (maag)	GSP	GP	GS	GB	GSB
1	Rasa nyeri dan rasa tidak nyaman pada perut	✓				
2	Rasa penuh didaerah lambung / daerah perut		✓			
3	Perut kembung		✓			
4	Rasa mual	✓				
5	Muntah	✓				
6	Rasa nyeri pada uluhati	✓				
7	Selera makan berkurang		✓			
8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas	✓				
9	Sering sendawa				✓	
10	Sering merdengar suara usus yang keras (borborigmi)				✓	
11	Sembelit				✓	
12	Diare		✓			
13	Kehilangan berat badan secara mendadak					✓
14	Demam atau meriang	✓				
15	Kejang perut					✓
16	Keluarnya BAB warna hitam pekat					✓
17	Pendarahan (BAB darah atau Muntah darah)					✓
18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher					✓
19	Rasa ada makanan/minuman balik ke mulut					✓
20	Mulut terasa asam dan pahit					✓
21	Batuk menahun					✓
22	Serak dan tenggorakan sakit					✓
23	Lesu (mudah lelah)				✓	
24	Perasaan kenyang berlebihan walaupunhanya makan sedikit		✓			
25	Dehidrasi berat					✓
26	Rasa nyeri pada tukak deodenum					✓
27	Sakit pada tukak lambung	✓				

PENGISIAN TINGKAT KEPENTINGAN GEJALA JENIS PENYAKIT LAMBUNG
Dispepsia

No	Gejala Dispepsia	GSP	GP	GS	GB	GSB
1	Rasa nyeri dan rasa tidak nyaman pada perut	✓				
2	Rasa penuh didaerah lambung / daerah perut	✓				
3	Perut kembung	✓				
4	Rasa mual	✓				
5	Muntah		✓			
6	Rasa nyeri pada uluhati		✓			
7	Selera makan berkurang	✓				
8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas	✓				
9	Sering sendawa				✓	
10	Sering merdengar suara usus yang keras (borborigmi)				✓	
11	Sembelit					✓
12	Diare					✓
13	Kehilangan berat badan secara mendadak					✓
14	Demam atau meriang				✓	
15	Kejang perut					✓
16	Keluarnya BAB warna hitam pekat					✓
17	Pendarahan (BAB darah atau Muntah darah)					✓
18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher					✓
19	Rasa ada makanan/minuman balik ke mulut					✓
20	Mulut terasa asam dan pahit					✓
21	Batuk menahun					✓
22	Serak dan tenggorakan sakit					✓
23	Lesu (mudah lelah)			✓		
24	Perasaan kenyang berlebihan walaupunhanya makan sedikit	✓				
25	Dehidrasi berat					✓
26	Rasa nyeri pada tukak deodenum					✓
27	Sakit pada tukak lambung			✓		

PENGISIAN TINGKAT KEPENTINGAN GEJALA JENIS PENYAKIT LAMBUNG
Kanker Lambung (Gastric Cancer)

No	Gejala Kanker Lambung (Gastric Cancer)	GSP	GP	GS	GB	GSB
1	Rasa nyeri dan rasa tidak nyaman pada perut		✓			
2	Rasa penuh didaerah lambung / daerah perut	✓				
3	Perut kembung			✓		
4	Rasa mual			✓		
5	Muntah			✓		
6	Rasa nyeri pada uluhati				✓	
7	Selera makan berkurang	✓				
8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas	✓				
9	Sering sendawa				✓	
10	Sering merdengar suara usus yang keras (borborigmi)					✓
11	Sembelit				✓	
12	Diare				✓	
13	Kehilangan berat badan secara mendadak	✓				
14	Demam atau meriang	✓				
15	Kejang perut					✓
16	Keluarnya BAB warna hitam pekat	✓				
17	Pendarahan (BAB darah atau Muntah darah)	✓				
18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher					✓
19	Rasa ada makanan/minuman balik ke mulut				✓	
20	Mulut terasa asam dan pahit					✓
21	Batuk menahun					✓
22	Serak dan tenggorakan sakit					✓
23	Lesu (mudah lelah)	✓				
24	Perasaan kenyang berlebihan walaupunhanya makan sedikit	✓				
25	Dehidrasi berat	✓				
26	Rasa nyeri pada tukak deodenum			✓		
27	Sakit pada tukak lambung		✓			

PENGISIAN TINGKAT KEPENTINGAN GEJALA JENIS PENYAKIT LAMBUNG
Gastroesophageal Reflux Disease (GERD)

No	Gastroesophageal Reflux Disease (GERD)	GSP	GP	GS	GB	GSB
1	Rasa nyeri dan rasa tidak nyaman pada perut		✓			
2	Rasa penuh didaerah lambung / daerah perut			✓		
3	Perut kembung			✓		
4	Rasa mual	✓				
5	Muntah	✓				
6	Rasa nyeri pada uluhati		✓			
7	Selera makan berkurang		✓			
8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas		✓			
9	Sering sendawa	✓				
10	Sering merdengar suara usus yang keras (borborigmi)					✓
11	Sembelit					✓
12	Diare					✓
13	Kehilangan berat badan secara mendadak				✓	
14	Demam atau meriang				✓	
15	Kejang perut					✓
16	Keluarnya BAB warna hitam pekat					✓
17	Pendarahan (BAB darah atau Muntah darah)					✓
18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher	✓				
19	Rasa ada makanan/minuman balik ke mulut	✓				
20	Mulut terasa asam dan pahit	✓				
21	Batuk menahun					✓
22	Serak dan tenggorakan sakit	✓				
23	Lesu (mudah lelah)		✓			
24	Perasaan kenyang berlebihan walaupunhanya makan sedikit	✓				
25	Dehidrasi berat				✓	
26	Rasa nyeri pada tukak deodenum					✓
27	Sakit pada tukak lambung		✓			

PENGISIAN TINGKAT KEPENTINGAN GEJALA JENIS PENYAKIT LAMBUNG
Gastroenteritis

No	Gejala Gastroenteritis	GSP	GP	GS	GB	GSB
1	Rasa nyeri dan rasa tidak nyaman pada perut		✓			
2	Rasa penuh didaerah lambung / daerah perut		✓			
3	Perut kembung	✓				
4	Rasa mual	✓				
5	Muntah	✓				
6	Rasa nyeri pada uluhati			✓		
7	Selera makan berkurang			✓		
8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas	✓				
9	Sering sendawa				✓	
10	Sering merdengar suara usus yang keras (borborigmi)	✓				
11	Sembelit			✓		
12	Diare	✓				
13	Kehilangan berat badan secara mendadak					✓
14	Demam atau meriang	✓				
15	Kejang perut					✓
16	Keluarnya BAB warna hitam pekat					✓
17	Pendarahan (BAB darah atau Muntah darah)					✓
18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher					✓
19	Rasa ada makanan/minuman balik ke mulut					✓
20	Mulut terasa asam dan pahit					✓
21	Batuk menahun					✓
22	Serak dan tenggorakan sakit					✓
23	Lesu (mudah lelah)		✓			
24	Perasaan kenyang berlebihan walaupunhanya makan sedikit				✓	
25	Dehidrasi berat				✓	
26	Rasa nyeri pada tukak deodenum					✓
27	Sakit pada tukak lambung					✓

PENGISIAN TINGKAT KEPENTINGAN GEJALA JENIS PENYAKIT LAMBUNG
Gastroparesis

No	Gejala Gastroparesis	GSP	GP	GS	GB	GSB
1	Rasa nyeri dan rasa tidak nyaman pada perut			✓		
2	Rasa penuh didaerah lambung / daerah perut		✓			
3	Perut kembung		✓			
4	Rasa mual			✓		
5	Muntah			✓		
6	Rasa nyeri pada uluhati				✓	
7	Selera makan berkurang		✓			
8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas			✓		
9	Sering sendawa	✓				
10	Sering merdengar suara usus yang keras (borborigmi)		✓			
11	Sembelit		✓			
12	Diare					✓
13	Kehilangan berat badan secara mendadak					✓
14	Demam atau meriang					✓
15	Kejang perut	✓				
16	Keluarnya BAB warna hitam pekat					✓
17	Pendarahan (BAB darah atau Muntah darah)					✓
18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher					✓
19	Rasa ada makanan/minuman balik ke mulut			✓		
20	Mulut terasa asam dan pahit					✓
21	Batuk menahun					✓
22	Serak dan tenggorakan sakit					✓
23	Lesu (mudah lelah)				✓	
24	Perasaan kenyang berlebihan walaupunhanya makan sedikit	✓				
25	Dehidrasi berat					✓
26	Rasa nyeri pada tukak deodenum					✓
27	Sakit pada tukak lambung					✓

PENGISIAN TINGKAT KEPENTINGAN GEJALA JENIS PENYAKIT LAMBUNG
Tukak lambung

No	Gejala Tukak lambung	GSP	GP	GS	GB	GSB
1	Rasa nyeri dan rasa tidak nyaman pada perut	✓				
2	Rasa penuh didaerah lambung / daerah perut		✓			
3	Perut kembung				✓	
4	Rasa mual		✓			
5	Muntah		✓			
6	Rasa nyeri pada uluhati	✓				
7	Selera makan berkurang		✓			
8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas	✓				
9	Sering sendawa				✓	
10	Sering merdengar suara usus yang keras (borborigmi)					✓
11	Sembelit				✓	
12	Diare		✓			
13	Kehilangan berat badan secara mendadak					✓
14	Demam atau meriang		✓			
15	Kejang perut					✓
16	Keluarnya BAB warna hitam pekat			✓		
17	Pendarahan (BAB darah atau Muntah darah)					✓
18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher					✓
19	Rasa ada makanan/minuman balik ke mulut					✓
20	Mulut terasa asam dan pahit					✓
21	Batuk menahun					✓
22	Serak dan tenggorakan sakit					✓
23	Lesu (mudah lelah)				✓	
24	Perasaan kenyang berlebihan walaupunhanya makan sedikit				✓	
25	Dehidrasi berat					✓
26	Rasa nyeri pada tukak deodenum				✓	
27	Sakit pada tukak lambung	✓				

Lampiran H. Validasi hasil diagnosa Sistem CBR Diagnosa Penyakit Lambung

Validasi Hasil Diagnosa Sistem
CBR Diagnosa Lambung

Hari/Tanggal Pengisian	:	Jumat / 29 September 2017
Nama Narasumber	:	dr. Devi Chintya Kumalarani
Pekerjaan	:	Dokter Manajemen Rumah Sakit Citra Husada
Alamat	:	Jln. soekarno hatta no. 36 Jember
Tanda Tangan	:	

Hasil pengisian validasi sistem akan digunakan untuk keperluan penelitian dalam rangka penyusunan Skripsi dengan judul ***CASE BASED REASONING MENGGUNAKAN ALGORITMA K-NEAREST NEIGHBOR (K-NN) UNTUK DIAGNOSA PENYAKIT LAMBUNG.*** *Penelitian ini dilaksanakan oleh Andre Bhaskoro Suprayogi (Mahasiswa Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Jember), dibawah bimbingan* Prof. Drs. Slamin, M.Comp.Sc., Ph.D, Diah ayu Retnani W, ST ., M.Eng.

Validasi Data Hasil Diagnosa Sistem

CBR Diagnosa Lambung

No	Nama	Usia	Gejala		Hasil Diagnosa Sistem	Diagnosa Pakar (Dokter)	
			Kode	Gejala yang dirasakan		Sesuai	Tidak Sesuai
1	andi	21	G1	Rasa nyeri dan rasa tidak nyaman pada perut	Gastroenteritis	✓	
			G4	Rasa mual			
			G5	Muntah			
			G11	Sembelit			
			G12	Diare			
			G13	Kehilangan berat badan secara mendadak			
			G14	Demam atau meriang			
			G22	Serak dan tenggorakan sakit			
			G23	Lesu (mudah lelah)			
2	anton	22	G3	Perut kembung	Grastitis (maag)	✓	
			G4	Rasa mual			
			G5	Muntah			
			G6	Rasa nyeri pada uluhati			
			G7	Selera makan berkurang			
			G14	Demam atau meriang			
			G16	Keluarnya BAB warna hitam pekat			
			G20	Mulut terasa asam dan pahit			
			G22	Serak dan tenggorakan sakit			
			G23	Lesu (mudah lelah)			
3	Dinnar	22	G3	Perut kembung	Gastroenteritis	✓	
			G4	Rasa mual			
			G14	Demam atau meriang			
			G15	Kejang perut			
			G16	Keluarnya BAB warna hitam pekat			
			G20	Mulut terasa asam dan pahit			
			G22	Serak dan tenggorakan sakit			
			G23	Lesu (mudah lelah)			
4	Dessy	40	G5	Muntah	Kanker lambung	✓	
			G13	Kehilangan berat badan secara mendadak			
			G14	Demam atau meriang			
			G15	Kejang perut			
			G16	Keluarnya BAB warna hitam pekat			
			G21	Batuk menahun			
5	Ahmad	20	G2	Rasa penuh didaerah lambung / daerah perut	Dispepsia	✓	
			G3	Perut kembung			
			G4	Rasa mual			
			G7	Selera makan berkurang			
			G8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas			
			G9	Sering sendawa			
			G11	Sembelit			
			G12	Diare			

No	Nama	Usia	Gejala		Hasil Diagnosa Sistem	Diagnosa Pakar (Dokter)	
			Kode	Gejala yang dirasakan		Sesuai	Tidak Sesuai
6	Sulaiman	25	G1	Rasa nyeri dan rasa tidak nyaman pada perut	Grastitis (maag)	✓	
			G2	Rasa penuh didaerah lambung / daerah perut			
			G3	Perut kembung			
			G4	Rasa mual			
			G5	Muntah			
			G6	Rasa nyeri pada uluhati			
			G7	Selera makan berkurang			
			G8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas			
7	Firdaus	26	G1	Rasa nyeri dan rasa tidak nyaman pada perut	Dispepsia		✓
			G3	Perut kembung			
			G6	Rasa nyeri pada uluhati			
			G7	Selera makan berkurang			
			G8	Rasa nyeri dan rasa tidak nyaman pada perut bagian atas			
			G9	Sering sendawa			
			G10	Sering merdengar suara usus yang keras (borborigmi)			
			G11	Sembelit			
			G15	Kejang perut			
			G23	Lesu (mudah lelah)			
			G25	Dehidrasi berat			
8	Ilzam	27	G4	Rasa mual	Tukak Lambung	✓	
			G5	Muntah			
			G6	Rasa nyeri pada uluhati			
			G7	Selera makan berkurang			
			G25	Dehidrasi berat			
			G26	Rasa nyeri pada tukak deodenum			
			G27	Sakit pada tukak lambung			
9	Ruli	28	G3	Perut kembung	GERD	✓	
			G14	Demam atau meriang			
			G15	Kejang perut			
			G18	Rasa nyeri terbakar dibelakang tulang dada (heartburn) yang menyebar ke leher			
			G19	Rasa ada makanan/minuman balik ke mulut			
			G20	Mulut terasa asam dan pahit			
			G21	Batuk menahun			
			G22	Serak dan tenggorakan sakit			
10	Rusli	29	G1	Rasa nyeri dan rasa tidak nyaman pada perut	Gastroparesis	✓	
			G3	Perut kembung			
			G4	Rasa mual			
			G5	Muntah			
			G7	Selera makan berkurang			
			G15	Kejang perut			
			G24	Perasaan kenyang berlebihan walaupunhanya makan sedikit			