ANALISIS DAN DESAIN SISTEM PAKAR PENANAMAN CABAI MERAH KERITING (CAPSICUM ANNUM) DENGAN POLA PERANCANGAN MODEL VIEW CONTROLLER (MVC)

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Disusun oleh :

RINDANG MAZDARANI HAKIKI

NIM. 0910680040

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

PROGRAM STUDI INFORMATIKA

MALANG

2014

LEMBAR PERSETUJUAN

ANALISIS DAN DESAIN SISTEM PAKAR PENANAMAN CABAI MERAH KERITING (CAPSICUM ANNUM) DENGAN POLA PERANCANGAN MODEL VIEW CONTROLLER (MVC)

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :

RINDANG MAZDARANI HAKIKI

NIM. 0910680040

Dosen Pembimbing I	Dosen Pembimbing II
Arief Andy Soebroto, ST., M.Kom	**Issa Arwani, S.Kom, M.Sc.**
NIP 197204251999031002	**NIP 83092206110074**

LEMBAR PENGESAHAN

ANALISIS DAN DESAIN SISTEM PAKAR PENANAMAN CABAI MERAH KERITING (CAPSICUM ANNUM) DENGAN POLA PERANCANGAN MODEL VIEW CONTROLLER (MVC)

SKRIPSI
KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :
RINDANG MAZDARANI HAKIKI
NIM. 0910680040

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 20 Januari 2014

Penguji I	Penguji II
Dr. Eng Herman Tolle, ST, MT.	**Ismiarta Aknuranda, ST, M.Sc, Ph.D**
NIP 19740823 200012 1 001	**NIK 74071906110079**

Penguji III

Aryo Pinandito, ST, M.MT

NIK 83051916110374

Mengetahui,

Ketua Program Studi Informatika

Drs. Marji, MT.

NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 24 Januari 2014
Mahasiswa,

Rindang Mazdarani H
0910680040

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, penulis dapat menyelesaikan laporan skripsi yang berjudul **"ANALISIS DAN DESAIN SISTEM PAKAR PENANAMAN CABAI MERAH KERITING (CAPSICUM ANNUM) DENGAN POLA PERANCANGAN MODEL VIEW CONTROLLER (MVC)".**

Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar – besarnya kepada :

1. Bapak Drs. Marji, MT selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya Malang.
2. Bapak Arief Andy S, ST, M.Kom selaku dosen pembimbing I selama pelaksanaan skripsi.
3. Bapak Issa Arwani, S.Kom, M.Sc selaku dosen pembimbing II selama pelaksanaan skripsi.
4. Bapak Sujiono SE, MM dan Ibu Suryati yang selalu memberi support dan doa.
5. Bapak Dr. Eng Herman Tolle, ST, MT, Bapak Ismiarta Aknuranda ST, M.Sc, Ph.D, dan Bapak Aryo Pinandito ST, M.MT selaku penguji skripsi.
6. Seluruh pihak yang telah memberi support penulis dalam penyelesaian skripsi ini yaitu Iqbal Luthfillah S.Kom, Frendy Galih Arga Dwi Arsa, Rona Mercylina Pramita, Raihana Surga Firdausy, Novelia Kharisma S S.Kom. Dian Arisandy S.Kom. Hafidz Rozaq N.J S.Kom, Thierry Rahman A S.Kom. Mustikadewi Prihastuti S.Kom, Yusuf Oktofani S.Kom, seluruh angkatan 2009 dan civitas PTIIK yang banyak membantu yang tidak bisa disebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih banyak kekurangan. Oleh karena itu, saran dan kritik membangun dari para pembaca senantiasa diharapkan. Semoga laporan ini dapat memberikan manfaat bagi semua pihak, Amin.

Malang, 24 Januari 2014

Penulis

ABSTRAKSI

Rindang Mazdarani H. 2014 : Analisis dan Desain Sistem Pakar Penanaman Cabai Merah Keriting (Capsicum Annum) dengan Pola Perancangan Model View Controller (MVC). Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing: Arief Andy Soebroto, ST., M.Kom dan Issa Arwani, S.Kom, M.Sc.

Penelitian ini menerapkan design pattern Model View Controller (MVC) dalam pengembangannya. MVC adalah design pattern atau arsitektur yang memisahkan antara data (model) dengan user interface. Arsitektur MVC merupakan model yang banyak dipakai dalam pengembangan aplikasi Java EE yang sebenarnya dan merupakan arsitektur paling maju di antara arsitektur yang lain. Sistem pakar penanaman cabai menjadi obyek studi dalam penelitian ini. Dengan menerapkan arsitektur MVC pada aplikasi sistem pakar penanaman cabai, sehingga aplikasi dapat dipecah menjadi tiga bagian penting sesuai fungsinya. Aplikasi ini telah diuji pada responden yang pernah menerapkan MVC pada perancangan aplikasinya, MVC diakui memberi kemudahan dalam pengembangan aplikasi. Pada aplikasi sistem pakar penanaman cabai merah keriting ini sebanyak 86% responden menyatakan bahwa aplikasi ini telah memenuhi syarat sebagai aplikasi yang menerapkan arsitektur MVC.

Kata kunci : MVC, Java EE, sistem pakar, penanaman cabai

ABSTRACT

Rindang Mazdarani H. 2014 : Analysis and Design Expert System of Red Chili (Capsicum annuum) Curl Planting with Design Pattern Model View Controller (MVC). Undergraduated Thesis of Informatics Engineering Program, Information Technology and Computer Science Program, University of Brawijaya, Malang. Supervisor: Subroto Andy Arief, ST., M.Kom and Issa Arwani, Kom, M.Sc.

This research applies design pattern Model View Controller (MVC) in its development. MVC is a design pattern or architecture that separates the data (model) and user interface. MVC architecture is a model that is widely used in the development of Java EE applications and an architecture which is actually the most advanced among the other architectures. Chili cultivation expert system into an object of study in this research. By implementing MVC architecture in chili cultivation expert system application, so the application can be broken down into three essential parts according to its function. This app has been tested on respondents who had implemented the MVC design applications, MVC is recognized to provide convenience in application development. On the application of expert system's curly red chili cultivation by 86% of respondents stated that this application has been qualified as applications that implement the MVC architecture.

Keywords: MVC, Java EE, expert systems, planting chilies

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam aplikasi *web*, arsitektur yang selama ini digunakan dikenal dengan nama *Model* 1 dan Model2 (MVC). *Model* 1 yaitu teknik pengembangan *web* yang semua fungsionalitas diserahkan pada sebuah obyek atau file. Penelitian sebelumnya, perancangan dan implementasi Sistem Pakar Penanaman Cabai Keriting [RMH-12] telah menerapkan *model* 1 dimana *view* dan *controller* disatukan dalam satu obyek. Keuntungan *model* 1 dalam aplikasi *web* adalah dapat dikerjakan dengan cepat apabila aplikasi tersebut tidak terlalu kompleks. Akan tetapi *model* 1 sudah tidak sesuai lagi apabila aplikasi menjadi kompleks. Kode program akan menjadi lebih panjang, rumit,sulit dipelajari oleh *programmer* lain, serta menyulitkan dalam perawatannya [NWD-10].

Model 2 (MVC) telah banyak diterapkan, seperti sistem pengamanan *database* [TWD-09], perancangan *web* order [SKD-13], serta perancangan aplikasi m-learning [BMS-06]. Penerapan arsitektur MVC ini banyak digunakan dalam pengembangan aplikasi Java EE dan efektif dalam semua aplikasi perangkat lunak [NWD-10]. *Model View Controller* (MVC) adalah *desain pattern* atau arsitektur yang memisahkan antara data (*model*) dengan *user interface*. MVC memisahkan antara bagian bisnis dan bagian presentasi. Aplikasi *web* dibagi menjadi tiga bagian sesuai fungsinya yaitu *model, view* dan *controller*. Keuntungan dari *model* 2 ini adalah sesuai untuk aplikasi yang sederhana sampai ke aplikasi yang kompleks, membantu dalam perawatan dan pengembangannya [NWD-10].

Penulis akan membangun aplikasi sistem pakar penanaman cabai dengan menerapkan pola perancangan MVC. Dengan menerapkan arsitektur MVC pada aplikasi sistem pakar penanaman cabai keriting aplikasi dapat dipecah menjadi tiga bagian penting sesuai fungsinya. Diharapkan dengan penerapan arsitektur MVC pada sistem pakar penanaman cabai keriting membantu dalam perawatan dan pengembangan aplikasi.

1.2 Rumusan Masalah

Rumusan masalah-masalah yang akan dibahas adalah :

1. Bagaimana perancangan Sistem Pakar Penanaman Cabai Merah dengan metode MVC?
2. Bagaimana implementasi aplikasi Sistem Pakar Penanaman Cabai Merah Berbasis Java EE dengan metode MVC ?
3. Bagaimana hasil pengujian dari aplikasi Sistem Pakar Penanaman Cabai Merah Berbasis Java EE dengan metode MVC ?

1.3 Batasan Masalah

Agar tidak menyimpang dari masalah yang akan dibahas, maka diperlukan batasan – batasan masalah sebagai berikut :

1. Aplikasi sistem pakar penanaman cabai sebagai objek studi kasus dari penerapan arsitektur MVC.
2. Framework yang digunakan dalam pengembangan aplikasi sistem menggunakan bahasa pemrograman berorientasi objek, mengintegrasikan Servlet, Hibernate dan JSP berbasis Java EE.
3. DBMS yang digunakan untuk penyimpanan adalah MySQL.
4. Server di hosting menggunakan localhost.
5. Pengujian sistem meliputi :
   - Pengujian unit menggunakan metode White Box
   - Pengujian validasi menggunakan metode Black Box
   - Pengujian User Acceptance Test (UAT)

1.4 Tujuan

Adapun tujuan yang diinginkan penulis dalam penelitian ini adalah sebagai berikut :

1. Menerapkan arsitektur MVC pada aplikasi sistem pakar penanaman cabai.
2. Merancang dan membangun aplikasi sistem pakar penanaman cabai dengan arsitektur MVC.
3. Menguji aplikasi sistem pakar penanaman cabai dengan arsitektur MVC.

1.5 Manfaat

Adapun manfaat dari penelitian ini adalah sebagai berikut :

1.Untuk penulis

- Menambah pengetahuan tentang pembuatan aplikasi sistem pakar yang menerapkan arsitektur MVC.
- Menambah pengalaman dalam perancangan sistem perangkat lunak sehingga dapat merancang sistem yang bermanfaat bagi masyarakat.

2.Untuk pembaca

- Pembaca dapat menganalisa dan mengembangkan sistem yang lebih baik dengan menerapkan arsitektur MVC.
- Menambah wawasan pengetahuan tentang cara merancang aplikasi sistem pakar penanaman cabai dengan arsitektur MVC.

1.6 Sistematika Pembahasan

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan tugas akhir secara garis besar yang meliputi beberapa bab, sebagai berikut.

BAB I : Pendahuluan

Menguraikan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika pembahasan.

BAB II : Kajian Pustaka dan Dasar Teori

Menguraikan tentang dasar teori pembuatan aplikasi sistem pakar penanaman cabai dengan arsitektur MVC.

BAB III : Metode Penelitian

Metode penelitian menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan tugas akhir yang terdiri dari studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis serta pengambilan kesimpulan.

BAB IV : Perancangan

Perancangan menguraikan analisis kebutuhan serta perancangan sistem.

BAB V : Implementasi

Membahas implementasi dari Sistem Pakar Penanaman Cabai Keriting dengan Pola Perancangan MVC sesuai dengan perancangan sistem yang telah dibuat.

BAB VI : Pengujian dan analisis

Memuat hasil pengujian dan analisis terhadap Sistem Pakar Penanaman Cabai Keriting dengan Pola Perancangan MVC yang telah direalisasikan.

BAB VII : Penutup

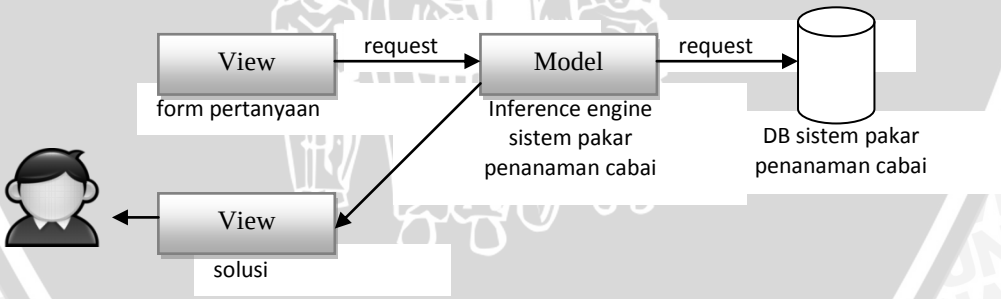
Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yag dikembangkan dalam skripsi ini serta saran – saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Pada bab ini membahas kajian pustaka dan landasan teori. Kajian pustaka membahas penelitian terdahulu dan yang diusulkan. Landasan teori diperlukan untuk membahas teori yang diperlukan dalam penelitian. Pada penelitian ini teori yang dibutuhkan adalah sistem pakar, *Model View Controller* (MVC), MySQL, *Hyper Text Markup Language* (HTML), dan Rekayasa Perangkat Lunak.

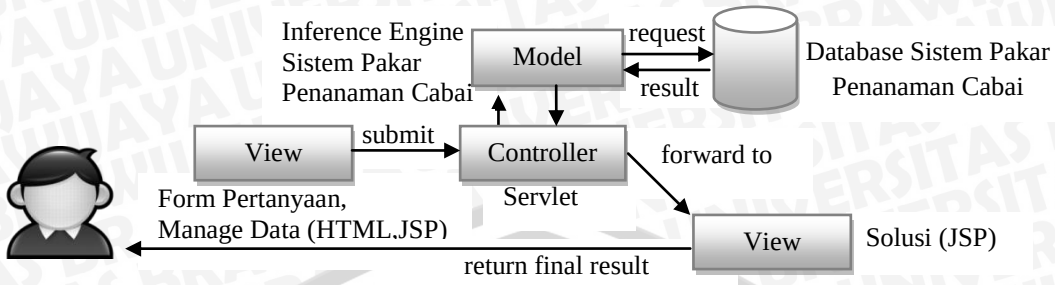
2.1 Kajian Pustaka

Pada penelitian terdahulu **"**Perancangan Aplikasi Sistem Pakar Teknik Budidaya Dalam Penanaman Dan Pemilihan Varietas Tanaman Cabai Merah Keriting" [RMH-12] telah merancang sistem pakar berbasis *web* namun belum menerapkan arsitektur MVC. Aplikasi yang telah dirancang pada penelitian terdahulu, hanya memisahkan *model* (data) dengan *view* (tampilan). *Logic request response* masih ditangani oleh *view*. Oleh karena itu penelitian tersebut menjadi obyek studi kasus untuk pengembangan penelitian ini dengan menerapkan arsitektur MVC. Berikut merupakan gambaran arsitektur dari penelitian terdahulu.



Gambar 2.1 Arsitektur Sistem Penelitian Terdahulu
Sumber : [RMH-12]

Pada gambar diatas, arsitektur sistem dari penelitian terdahulu terpisah menjadi *model dan view*. *Action request dan response* pada sistem ditangani langsung oleh *model* dan *view*. Sistem pada penelitian ini belum menerapkan arsitektur MVC. Oleh karena itu peneliti ingin menerapkan arsitektur MVC pada penelitian ini. Berikut arsitektur sistem yang menerapkan MVC.



Gambar 2.2 Arsitektur Sistem Penelitian Yang Diusulkan
Sumber : Perancangan

Pada gambar diatas, arsitektur sistem terbagi menjadi *model*, *view* dan *controller*. Sesuai bagiannya, *model* menangani data-data dan *business logic* dari sistem, *view* menangani tampilan dari sistem dan *controller* menangani *action request response* pada sistem. Penerapan arsitektur MVC pada penelitian ini bertujuan untuk memisahkan tugas dari masing-masing bagian sistem.

2.2 Sistem Pakar

Sistem pakar secara umum adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Atau dengan kata lain sistem pakar adalah sistem yang didesain dan diimplementasikan dengan bantuan bahasa pemrograman tertentu untuk dapat menyelesaikan masalah seperti yang dilakukan oleh para ahli. Diharapkan dengan sistem ini, orang awam dapat menyelesaikan masalah tertentu baik ‘sedikit’ rumit ataupun rumit sekalipun ‘tanpa’ bantuan para ahli dalam bidang tersebut. Sedangkan bagi para ahli, sistem ini dapat digunakan sebagai asisten yang berpengalaman [KUS-07].

Sistem pakar merupakan cabang dari *Artificial Intelligence* (AI) yang cukup tua karena sistem ini telah mulai dikembangkan pada pertengahan tahun 1960. Sistem pakar yang muncul pertama kali adalah *General-purpose problem solver* (GPS) yang dikembangkan oleh Newl dan Simon. Sampai saat ini sudah banyak sistem pakar. yang dibuat, seperti MYCIN, DENDRAL, XCON & XSEL, SOPHIE, Prospector, FOLIO, DELTA, dan sebagainya [KUS-07].

Berikut perbandingan antara sistem konvensional dengan sistem pakar [KUS-07].:

a. Sistem Konvensional
   1. Informasi dan pemrosesan umumnya digabung dalam satu program *sequential*.
   2. Program tidak pernah salah (kecuali pemrogramnya yang salah).
   3. Tidak menjelaskan mengapa *input* dibutuhkan atau bagaimana hasil diperoleh.
   4. Data harus lengkap.
   5. Perubahan pada program merepotkan.
   6. Sistem bekerja jika sudah lengkap.

b. Sistem Pakar
   1. *Knowledge base* terpisah dari mekanisme pemrosesan *(inference)*
   2. Program bisa melakukan kesalahan
   3. Penjelasan *(explanation)* merupakan bagian dari ES
   4. Data tidak harus lengkap
   5. Perubahan pada *rules* dapat dilakukan dengan mudah
   6. Sistem bekerja secara heuristik dan logic.

Tujuan pengembangan sistem pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk mensubstitusikan pengetahuan manusia ke dalam bentuk sistem, sehingga dapat digunakan oleh orang banyak.

2.3 Hyper Text Markup Language (HTML)

HTML merupakan protokol yang digunakan untuk mentransfer data atau document dari *web* server ke browser kita (Internet Explorer, Netscape Navigator, NeoPlanet, dll). HTML inilah yang memungkinkan Anda menjelajah internet dan melihat halaman *web* yang menarik [KKN-07:1].

HTML dokumen tersebut mirip dengan dokumen teks biasa, hanya dalam dokumen ini sebuah teks bisa memuat instruksi yang ditandai dengan kode atau lebih dikenal dengan TAG tertentu. Sebagai contoh jika ingin membuat teks ditampilkan menjadi tebal seperti: TAMPIL TEBAL, maka penulisannya dilakukan dengan cara: `<b>TAMPIL TEBAL</b>`. Tanda `<b>` digunakan untuk

mengaktifkan instruksi cetak tebal, diikuti oleh teks yang ingin ditebalkan, dan diakhiri dengan tanda untuk menonaktifkan cetak tebal tersebut.

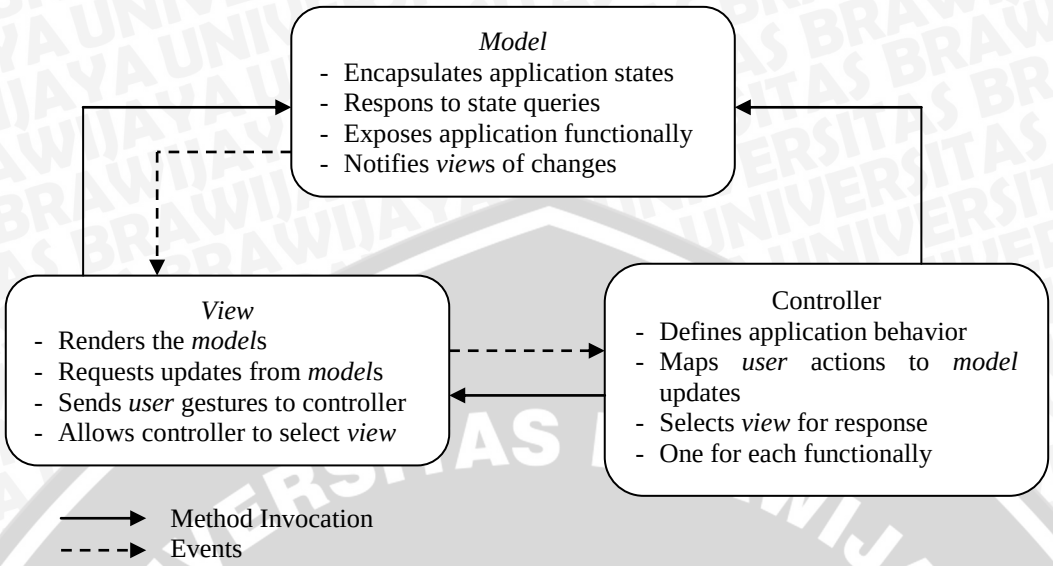
Secara garis besar, terdapat 4 jenis elemen dari HTML:

- *structural*. tanda yang menentukan level atau tingkatan dari sebuah teks (contoh, <h1>Golf</h1> akan memerintahkan browser untuk menampilkan "Golf" sebagai teks tebal besar yang menunjukkan sebagai Heading 1.
- *presentational*. tanda yang menentukan tampilan dari sebuah teks tidak peduli dengan level dari teks tersebut (contoh, boldface akan menampilkan bold. Tanda presentational saat ini sudah mulai digantikan oleh CSS dan tidak direkomendasikan untuk mengatur tampilan teks.
- *hypertext*. tanda yang menunjukkan pranala ke bagian dari dokumen tersebut atau pranala yang menunjukkan ke dokumen lain (contoh, Sample kode text tersebut menampilkan Sample sebagai sebuah hyperlink ke URL tertentu).
- Elemen *widget* yang membuat objek-objek lain seperti tombol (<button>), list (<li>), dan garis horizontal (<hr>).

2.4 *Model View* Controller (MVC)

MVC adalah *design pattern* atau arsitektur yang digunakan dala rekayasa perangkat lunak, dimana terjadi pemisahan yang jelas antara data *(model*) dengan *user interface* (*view*) [NWD-10]. Arsitektur MVC adalah sebuah pola yang terbukti membangun proyek secara lebih efektif. Hal itu dilakukan dengan memilah komponen antara *Model, View dan Controller* pada bagian – bagian dalam proyek [JMV-11].

Aplikasi apapun, bagian dalam kode yang sering mengalami perubahan adalah bagian *user interface*. *User interface* adalah bagian yang paling terlihat oleh *user* dan bagaimana ia berinteraksi dengan aplikasi, membuatnya menjadi titik fokus pengubahan berdasar kemudahan penggunaan. Pola MVC menyediakan sebuah solosi terhadap permasalahan tersebut dengan membagi aplikasi menjadi bagian – bagian tersendiri, *Model*, *View* dan *Controller*, memisahkan antar bagian tersebut dan membuat tata interaksi diantaranya.



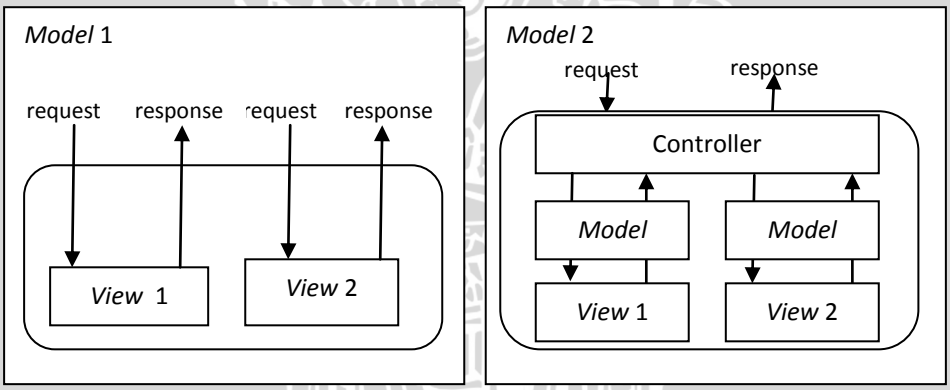
Gambar 2.3 Arsitektur MVC
Sumber : [JMV-11]

Model merupakan sesuatu yang merepresentasikan data, misalnya tabel database yang diolah (simpan, ubah, hapus) oleh *Controller* untuk ditampilkan (*View*). *Model* ini mengatur respon terhadap permintaan, serta memberi hak akses untuk memanipulasi data. *Framework* berbasis MVC menekankan pada pentingnya desain database yang valid, menggunakan *Model* ini memungkinkan *developer* melakukan *query* antar *database* bila diperintahkan oleh *Controller* (*logic*). Beberapa kelebihan menggunakan *Model*, yaitu dalam proses *maintenance* aplikasi yang lebih menguntungkan karena detail dari data dan operasinya dapat ditempatkan pada area yang ditentukan oleh *Model*. Keuntungan lainnya komponen *Model* dapat digunakan kembali oleh aplikasi lain yang memiliki kegunaan / fungsi yang hampir sama, karena telah dipisahkan secara total antara data dengan *interface*-nya.

View merupakan informasi yang ditampilkan ke *user* sebagai media *interface* (menggunakan HTML, CSS, javasript), jadi tidak berisi proses bisnis yang berhubungan dengan data pada *database*. Komponen *grafis* menyediakan *representasi* proses internal aplikasi dan menuntun alur interaksi *user* terhadap aplikasi yang ada. Contoh *View* adalah template dari tampilan aplikasi atau pada *web*site yang kita lihat, sehingga tidak ada layer lain yang berinteraksi dengan *user* kecuali pada layer *View* ini. Keuntungan *View* antara lain, memudahkan

desaigner sehingga bisa berkonsentrasi penuh pada desain tanpa harus memperhatikan hal – hal detail lainnya. Dan juga, memungkinkan ketersediaan multiple *interface* (Swing, *Web*, Console) dalam aplikasi namun mengeksekusi komponen *Model* sesuai fungsionalitas yang diharapkan.

Controller akan melakukan segala aktifitas proses bisnis dan aktifitas control lainnya seperti mengolah data dari *Model*, menyimpannya dalam variabel – variabel (manipulasi data) lalu menampilkannya pada *View*, benar atau tidaknya hasil olahan data akan sangat tergantung dari logika kerja aplikasi yang tersusun pada bagian *Controller* ini, sehingga *Controller* bisa disebut sebagai bagian yang paling signifikan dari aplikasi berbasis MVC. Dalam *Controller* ini menyediakan detail alur program dan bertanggung jawab menampung events dari *user* melalui *View* dan mengupdate komponen *Model* menggunakan data yang dimasukkan *user*.



Gambar 2.4 Perbedaan *Model* 1 dengan *Model* 2
Sumber : [NWD-10]

Gambar 2.4 merepresentasikan perbedaan *model* 1 dengan *model* 2. Dimana *model* 1 tampilan *web* serta penanganan request response ditangani sekaligus oleh satu obyek. Ketergantungan kode program tinggi, kode program dalam file akan sangat panjang, rumit sulit dipelajari programmer lain serta sulit dalam perawatannya apalagi untuk aplikasi yang kompleks. Sedangkan *model* 2 tugas terbagi sesuai dengan fungsinya, request dan response ditangani oleh controller, dan obyek *view* menangani *user interface*. Keuntungan dari penggunaan *model* 2 ini adalah lebih mudah dalam debugging sesuai untuk

aplikasi sederhana maupun kompleks, lebih mudah dirawat dan dikembangkan. Selain itu tingkat reusability kode program juga cukup tinggi.

2.4.1 Java Server Pages (JSP)

Java Server Pages (JSP) merupakan sebuah tenologi servlet-based yang digunakan pada *web* tier untuk menghadirkan *dynamic* dan *static content*. JSP merupakan *text-based* dan kebanyakan berisi *template* teks HTML yang digabungkan dengan spesifik *tags dynamic content* [ILM-12]. Dalam MVC JSP lebih sering digunakan sebagai komponen *view*. Kenapa menggunakan JSP? [JEN-11 : 1]

- JSPs merupakan dokumen text seperti HTML, para pengembang menghindari format dan manipulasi yang memungkinkan String yang sangat panjang untuk menghasilkan *output*. *Content* HTML sekarang tidak ditempelkan dengan berbagai macam kode dari Java. Hal ini membuatnya lebih mudah untuk dipelihara.
- JSPs lebih dikenal oleh semua orang dengan pengetahuan dari HTML, hanya dengan mempelajari *markup dynamic*. Hal ini membuatnya mungkin untuk para desainer site untuk membuat template HTML dari sebuah site, dengan para pengembang memprosesnya suatu saat nanti untuk memasukkan tags yang menghasilkan dynamic content. Hal ini juga memudahkan dalam pengembangan *web* page.
- JSPs memiliki built-in yang mendukung untuk penggunaan komponen software yang dapat digunakan kembali (JavaBeans). Hal ini tidak hanya membiarkan para pengembang menghindari kemungkinan menemukan kembali inti/kemudi dari tiap aplikasi, mempunyai software pendukung untuk memisahkan komponen software untuk menghandle logic promotes separation dari presentasi dan business logic yang compatible, dengan mengabaikan vendor atau sistem operasinya.
- Dalam kaitannya dengan cara kerja JSPs, mereka tidak membutuhkan kompilasi dari para pengembang. Kompilasi ini telah ada untuk kita pada

kontainer servlet. Modifikasi JSPs dideteksi secara otomatis. Hal ini secara relatif membuatnya mudah untuk dibangun.

2.4.2 Servlet

Servlet adalah sebuah class dalam bahasa pemgrograman Java yang digunakan untuk meningkatkan kapabilitas dari server sebagai host dari aplikasi yang diakses melalui request-response programming model (Diadaptasi dari tutorial J2EE). Servlet adalah sebuah class java yang meng-implement interface Servlet dan menerima request yang berasal dari class Java, web client, atau servlet lain yang membangkitkan response. Untuk memulai pembuatan servlet. Anda diharapkan mengerti mengenai pemrograman, konsep client-server, dasar-dasar HTML dan HTTP (HyperText Transfer Protocol). Untuk menciptakan sebuah servlet, Anda perlu untuk meng-import standard extension class dari javax.servlet dan javax.servlet.http ke program java Anda. Javax.servlet berisi framework dasar servlet dimana javax.servlet.http digunakan sebagai ekstensi dari framework servlet bagi servlet yang akan menjawab HTTP request [JSB-11].

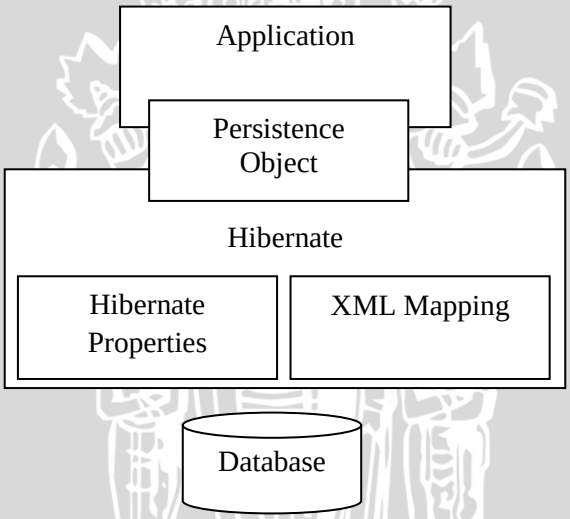
Java Servlet adalah komponen berbasis java yang digunakan untuk berkomunikasi dengan *web* server dengan *model* pemrograman request response. Tujuan utama spesifikasi servlet adalah menyediakan mekanisme yang handal untuk menangani request response dalam aplikasi berbasis *web*. Dalam desain pattern MVC Servlet lebih sering digunakan untuk komponen controller [NWD-10].

2.4.3 Framework Hibernate

Hibernate adalah suatu alat yang digunakan untuk melakukan *Object Relational Mapping (ORM)* pada lingkungan java. Istilah *ORM* mengacu pada teknik untuk memetakan data yang terletak pada objek ke database relational. ORM sendiri adalah untuk mengkonversi tipe data antara dua hal yang tidak kompatibel yaitu database relasional dan pemrograman berorientasi obyek bahasa. Dimana database relational berbicara mengenai kolom, record sedangkan pada pemrograman berorientasi objek pada attribut dan objek.

Hibernate tidak hanya mengatur pemetaan antara kelas kelas di java ke tabel di database, tapi juga memberikan mekanisme pengolahan data. Hal ini akan mempercepat proses pengembangan dibandingkan cara manual dengan melakukan penanganan data melalui *SQL* dan *JDBC*. Merupakan sebuah proyek Open Source profesional dan komponen terpenting dari JBoss Enterprises Middleware System. Berikut fitur-fitur yang ada pada hibernate :

1. Fitur pemetaan yang fleksibel
2. Fitur pemetaan seperti inheritance dan polymorphism
3. HQL yang bebas vendor database
4. Assosiasi join secara otomatis
5. Tidak akan mengupdate objek yang tidak dimodifikasi
6. Mengurangi penulisan baris kode program



Gambar 2.5 Struktur Hibernate
Sumber : [UNI-13]

2.5 MySQL

Sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management Sistem*) atau DBMS yang multithread, multi-*user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL [ACS-10 : 10].

Fitur-fitur MySQL antara lain [SEN-12] :

- Relational Database Sistem. Seperti halnya software database lain yang ada di pasaran, MySQL termasuk RDBMS.
- Arsitektur Client-Server. MySQL memiliki arsitektur client-server dimana server database MySQL terinstal di server. Client MySQL dapat berada di komputer yang sama dengan server, da n dapat juga di komputer lain yang berkomunikasi dengan server melalui jaringan bahkan internet.
- Mengenal perintah SQL standar. SQL (Structured Query Language) merupakan suatu bahasa standar yang berlaku di hampir semua software database. MySQL mendukung SQL versi SQL:2003.
- Mendukung Sub Select. Mulai versi 4.1 MySQL telah mendukung select dalam select (sub select).
- Mendukung *View*s. MySQL mendukung *view*s sejak versi 5.0.
- Mendukung Stored Prosedured (SP). MySQL mendukung SP sejak versi 5.0.
- Mendukung Triggers. MySQL mendukung trigger pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampua trigger pada versi 5.1.
- Mendukung replication.
- Mendukung foreign key.

2.6 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan. Pada definisi ini, ada dua istilah kunci [SOM-03:7] :

1. Disiplin rekayasa

Perekayasa membuat suatu alat bekerja. Perekayasa menerapkan teori, metode, dan alat bantu yang sesuai. Perekayasa menggunakannya dengan selektif dan selalu mencoba mencari solusi terhadap permasalahan, walaupun tidak ada teori atau metode yang mendukung. Perekayasa juga menyadari bahwa mereka harus bekerja dalam batasan organisasi dan keuangan, sehingga mereka berusaha mencari solusi dalam batasan – batasan ini.

2. Semua aspek produksi perangkat lunak

Rekayasa perangkat lunak tidak hanya berhubungan dengan proses teknis dari pengembangan perangkat lunak tetapi juga dengan kegiatan eperti manajemen proyek perangkat lunak dan pengembangan alat bantu, metode dan teori untuk mendukung produksi perangkat lunak. Rekayasa perangkat lunak memakai pendekatan yang sistematis dan terorganisir terhadap pekerjaan mereka karena cara ini seringkali paling efektif untuk menghasilkan perangkat lunak berkualitas tinggi. Rekayasa ini mencakup masalah pemilihan metode yang paling sesuai untuk satu set keadaan dan pendekatan yang lebih kreatif, informal terhadap pengembangan yang mungkin efektif pada beberapa keadaan [SOM-03:7].

***System Development Life Cycle* (SDLC)**

SDLC adalah keseluruhan proses dalam membangun sistem melalui beberapa langkah. Ada beberapa model SDLC. Model yang cukup populer dan banyak digunakan adalah waterfall. Beberapa model lain SDLC misalnya fountain, spiral, rapid, prototyping, incremental, build & fix, dan synchronize & stabilize. Dengan siklus SDLC, proses membangun sistem dibagi menjadi beberapa langkah dan pada sistem yang besar, masing-masing langkah dikerjakan oleh tim yang berbeda.

Dalam sebuah siklus SDLC, terdapat enam langkah. Jumlah langkah SDLC pada referensi lain mungkin berbeda, namun secara umum adalah sama. Langkah tersebut adalah [YUL-13].

- Analisis sistem, yaitu membuat analisis aliran kerja manajemen yang sedang berjalan
- Spesifikasi kebutuhan sistem, yaitu melakukan perincian mengenai apa saja yang dibutuhkan dalam pengembangan sistem dan membuat perencanaan yang berkaitan dengan proyek sistem
- Desain sistem, yaitu membuat desain aliran kerja manajemen dan desain pemrograman yang diperlukan untuk pengembangan sistem informasi

- Pengembangan sistem, yaitu tahap pengembangan sistem informasi dengan menulis program yang diperlukan
- Pengujian sistem, yaitu melakukan pengujian terhadap sistem yang telah dibuat
- Implementasi dan pemeliharaan sistem, yaitu menerapkan dan memelihara sistem yang telah dibuat

Gambar 2.6 Cycle SDLC
Sumber : [MNU-08]

Siklus SDLC dijalankan secara berurutan, mulai dari langkah pertama hingga langkah keenam. Setiap langkah yang telah selesai harus dikaji ulang, kadang-kadang bersama expert user, terutama dalam langkah spesifikasi kebutuhan dan perancangan sistem untuk memastikan bahwa langkah telah dikerjakan dengan benar dan sesuai harapan. Jika tidak maka langkah tersebut perlu diulangi lagi atau kembali ke langkah sebelumnya.

2.7 Pengujian Perangkat Lunak

Ujicoba software merupakan elemen yang kritis dari SQA dan merepresentasikan tinjauan ulang yang menyeluruh terhadap spesifikasi, desain dan pengkodean. Ujicoba merepresentasikan ketidaknormalan yang terjadi pada pengembangan software. Selama definisi awal dan fase pembangunan, pengembang berusaha untuk membangun software dari konsep yang abstrak sampai dengan implementasi yang memungkinkan. Para pengembang membuat serangkaian uji kasus yang bertujuan untuk ”membongkar” software yang mereka

bangun. Kenyataannya, ujicoba merupakan salah satu tahapan dalam proses pengembangan software yang dapat dilihat (secara psikologi) sebagai destruktif, dari pada sebagai konstruktif. Pengembang software secara alami merupakan orang konstruktif. Ujicoba yang diperlukan oleh pengembang adalah untuk melihat kebenaran dari software yang dibuat dan konflik yang akan terjadi bila kesalahan tidak ditemukan. Dari sebuah buku, Glen Myers menetapkan beberapa aturan yang dapat dilihat sebagai tujuan dari ujicoba [AYT-11]:

1. Ujicoba merupakan proses eksekusi program dengan tujuan untuk menemukan kesalahan.
2. Sebuah ujicoba kasus yang baik adalah yang memiliki probabilitas yang tinggi dalam menemukan kesalahan-kesalahan yang belum terungkap.
3. Ujicoba yang berhasil adalah yang mengungkap kesalahan yang belum ditemukan.

Sehingga tujuan dari ujicoba ini adalah mendesain serangkaian tes yang secara sistematis mengungkap beberapa jenis kesalahan yang berbeda dan melakukannya dalam waktu dan usaha yang minimum.

Whitebox Testing

Ujicoba *Whitebox* merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk menghasilkan kasus-kasus uji. Dengan menggunakan metode ujicoba *whitebox*, para pengembang software dapat menghasilkan kasus-kasus uji yang [AYT-11] :

1. Menjamin bahwa seluruh *independent paths* dalam modul telah dilakukan sedikitnya satu kali,
2. Melakukan seluruh keputusan logikal baik dari sisi benar maupun salah,
3. Melakukan seluruh perulangan sesuai batasannya dan dalam batasan operasionalnya
4. Menguji struktur data internal untuk memastikan validitasnya

Kegunaan dari pengujian software dikarenakan sifat kerusakan alami dari software itu sendiri, yaitu [AYT-11]:

1. Kesalahan logika dan kesalahan asumsi secara proposional terbalik dengan kemungkinan bahwa alur program akan dieksekusi. Kesalahan akan selalu ada

ketika mendesain dan implementasi fungsi, kondisi atau kontrol yang keluar dari alur utama. Setiap harinya pemrosesan selalu berjalan dengan baik dan dimengerti sampai bertemu ”kasus spesial” yang akan mengarahkannya kepada kehancuran.

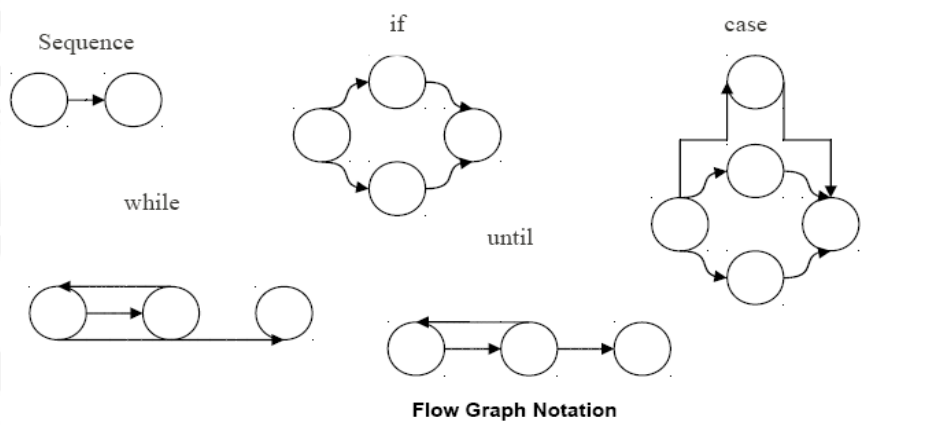
2. Sering percaya bahwa alur logikal tidak akan dieksekusi ketika dikenyataannya, mungkin akan dieksekusi dengan basis regular. Alur logika program biasanya berkebalikan dari intuisi, yaitu tanpa disadari asumsi mengenai alur kontrol dan data dapat mengarahkan pada kesalahan desain yang tidak dapat terlihat hanya dengan satu kali ujicoba.
3. Kesalahan *typographical* (cetakan) bersifat *random*. Ketika program diterjemahkan kedalam kode sumber bahasa pemrograman, maka akan terjadi kesalahan pengetikan. Banyak yang terdeteksi dengan mekanisme pemeriksaan sintaks, tetapi banyak juga yang tidak terdeteksi sampai dengan dimulainya ujicoba.

Karena alasan tersebut diatas, maka ujicoba whitebox testing perlu dilakukan selain blackbox testing.

Ujicoba Berbasis Alur (Basis Path Testing)

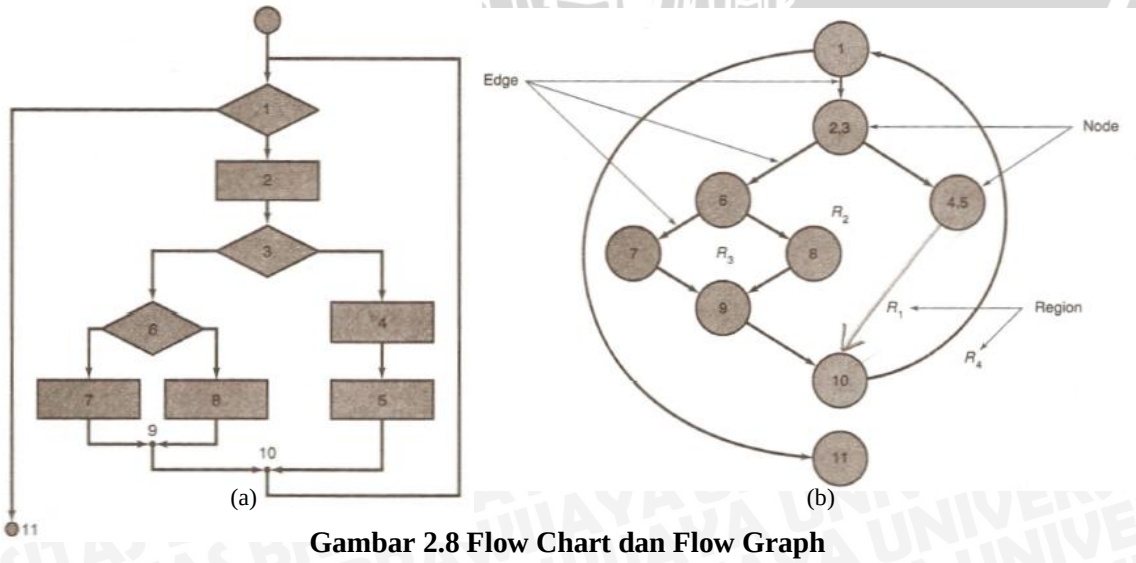
Ujicoba berbasis alur merupakan teknik ujicoba *whitebox* pertama yang diusulkan oleh Tom McCabe. Metode berbasis alur memungkinkan perancang kasus uji untuk menghasilkan ukuran kompleksitas logikal dari desain prosedural dan menggunakan ukuran ini untuk mendefinisikan himpunan basis dari alur eksekusi. Kasus uji dihasilkan untuk melakukan sekumpulan basis yang dijamin untuk mengeksekusi setiap perintah dalam program, sedikitnya satu kali selama ujicoba.

Notasi sederhana untuk merepresentasikan alur kontrol disebut graf alur (*flow graph*), digambarkan seperti gambar dibawah ini :



Gambar 2.7 Notasi Graf Alur
Sumber : [AYT-11]

Untuk mengilustrasikan kegunaan dari diagram alir dapat dilihat pada gambar dibawah ini. Gambar bagian (a) digunakan untuk menggambarkan struktur kontrol program, sedangkan gambar bagian (b) setiap lingkaran disebut dengan *flow graph node*, merepresentasikan satu atau lebih perintah prosedural. Urutan dari simbol proses dan simbol keputusan dapat digambarkan menjadi sebuah *node*, sedangkan anak panah disebut *edges*, menggambarkan aliran dari kontrol sesuai dengan diagram alir. Sebuah edge harus berakhir pada sebuah node walaupun tidak semua node merepresentasikan perintah prosedural. Area yang dibatasi oleh edge dan node disebut *region*, area diluar graph juga dihitung sebagai region.



Gambar 2.8 Flow Chart dan Flow Graph
Sumber : [AYT-11]

Langkah – langkah dari pengujian berbasis alur (Basis Path Testing) adalah sebagai berikut :

1) Gunakan rancangan atau kode program sebagai dasar, kemudian buat *flow graphnya*. Gambar 2.7 menunjukan tranformasi *flow chart* prosedur ke bentuk *flow graph*.
2) Tentukan kompleksitas siklomatik dari *flow graph* yang dibuat. *Cyclomatic complexity* adalah metriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program. Bila metriks ini digunakan dalam konteks metode pengujian *Basis Path*, maka nilai yang terhitung untuk *cyclomatic complexity* menentukan jumlah jalur independen (*independent path*) dalam *basis set* suatu program dan memberi batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua *statement* telah dieksekusi sedikitnya satu kali. Jalur independen adalah jalur yang melalui program yang mengenalkan sedikitnya satu rangkaian *statement* proses baru atau suatu kondisi baru. Untuk menentukan *cyclomatic complexity* bisa dilakukan dengan beberapa cara, diantaranya [AYT-11]:
   a) Jumlah *region* pada *flow graph* sesuai dengan *cyclomatic complexity*.
   b) *Cyclomatic complexity* V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node.*
   c) V(G) = P + 1, dimana P adalah jumlah *predicate node* yaitu *node* yang merupakan kondisi (ada 2 atau lebih *edge* akan keluar *node* ini).
3) Tentukan himpunan basis dari jalur – jalur yang independen secara linear.
4) Persiapkan kasus pengujian yang akan memaksa eksekusi masing – masing jalur di himpunan basis.

Pada penelitian ini menggunakan teknik *Basis Path Testing*.

BlackBox Testing

Pengujian blackbox berfokus pada persyaratan fungsional perangkat lunak. Disebut juga pengujian behavioral atau pengujian partisi. Pengujian blackbox memungkinkan perekayasa perangkat lunak mendapatkan serangkaian input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian blackbox berusaha menemukan [HAR-09] :

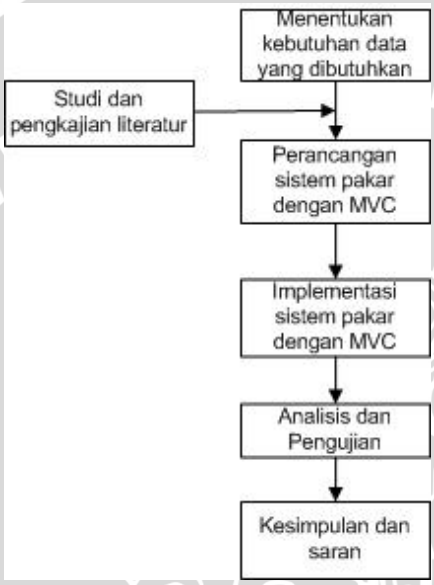
· Fungsi – fungsi yang tidak benar atau hilang
· Kesalahan interface
· Kesalahan dalam struktur data atau akses database eksternal.
· Kesalahan kinerja
· Inisialisasi dan kesalahan terminasi.

Dengan mengaplikasikan teknik blackbox maka kita menarik serangkaian test case yang memenuhi kriteria berikut [HAR-09] :

· Test case yang mengurangi, dengan harga lebih dari satu, jumlah test case tambahan yang harus di desain untuk mencapai pengujian yang dapat dipertanggungjawabkan.
· Test case yang member tahu kita sesuatu mengenai kehadiran atau ketidakhadiran kelas kesalahan, daripada member tahu kesalahan yang berhubungan hanya dengan pengujian spesifik.

BAB III
METODOLOGI PENELITIAN

Bab ini menjelaskan langkah-langkah kajian ilmiah yang akan dilakukan dalam penyusunan skripsi, yaitu studi literatur, metode pengambilan data, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis.



Gambar 3.1 Diagram alir metode penelitian
Sumber : Perancangan

3.1 Studi Literatur

Diperlukan adanya pembelajaran dan pemahaman terhadap teori – teori dasar tentang segala aspek yang mendukung dalam perancangan sistem. Teori – teori tersebut antara lain :

1.Penelitian terdahulu tentang sistem pakar penamanan cabai dan penerapan MVC pada aplikasi web.
2.Sistem Pakar.
3.HTML.
4.*Model View Controller* (MVC).
5.MySQL.
6.Rekayasa Perangkat Lunak.

3.2 Analisis Kebutuhan Sistem

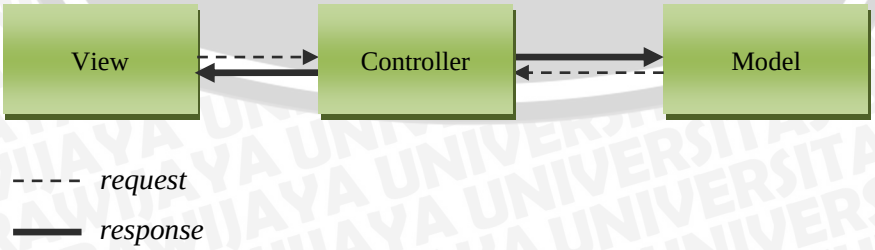
Analisis kebutuhan sistem bertujuan untuk mengetahui semua kebutuhan yang dibutuhkan oleh sistem. Metode analisis kebutuhan fungsional dari sistem menggunakan bahasa pe*model*an UML (*Unified Modelling Language*). UML adalah sebuah bahasa yang mendokumentasikan gambaran perancangan dari sebuah perangkat lunak berbasis *Object-Oriented*. Penelitian ini akan merancang sistem pakar penanaman cabai keriting dengan menerapkan pola perancangan MVC.

Pada penelitian sebelumnya penulis telah merancang sistem pakar dengan gambaran arsitektur seperti berikut :

Gambar 3.2 Arsitektur Sistem Pakar non MVC
Sumber : Perancangan

Request dan *response* pada sistem masih ditangani oleh masing-masing *view* dan *model*. Tidak ada perantara yang menjembatani antara *view* dan *model* dalam menangani action *request* dan *response* sehingga tugas *view* dan *model* selain menangani tugas masing-masing sebagai tampilan dan data, sekaligus berperan sebagai *controller*.

Dalam penelitian ini penulis merancang sistem pakar dengan menerapkan desain MVC pada sistem, dimana arsitektur dari sistem tersebut digambarkan sebagai berikut :

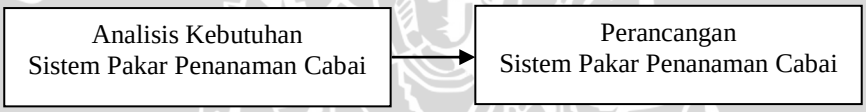


Gambar 3.3 Arsitektur Sistem Pakar dengan MVC
Sumber : Perancangan

Controller bertugas untuk menangani *action request response* dari *view* ke *model* dan sebaliknya. Bagian-bagian dari sistem akan terpisah sesuai dengan tugas masing – masing dengan adanya *controller* pada sistem, serta dengan didesainnya sistem dengan menerapkan arsitektur MVC. *View* bertugas untuk menangani *user interface*, *controller* bertugas menangani *action request response*, dan *model* menangani data dan *bussines logic* sistem.

3.3 Perancangan Perangkat Lunak

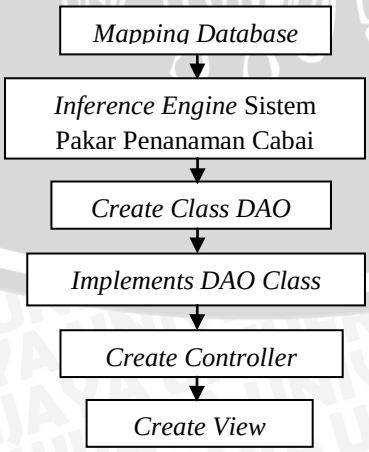
Perancangan perangkat lunak akan mendeskripsikan rancangan kebutuhan-kebutuhan dalam pembangunan aplikasi sistem. Diantaranya perancangan site map, perancangan data, perancangan class diagram, perancangan interface, perancangan inference engine, perancangan algoritma, dan perancangan arsitektur aplikasi sistem.



Gambar 3.4 Diagram Blok Perancangan Sistem
Sumber : Perancangan

3.4 Implementasi

Implementasi penelitian ini dimulai dari pembangunan aplikasi sistem pakar berdasarkan inference engine dan knowledge base yang telah dirancang.



Gambar 3.5 Diagram Blok Implementasi
Sumber : Perancangan

Kemudian menerapkan pola perancangan MVC pada sistem pakar tersebut. Implementasi selanjutnya adalah pembangunan web sesuai dengan perancangan user interface yang sudah direncanakan sehingga sistem dapat diaplikasikan untuk user.

3.5 Pengujian Sistem

Pengujian dari aplikasi sistem pakar, melalui dua jenis pengujian yaitu :

- Pengujian Unit Test

  Pengujian unit test dengan *White Box* dilakukan pada beberapa metode pada kelas dasar. Metode *White Box* yang digunakan adalah Basic Path. Metode ini akan mencari ukuran kompleksitas suatu prosedur serta mendefinisikan jalur eksekusi dasar.

- Pengujian Validasi

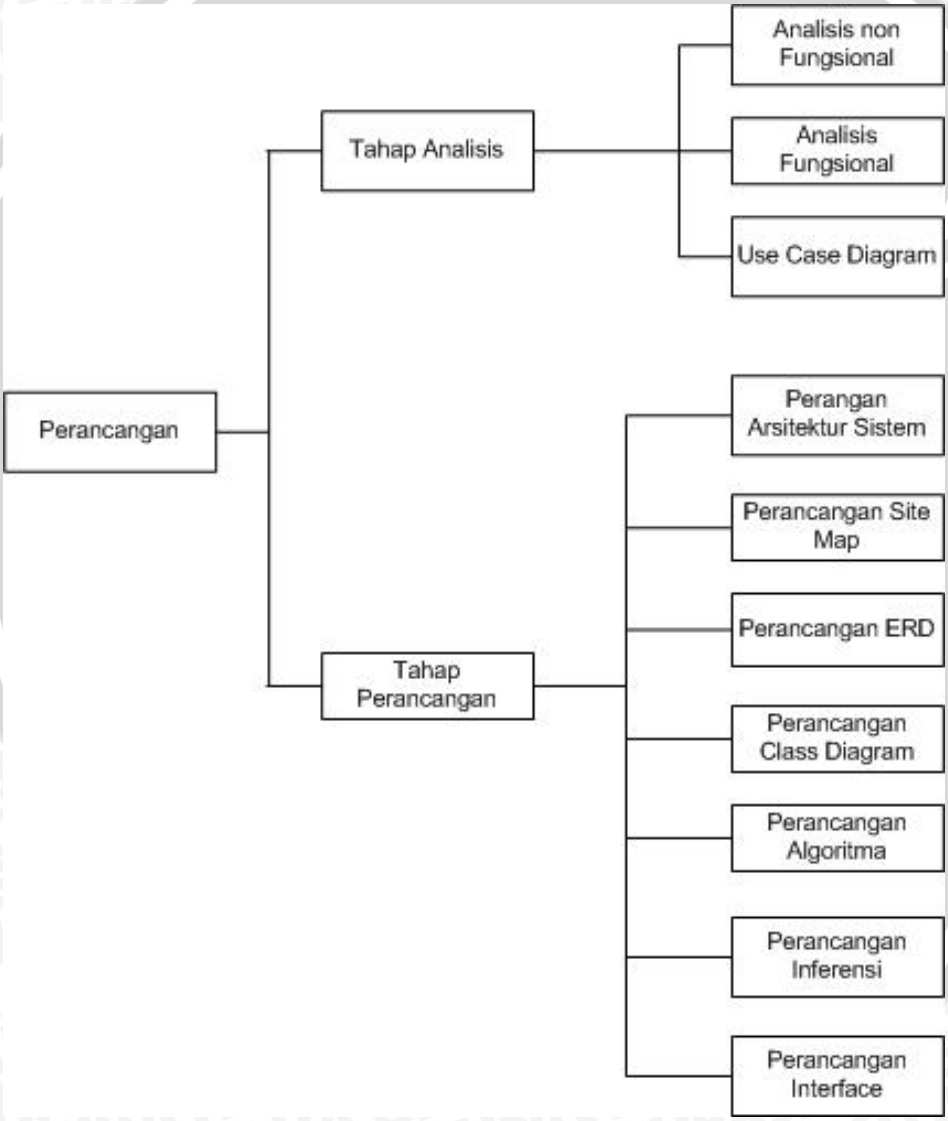
  Pengujian fungsionalitas dari aplikasi sistem menggunakan metode *Black Box*. Pengujian ini bertujuan untuk menunjukkan operasi aplikasi sistem tentang, apakah pemasukan data keluaran telah berjalan dengan baik sesuai yang di harapkan.

3.6 Pengambilan Kesimpulan dan Saran

Hasil dari metodologi penelitian dapat dilakukan dengan pengambilan kesimpulan setelah tahapan perancangan, implementasi dan pengujian perangkat lunak selesai dilakukan. Kesimpulan diperoleh dari hasil pengujian dan analisis sistem. Pengambilan saran bertujuan untuk memperbaiki kesalahan serta pengembangan perangkat lunak selanjutnya.

BAB IV
PERANCANGAN

Dalam sub bab ini akan dibahas mengenai analisis dan perancangan perangkat lunak pada aplikasi sistem. Analisis berisi tentang proses analisis kebutuhan sistem, analisis kebutuhan data, serta pemodelan. Sedangkan perancangan berisi perancangan sistem perangkat lunak.



Gambar 4.1 Diagram alir analisis dan perancangan
Sumber : Perancangan

4.1 Analisa Kebutuhan Perangkat Lunak

Dalam analisis kebutuhan perangkat lunak menspesifikasikan kebutuhan-kebutuhan yang diperlukan dalam pembangunan perangkat lunak.

4.1.1 Deskripsi Umum Sistem

Sistem yang akan dibangun adalah Sistem Pakar Penanaman Cabai Merah Keriting dengan Pola Perancangan MVC yang memisahkan dalam 3 bagian fungsi menjadi Model, View, dan Controller. Bagian Model berisi data-data yang dibutuhkan sistem pakar, serta inferensi sistem pakar. Data-data tersebut membutuhkan penyimpanan dalam database. Bagian View berisi *user interface*. Sedangkan bagian Controller berisi obyek yang menangani *request* dan *response*.

4.1.2 Fitur Utama Perangkat Lunak

Beberapa fitur dari sistem yaitu :

- Ubah data sistem pakar
- Tambah data sistem pakar
- Hapus data sistem pakar
- Inferensi sistem pakar

4.1.3 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional mendeskripsikan tentang kebutuhan dalam bentuk *identifier* untuk memudahkan pelacakan. *Identifier* dibedakan menjadi *Software Requirements Spesification (*SRS*)* dan *Interface Requirements Spesification (*IRS*)*. Tabel dibawah mendeskripsikan kebutuhan fungsional sistem dari admin serta user.

- **Admin**

Admin adalah orang yang berperan sebagai operator dalam sistem. Admin bertugas menambah, menghapus serta mengubah data pada sistem pakar. Berikut detail kebutuhan fungsional dari admin.

Tabel 4.1 Kebutuhan Fungsional Admin

Kode	Kebutuhan Fungsional	Keterangan
SRS_001	Admin harus melakukan login sebelum masuk ke halaman admin	Merupakan tampilan awal saat admin memilih menu admin yang akan masuk ke halaman admin
IRS_001	Perangkat lunak menerima input password	Masuk ke halaman admin
SRS_002	Admin bertugas menambah data sistem pakar	Merupakan tampilan yang berfungsi menambahkan data sistem pakar yang kemudian disimpan dalam database
SRS_003	Admin bertugas mengubah data sistem pakar	Merupakan tampilan yang berfungsi mengubah data sistem pakar yang kemudian disimpan dalam database
SRS_004	Admin bertugas menghapus data sistem pakar	Merupakan tampilan yang berfungsi menghapus data sistem pakar yang kemudian disimpan dalam database
IRS_002	Perangkat lunak menerima input data tambah, ubah dan hapus	Kembali ke halaman admin

Sumber : Perancangan

- **User**

User adalah orang yang menggunakan sistem. User dapat diperankan oleh siapa saja yang menggunakan sistem pakar penanaman cabai. User memiliki kewenangan untuk mendapatkan solusi dari pertanyaan yang telah diinputkan. Detail kebutuhan fungsional user dijelaskan pada tabel berikut.

Tabel 4.2 Kebutuhan Fungsional User

Kode	Kebutuhan Fungsional	Keterangan
SRS_005	User masuk ke sistem pakar	Merupakan tampilan awal yang terdapat pilihan dimana tombol sistem pakar yang dipilih maka akan masuk ke halaman sistem pakar
IRS_003	Perangkat lunak menerima input user	Ke halaman sistem pakar
SRS_006	User menginputkan data-data untuk memperoleh solusi	Merupakan tampilan halaman sistem pakar dimana terdapat data-data yang akan diolah inferensi sistem pakar untuk mendapatkan solusi dari data-data yang telah diinputkan
IRS_004	Perangkat lunak menerima input data dari user	Tampil solusi

Sumber : Perancangan

4.1.4 Analisis Kebutuhan non Fungsional

Pada analisis kebutuhan *non* fungsional mendeskripsikan kebutuhan-kebutuhan *non* fungsional yang harus dipenuhi oleh sistem. Tabel berikut mendeskripsikan kebutuhan non fungsional dari sistem

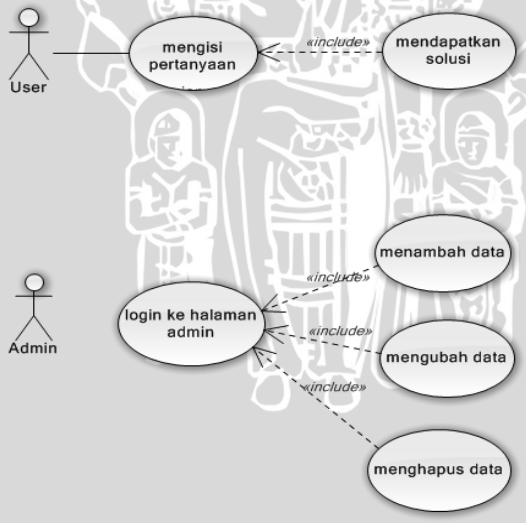
Tabel 4.3 Kebutuhan non Fungsional

Parameter	Deskripsi Kebutuhan
Avaliability	Sistem harus dapat beroperasi selama waktu yang ditentukan.
Usability	Sistem harus memberi kemudahan pada user dalam penggunaan
Security	Sistem harus aman, karena terdapat data penting. *Security* pada sistem ini menggunakan fungsi Login.
Maintanability	Sistem harus mudah direvisi jika diperlukan

Sumber:Perancangan

4.1.5 Use Case Sistem

Use case diagram mendeskripsikan kegiatan fungsional dari sistem dengan user. Aktor terdiri dari *user* dan *admin*, dimana kegiatan fungsional tersebut dideskripsikan pada gambar berikut.



Gambar 4.2 Use Case Diagram
Sumber : Perancangan

Aktor terdiri dari dua yaitu *user* dan *admin*. *Admin* adalah orang yang bertugas dalam mengelola data sistem pakar, seperti menambah, menghapus dan menambah data. *User* adalah semua orang yang menggunakan aplikasi sistem

pakar penanaman cabai. Tabel 4.4 mendeskripsikan definisi aktor dari *use case* pada gambar 4.2 :

Tabel 4.4 Definisi Aktor

Nama Aktor	Deskripsi Tugas
User	- Mengisi pertanyaan untuk memperoleh solusi
Admin	- Menambah data pengetahuan sistem pakar - Mengubah data sistem pakar - Menghapus data sistem pakar

Sumber : Perancangan

Sistem pakar penanaman cabai memiliki 5 *use case*, 2 merupakan *use case user* dan sisanya merupakan *use case admin*. Detail dari *use case* tersebut akan dideskripsikan pada skenario *use case*. Tabel 4.5 mendeskripsikan definisi *use case* pada gambar 4.2 :

Tabel 4.5 Definisi Use Case

Use Case	Deskripsi	Aktor
Mengisi pertanyaan	mengisikan pertanyaan untuk mendapatkan solusi	User
Login	masuk ke halaman admin	Admin
Menambah data	memasukkan data sistem pakar	Admin
Mengubah data	mengubah data sistem pakar	Admin
Menghapus data	menghapus data sistem pakar	Admin

Sumber : Perancangan

Definisi *use case* yang telah diuraikan pada tabel 4.5 diatas akan dijabarkan menjadi skenario pada tiap-tiap *use case*. Berikut uraian skenario *use case* dari tabel 4.5 :

- **Use Case Mengisi Pertanyaan**

Use case mengisi pertanyaan diperankan oleh user. Tujuan dari pengisian pertanyaan ini adalah untuk mendapatkan solusi berdasarkan fakta-fakta yang telah diinputkan oleh user. Berikut skenario pada *use case* mengisi pertanyaan.

Kondisi Awal : User berada pada halaman awal

Kondisi Akhir : User mendapat solusi dari pertanyaan yang diinputkan

Tabel 4.6 Use Case Mengisi Pertanyaan

User	Sistem
1.User menekan tombol sistem pakar	
	2. Sistem menampilkan halaman pertanyaan untuk user
3.User mengisi pertanyaan	
	4.Sistem memproses pertanyaan dari user
	5.Sistem menampilkan solusi
6.User mendapatkan solusi	

Sumber : Perancangan

- **Use Case Login**

Use case login diperankan oleh admin. Login bertujuan untuk mengakses halaman *manage data* dari sistem pakar penanaman cabai. Berikut skenario dari *use case login.*

Kondisi Awal : Admin berada pada halaman awal

Kondisi Akhir : Admin masuk ke halaman admin

Tabel 4.7 Use Case Login

Admin	Sistem
1.Admin masuk ke menu admin	
	2.Sistem menampilkan form login
3.Admin memasukkan password pada form	
	4.Sistem menampilkan halaman admin

Sumber : Perancangan

- **Use Case Menambah Data**

Use case menambah data diperankan oleh admin. *Use case* menambah data bertujuan untuk menambah data yang mengacu pada pohon inferensi sistem pakar yaitu meliputi penambahan data gejala dan data rule Berikut skenario dari *use case* menambah data.

Kondisi Awal : Admin berada pada halaman admin

Kondisi Akhir : Admin kembali lagi pada halaman admin

Tabel 4.8 Use Case Menambah Data

Admin	Sistem
1.Admin memilih data tabel yang akan ditambah	
	2.Sistem menampilkan data tabel
3.Admin menginputkan data yang ditambah	
4.Admin menekan tombol tambah	
	5.Sistem menerima input
	6.Sistem menyimpan data melalui perantara controller
	7.Sistem menyimpan data ke database
	8.Sistem menampilkan halaman admin

Sumber : Perancangan

- **Use Case Mengubah Data**

Use case mengubah data diperankan oleh admin. *Use case* mengubah data bertujuan untuk mengubah data-data yang ingin diubah dari sistem pakar penanaman cabai. Berikut skenario dari *use case* mengubah data.

Kondisi Awal : Admin berada pada halaman admin

Kondisi Akhir : Admin kembali pada halaman admin

Tabel 4.9 Use Case Mengubah Data

Admin	Sistem
1.Admin memilih data tabel yang akan diubah	
	2.Sistem menampilkan data tabel
3.Admin menginputkan data yang diubah	
4.Admin menekan tombol ubah	
	5.Sistem menerima input
	6.Sistem menyimpan data melalui perantara controller
	7.Sistem menyimpan data ke database
	8.Sistem menampilkan halaman admin

Sumber : Perancangan

- **Use Case Menghapus Data**

Use case menghapus data diperankan oleh admin. *Use case* menghapus data bertujuan untuk menghapus data-data yang diinginkan pada sistem pakar penanaman cabai. Berikut skenario dari *use case* menghapus data.

Kondisi Awal : Admin berada pada halaman admin

Kondisi Akhir : Admin kembali pada halaman admin

Tabel 4.10 Use Case Menghapus Data

Admin	Sistem
1.Admin memilih data tabel yang akan diubah	
	2.Sistem menampilkan data tabel
3.Admin memilih kolom data yang dihapus	
4.Admin menekan tombol hapus	
	5.Sistem menerima input
	6.Sistem menyimpan data melalui perantara controller
	7.Sistem menyimpan data ke database
	8.Sistem menampilkan halaman admin

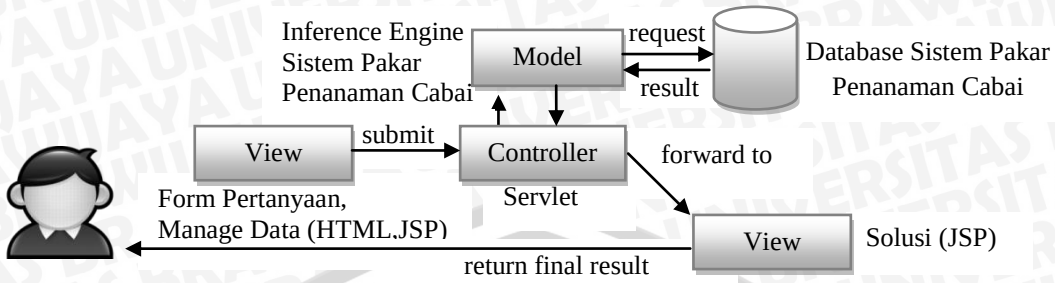
Sumber : Perancangan

4.2 Perancangan Perangkat Lunak

Pada bagian perancangan perangkat lunak mendeskripsikan tentang perancangan-perancangan kebutuhan dalam pembangunan perangkat lunak seperti perancangan arsitektur sistem pakar, perancangan site map, perancangan interface, perancangan class diagram, serta perancangan inferensi.

4.2.1 Perancangan Arsitektur Sistem

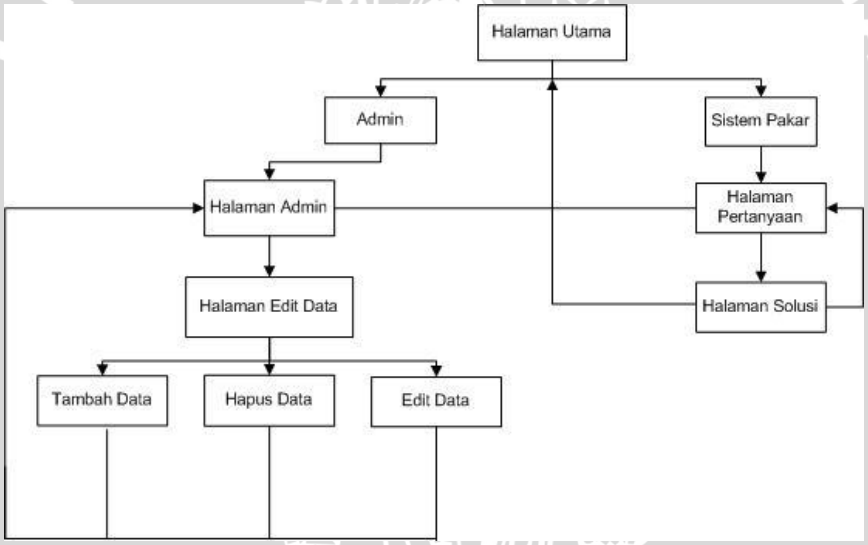
User mengisikan data form selanjutnya akan di submit ke servlet. Servlet mengirimkan request ke business logic, kemudian bussines logic request ke database, hasil disampaikan kembali ke servlet. Servlet merespon logic, kemudian forward ke view/halaman jsp sebagai final result yang akan ditampilkan ke user.



Gambar 4.3 Arsitektur Sistem Pakar Penanaman Cabai
Sumber : Perancangan

4.2.2 Perancangan Site Map

Perancangan sitemap bertujuan untuk menggambarkan alur dari halaman website sistem. Gambar 4.4 dibawah ini merupakan perancangan sitemap dari sistem pakar penanaman cabai merah keriting.



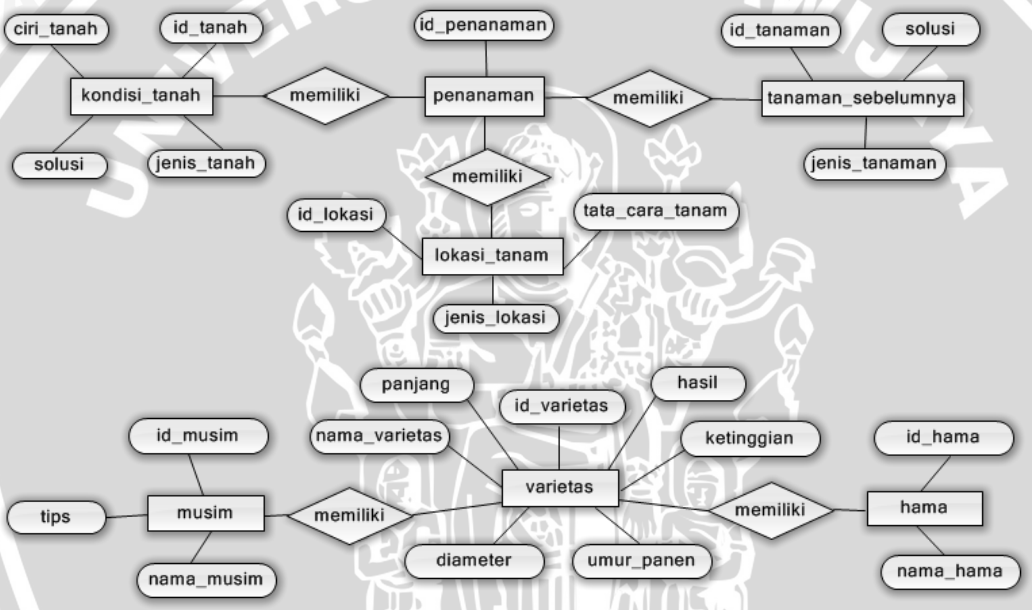
Gambar 4.4 Perancangan Site Map Sistem Pakar Penanaman Cabai
Sumber : Perancangan

Pada halaman utama terdapat menu admin dan tombol masuk ke sistem pakar. Jika pilihan admin maka akan dimintai password untuk masuk ke halaman admin. Ketika masuk pada halaman admin, admin dapat melakukan edit data seperti tambah data, hapus data dan edit data, yang kemudian apabila berhasil akan di *redirect* kembali ke halaman admin. Jika pilihan sistem pakar, maka akan muncul halaman pertanyaan yang kemudian user diminta untuk menginputkan jawaban pada pertanyaan yang telah tersedia, kemudian ketika submit jawaban

maka user akan mendapatkan solusi dari data-data yang telah diinputkan. Kemudian user dapat kembali lagi ke halaman pertanyaan untuk memperoleh solusi yang diinginkan, atau jika tidak ingin melanjutkan user dapat kembali ke halaman utama sistem pakar.

4.2.3 Perancangan Entity Relationship Diagram (ERD)

Pemetaan data didalam database direpresentasikan dalam bentuk permodelan ERD (entity relationship diagram). Struktur data dari aplikasi sistem pakar ditunjukan pada ERD berikut :



Gambar 4.5 Perancangan ERD Sistem Pakar Penanaman Cabai
Sumber : Perancangan

Berdasarkan struktur data dari gambar 4.5, tabel yang dibuat terdiri dari :

a). Tabel penanaman

b). Tabel varietas

c). Tabel tanaman sebelumnya

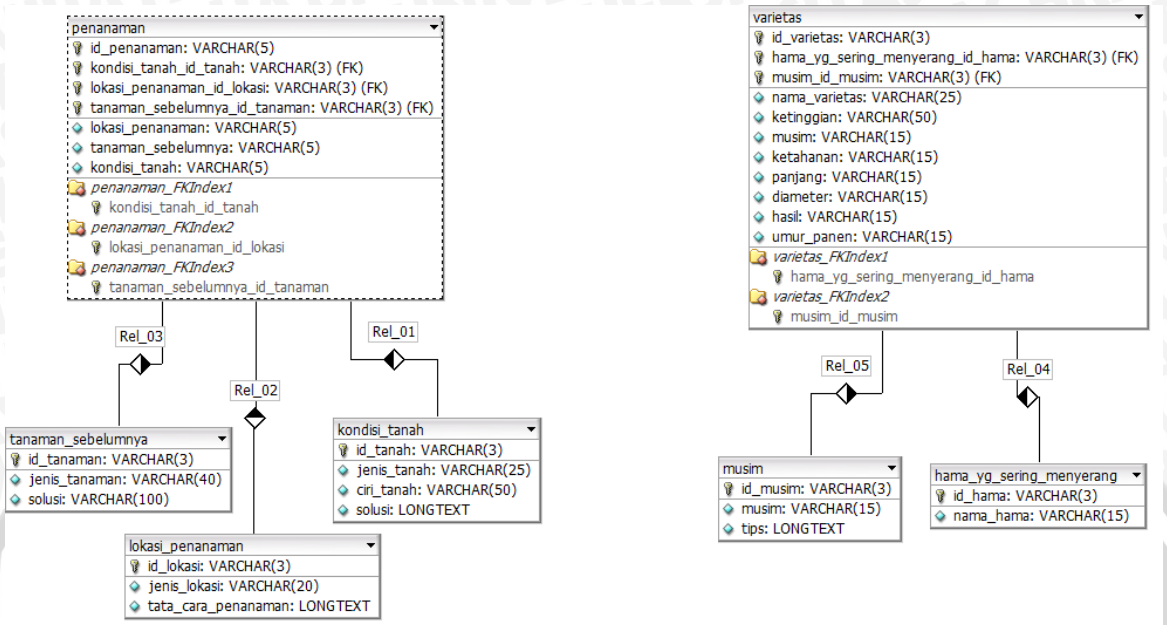
d). Tabel lokasi penanaman

e). Tabel kondisi tanah

f). Tabel musim

g). Tabel hama

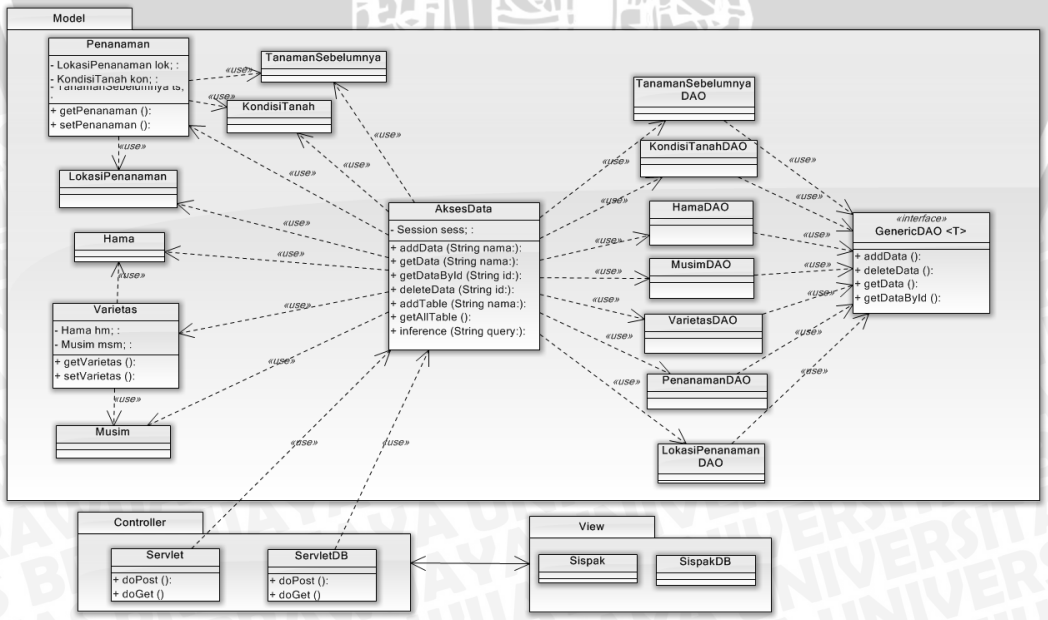
Gambar 4.6 akan merepresentasikan *physical database* berdasarkan struktur ERD yang telah dirancang pada gambar 4.5 sebagai berikut :



Gambar 4.6 Perancangan Physical Database
Sumber : Perancangan

4.2.4 Perancangan Class Diagram

Perancangan class diagram bertujuan untuk mendeskripsikan kelas-kelas yang akan dibuat pada pemrograman berorientasi objek.



Gambar 4.7 Perancangan Class Diagram
Sumber : Perancangan

Class diagram menjelaskan kelas-kelas serta relasi masing-masing kelas. Berikut merupakan perancangan class diagram dari sistem pakar penanaman cabai yang menerapkan pola perancangan MVC. Pada gambar class diagram diatas telah dipisahkan masing-masing class menjadi tiga bagian sesuai fungsinya yaitu package model, view dan controller.

Tabel dibawah ini merupakan penjelasan nama kelas serta keterangan dari class diagram gambar 4.7 sebagai berikut :

Tabel 4.11 Keterangan Class Diagram

Nama Class	Keterangan
TanamanSebelumnya, KondisiTanah, LokasiPenanaman, Hama, Musim, Penanaman dan Varietas	Merupakan kelas generate database menggunakan framework hibernate
GenericDAO<T>	Merupakan interface CRUD sistem pakar
TanamanSebelumnyaDAO, KondisiTanahDAO, LokasiPenanamanDAO, HamaDAO, MusimDAO, PenanamanDAO dan VarietasDAO	Merupakan kelas yang mengimplementasikan method dari interface GenericDAO<T>
AksesData	Merupakan class yang berisi inference serta kumpulan method action untuk akses ke database
Servlet, servletdb	Merupakan kelas yang berfungsi menjadi controller pada sistem pakar

Sumber : Perancangan

Dari rincian class diagram serta tabel diatas, terdapat 17 kelas dan 1 interface. Dimana terdapat 7 kelas hasil dari *generate database* yang dilakukan oleh hibernate, 1 interface GenericDAO, 7 kelas yang mengimplementasikan kelas GenericDAO serta kelas pendukung lainnya.

4.2.5 Perancangan Algoritma

Pada perancangan algoritma, akan dibahas pseudocode dari kode-kode yang telah dirancang dari perancangan class diagram. Berikut algoritma yang telah dirancang.

1. Algoritma Inference

```
NAMA ALGORITMA : Inferensi

DEKLARASI
String -> kondisi terpilih, Sistem pakar

DESKRIPSI

   MASUKAN : pilihan sistem pakar (penanaman atau varietas)

   PROSES
1  Jika sistem pakar yang dipilih penanaman maka
   Dapatkan pertanyaan dari rule penanaman
   a. Selama masih ada pertanyaan
            • Tampilkan pertanyan untuk setiap kondisi
            • Tampilkan pilihan jawaban ya atau tidak
            • Jika jawaban sama dengan ya maka
                i. Jawaban disimpan ke variable kondisi
                   terpilih
            • Jika Tidak
                i. Tampilkan pertanyaan selanjutnya
   b. Jawaban di kirim ke controller dan oleh controller jawaban
      diubah menjadi query dan dieksekusi
   c. Solusi ditampilkan
2  Jika sistem pakar yang dipilih varietas maka
   Dapatkan pertanyaan dari rule varietas
   a. Selama masih ada pertanyaan
            • Tampilkan pertanyan untuk setiap kondisi
            • Tampilkan pilihan jawaban ya atau tidak
            • Jika jawaban sama dengan ya maka
                i. Jawaban disimpan ke variable kondisi
                   terpilih
            • Jika Tidak
                i. Tampilkan pertanyaan selanjutnya
   b. Jawaban di kirim ke controller dan oleh controller jawaban
      diubah menjadi query dan dieksekusi
   c. Solusi ditampilkan
```

Gambar 4.8 Algoritma Inference
Sumber : Perancangan

Algoritma inference merupakan algoritma yang bertujuan untuk memperoleh solusi penanaman cabai. Algoritma inference bekerja dengan cara menampilkan pertanyaan yang berkaitan dengan penanaman cabai yang telah tersimpan pada tabel aturan. Jawaban kondisi terpilih akan dikirimkan ke controller untuk diproses dan hasil solusi nya ditampilkan

2. Algoritma Add Rule Sistem Pakar

```
NAMA ALGORITMA : Add Rule Sistem Pakar

DEKLARASI
Array -> gejala, solusi


DESKRIPSI

   MASUKAN : pilihan sistem pakar (penanaman atau varietas)

   PROSES
1  Jika sistem pakar yang dipilih penanaman maka
   a. Meng*generate* kondisi yang ada pada sistem pakar penanaman
   b. Mengisi kolom kondisi dan solusi penanaman cabai
   c. Data dikirmkan ke controller
   d. Kontroller memasukkan rule baru ke database
2  Jika sistem pakar yang dipilih varietas maka
   a. Meng*generate* kondisi yang ada pada sistem pakar varietas
   b. Mengisi kolom kondisi dan solusi varietas
   c. Data dikirmkan ke controller
   d. Kontroller memasukkan rule baru ke database

   KELUARAN : Data rule baru telah ditambahkan ke database rule
```

Gambar 4.9 Algoritma Add Rule Sistem Pakar
Sumber : Perancangan

Algoritma add rule berfungsi menambah rule sistem pakar penanaman cabai merah yang dibagi ke dalam dua bagian yaitu sistem pakar penanaman dan pemilihan varietas. Data dari rule yang ingin ditambahkan ke dalam knowledge sistem pakar diisi oleh admin. Kemudian data yang telah terisi dikirimkan ke controller untuk selanjutnya disisipkan ke dalam tabel aturan.

3. Algoritma Add Tabel (Gejala)

```
NAMA ALGORITMA : addGejala

DEKLARASI
namatabel -> String
sispak -> int

DESKRIPSI

MASUKAN : String, int

PROSES
1 Membuat tabel dengan parameter namatabel
2 Jika sispak = 1 (penanaman)
   Menambah data pada tabel data, sispak = penanaman, tabel =
   namatabel
   Menambah kolom namatabel pada tabel penanaman
```

```
3  Jika sispak = 2 (varietas)
   Menambah data pada tabel data, sispak = varietas, tabel =
   namatabel
   Menambah kolom namatabel pada tabel varietas

   KELUARAN : Gejala baru telah ditambahkan
```

Gambar 4.10 Algoritma add Gejala
Sumber : Perancangan

Algoritma add Table (Gejala) merupakan algoritma untuk menambah gejala sistem pakar. Secara dinamis gejala/tabel dari sistem pakar dapat ditambahkan pada aplikasi. Sistem pakar penanaman cabai merah keriting terdiri dari dua sistem pakar. Ketika melakukan penambahan gejala/tabel, terdapat pilihan sistem pakar penanaman atau varietas. Jika yang dipilih adalah penanaman, maka secara otomatis akan ditambahkan pada tabel data dimana kolom sispak berisi penanaman dan tabel berisi sesuai dengan parameter `namatabel` yang diinputkan. Kemudian secara otomatis ditambahkan kolom dengan nama sesuai parameter `namatabel` pada tabel penanaman. Sama halnya jika yang dipilih adalah varietas.

4. Algoritma Add Data

```
NAMA ALGORITMA : addData

DEKLARASI
Session -> sess
Object -> object

DESKRIPSI

  MASUKAN : sess, object

  PROSES
  Berdasarkan parameter, variable sess memanggil method
  saveOrUpdate untuk menyimpan data dalam bentuk object.

  KELUARAN : Data disimpan/diupdate dalam bentuk object
```

Gambar 4.11 Algoritma Add Data
Sumber : Perancangan

Algoritma addData merupakan algoritma yang bertujuan menyimpan dan mengupdate data pada sistem pakar. Algoritma diatas menjelaskan pemanggilan method `saveorUpdate` dengan parameter object yang dilakukan oleh variable `sess` untuk menyimpan/mengupdate data dalam bentuk object.

5. Algoritma Delete Data

```
NAMA ALGORITMA : deleteData

DEKLARASI
Session -> sess
String -> id
Query -> q

DESKRIPSI

   MASUKAN : sess, id

   PROSES
1  Variabel sess memanggil method createQuery dengan parameter
   id untuk mendapatkan id data yang akan dihapus kemudian query
   disimpan pada variabel q
2  Kemudian variabel q memanggil method ExecuteUpdate untuk
   mengupdate perubahan data.

   KELUARAN : Data dihapus sesuai id yang dimasukkan
```

Gambar 4.12 Algoritma Delete Data
Sumber : Perancangan

Algoritma deleteData bertujuan untuk menghapus data sistem pakar berdasarkan id. Algoritma diatas menjelaskan pemanggilan method `createQuery` dengan parameter id untuk mendapatkan id data yang akan dihapus kemudian disimpan pada variabel q. Kemudian variabel q memanggil method `ExecuteUpdate` untuk mengupdate perubahan data pada sistem pakar penanaman cabai.

6. Algoritma Get Data Nama Tabel

```
NAMA ALGORITMA : GetData PARAMETER

DEKLARASI :
Session -> sess

DESKRIPSI

   MASUKAN : sess

   PROSES
   Variabel sess memanggil method createQuery untuk mendapatkan
   seluruh data dan mengembalikan nilai dalam bentuk list

   KELUARAN : Nilai kembalian dalam bentuk list
```

Gambar 4.13 Algoritma Get Data
Sumber : Perancangan

Algoritma getData bertujuan untuk mendapatkan seluruh data pada sistem pakar penanmab cabai. Algoritma diatas menjelaskan proses pengambilan data tabel secara keseluruhan, data dikembalikan dalam bentuk list yang diterima oleh parameter `sess`.

7. Algoritma Get Data By Table Name

```
NAMA ALGORITMA : GetData PARAMETER sess IS Session, tabel IS
String

DEKLARASI :
Session -> Sess
String -> Id

DESKRIPSI

   MASUKAN : sess, nama tabel

   PROSES
     Variabel sess memanggil method createQuery dengan parameter
     nama tabel  untuk memperoleh data sesuai dengan nama tabel
     kemudian mengembalikannya dalam bentuk list

     KELUARAN : Nilai kembalian dalam bentuk list
```

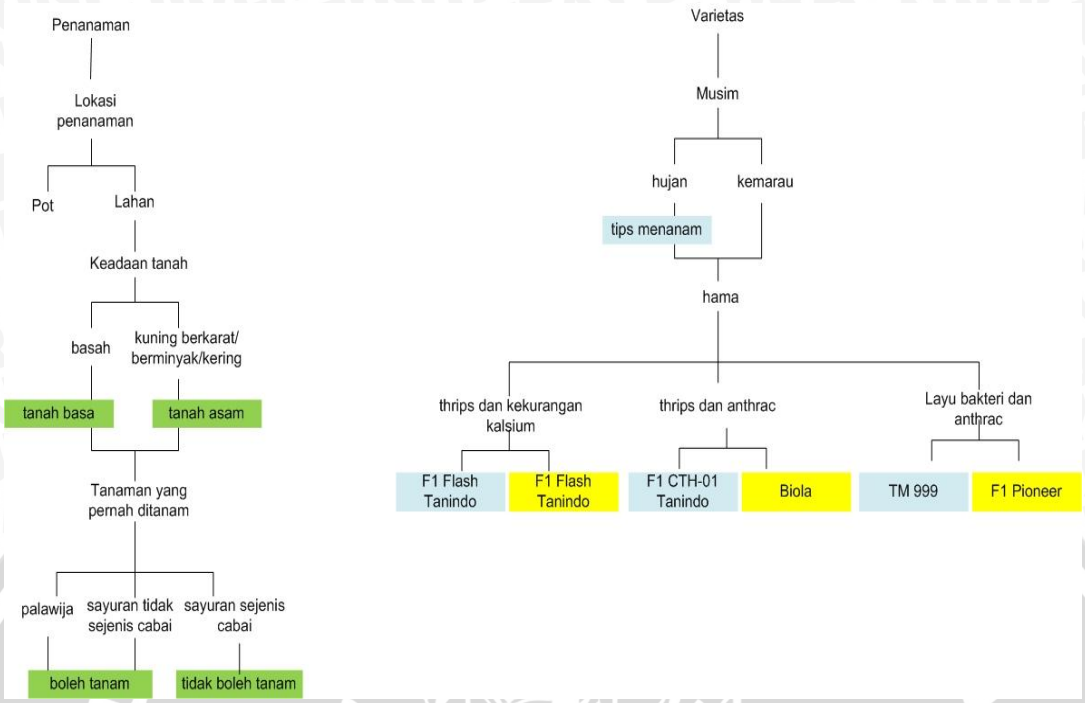
Gambar 4.14 Algoritma Get Data By Table Name
Sumber : Perancangan

Algoritma getDataByTableName bertujuan untuk memperoleh data sistem pakar berdasarkan nama tabel. Algoritma diatas menjelaskan proses pengambilan data tabel menurut parameter `namatabel`, data dikembalikan dalam bentuk list yang diterima oleh parameter `sess`.

4.2.6 Perancangan Inferensi

Perancangan inferensi bertujuan untuk mendeskripsikan alur kerja dari sistem pakar. Metode inferensi sistem pakar penanaman cabai menggunakan *forward chaining* yaitu pengambilan kesimpulan dari fakta-fakta yang telah ada. Berikut pohon perancangan inferensi dari sistem pakar penanaman cabai.

Gambar 4.15 merepresentasikan proses inferensi sistem pakar. Dimana proses inferensi terbagi menjadi 2 yaitu inferensi penanaman dan varietas. Pada inferensi penanaman ada tiga faktor yang menentukan solusi yaitu lokasi tanam, kondisi tanah dan tanaman yang pernah ditanam sebelumnya. Sedangkan pada inferensi varietas ada dua faktor yang menentukan solusi yaitu hama dam musim.



Gambar 4.15 Perancangan Inferensi Sistem Pakar Penanaman Cabai
Sumber : Perancangan

Dari gambar 4.15 pohon inferensi sistem pakar, dapat dibuat tabel sebagai alir dari aturan-aturan sistem pakar. Tabel aturan dibagi menjadi dua yaitu penanaman dan pemilihan varietas, sesuai pada pohon inferensi. Tabel dibawah merupakan tabel aturan dari penanaman.

Tabel 4.12 Rule Base Penanaman Cabai

No Aturan	Aturan	Kesimpulan
R01	IF Lokasi Penanaman lahan AND Keadaan Tanah garis kuning berkarat / berminyak ketika berair / kering AND Tanaman Sebelumnya palawija	Tanah asam Boleh tanam cabai
R02	IF Lokasi Penanaman lahan AND Keadaan Tanah garis kuning seperti berkarat berminyak ketika berair / kering AND Tanaman Sebelumnya sayuran sejenis cabai	Tanah asam Tidak boleh tanam cabai
R03	IF Lokasi Penanaman lahan AND Keadaan Tanah garis kuning seperti berkarat berminyak ketika berair / kering AND Tanaman Sebelumnya sayuran tidak sejenis cabai	Tanah asam Boleh tanam cabai
R04	IF Lokasi Penanaman lahan AND Keadaan Tanah basah AND Tanaman Sebelumnya palawija	Tanah basa Boleh tanam cabai

R05	IF Lokasi Penanaman lahan AND Keadaan Tanah basah AND Tanaman Sebelumnya sayuran sejenis cabai	Tanah basa Tidak boleh tanam cabai
R06	IF Lokasi Penanaman lahan AND Keadaan Tanah basah AND Tanaman Sebelumnya sayuran tidak sejenis cabai	Tanah basa Boleh tanam cabai

Sumber : Perancangan

Dari tabel diatas, terdapat 6 aturan berserta kesimpulan yang didapat dari fakta-fakta yang telah diperoleh. Alur dari sistem pakar penanaman cabai dideskripsikan pada tabel aturan penanaman diatas. Aturan diatas merupakan contoh sample dari data-data yang telah ada pada sistem. Tabel dibawah ini merupakan aturan dari pemilihan varietas. Aturan pemilihan varietas pada tabel dirancang sesuai pohon inferensi. Berikut tabel aturan pada pemilihan varietas.

Tabel 4.13 Rule Base Pemilihan Varietas Cabai

No Aturan	Aturan	Kesimpulan
R07	IF musim hujan AND hama thrips dan kekurangan kalsium	F1 Flash 750 Tanindo
R08	IF musim hujan AND hama thrips dan anthrac	F1 CTH-01 Tanindo
R09	IF musim hujan AND hama layu bakteri dan anthrac	TM 999 Seminis
R10	IF musim kemarau AND hama layu bakteri dan anthrac	F1 Pioneer
R11	IF musim kemarau AND hama thrips dan kekurangan kalsium	F1 Flash Tanindo
R12	IF musim kemarau AND hama thrips dan anthrac	Biola

Sumber : Perancangan

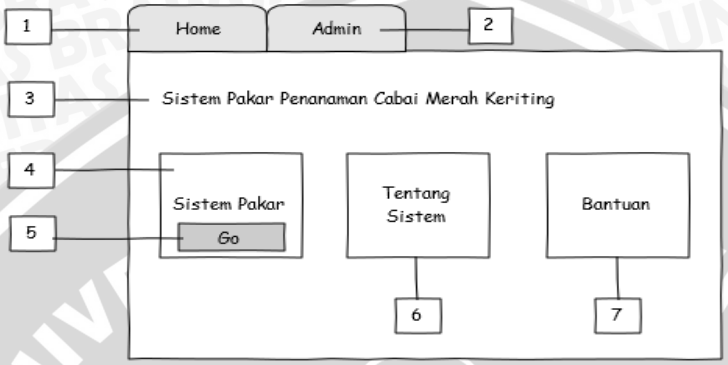
Dari tabel diatas, terdapat 6 aturan berserta kesimpulan yang didapat dari fakta-fakta yang telah diperoleh. Alur dari sistem pakar pemilihan varietas dideskripsikan pada tabel aturan pemilihan varietas diatas. Aturan diatas merupakan contoh sample dari data-data yang telah ada pada sistem.

Kelebihan dari sistem pakar penanaman cabai merah keriting dengan pola perancangan MVC ini adalah sistem dapat melakukan penambahan tabel (gejala) untuk membuat *rule* baru secara otomatis pada aplikasi tanpa melakukan perubahan struktur kode inference.

4.2.7 Perancangan Interface

Pada sub bab ini akan merepresentasikan perancangan interface dari sistem, berikut rincian dari interface sistem :

1. Halaman Utama

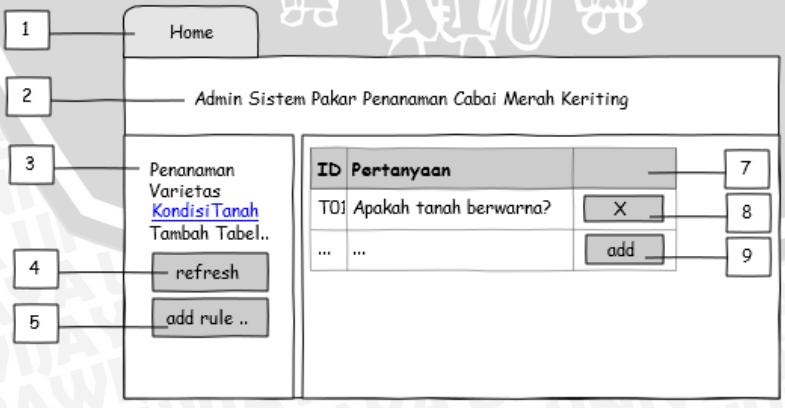


Gambar 4.16 Perancangan Halaman Utama
Sumber : Perancangan

Keterangan :

1. Merupakan menu untuk ke halaman utama.
2. Merupakan menu untuk masuk ke halaman admin.
3. Title dari aplikasi sistem.
4. Merupakan kotak sistem pakar, berisi pengertian umum sistem pakar.
5. Merupakan tombol untuk masuk ke halaman sistem pakar.
6. Berisi deskripsi tentang sistem pakar penanaman cabai merah keriting.
7. Berisi petunjuk untuk menggunakan sistem pakar.

2. Halaman Admin

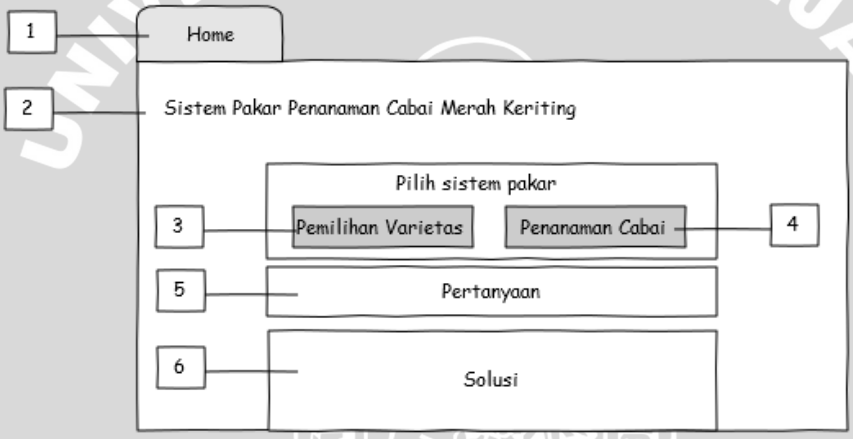


Gambar 4.17 Perancangan Halaman Admin
Sumber : Perancangan

Keterangan :

1. Merupakan menu untuk masuk ke halaman utama.
2. Title dari aplikasi sistem.
3. List tampilan tabel sistem pakar penanaman cabai.
4. Tombol untuk refresh halaman admin.
5. Tombol untuk menambah rule sistem pakar.
6. Tampilan dari data tabel yang dipilih.
7. Tombol untuk menghapus data tabel.
8. Tombol untuk menambah data tabel.

3. Halaman Sistem Pakar



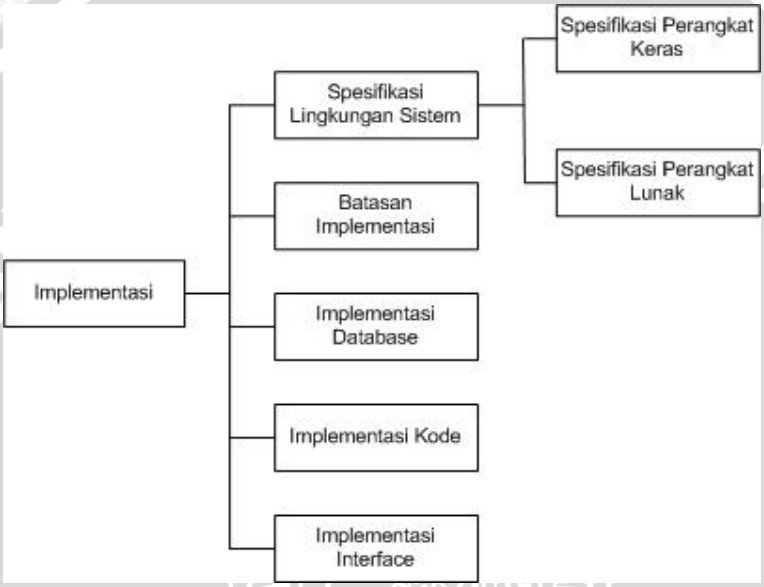
Gambar 4.18 Perancangan Halaman Sistem Pakar
Sumber : Perancangan

Keterangan :

1. Menu untuk masuk ke halaman utama.
2. Title aplikasi sistem.
3. Tombol untuk masuk ke sistem pakar pemilihan varietas.
4. Tombol untuk masuk ke sistem pakar penanaman cabai.
5. Tampilan untuk pertanyaan sistem pakar.
6. Tampilan untuk halaman solusi.

BAB V
IMPLEMENTASI

Pada bab implementasi membahas implementasi perangkat lunak berdasarkan perancangan yang telah dilakukan pada bab sebelumnya. Pada bab ini akan membahas tentang spesifikasi sistem, batasan implementasi, implementasi algoritma, implementasi database, serta implementasi interface.



Gambar 5.1 Diagram Implementasi
Sumber : Implementasi

5.1 Spesifikasi Lingkungan Sistem

Sistem aplikasi ini dibangun pada lingkungan implementasi perangkat keras dan perangkat lunak. Berikut uraian dari spesifikasi perangkat keras dan perangkat lunak yang digunakan oleh sistem pakar penanaman cabai merah keriting.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras mendeskripsikan kebutuhan apa saja yang diperlukan dalam implementasi sistem pakar penanaman cabai merah keriting. Berikut spesifikasi perangkat keras dalam implementasi sistem diuraikan pada tabel berikut :

Tabel 5.1 Spesifikasi Perangkat Keras

Notebook VAIO VPC-CW21FX	
Processor	Intel® Core™ i3 CPU M 330 @2.13GHz
Memory(RAM)	4 GB
Harddisk	500 GB
Monitor	14.0" (1366x768)

Sumber : Implementasi

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat keras mendeskripsikan kebutuhan apa saja yang diperlukan dalam implementasi sistem pakar penanaman cabai merah keriting. Spesifikasi perangkat lunak yang digunakan dalam implementasi sistem akan diuraikan pada tabel berikut :

Tabel 5.2 Spesifikasi Perangkat Lunak

Notebook VAIO VPC-CW21FX	
Operating System	Windows 7 Home Premium
Programming Language	Java, HTML
Software Development Kit	Netbeans IDE 7.2.1, Hibernate
DBMS	MySQL
Host	Apache
Programming Environment	Java Runtime Environment

Sumber : Implementasi

5.2 Batasan Implementasi

Berikut batasan-batasan dalam implementasi sistem :

1. Pembuatan aplikasi menggunakan Netbeans IDE 7.2.1
2. Pemetaan database menggunakan framework Hibernate
3. Implementasi dibatasi sampai hosting server localhost
4. DBMS sistem pakar penanaman cabai merah keriting menggunakan MySQL.

5.3 Implementasi Database

Dalam implementasi, tahap pertama yang dilakukan adalah pembuatan database yang berisi data knowledge base dari sistem pakar. Berikut struktur table-tabel berdasarkan perancangan ERD pada gambar 4.5

1. Tabel Hama

Tabel 5.3 merupakan tabel hama yang sering menyerang. Pada tabel ini berisi data-data hama yang sering menyerang cabai.

Tabel 5.3 Hama Yang Sering Menyerang

Field	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra	Aksi
id_hama	varchar(3)	latin1_swedish_ci		Tidak	None		
nama_hama	varchar(15)	latin1_swedish_ci		Tidak	None		

Sumber : Implementasi

2. Tabel Kondisi Tanah

Tabel 5.4 merupakan tabel kondisi tanah. Pada tabel ini berisi data – data kondisi tanah beserta ciri-ciri dan solusinya.

Tabel 5.4 Kondisi Tanah

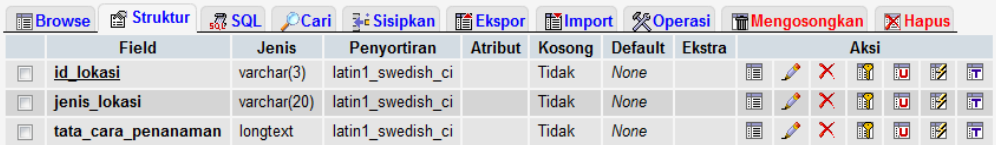
Field	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra	Aksi
id_tanah	varchar(3)	latin1_swedish_ci		Tidak	None		
jenis_tanah	varchar(25)	latin1_swedish_ci		Tidak	None		
ciri_tanah	varchar(50)	latin1_swedish_ci		Tidak	None		
solusi	longtext	latin1_swedish_ci		Tidak	None		

Sumber : Implementasi

3. Tabel Lokasi Penanaman

Tabel 5.5 merupakan tabel lokasi penanaman. Pada tabel ini berisi data-data lokasi penanaman cabai.

Tabel 5.5 Lokasi Penanaman



Field	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra	Aksi
id_lokasi	varchar(3)	latin1_swedish_ci		Tidak	None		
jenis_lokasi	varchar(20)	latin1_swedish_ci		Tidak	None		
tata_cara_penanaman	longtext	latin1_swedish_ci		Tidak	None		

Sumber : Implementasi

4. Tabel Musim

Tabel 5.6 merupakan tabel musim. Pada tabel ini berisi data-data musim serta tips-tips yang harus dilakukan pada suatu musim tertentu untuk menanam cabai yang baik.

Tabel 5.6 Musim

Browse | Struktur | SQL | Cari | Sisipkan | Ekspor | Import | Operasi | Mengosongkan | Hapus

	Field	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra	Aksi
☐	id_musim	varchar(3)	latin1_swedish_ci		Tidak	None		
☐	musim	varchar(15)	latin1_swedish_ci		Tidak	None		
☐	tips	longtext	latin1_swedish_ci		Tidak	None		

Sumber : Implementasi

5. Tabel Penanaman

Tabel 5.7 merupakan tabel penanaman. Pada tabel ini berisi data-data penanaman meliputi lokasi tanam beserta kondisi-tanah dan tanaman sebelumnya.

Tabel 5.7 Penanaman

Browse | Struktur | SQL | Cari | Sisipkan | Ekspor | Import | Operasi | Mengosongkan | Hapus

	Field	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra	Aksi
☐	lokasi_penanaman	varchar(5)	latin1_swedish_ci		Tidak	None		
☐	tanaman_sebelumnya	varchar(5)	latin1_swedish_ci		Tidak	None		
☐	kondisi_tanah	varchar(5)	latin1_swedish_ci		Tidak	None		
☐	id_penanaman	varchar(5)	latin1_swedish_ci		Tidak	None		

Sumber : Implementasi

6. Tabel Tanaman Sebelumnya

Tabel 5.8 merupakan tabel tanaman sebelumnya. Pada tabel ini berisi data-data jenis tanaman yang pernah ditanam sebelum menanam cabai.

Tabel 5.8 Tanaman Sebelumnya

Browse | Struktur | SQL | Cari | Sisipkan | Ekspor | Import | Operasi | Mengosongkan | Hapus

	Field	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra	Aksi
☐	id_tanaman	varchar(3)	latin1_swedish_ci		Tidak	None		
☐	jenis_tanaman	varchar(40)	latin1_swedish_ci		Tidak	None		
☐	solusi	varchar(100)	latin1_swedish_ci		Tidak	None		

Sumber : Implementasi

7. Tabel Varietas

Pada tabel 5.9 berisi data-data varietas beserta hama yang pernah menyerang, musim yang cocok untuk menanam varietas, serta rasa tiap-varietas.

Tabel 5.9 Varietas

Browse | Struktur | SQL | Cari | Sisipkan | Ekspor | Import | Operasi | Mengosongkan | Hapus

	Field	Jenis	Penyortiran	Atribut	Kosong	Default	Ekstra	Aksi
	id_varietas	varchar(3)	latin1_swedish_ci		Tidak	None		
	nama_varietas	varchar(25)	latin1_swedish_ci		Tidak	None		
	ketinggian	varchar(50)	latin1_swedish_ci		Tidak	None		
	musim	varchar(15)	latin1_swedish_ci		Tidak	None		
	ketahanan	varchar(50)	latin1_swedish_ci		Tidak	None		
	panjang	varchar(15)	latin1_swedish_ci		Tidak	None		
	diameter	varchar(15)	latin1_swedish_ci		Tidak	None		
	hasil	varchar(15)	latin1_swedish_ci		Tidak	None		
	umur panen	varchar(15)	latin1_swedish_ci		Tidak	None		

Sumber : Implementasi

5.4 Implementasi Kode

Pada implementasi kode, akan dibahas penerapan kode dari algoritma yang telah dibahas pada perancangan serta pada perancangan class diagram pada bab perancangan. Berikut kode yang telah di implementasikan.

1. Implementasi Inference

```
1   <% if (request.getParameter("sispak") != null) {
            if (request.getAttribute("gejala") == "Solusi") {
2            List<Object[]> hasil = (List)
    request.getAttribute("hasil");

3                   for (Object[] data : hasil) {
                        int loop = data.length;
                        int a = 1;
                        while (a < loop) {
                            out.print(data[a]+"<br>");
                            a++;
4                       }
                    }
5           } else {
6                   out.print(request.getAttribute("gejala"));
            }
7   int pertanyaan = Integer.parseInt (request.getParameter (
    "pertanyaan"));
8   int tabel = Integer.parseInt(request.getParameter("tabel"));
    %>

9   <form action="servlet" method="get">
       <input type="hidden" name="tabel" value="<%
       out.print(tabel = tabel + 1);%>"></input>
       <button type="submit" name="pertanyaan"
       value="0">IYA</button>
       <input type="hidden" name="sispak" value=
       "<%out.print(request.getParameter("sispak"));%>"> </input>
```

```
    <input type="hidden" name="savenilai" value="<%
    out.print((tabel - 1) + "_" + pertanyaan);%>"></input>
    <input type="hidden" name="save" value="true">
</form>
<form action="servlet" method="get">
    <input type="hidden" name="sispak"
    value="<%out.print(request.getParameter("sispak"));%>">
    </input>
    <input type="hidden" name="tabel" value="<%
    out.print(tabel - 1);%>"></input>
    <button type="submit" name="pertanyaan" value="<%
     out.print(pertanyaan = pertanyaan + 1);%>">TIDAK</button>
</form>
```

Gambar 5.2 Implementasi Inference Engine
Sumber : Implementasi

Gambar 5.2 merupakan implementasi Inference Engine sistem pakar cabai berdasarkan perancangan algoritma pada gambar 4.8. Baris 2 digunakan untuk menangkap respon solusi sistem pakar dari controller dalam bentuk List . Baris 3-4 berfungsi untuk menampilkan setiap data dari List solusi. Baris 6 berfungsi untuk menampilkan pertanyaan tentang kondisi persiapan sebelum menanam cabai. Baris 7-8 berfungsi untuk mendapatkan jumlah pertanyaan yang mungkin terdapat dalam suatu rule. Baris 9-10 menampilkan pilihan jawaban 'YA' ketika jawaban ya dipilih maka kondisi terpilih akan disimpan dan menampilkan pertanyaan selanjutnya dari bagian lain contohnya dari pertanyaan hama ke lokasi lahan. Baris 11-12 menampilkan jawaban 'TIDAK'. Jika jawaban tidak terpilih maka pertanyaan akan berganti kekondisi selanjutnya dalam bagian yg sama.Contohnya pertanyaan tentang tanah asam jika tidak maka muncul pertanyaan tentang tanah basa.

2. Implementasi Add *Rule* Varietas

```
1   else if (request.getAttribute("halaman") == "addvarietas") {

2   List<String> getdatakolom = (List)
    request.getAttribute("getdatakolom");

3   out.print("<form action='servletdb' method='get'>");
    out.print("<table>");
    out.print("<tr><td style='width:80px'>Id Varietas</td>
    <td><input type='text' name='idvarietas'></td></tr>");
    out.print("<tr><td style='width:80px'>Nama Varietas</td>
    <td><input type='text' name='varietas'></td></tr>");
    out.print("<tr><td style='width:80px'>Ketinggian</td>
    <td><input type='text' name='ketinggian'></td></tr>");
```

```
    out.print("<tr><td style='width:80px'>Diameter</td>
    <td><input type='text' name='diameter'></td></tr>");
    out.print("<tr><td style='width:80px'>Panjang</td><td><input
    type='text' name='panjang'></td></tr>");
    out.print("<tr><td style='width:80px'>Hasil</td><td><input
    type='text' name='hasil'></td></tr>");
    out.print("<tr><td style='width:80px'>Lama
    Panen</td><td><input type='text'
    name='lamapanen'></td></tr>");
    out.print("<tr><td style='width:80px'>Solusi</td><td><input
    type='text' name='solusi'></td></tr>");

    int jumlahadd = 1;
        for (String data : getdatakolom) {
            List<Object[]> get = (List)
            request.getAttribute(data);
            out.print("<td style='width:80px'>" + data +
            "</td>");
            out.print("<td><select name='kolom-" + jumlahadd +
            "'>");

        for (Object[] data2 : get) {
                out.print("<option value='" + data2[0] + "'>" +
            data2[1] + "</option>");
            }
            out.print("</select></td></tr>");
            jumlahadd++;
        }
4   out.print("</table>");
    out.print("<input type='hidden' name='jml' value ='" +
    jumlahadd + "' > ");
5   out.print("<button type ='submit'value ='addvarietas'name
    ='addrule'>Save</button >");
    out.print("</form>");
    }

    // tambah rule penanaman
6   else if (request.getAttribute("halaman") == "addpenanaman")
    {
7       List<String> getdatakolom = (List)
    request.getAttribute("getdatakolom");
    out.print("<form action='servletdb' method='get'>");
    out.print("<table>");
    out.print("<tr><td style='width:80px'>Id
8   Penanaman</td><td><input type='text'
    name='idpenanaman'></td></tr>");
    out.print("<tr>");

        int jumlahadd = 1;
        for (String data : getdatakolom) {
            List<Object[]> get = (List)
            request.getAttribute(data);
            out.print("<td style='width:80px'>" + data +
            "</td>");
            out.print("<td><select name='kolom-" + jumlahadd +
            "'>");
```

```
            for (Object[] data2 : get) {
                out.print("<option value='" + data2[0] + "'>" +
                data2[1] + "</option>");
            }
            out.print("</select></td></tr>");
            jumlahadd++;
9       }
    out.print("<input type='hidden' name='jml' value='" +
    jumlahadd + "'>");
    out.print("<tr><td style='width:80px'>Solusi</td><td><input
    type='textarea' name='solusi'></td></tr>");
    out.print("</table>");
    out.print("<button type='submit' value='addpenanaman'
10  name='addrule'>Save</button>");
    out.print("</form>");
    }
```

Gambar 5.3 Implementasi Add Rule Varietas
Sumber : Implementasi

Gambar 5.3 merupakan kode add rule untuk pemilihan varietas berdasarkan perancangan algoritma pada gambar 4.9. Baris 1 untuk menyeleksi apakah penambahan rule terjadi di sistem pakar pemilihan varietas. Baris 2 untuk mendapatkan jumlah kolom yang harus disi dalam penambahan rule sistem pakar pemilihan varietas cabai. Baris ke 3-4 menampilkan kolom yang harus diisi. Baris ke 5 berfungsi sebagai tombol, jika *button* tersebut di tekan maka data data kolom rule akan dikirimkan ke controller dan disimpan dalam database

Baris 6 untuk menyeleksi apakah penambahan rule terjadi di sistem pakar pemilihan penanaman cabai. Baris 7 untuk mendapatkan jumlah kolom yang harus disi dalam penambahan rule sistem pakar penanaman cabai. Baris ke 8-9 menampilkan kolom yang harus diisi. Baris ke 10 berfungsi sebagai tombol, jika *button* tersebut di tekan maka data data kolom rule akan dikirimkan ke controller dan disimpan dalam database.

3. Implementasi Add Table (Gejala)

```
    public void addTable(Session sess, String namatabel, int
    sispak) {
1           try {
                Query q = sess.createSQLQuery("CREATE TABLE IF NOT
    EXISTS  " + namatabel + "(id varchar(4) primary key,pertanyaan
    varchar(100))");
                Query y, z;
                //jika pilihan sispak = penanaman, memasukkan
    namatabel ke tabel data dan menyisipkan kolom ke tabel
```

```
    penanaman dst
2                   if (sispak == 1) {
                        y = sess.createSQLQuery("insert into data
    values ('penanaman','" + namatabel + "')");
                        z = sess.createSQLQuery("alter table
    penanaman add column " + namatabel + " varchar(4)");
3                   } else {
                        y = sess.createSQLQuery("insert into data
    values ('varietas','" + namatabel + "')");
                        z = sess.createSQLQuery("alter table
    varietas add column " + namatabel + " varchar(4)");
                    }
4                   q.executeUpdate();
5                   y.executeUpdate();
6                   z.executeUpdate();

                } catch (Exception e) {
                }
        }
```

Gambar 5.4 Implementasi Add Gejala
Sumber : Implementasi

Gambar 5.4 merupakan kode add gejala sistem pakar penanaman cabai merah keriting berdasarkan perancangan algoritma pada gambar 4.10 . Baris 1 untuk membuat tabel baru berdasarkan parameter namatabel. Baris 2 jika sispak yang dipilih adalah penanaman, maka akan ditambahkan data dengan isi kolom sispak penanaman dan kolom tabel sesuai parameter namatabel pada tabel data, serta penambahan kolom pada tabel penanaman dengan nama kolom sesuai parameter namatabel.

Sama dengan halnya baris 2, baris 3 jika sispak yang dipilih adalah varietas maka akan ditambahkan data dengan isi kolom sispak varietas dan kolom tabel sesuai parameter namatabel pada tabel data, serta penambahan kolom pada tabel varietas dengan nama kolom sesuai parameter namatabel. Pada baris 4,5,6 merupakan eksekusi query untuk melakukan update query ke database.

4. Implementasi Add Data

```
1    public void addData(Session sess, T entity) {
2         sess.saveOrUpdate(entity);
     }
```

Gambar 5.5 Implementasi Add Data
Sumber : Implementasi

Gambar 5.5 merupakan implementasi kode Add Data berdasarkan perancangan algoritma pada gambar 4.11. Variable `sess` melakukan pemanggilan method `saveorUpdate` dengan parameter object oleh untuk menyimpan/mengupdate data dalam bentuk object.

5. Implementasi Delete Data

```
1  public void deleteData(Session sess, String id) {
2  Query q = sess.createQuery("delete from Hama where id='"+id+"'");
3  q.executeUpdate();
       }
```

Gambar 5.6 Implementasi Delete Data
Sumber : Implementasi

Gambar 5.6 merupakan implementasi dari delete Data berdasarkan perancangan algoritma pada gambar 4.12. Pada baris 2 variable `q` menyimpan query yang mengambil data tabel melalui variable `sess` sesuai parameter `id`. Pada baris 3 variable `q` memanggil method `ExecuteUpdate()` untuk mengupdate query yang telah dilakukan pada baris ke 2.

6. Implementasi Get Table Name

```
1   public List<String> getAllTable(Session sess) {
2   List<String> allTables = sess.createSQLQuery("SELECT TABLE_NAME
    FROM information_schema.TABLES WHERE
    TABLE_SCHEMA=DATABASE()").list();

    return allTables;
      }
```

Gambar 5.7 Implementasi Get Table Name
Sumber : Implementasi

Gambar 5.7 merupakan implementasi *get table name* berdasarkan perancangan algoritma pada gambar 4.13. Pada baris 2 merupakan proses pengambilan nama tabel secara keseluruhan, data dikembalikan dalam bentuk list.

7. Implementasi Get Data By Table Name

```
1  public List getDataNew(Session sess, String namatable) {
2   List q = sess.createSQLQuery("select * from " +
   namatable).list();
   return q;
      }
```

Gambar 5.8 Implementasi Get Data By Tabel Name
Sumber : Implementasi

Gambar 5.8 merupakan implementasi get Data by table name berdasarkan perancangan algoritma pada gambar 4.14. Pada baris 2 merupakan proses pengambilan data tabel menurut parameter `namatabel`.

5.5 Implementasi Interface

Pada implementasi interface akan dijelaskan hasil implementasi user interface yang sebelumnya telah dirancang pada bab perancangan.

1. Halaman Utama

Gambar 5.9 merupakan tampilan halaman utama sistem pakar sesuai pada perancangan gambar 4.15. Dimana sesuai tampilan tersebut terdapat menu halaman utama, dan halaman admin. Serta tombol untuk masuk ke sistem pakar, bantuan, dan deskripsi tentang sistem.

Gambar 5.9 Implementasi Halaman Utama
Sumber : Implementasi

2. Halaman Admin

Apabila memilih menu admin, maka akan ditampilkan halaman seperti gambar 5.10 sesuai perancangan interface pada gambar 4.16

Gambar 5.10 Implementasi Halaman Admin
Sumber : Implementasi

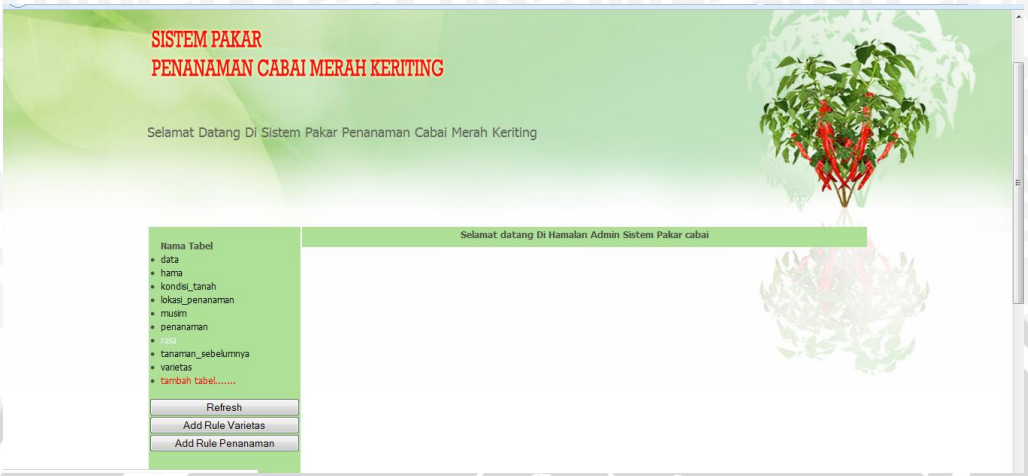
Dimana terdapat tampilan seluruh tabel beserta datanya, serta penambahan tabel, penambahan rule, dan refresh halaman.

3. Halaman Add Tabel (Gejala)

Halaman add Tabel berfungsi untuk menambah gejala baru pada sistem pakar penanaman cabai merah keriting. Berikut penjelasan proses penambahan datanya.

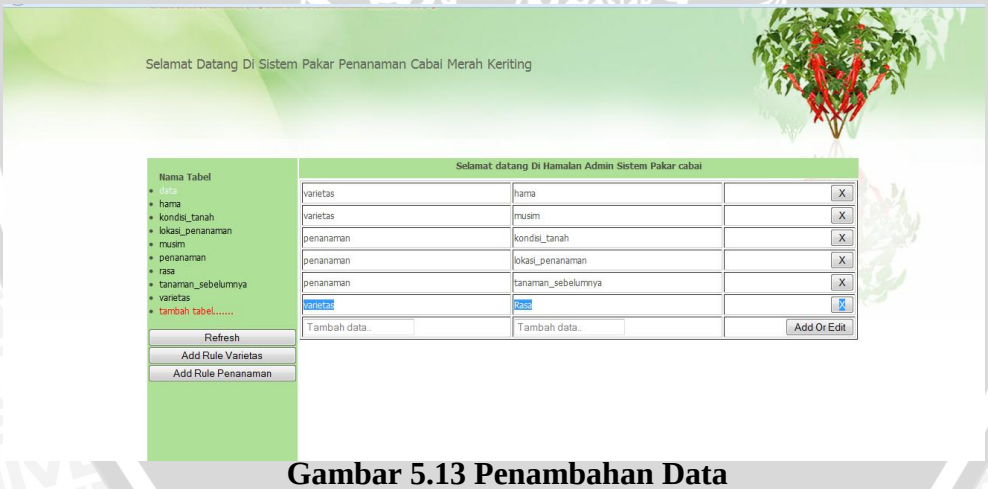
Gambar 5.11 Implementasi Penambahan Tabel
Sumber : Implementasi

Pada gambar 5.11 menjelaskan proses penambahan tabel/gejala. Pada tampilan tersebut terdapat pilihan sistem pakar yaitu penanaman/pemilihan varietas, dan input text nama tabel baru yang akan ditambahkan.



Gambar 5.12 Tampilan Penambahan Tabel
Sumber : Implementasi

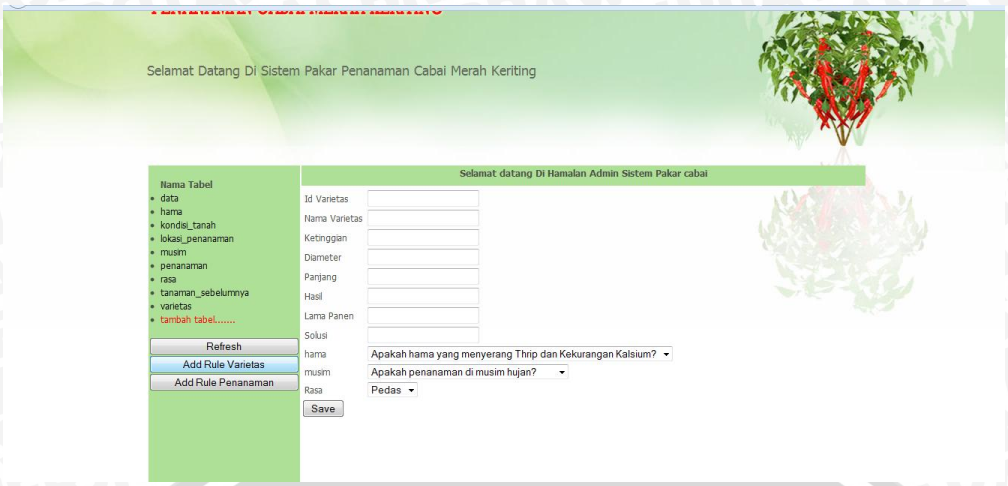
Pada gambar 5.12 ditunjukkan bahwa tabel/gejala yang ditambahkan otomatis tampil pada sistem setelah dilakukan refresh page.



Gambar 5.13 Penambahan Data
Sumber : Implementasi

Pada gambar 5.13 ditunjukkan bahwa tabel/gejala yang ditambahkan otomatis ditambahkan pada tabel data. Tabel data bertugas untuk menyimpan data apakah suatu tabel/gejala termasuk pada sistem pakar penanaman cabai atau sistem pakar varietas

.



Gambar 5.14 Penambahan Gejala Baru Pada Rule
Sumber : Implementasi

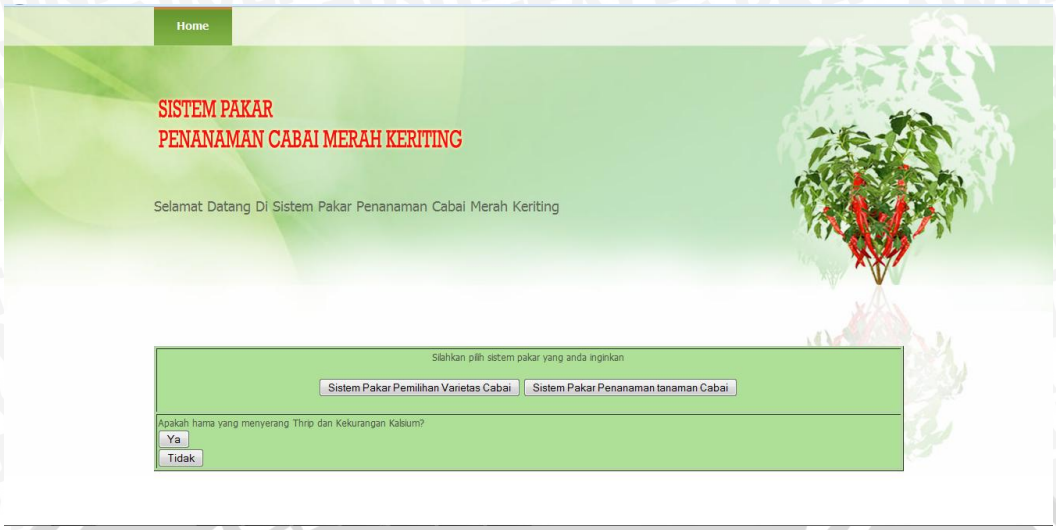
Tabel/gejala yang baru ditambahkan, secara otomatis langsung disisipkan pada rule sistem pakar. Sesuai dengan contoh gambar diatas, tabel/gejala yang ditambahkan adalah tabel rasa dan dimasukkan pada sistem pakar varietas. Maka otomatis pada rule sistem pakar pemilihan varietas terdapat penambahan gejala baru yaitu rasa.

4. Halaman Sistem Pakar

Ketika masuk ke halaman sistem pakar, maka akan ditampilkan halaman seperti pada gambar 5.15 sesuai perancangan pada gambar 4.17. Dimana terdapat pilihan sistem pakar pemilihan varietas cabai, dan sistem pakar penanaman cabai.

Gambar 5.15 Implementasi Halaman Sistem Pakar
Sumber : Implementasi

Ketika dipilih dari salah satu sistem pakar, maka akan ditampilkan halaman pertanyaan seperti gambar 5.16 berikut ini.



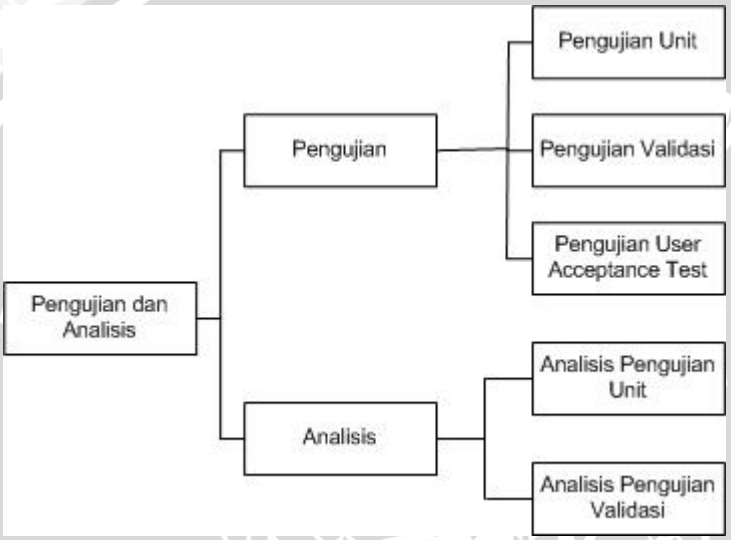
Gambar 5.16 Implementasi Halaman Pertanyaan Sistem Pakar
Sumber : Implementasi

Ketika pertanyaan – pertanyaan tersebut telah dipenuhi maka akan ditampilkan solusi seperti pada gambar 5.17 seperti berikut.

Gambar 5.17 Implementasi Halaman Solusi
Sumber : Implementasi

BAB VI
PENGUJIAN DAN ANALISIS

Pada bab ini menjelaskan pengujian dan analisis yang dilakukan pada aplikasi sistem pakar penanaman cabai yang telah dibangun. Pengujian yang dilakukan yaitu pengujian unit dan validasi. Dimana pengujian unit menggunakan metode *White Box* dan pengujian validasi menggunakan metode *Black Box*.



Gambar 6.1 Diagram Alir Pengujian dan Analisis
Sumber : Pengujian dan Analisis

6.1 Pengujian

Pada tahap pengujian terdapat tiga pengujian yang dilakukan yaitu pengujian unit, validasi, dan User Acceptance Test (UAT), masing-masing tahapan dari pengujian tersebut akan dijelaskan pada sub bab berikut.

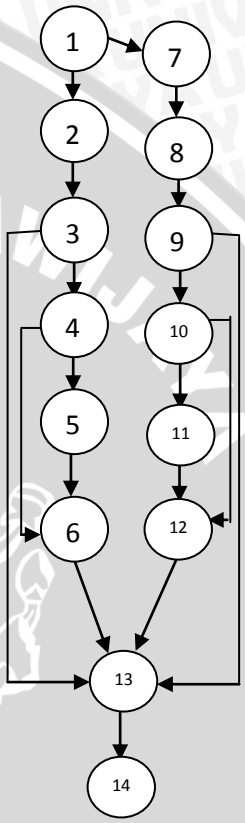
6.1.1 Pengujian Unit

Pada pengujian unit menggunakan metode *White Box* dengan teknik *basic path*. Teknik *Basis Path Testing* melakukan pengujian dengan menggambarkan algoritma pada sebuah *flow graph*, kemudian menentukan *cyclometic complexcity* dan melakukan uji kasus untuk setiap path yang ada.

6.1.1.1 Pengujian Unit Inferensi

```
INFERENSI parameter Penanaman, Varietas
Is String

DECLARATION :
  jawaban, Sistem pakar Is String
  Pertanyaan is Int
  Solusi is List

DESCRIPTION :

1  If (sistempakar == 'penanaman'){

2  Pertanyaan = getJumlah('penanaman');

3  While(pertanyaan > 0){
4           PRINT(pertanyaan)
5           PRINT(jawaban ya dan tidak)

6           IF(jawaban == 'ya'){
7            jawaban = pertanyaan terpilih}

8           ELSE IF(jawaban =='tidak'){
9            Pertanyaan.next()
}
10 Solusi = Sendjawaban(jawaban)
   }

11 If (sistempakar == 'varietas){

12 Pertanyaan = getJumlah('varietas');

13 While(pertanyaan > 0){
14          PRINT(pertanyaan)
15          PRINT(jawaban ya dan tidak)

16          IF(jawaban == 'ya'){
17           jawaban= pertanyaan terpilih}

18          ELSE IF(jawaban =='tidak'){
19           Pertanyaan.next()
}
21 Solusi = Sendjawaban(jawaban)
   }

22 Solusisispak = getsolusi(solusi)
23 If(isset(Solusi)){
24   While(solusi.hasNext()){
25    PRINT(solusi)
26    Solusi.next()
     }
```

Node(N) = 14
Edge(E) = 18



Gambar 6.2 Pengujian Unit Inferensi
Sumber : Pengujian dan Analisis

Pada tabel diatas Inferensi() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2

= 18 – 14 + 2 = 6

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan dua buah basis set dari jalur independent yaitu:

Jalur 1: 1 – 2 – 3 – 4 – 5 – 6 – 13 – 14

Jalur 2: 1 – 2 – 3 – 13 – 14

Jalur 3: 1 – 2 – 3 – 4 – 6 – 13 – 14

Jalur 4: 1 – 7 – 8 – 9 – 10 – 11 – 12 – 13 – 14

Jalur 5: 1 – 7 – 8 – 9 – 13 – 14

Jalur 6: 1 – 7 – 8 – 9 – 10 – 12 – 13 – 14

Penentuan test case untuk masing-masing jalur dan hasil dari eksekusi dari masing-masing test case akan dijelaskan pada tabel 6.1.

Tabel 6.1 Test Case Pengujian Unit Inferensi()

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Memberikan kondisi terhadapa sistem pakar penanaman dengan kondisi selalu yg pertama	Solusi sistem pakar penanaman	Solusi sistem pakar penanaman
2	Pilihan sistem pakar penanaman namun tidak terdapat pertanyaan / rule	Solusi tidak mendapatkan hasil	Solusi tidak mendapatkan hasil
3	Memberikan kondisi terhadapa sistem pakar penanaman	Solusi sistem pakar penanaman	Solusi sistem pakar penanaman
4	Memberikan kondisi terhadapa sistem pakar varietas	Solusi sistem pakar varietas	Solusi sistem pakar varietas
5	Pilihan sistem pakar varietas namun tidak terdapat pertanyaan / rule	Solusi tidak mendapatkan hasil	Solusi tidak mendapatkan hasil
6	Memberikan kondisi terhadapa sistem pakar varietas dengan kondisi selalu yg pertama	Solusi sistem pakar varietas	Solusi sistem pakar varietas

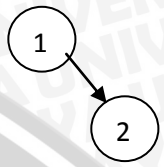
Sumber : Pengujian dan Analisis

Dari hasil test case diatas hasil yang didapatkan telah sesuai dengan hasil yang diharapkan.

6.1.1.2 Pengujian Unit Add Rule Sistem Pakar

```
Add rule sistem pakar PARAMETER sistem pakar IS String

DECLARATION :
kondisi, solusi IS ARRAY

DESCRIPTION

1  IF (sistem pakar == 'penanaman')
2    INPUT kondisi
3    Sendrulepenanaman(kondisi)

4  IF (sistem pakar == 'varietas)
5    INPUT kondisi
6    Sendrulevarietas(kondisi)
```

Node(N) = 2
Edge(E) = 1



Gambar 6.3 Pengujian Unit addRule()
Sumber : Pengujian dan Analisis

Pada tabel diatas addRule() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2

= 1 – 2 + 2 = 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan dua buah basis set dari jalur independent yaitu:

Jalur 1: 1 – 2

Penentuan test case untuk masing-masing jalur dan hasil dari eksekusi dari masing-masing test case akan dijelaskan pada tabel 6.2.

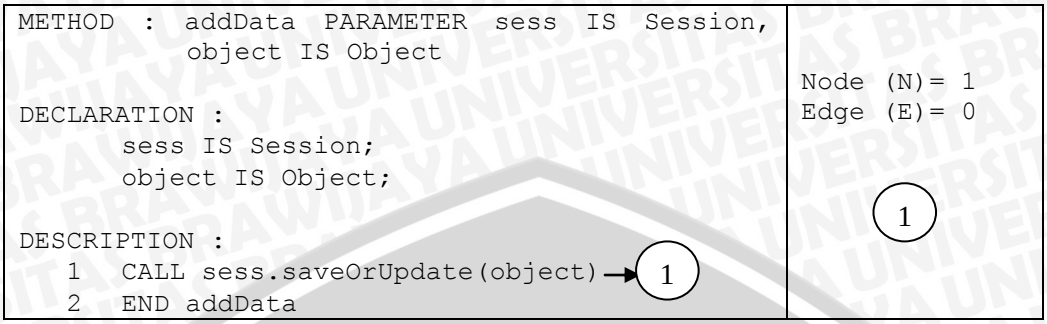
Tabel 6.2 Test Case Pengujian Unit addRule()

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Tambah rule dengan kode penanaman	Rule penanaman berhasil ditambahkan ke database	Rule penanaman berhasil ditambahkan ke database
2	Tambah rule dengan kode varietas	Rule varietas berhasil ditambahkan ke database	Rule varietas berhasil ditambahkan ke database

Sumber : Pengujian dan Analisis

Dari hasil test case diatas hasil yang didapatkan telah sesuai dengan hasil yang diharapkan.

6.1.1.3 Pengujian Unit addData()



Gambar 6.4 Pengujian Unit addData()
Sumber : Pengujian dan Analisis

Pada tabel diatas addData() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2

= 0 – 1 + 2 = 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1: 1

Penentuan test case untuk masing-masing jalur dan hasil dari eksekusi dari masing-masing test case akan dijelaskan pada tabel 6.3.

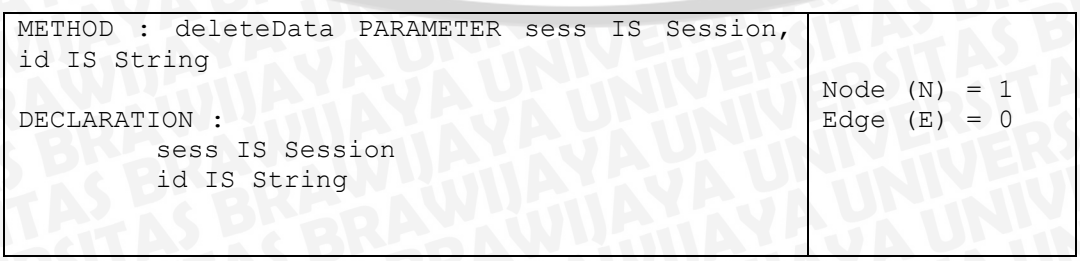
Tabel 6.3 Test Case Pengujian Unit addData()

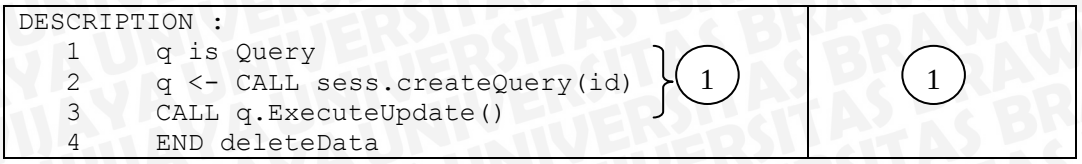
Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Menyimpan data dalam bentuk objek	Objek berhasil disimpan/diupdate dalam database	Objek berhasil disimpan/diupdate dalam database

Sumber : Pengujian dan Analisis

Dari hasil test case diatas hasil yang didapatkan telah sesuai dengan hasil yang diharapkan.

6.1.1.4 Pengujian Unit deleteData()





Gambar 6.5 Pengujian Unit deleteData()
Sumber : Pengujian dan Analisis

Pada tabel diatas deleteData() menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2

= 0 – 1 + 2 = 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1: 1

Penentuan test case untuk masing-masing jalur dan hasil dari eksekusi dari masing-masing test case akan dijelaskan pada tabel 6.4.

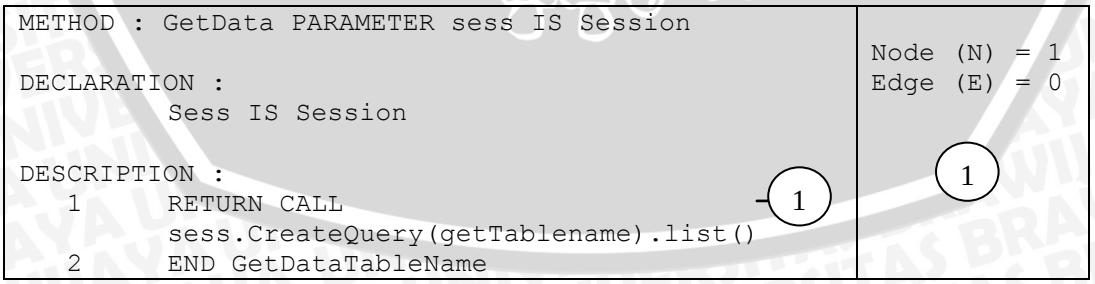
Tabel 6.4 Test Case Pengujian Unit deleteData()

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Menghapus data	Data berhasil dihapus dari database	Data berhasil dihapus dari database

Sumber : Pengujian dan Analisis

Dari hasil test case diatas hasil yang didapatkan telah sesuai dengan hasil yang diharapkan.

6.1.1.5 Pengujian Unit Get Data Table Name



Gambar 6.6 Pengujian Unit Get Data Table Name
Sumber : Pengujian dan Analisis

Pada tabel diatas get data Table name menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2 = 0 – 1 + 2 = 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1: 1

Penentuan test case untuk masing-masing jalur dan hasil dari eksekusi dari masing-masing test case akan dijelaskan pada tabel 6.5

.Tabel 6.5 Test Case Pengujian Unit Get Data Table Name

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Mendapatkan seluruh data dalam bentuk list	Data dikembalikan dalam bentuk list	Data dikembalikan dalam bentuk list

Sumber : Pengujian dan Analisis

Dari hasil test case diatas hasil yang didapatkan telah sesuai dengan hasil yang diharapkan.

6.1.1.6 Pengujian Unit Get Data By Table Name

```
METHOD : GetData PARAMETER sess IS Session,
tablename IS String

DECLARATION :
        Sess IS Session
        tablename IS String

DESCRIPTION :
1   1   RETURN CALL
        sess.CreateQuery(getdatatabel by
        tablename).list()
    2   END GetDataByname
```

```
Node (N) = 1
Edge (E) = 0

    1
```

Gambar 6.7 Pengujian Unit get Data By Table Name
Sumber : Pengujian dan Analisis

Pada tabel diatas get Data by Table Name menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2

= 0 – 1 + 2 = 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1: 1

Penentuan test case untuk masing-masing jalur dan hasil dari eksekusi dari masing-masing test case akan dijelaskan pada tabel 6.6.

Tabel 6.6 Test Case Pengujian Unit get Data by Table Name

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan
1	Mendapatkan data menurut nama tabel dalam bentuk list	Data dikembalikan sesuai nama tabeldalam bentuk list	Data dikembalikan sesuai nama tabel dalam bentuk list

Sumber : Pengujian dan Analisis

Dari hasil test case diatas hasil yang didapatkan telah sesuai dengan hasil yang diharapkan.

6.1.2 Pengujian Validasi

Pengujian validasi menggunakan metode *Black Box* dimana pengujian yang dilakukan adalah pengujian fungsionalitas dari sistem, apakah sistem berfungsi dengan hasil yang diinginkan atau tidak. Berikut tabel hasil pengujian validasi sistem.

Tabel 6.7 Pengujian Validasi

No	Kasus Uji	Prosedur dan Input	Kondisi yang diharapkan	Hasil
1.	Login admin	Memilih menu admin, lalu memasukkan password	Jika login sebagai admin maka akan dimintai password, jika benar akan ditampilkan halaman admin.	Valid
2.	Masuk ke sistem pakar	Menekan tombol masuk ke sistem pakar	Akan ditampilkan halaman sistem pakar	Valid
3.	Menggunakan sistem pakar	Memilih sistem pakar yang diinginkan dengan menekan tombol yang tersedia	Akan ditampilkan halaman pertanyaan sistem pakar sesuai pilihan	Valid
4.	Mendapatkan solusi	Menjawab petanyaan sampai selesai	Akan ditampilkan solusi	Valid
5.	Menambah data	Admin menginputkan data-data yang akan di simpan ke database	Akan ditampilkan halaman admin, dan data yang diinputkan berhasil masuk ke database	Valid

6.	Mengubah data	Admin menginputkan data-data yang akan di ubah ke database	Akan ditampilkan halaman admin, dan data yang diubah berhasil masuk ke database	Valid
7.	Menghapus data	Admin menekan tombol hapus pada baris yang ingin dihapus	Akan ditampilkan halaman admin, dan data yang dihapus berhasil terhapus dari database	Valid
8.	Menambah tabel	Admin menekan tombol tambah tabel, lalu menginputkan tabel yang ingin ditambahkan ke database	Akan ditampilkan halaman admin, dan tabel yang ditambah berhasil tersimpan ke database	Valid
9.	Menambah rule	Admin menekan tombol tambah rule, kemudian menginputkan rule baru	Akan ditampilkan halaman admin, dan rule yang ditambah berhasil tersimpan ke database	Valid
10.	Kembali ke halaman utama	Memilih menu home	Kembali ke halaman utama	Valid

Sumber : Pengujian dan Analisis

6.1.3 Pengujian Non Fungsional

Berikut ini adalah hasil identifikasi dari pengujian kebutuhan non fungsional.

Tabel 6.8 Pengujian Non Fungsional

No	Parameter	Deskripsi Kebutuhan	Kondisi yang Terjadi	Bukti
1.	***Availability***	Sistem harus beroperasi selama waktu yang ditentukan	Aplikasi sistem dihosting di localhost sehingga tidak dapat diukur	Tidak Valid
2.	***Usability***	Sistem harus memberi kemudahan pada user dalam penggunaan	Telah dilakukan UAT pada bab 6.1.4, dan 64% dari responden menyatakan bahwa tampilan dari sistem bagus	Valid
3.	***Security***	Sistem harus aman. Pada sistem ini menggunakan Login.	Telah dilakukan pengujian validasi pada bab 6.1.2 tentang test case login admin, dan	Valid

			terbukti berfungsi dengan baik	
4.	***Maintanability***	Sistem harus mudah direvisi jika diperlukan	Telah dilakukan pengujian UAT pada bab 6.1.4. Dari 14 responden, tidak ada yang menyatakan sulit dalam memahami source code	Valid

Sumber : Pengujian dan Analisis

6.1.4 Pengujian User Acceptance Test (UAT)

Bertujuan untuk menguji apakah sistem sudah sesuai dengan spesifikasi fungisonal sistem (validation). Berikut kerangka kuisioner dari pengujian UAT :

Tujuan:

Untuk mendapatkan respon pengguna/pengembang aplikasi sistem pakar penanaman cabai merah keriting yang menerapkan pola MVC. Sasaran responden /penguji sistem adalah dosen/mahasiswa yang memiliki pengalaman/pengetahuan dalam mengembangkan aplikasi yang menerapkan arsitektur MVC.

Respon:

1. Apakah responden memahami MVC?
2. Apakah aplikasi mudah untuk dalam pengembangan dan perawatannya ?
3. Apakah aplikasi telah memenuhi syarat sebagai aplikasi yang berbasis MVC ?
4. Apakah aplikasi ini memudahkan pengguna?
5. Apakah tanggapan/usulan dari responden terhadap aplikasi?

Penjelasan :

1. Apakah anda pernah menerapkan arsitektur MVC pada aplikasi yang anda kembangkan?
   a. Pernah b. Tidak pernah

Keterangan :

Pertanyaan diatas bertujuan untuk mengetahui apakah responden pernah menerapkan arsitektur MVC pada aplikasi yang pernah dikembangkan untuk menelaah sejauh mana pemahaman responden terhadap konsep dari arsitektur MVC.

2. Jika anda pernah menerapkan MVC apakah MVC memudahkan dalam pengembangan aplikasi anda?

   a. Iya b. Tidak c. Bisa Jadi

   Keterangan :

   Pertanyaan diatas bertujuan untuk mengetahui apakah arsitektur MVC yang telah diterapkan pada aplikasi yang dikembangkan memudahkan responden dalam pengembangan aplikasi. Jika tidak yakin arsitektur MVC yang diterapkan telah memberi kemudahan dalam pengembangan aplikasi maka responden sebaiknya memilih jawaban c.

3. Apakah anda memahami manfaat dari arsitektur MVC pada aplikasi sistem pakar penanaman cabai merah keriting?

   a. Paham b. Kurang paham c. Tidak paham

Keterangan :

Pertanyaan diatas bertujuan untuk mengetahui apakah responden merasakan dan memahami manfaat/pengaruh dari arsitektur MVC pada aplikasi sistem pakar penanaman cabai merah keriting seperti pada tujuan daripada dirancangnya sistem ini dengan arsitektur MVC yaitu memisahkan tugas dari aplikasi menjadi tiga bagian yaitu Model, View, dan Controller sehingga memudahkan dalam pengembangan aplikasi selanjutnya.

4. Apakah anda mudah dalam memahami keseluruhan dari source code aplikasi sistem pakar penanaman cabai merah keriting?

   a. Sangat mudah b. Mudah c. Sulit

Keterangan :

Pertanyaan diatas bertujuan untuk mengetahui apakah responden mudah dalam memahami seluruh source code dalam aplikasi sistem pakar penanaman cabai merah keriting ini, hal ini bertujuan untuk mengetahui apakah kode program masih rumit sehingga susah dipahami, atau kode sederhana sehingga mudah dipahami oleh responden. Tujuan dari mengetahui pemahaman source code untuk mengukur kerumitan dari kode sehingga memudahkan pengembangan sistem.

5. Bagaimana tampilan dari aplikasi sistem pakar penanaman cabai merah keriting ini?

   a. Bagus b. Cukup c. Kurang

Keterangan :

Pertanyaan diatas bertujuan untuk mengetahui apakah tampilan/user interface pada sistem pakar penanaman cabai sudah baik atau belum. Pernyataan bagus, cukup atau kurang berdasarkan parameter user interface dari sistem kontras warna, dan pemilihan warna pada tampilan aplikasi, dan memberikan kemudahan pengguna dalam menggunakan aplikasi.

6. Apakah aplikasi sistem pakar penanaman cabai merah keriting ini telah memenuhi syarat sebagai aplikasi yang menerapkan arsitektur MVC?

   a. Ya b. Kurang c. Tidak

Keterangan :

Pertanyaan diatas bertujuan untuk mengetahui apakah aplikasi sistem pakar penanaman cabai merah keriting telah memenuhi syarat sebagai aplikasi yang menerapkan arsitektur MVC.

7. Apa tanggapan anda tentang aplikasi sistem pakar penanaman cabai merah keriting ini?

Keterangan :

Pertanyaan diatas bertujuan untuk mendapatkan tanggapan dari responden mengenai saran dan kritik terhadap sistem pakar penanaman cabai ini sehingga dapat dikembangkan dengan lebih baik lagi.

Berdasarkan UAT yang telah dilakukan, berikut hasil dari pengujian UAT :

Tabel 6.9 Kalkulasi Pengujian UAT

Gabungan Jawaban				Responden
No	a	b	c	
1	14	5		19
2	11	1	2	14
3	9	5		14
4	6	8		14
5	9	3	2	14
6	12	2		14

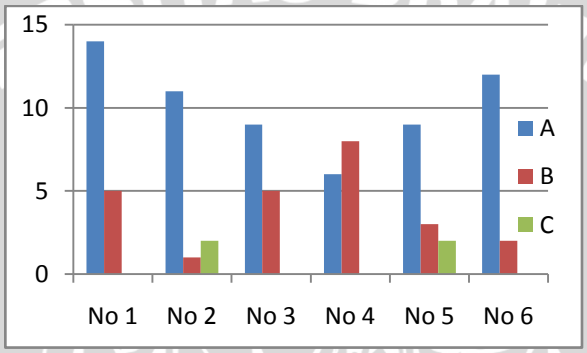
Sumber : Pengujian dan Analisis

Tabel 6.10 Persentase Pengujian UAT

Gabungan Jawaban				Responden
No	a	b	c	
1	74%	26%	0%	1
2	79%	7%	14%	1
3	64%	36%	0%	1
4	43%	57%	0%	1
5	64%	22%	14%	1
6	86%	14%	0%	1

Sumber : Pengujian dan Analisis

Dari tabel diatas dapat digambarkan grafik dari respon pertanyaan, seperti berikut



Gambar 6.8 Grafik Pengujian UAT
Sumber : Analisis dan Pengujian

Ringkasan :

- Pada pengujian UAT ini terdapat 6 pertanyaan pilihan, detail dari pertanyaan-pertanyaan tersebut telah dijelaskan sebelumnya.
- Pada pertanyaan no 1, responden 74% memilih jawaban a dimana berarti responden pada pengujian ini pernah menerapkan MVC pada aplikasi yang dikembangkan, sedangkan 26% menyatakan belum pernah menerapkan MVC.
- Pada pertanyaan no 2, responden 79% memilih jawaban a dimana berarti responden merasakan kemudahan yang diberikan dari penerapan MVC pada aplikasi yang telah dikembangkan, 7% menyatakan MVC tidak memberikan kemudahan, sedangkan ada 14% yang tidak yakin bahwa MVC memberikan kemudahan pada aplikasi yang dikembangkan.
- Pada pertanyaan no 3, 64% persen responden menyatakan paham akan manfaat dari penerapan MVC pada sistem pakar penanaman cabai merah keriting, sedangkan 36% sisanya masih kurang paham dikarenakan sistem yang

sederhana sehingga belum terasa manfaat dari MVC yang telah diterapkan pada sistem pakar penanaman cabai merah keriting.

- Pada pertanyaan no 4, 43% responden menyatakan bahwa sangat mudah untuk memahami source code dari sistem pakar penanaman cabai merah keriting dikarenakan sistem yang masih sederhana sehingga mudah untuk dipahami, sedangkan 57% lainnya menyatakan mudah dalam memahami source code.
- Pada pertanyaan no 5, 64% responden menyatakan bahwa tampilan dari aplikasi sistem pakar penanaman cabai merah keriting sudah bagus, sedangkan 22% diantaranya menyatakan cukup, dan 14% sisanya menyatakan kurang
- Pada pertanyaan no 6, 86% responden menyatakan bahwa sistem pakar penanaman cabai merah keriting ini telah memenuhi syarat sebagai aplikasi yang menerapkan MVC, sedangkan 14% sisanya menyatakan kurang.

6.2 Analisis

Analisis bertujuan untuk mendapatkan kesimpulan dari pengujian yang telah dilakukan pada sistem. Berikut uraian analisis dari tiap-tiap pengujian yang telah dilakukan.

6.2.1 Analisis Pengujian Unit

Pada pengujian unit telah dilakukan pengujian pada operasi-operasi yang berada pada kelas. Dari hasil pengujian, fungsi inference() memiliki kasus uji terbanyak yaitu 6 kasus uji. Hasil yang diharapkan sesuai dengan hasil yang didapat. Maka dapat diambil kesimpulan bahwa operasi unit yang terdapat pada program sesuai dengan hasil yang diharapkan.

6.2.2 Analisis Pengujian Validasi

Pada pengujian validasi dilakukan pengujian fungsionalitas sistem, apakah berjalan sesuai dengan yang diharapkan atau tidak sesuai dengan analisis kebutuhan fungsional yang telah dirancang. Dari hasil pengujian seluruh kondisi yang diharapkan dengan hasil menunjukkan bahwa apa yang diharapkan telah sesuai dengan hasil. Maka dapat diambil kesimpulan bahwa hasil yang diharapkan valid sesuai program yang telah dibangun.

6.2.3 Analisis Pengujian Non Fungsional

Pada pengujian non fungsional dilakukan pengujian apakah sistem telah sesuai pada parameter-parameter yang telah dirancang. Dari hasil pengujian dapat ditarik kesimpulan bahwa keseluruhan parameter terpenuhi, akan tetapi parameter *availability* tidak terpenuhi dikarenakan kendala sistem yang hanya terhosting di localhost.

6.2.4 Analisis Pengujian User Acceptance Test (UAT)

Pada pengujian UAT dilakukan pengujian fungsionalitas sistem, apakah berjalan sesuai dengan yang diharapkan atau tidak. Dari hasil pengujian UAT yang telah dilakukan, sistem ini mendapat beberapa tanggapan negative diantaranya :

- Sistem masih terlalu sederhana sehingga belum terlihat manfaatnya.
- Pemilihan kontras warna tampilan aplikasi sistem kurang tepat.
- Penerapan MVC kurang terasa karena inference statis, sehingga harus melakukan perubahan code jika ada inference baru.

Akan tetapi tanggapan tersebut telah diperbaiki penulis, sehingga aplikasi sistem pakar penanaman cabai merah keriting memiliki inference yang dinamis, sehingga tidak perlu dilakukan perubahan code ketika terdapat inference baru. Tampilan dari aplikasi juga telah diperbaiki untuk memenuhi komposisi pemilihan warna yang baik, dan tampilan yang memudahkan user dalam menggunakan aplikasi.

Sistem ini mendapat tanggapan dari 86% responden bahwa aplikasi sistem pakar penanaman cabai merah keriting ini telah menerapkan arsitektur MVC.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan membahas kesimpulan dan saran dari penelitian yang telah dilakukan. Kesimpulan bertujuan untuk mengambil ringkasan dari penelitian, dan saran bertujuan untuk memberikan masukan untuk pengembangan penelitian.

7.1 Kesimpulan

Dari hasil perancangan, implementasi, serta proses pengujian sistem yang telah dilakukan, maka diambil kesimpulan sebagai berikut :

1. Sistem pakar Penanaman Cabai Merah Keriting merupakan aplikasi sistem pakar yang bertujuan memberikan solusi untuk penanaman cabai merah keriting yang baik.
2. Sistem Pakar Penanaman Cabai Merah Keriting menerapkan desain pola perancangan *Model View Controller* dalam pengembangannya.
3. Perancangan Sistem Pakar Penanaman Cabai Keriting meliputi perancangan arsitektur, sitemap, *Entity Relationship Diagram, Class Diagram, Inference Engine,* algoritma, serta *User Interface.*
4. Pengujian unit Sistem Pakar Penananaman Cabai Keriting dengan metode *White box* menghasilkan kesimpulan bahwa unit modul dari program sudah sesuai dengan output yang diharapkan dan kode unit dalam pengujian yang menghasilkan kasus uji terbanyak adalah kode operasi Inference() yaitu sebanyak 6 kasus uji.
5. Berdasarkan hasil pengujian validasi menggunakan metode *blackbox testing*, didapatkan keseluruhan fungsional aplikasi permainan dapat berjalan sesuai daftar kebutuhan yang telah dibuat dan dimodelkan dalam *use case*.
6. Berdasarkan hasil pengujian User Acceptance Test (UAT) sistem pakar penanaman cabai merah keriting ini, dapat diambil kesimpulan :

- 74% responden pada pengujian ini pernah menerapkan MVC pada aplikasi yang dikembangkan.
- 79% responden merasakan kemudahan yang diberikan dari penerapan MVC pada aplikasi yang telah dikembangkan.
- 64% responden merasakan manfaat dari MVC yang telah diterapkan pada sistem pakar penanaman cabai merah keriting.
- 57% responden menyatakan bahwa mudah untuk memahami source code dari sistem pakar penanaman cabai merah keriting.
- 64% responden menyatakan bahwa tampilan dari aplikasi sistem pakar penanaman cabai merah keriting sudah bagus.
- 86% responden menyatakan bahwa sistem pakar penanaman cabai merah keriting ini telah memenuhi syarat sebagai aplikasi yang menerapkan MVC.
- Dengan penerapan pola perancangan MVC pada sistem pakar penanaman cabai merah keriting ini, inference dapat ditambahkan langsung melalui sistem tanpa melakukan perubahan kode.

7.2 Saran

Saran untuk pengembangan Sistem Pakar Penananaman Cabai Keriting lebih lanjut antara lain :

1. Pengembangan Sistem Pakar Penananaman Cabai Keriting selanjutnya dapat dilakukan dengan menggunakan framework Spring.
2. Perbaikan user interface yang lebih bagus, cepat dan ringan dengan menggunakan velocity.

DAFTAR PUSTAKA

[ACS-10] Solichin, Achmad. 2010. “MySQL Dari Pemula Hingga Mahir”. Universitas BudiLuhur : Jakarta.

[NWD-10] Widiyanto, Nur. 2010. "Membangun Aplikasi Java Enterprise dengan Arsitektur Model View Controller (MVC)". Penerbit : Andi Offset.

[RMH-12] Hakiki, Rindang. 2012. "Perancangan Aplikasi Sistem Pakar Teknik Budidaya dalam Penanaman dan Pemilihan Varietas Cabai Keriting berbasis Web Service". KKN-P Teknik Informatika : Universitas Brawijaya.

[BMS-06] Riyanto, Bambang dkk. 2006. "Perancangan Aplikasi M-Learning berbasis Java". STEI Institut Teknologi Bandung

[JEN-11] JENI. 2011. "Dasar JSP". Jeni Web Programming

[JMV-11] JENI. 2011. "MVC Introduction". Jeni Web Programming

[JSB-11] JENI. 2011. "Basis Servlet". Jeni Web Programming

[KKN-07] klikkanan.com. 2007. “Tutorial Belajar Dasar-dasar Hypertext Markup Language” Penerbit : DuniaPustaka.com

[KUS-07] Kusumadewi, Sri, 2007, Artificial Intelligence: Teknik dan Aplikasinya, Yogyakarta, Graha Ilmu.

[SEN-12] http://senaharimuda.wordpress.com/2012/06/11/mysql-dan-basis-data-4/

[SKD-13] David, Sandi Kosasi. 2013. "Penerapan Design Pattern Dalam Perancangan Web Order". STMIK Pontianak.

[SOM-03] Sommerville, Ian. 2003. Software Engineering (Rekayasa Perangkat Lunak) / Edisi 6 / Jilid 1, Jakarta: Penerbit Erlangga.

[TWD-09] Wardhana, Teddy. 2009 "Pengamanan Database Yang Diimplementasikan ke dalam Arsitektur MVC menggunakan Hibernate Framework". Universitas Gunadarma : Jakarta.

[UNI-13] Hibernate Framework. 2013. elearning.amikom.ac.id

[AYT-11] Liliana, Ayu. 2011. "Testing dan Implementasi". Universitas Gunadarma : Jakarta.

[PRE-10] Pressman, Roger S. 2010. Software Engineering: a practitioner's approach,7th edition. Mc Graw Hill.

[HAR-09] Harsiti. 2009. "Teknik Pengujian Perangkat Lunak". harsiti09.files.wordpress.com

[YUL-13] Yulia, group. 2013. "System Development aLife Cycle". www.blogspot.com/yuliagroup

[MNU-08] R. Mulyanto, Aunur. 2008. "Rekayasa Perangkat Lunak". Direktorat Pembinaan Sekolah menengah kejuruan.

LAMPIRAN

KUISIONER PENERAPAN MVC (MODEL VIEW CONTROLLER) PADA APLIKASI SISTEM PAKAR PENANAMAN CABAI MERAH KERITING

Nama : Dewy Tanda Tangan : 

1. Apakah anda pernah menerapkan arsitektur MVC pada aplikasi yang anda kembangkan?
   a. Pernah b. Tidak pernah
2. Jika anda pernah menerapkan MVC apakah MVC memudahkan dalam pengembangan aplikasi anda?
   a. Iya b. Tidak c. Bisa Jadi
3. Apakah anda memahami manfaat dari arsitektur MVC pada aplikasi sistem pakar penanaman cabai merah keriting?
   a. Paham b. Kurang paham c. Tidak paham
4. Apakah anda mudah dalam memahami keseluruhan dari source code aplikasi sistem pakar penanaman cabai merah keriting?
   a. Sangat mudah b. Mudah c. Sulit
5. Bagaimana tampilan dari aplikasi sistem pakar penanaman cabai merah keriting ini?
   a. Bagus b. Cukup c. Kurang
6. Apakah aplikasi sistem pakar penanaman cabai merah keriting ini telah memenuhi syarat sebagai aplikasi yang menerapkan arsitektur MVC?
   a. Ya b. Kurang c. Tidak
7. Apa tanggapan anda tentang aplikasi sistem pakar penanaman cabai merah keriting ini?

Aplikasi Sistem Pakar yang dibangun diharapkan bisa mempermudah kelompok pengguna sistem, terutama dalam mengaplikasikan konsep / pendekatan MVC bisa memaksimalkan untuk pengembangan lebih lanjut menjadi lebih maksimal oleh karena perlu diuji kehandalan MVC