**UNIVERSITAS INDONESIA**

# IMPLEMENTASI *KNOWLEDGE MANAGEMENT SYSTEM* BERBASIS *SEMANTIC* MEDIAWIKI PADA DIVISI OPERASIONAL PERUSAHAAN TELEKOMUNIKASI

**SKRIPSI**

**IRVANDA KURNIADI VIRDAUS**

**NPM 0806339162**

**FAKULTAS TEKNIK**
**PROGRAM SARJANA**
**DEPOK**
**DESEMBER 2011**

**UNIVERSITAS INDONESIA**

# IMPLEMENTASI *KNOWLEDGE MANAGEMENT SYSTEM* BERBASIS *SEMANTIC* MEDIAWIKI PADA DIVISI OPERASIONAL PERUSAHAAN TELEKOMUNIKASI

**SKRIPSI**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana**

**IRVANDA KURNIADI VIRDAUS**

**NPM 0806339162**

**FAKULTAS TEKNIK**
**PROGRAM STUDI TEKNIK KOMPUTER**
**DEPOK**
**DESEMBER 2011**

## HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.**

| | |
|---|---|
| **Nama** | **: Irvanda Kurniadi Virdaus** |
| **NPM** | **: 0806339162** |
| **Tanda Tangan** | **:** |
| **Tanggal** | **: 27 Desember 2011** |

## HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

| | |
|---|---|
| Nama | : Irvanda Kurniadi Virdaus |
| NPM | : 0806339162 |
| Program Studi | : Teknik Komputer |
| Judul Skripsi | : Implementasi *Knowledge Management System* Berbasis *Semantic* Mediawiki pada Divisi Operasional Perusahaan Telekomunikasi |

**Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia**

### DEWAN PENGUJI

Pembimbing : Prima Dewi Purnamasari, S.T., M.T., M.Sc. ( )

Penguji : Ir. Endang Sriningsih, MT, Si ( )

Penguji : I Gde Dharma Nugraha, S.T., M.T. ( )

Ditetapkan di : Depok
Tanggal : 16 Januari 2012

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada :

1. Ibu Prima Dewi Purnamasari, S.T., M.T., M.Sc sebagai dosen pembimbing atas segala bimbingan, ilmu, dan arahan baik dalam penulisan skripsi maupun selama masa studi di Teknik Komputer.
2. Ayah, Ibu, dan adik-adik yang selalu menjadi sumber inspirasi dan semangat.
3. Bapak Muhammad Salman S.T., M.I.T selaku pembimbing akademis yang selalu memberikan arahan akademis selama masa studi di Teknik Komputer.
4. Bapak Johan Eko Prasetyo selaku asisten manajer dinas IPN divisi Infratel PT Telekomunikasi Indonesia yang memberikan arahan dan bimbingan dalam pengerjaan *Knowledge Management System* yang diterapkan pada dinas tersebut.
5. Salman El Farisi, Qorib Munajat, dan Eskaning Arum Pawestri selaku teman yang membimbing dan memberikan arahan dalam menyusun skripsi mengenai *Knowledge Management System* ini.
6. Teman-teman di program studi Teknik Komputer, Teknik Elektro, dan Universitas Indonesia atas segala dukungan dan kerjasamanya

sehingga penulisan skripsi ini dapat diselesaikan dengan baik.

Depok, 27 Desember 2011

Penulis,

Irvanda Kurniadi Virdaus
NPM 0806339162

## HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Irvanda Kurniadi Virdaus

NPM : 0806339162

Program Studi : Teknik Komputer

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis karya : Tugas akhir

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif** (*Non-exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul:

**IMPLEMENTASI *KNOWLEDGE MANAGEMENT SYSTEM* BERBASIS SEMANTIC MEDIAWIKI PADA DIVISI OPERASIONAL PERUSAHAAN TELEKOMUNIKASI**

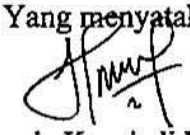
Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini, Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok

Pada tanggal: 27 Desember 2011

Yang menyatakan,



(Irvanda Kurniadi Virdaus)

# ABSTRAK


Nama : Irvanda Kurniadi Virdaus

Program Studi : Teknik Komputer


Judul : Implementasi Knowledge Management System Berbasis Semantic Mediawiki Pada Divisi Operasional Perusahaan Telekomunikasi.


Pembimbing : Prima Dewi Purnamasari, S.T., M.T., M.Sc.



*Knowledge management system* merupakan sistem yang dibuat untuk mengelola *knowledge* yang ada, baik yang bersifat *tacit*, maupun ekplisit. *Knowledge management system* banyak dibutuhkan dalam organisasi karena *knowledge* yang ada pada organisasi tersebut harus dikelola agar termanfaatkan dengan baik. Untuk membangun sistem ini digunakan metode *Software Development Life Cycle* dengan model *waterfall* sebagai metode *Software Engineering*. Adapun tahapan SDLC antara lain *user requirement, design analysis*, *implementation*, dan *testing*. Dalam penerapannya, *knowledge management system* ini dibangun menggunakan Mediawiki sebagai *framework* untuk membangunnya. Dalam skripsi ini dilakukan tiga pengujian, yaitu *unit testing, load testing*, dan *usability testing*. Dari pengujian yang dilakukan didapatkan hasil *unit testing* keberhasilan sebesar 96,47% yang menunjukkan bahwa fitur-fitur yang terdapat pada sistem sudah berjalan dengan baik. Dari *load teting* didapatkan hasil rata-rata *response time* untuk akses halaman utama sebesar 4705,1 ms dan rata-rata *response time* untuk akses halaman artikel sebesar 2173,5 ms, dimana hasil tersebut menunjukkan *response time* yang wajar. *Untuk load testing* dengan variasi *virtual user* menunjukkan kenaikan *response time* yang linear. Dari *usability testing* didapatkan hasil untuk pengguna rata-ratanya sebesar 4,73 dan untuk responden acak menghasilkan nilai rata-rata sebesar 4,14 dari skala 1-6. Hal tersebut menunjukkan sistem yang diterapkan sudah sesuai terhadap perusahaan dan cukup efektif dalam penggunaannya.



Kata kunci: *Knowledge Management System , Mediawiki, Semantic, software engineering*.

# ABSTRACT


Name : Irvanda Kurniadi Virdaus

Study Program : Computer Engineering


Title : Implementation of Knowledge Management System Using Semantic Mediawiki for Operasional Division in Telecomunication Company


Supervisor : Prima Dewi Purnamasari, S.T., M.T., M.Sc.



KMS is built to manage knowledge, either tacit or explicit knowledge. KMS is needed to manage knowledge so that organization can use their knowledge properly.( versi lain: Many organization need KMS because their knowledge need to be managed so that the knowledge can be used properly ). Software Engineering Development Life Cycle with Waterfall model was used in the KMS developing process of this research. The SDLC phase of Waterfall model consists of *user* requirement gathering, design analysis, implementation, and testing phase. In the implementation phase, Mediawiki was used as the framework to build the KMS. Three kind of testing was used in this research, which were unit testing, load testing, and usability testing. It can be derived from unit testing that the success rate to use the system was 96,47% which means the system features was running well. From load testing, showed that the average response time to access the main page was 4705,1 ms and the average response time to access article page was 2173,5 ms, which means the system had acceptable response time. Load testing with virtual *user* variation showed that there was linear response time escalation. The result of usability testing was 4,73 for average *user* and 4,14 for random respondents in 1-6 scale. In conclusion, the testing result showed that the system had been implemented properly according to corporate needs.



Keywords: *Knowledge Management System , Mediawiki, Semantic, Software Engineering.*

# DAFTAR ISI

## DAFTAR GAMBAR

# DAFTAR TABEL

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Dewasa ini pengkajian terhadap *knowledge management* sudah banyak dilakukan utamanya untuk implementasi terhadap organisasi-organisasi atau divisi-divisi pada suatu perusahaan, bahkan untuk penggunaan perusahaan tersebut secara umum. Pengkajian terhadap *knowledge management* ini dititikberatkan kepada seberapa besar pengaruh *knowledge management* terhadap utilisasi proses bisnis suatu perusahaan. Mengelola *knowledge* berarti mendayagunakan *knowledge* yang ada, baik *knowledge* dari masing-masing individu maupun hasil *knowledge* dari hasil diskusi kelompok yang digunakan untuk kebutuhan proses bisnis suatu perusahaan. Pengelolaan *knowledge* tersebut dimodelkan dalam suatu sistem yang dinamakan *Knowledge Management System*.

Pada perusahaan yang bergerak di bidang telekomunikasi juga membutuhkan *Knowledge Management System* untuk meningkatkan utilisasi proses bisnis divisi *operational* dan *maintenance* yang lingkup kerjanya berada di sekitar penyelesaian masalah gangguan dan *maintenance*. Dalam menjalankan proses bisnisnya, divisi operasional harus memiliki tiga aspek utama yang harus dikuasai, yaitu :

1. Kecepatan respon
2. Kecepatan mengerti dan memahami
3. Kecepatan penanganan masalah (*troubleshoot*)

Dari ketiga aspek tersebut, hal utama yang ingin ditingkatkan dengan adanya *Knowledge Management System* ini adalah kecepatan mengerti dan memahami serta kecepatan *troubleshoot*. Sebelum adanya *Knowledge Management System* ini sudah diciptakan model pembelajaran berbasis *training* sehingga kecepatan mengerti dan memahami serta kecepatan *troubleshoot* mampu ditingkatkan pada operator-operator yang bekerja pada divisi operasional. Namun banyak keterbatasan dari model yang diterapkan ini, yaitu tingkat kadaluarsa yang sangat tinggi. Dengan adanya model training ini, operator mau tidak mau harus di-

*training* secara berkala untuk mengatasi kekadaluarsaan *knowledge* yang dimiliki sehingga model ini kurang efektif diterapkan. Oleh karena itu, dengan kehadiran model *Knowledge Management System* ini, dua dari tiga aspek tersebut dapat ditingkatkan dengan model yang efektif dan efisien. *Knowledge Management System* dikatakan model yang efektif dan efisien karena model ini memberikan pengelolaan *knowledge* yang tak terbatas kepada penggunanya.

Paparan di atas merupakan ide dasar mengapa dibutuhkan sebuah *Knowledge Management System* (Sistem Manajemen Pengetahuan), utamanya pada divisi operasional. *Knowledge Management System* tersebut berisi kumpulan *knowledge* yang dibutuhkan, tidak hanya *knowledge* mengenai terminologi dan konsep, tetapi juga kasus-kasus yang terjadi pada saat *troubleshooting*. Kasus-kasus yang pernah terjadi dan terselesaikan akan terekam dalam sebuah sistem sehingga orang lain yang memiliki masalah yang sama mampu menggunakan *knowledge* tersebut untuk kebutuhan menentukan solusi *troubleshoot* yang tepat dalam waktu yang lebih singkat.

Pada tulisan ini, permasalahan yang menjadi fokus adalah perancangan dan implementasi *Knowledge Management System* pada sebuah perusahaan telekomunikasi pada divisi operasional berdasarkan *user requirement* yang diberikan menggunakan prinsip-prinsip *Knowledge Life Cycle* serta *Decision Support System* yang dikerjakan menggunakan kaidah-kaidah *System Development Life Cycle*.

Masalah yang diidentifikasi adalah perusahaan ingin meningkatkan efisiensi dan efektivitas proses bisnis pada divisi operasional dengan cara mengintegrasikan kumpulan *knowledge* yang berisi terminologi serta konsep dengan kasus-kasus yang pernah terjadi serta solusi yang diberikan. Dalam integrasi *knowledge* tersebut terdapat aspek *Decision Support System* yang mampu menganalisis *knowledge* terkait sehingga menjadi informasi baru yang dapat menjadi acuan *decision making*.

## 1.2 **Tujuan Penulisan**

Tujuan perancangan dan implementasi Mediawiki menggunakan konsep *Knowledge Management System* yang tertuang dalam penulisan skripsi ini secara

umum adalah untuk meningkatkan efektivitas serta kecepatan kerja dari divisi operasional. Dari tujuan umum tersebut dapat dirinci kembali menjadi beberapa tujuan khusus sebagai berikut:

1. Mengimplementasi model *Knowledge Management System* pada divisi operasional Perusahaan Telekomunikasi.
2. Mengimplementasi Mediawiki sebagai alat untuk menerapkan sistem yang telah dirancang.
3. Mengukur performansi sistem yang diterapkan pada perusahaan.
4. Mengukur efektivitas sistem serta kegunaan terhadap pengguna.

## 1.3 Batasan Masalah

Permasalahan dalam penulisan skripsi ini dibatasi pada perancangan bentuk *Knowledge Managemet System* yang tepat serta implementasi Mediawiki sebagai *tools* yang digunakan untuk menerapkan konsep *Knowledge Management System* dengan menerapkan *case-based reasoning* pada konsep *Knowledge-Based System*. Pada proses pelaksanaannya, perancangan dan implementasi ini meliputi hal-hal berikut:

1. Desain dan analisis
2. Perencanaan sistem
3. Pembuatan model sistem
4. Implementasi model
5. Pengukuran performansi sistem dan keefektivan sistem

## 1.4 Metode Penelitian

Metode penelitian yang digunakan dalam penulisan skripsi ini adalah studi literatur serta implementasi dari *tools* yang ada dan integrasi modul berdasarkan desain sistem dengan menggunakan kaidah *Software Development Life Cycle*. Studi literatur dilakukan untuk mendapatkan latar belakang masalah dan merumuskan solusi dari masalah tersebut. Selain itu, studi literatur juga sebagai dasar teori dan teknis dalam implementasi *tools* dan modul. Kaidah *Software Development Life Cycle* meliputi:

1. *Requirement definition*

2. *System and software design*
3. *Implementation and unit testing*
4. *Integration and system testing*
5. *Operation and maintenance*

Penulisan skripsi ini mencakup tahapan *user requirement, system design, implementation,* dan *testing*. Dalam perancangan sistem yang telah dilakukan, digunakan pula *Unified Modelling Language* (UML) sebagai bahasa standar dalam pemodelan rancangan perangkat lunak.

## 1.5 **Sistematika Penulisan**

Sistematika dari penulisan skripsi ini adalah sebagai berikut :

Bab 1 Pendahuluan

Bab Pendahuluan berisi Latar Belakang, Rumusan Masalah, Tujuan Penulisan, Batasan Masalah, Metode Penelitian, dan Sistematika Penulisan.

Bab 2 *Knowledge Management System*

Bab *Knowledge Management System* berisi tentang literatur yang berkaitan dengan teori merancang *Knowledge Management System*, teori mengenai *System Development Life Cycle*, kemudian juga mengenai *framework* yang digunakan yatitu Mediawiki.

Bab 3 Perancangan *Knowledge Management System* dengan *Semantic* Mediawiki

Bab Perancangan *Knowledge Management System* dengan *Semantic* Mediawiki berisi tentang perancangan desain sistem yang terdiri dari *user requirement* (berisi *functional requirement* dan *knowledge audit*), serta analisis desain yang berisi tentang pemodelan sistem kedalam bentuk *class diagram* dan *object diagram*, dan terakhir adalah *deployment diagram.*

Bab 4 Implementasi dan Integrasi *Knowledge Management System* Berbasis *Semantic* Mediawiki

Bab Implementasi dan Integrasi *Knowledge Management System* Berbasis *Semantic* Mediawiki berisi tentang implementasi sistem dengan

menggunakan Mediawiki, integrasi desain sistem dengan *semantic* Mediawiki, serta implementasi sistem pada *server*.

Bab 5 Pengujian dan Analisis

Bab Pengujian dan Analisis berisi tentang skenario pengujian, hasil pengujian dan analisis, serta analisis representasi *knowledge* terhadap *Knowledge Management System*.

Bab 6 Kesimpulan

Bab Kesimpulan berisi tentang kesimpulan penulisan.

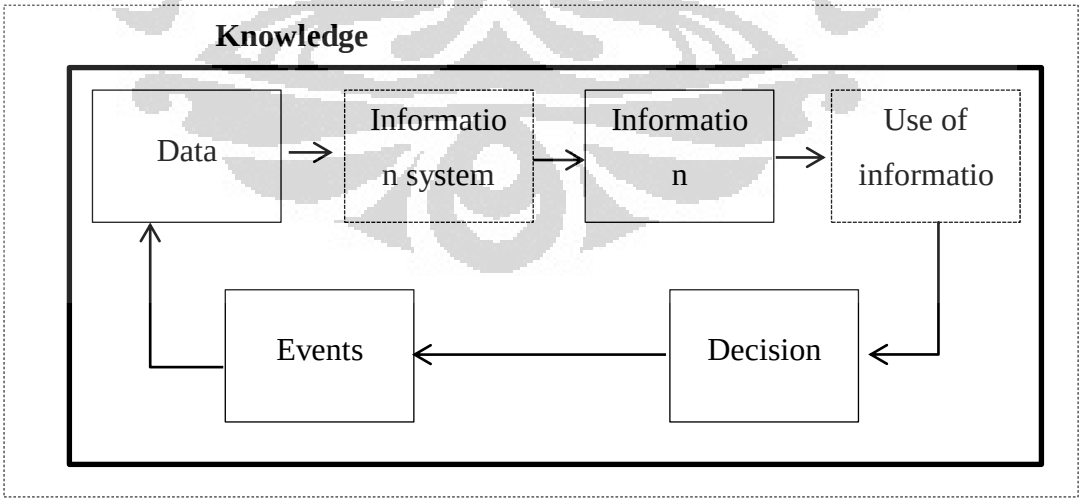
# BAB 2
# *KNOWLEDGE MANAGEMENT SYSTEM*

Bab ini berisikan teori-teori pendukung penelitian yang didapatkan dari hasil studi literatur. Setiap teori yang ada pada bab ini dijadikan sebagai landasan dalam perancangan dan implementasi sistem.

## 2.1 *Knowledge*

*Knowledge* merupakan kemampuan seseorang/individu dalam menghubungkan dan mengaitkan setiap informasi yang dimiliki olehnya dengan konsep-konsep lain yang relevan dengan area tertentu untuk digunakan dalam proses pengambilan keputusan. Berbeda dengan data informasi, sebuah *knowledge* erat kaitannya dengan konteks yang menentukan relevansi sebuah informasi terhadap situasi atau kondisi tertentu [9].

Dalam buku yang ditulis oleh Becerra-Fernandez, disebutkan bahwa *knowledge* tidak jauh dari data dan informasi walaupun ketiganya dapat dipertukarkan dalam penggunaannya seperti yang ditunjukkan pada Gambar 2.1 [1]. *Knowledge* merujuk kepada informasi yang mengandung *actions* dan *decisions.*




Gambar 2.1. Hubungan data, informasi, dan *knowledge* [1]

**Data** merupakan unit terkecil yang bersifat statis dan merupakan representasi dari fakta, observasi, dan persepsi (bisa benar ataupun salah) yang ditemukan dalam aktivitas sehari-hari. **Informasi** adalah hasil pengolahan dari data yang dapat memberikan gambaran lebih jelas terhadap suatu trend atau pola dari data tersebut [1]. Informasi bersifat dinamis. Bagi operator pada perusahaan telekomunikasi, data yang terekam dalam suatu aktivitas bisnis (*trouble* dan solusi) merupakan contoh dari informasi. Manajer dapat menggunakan informasi tersebut untuk membuat keputusan dalam menentukan permasalahan yang sering diakses dan mencari solusi permasalahan tersebut.

*Knowledge* memiliki keterkaitan antara data dan informasi dimana *knowledge* didefinisikan sebagai data dan informasi yang digabung dengan kemampuan, intuisi, pengalaman, gagasan, motivasi dari sumber yang kompeten. *Knowledge* merupakan level tertinggi, sedangkan informasi pada level menengah, dan data pada level rendah. *Knowledge* dapat merujuk pada suatu informasi yang memiliki arah, aksi, dan membuat keputusan, dimana aksi dan keputusan menjadi poin penting dalam sebuah *knowledge* yang dirujuk dari sebuah informasi.

### 2.1.1 Tipe *Knowledge*

*Knowledge* dapat diklasifikasikan ke dalam beberapa tipe. Merupakan hal penting dalam memahami tipe-tipe dari *knowledge* karena setiap tipe dari *knowledge* membutuhkan penanganan yang berbeda.

#### *2.1.1.1 Procedural* atau Deklaratif *Knowledge*

Pengetahuan prosedural (*procedural knowledge*) adalah pengetahuan mengenai bagaimana melakukan sesuatu. Pengetahuan deklaratif (*declarative knowledge*) merupakan pengetahuan yang sifatnya jelas [1]. Pengetahuan prosedural berbeda dari pengetahuan deklaratif. Pengetahuan prosedural lebih banyak diterapkan pada tugas. Misalnya pengetahuan prosedural yang digunakan untuk memecahkan masalah berbeda dari pengetahuan deklaratif yang dimiliki seseorang tentang pemecahan masalah tersebut. Pengetahuan deklaratif biasa dikaitkan dengan "*know-what*", sedangkan pengetahuan prosedural biasa dikaitkan dengan "*know-how*".

2.1.1.2 *Knowledge* yang bersifat *Tacit* atau *Explicit*

Klasifikasi penting lainnya dari *knowledge* adalah *knowledge* dipandang sebagai *Tacit* atau *Explicit*. *Explicit knowlege* merujuk kepada suatu *knowledge* yang direpresentasikan ke dalam sebuah angka dan huruf. Sehingga *knowledge* semacam ini dapat dibagi secara sistematis dalam bentuk data, spesifikasi, suara, gambar, program komputer, dan sebagainya.

Sebaliknya, *Tacit knowledge* mencakup wawasan, intuisi, dan dugaan. Tipe *knowledge* ini sulit untuk di ekspresikan dan dirumuskan, sehingga *knowledge* ini sulit untuk dibagi. Sebagai contoh, pada penanganan suatu permasalahan jaringan, seorang *network engineer* menggunakan intuisinya dalam menganalisis masalah yang ada. Setiap individu pasti memiliki intuisi yang berbeda dan terkadang intuisi tersebut yang menyebabkan penyelesaian masalah lebih cepat dikerjakan.

Dalam aplikasinya, *tacit knowledge* dan *explicit knowledge* memiliki penanganan yang berbeda karena keduanya jelas berbeda. Namun, agar *tacit knowledge* dapat ditangkap menjadi data yang kemudian dapat dipadukan menjadi sebuah informasi, dapat dilakukan konversi dari *tacit knowledg*e ke *explicit knowledge*.

2.1.1.3 *Knowledge* yang Umum atau Spesifik

Klasifikasi ketiga merupakan klasifikasi *knowledge* berdasarkan fokus apakah *knowledge* ini bersifat general atau spesifik. *General knowledge* dimiliki oleh kumpulan individu dan dapat dibagi dengan mudah antarindividu. Misalnya, peraturan permainan bola basket dapat dikategorikan sebagai *general knowledge*, khususnya untuk penonton yang ada di tribun penonton. Contoh kasus yang menerapkan *general knowledge* yaitu ketika seorang *network engineer* melakukan penyelesaian masalah terhadap jaringan yang mati. Jika permasalahan tersebut disebabkan oleh kabel yang putus, pelanggan akan langsung mengerti bahwa penyebab masalahnya terdapat pada kabel.

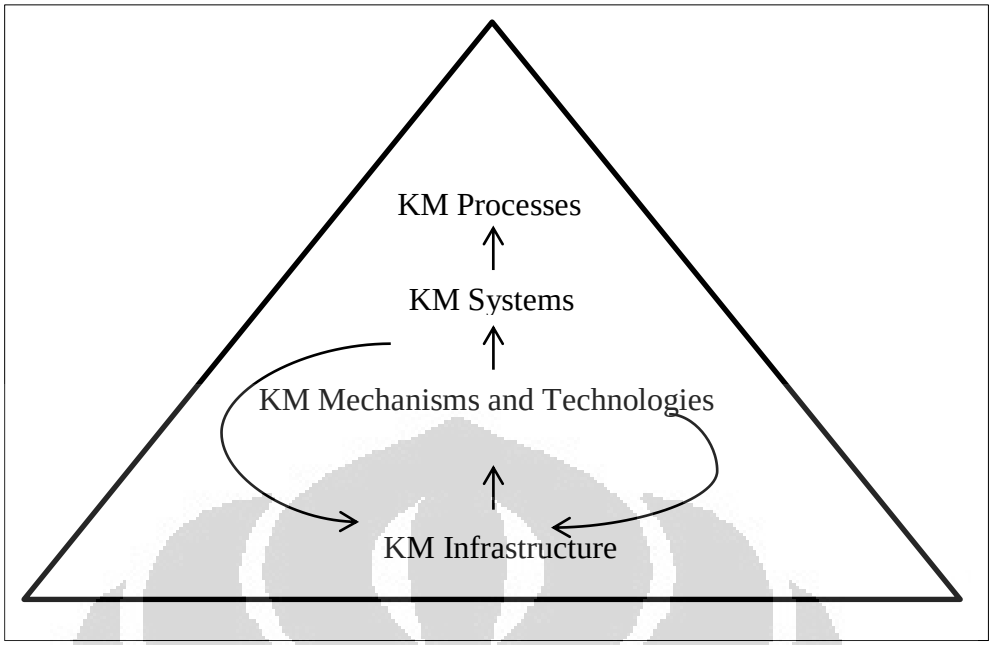
Berbeda dengan *specific knowledge*, *knowledge* ini hanya dimengerti oleh sekumpulan kecil individu sehingga sulit untuk dibagi. Contoh dari *knowledge* ini adalah pengetahuan mengenai istilah dalam bidang telekomunikasi. Ketika para *network engineer* berbicara mengenai *troughput* suatu *traffic* pada jaringan yang

sedang ditanganinya, hanya sesama *network engineer* yang memahami istilah tersebut. Orang lain yang mungkin tidak sengaja mendengarnya tidak akan mengerti maksud istilah tersebut. Hal ini yang dikatakan spesifik. *Specific knowledge* dapat dikategorikan dalam dua bentuk, yaitu secara teknis (*technically specific knowledge*) dan secara konteks (*contextually specific knowledge*). *Specific knowledge* secara teknis mencakup *knowledge* tentang alat-alat dan teknik untuk memecahkan suatu masalah, sedangkan secara konteks merujuk pada *knowledge* pada keadaan khusus dari waktu dan tempat dimana kerja dilakukan. Jadi secara kontekstual, *knowledge* merujuk pada sesuatu yang dikerjakan.

## 2.2 *Knowledge Management*

Pembahasan mengenai *knowledge* sudah diterangkan pada bahasan di atas. *Management* berarti merencanakan, mengumpulkan dan mengorganisir, mengoordinasikan *resource* untuk suatu tujuan. *Knowledge management* dapat diartikan sebagai proses yang mengoordinasikan penggunaan informasi, pengetahuan, dan pengalaman.

*Knowledge management* meliputi strategi manajemen, metode, dan teknologi untuk melindungi modal intelektual perusahaan dan langkah-langkah pengerjaan untuk mencapai hasil optimal dalam unjuk kerja dan daya saing. *Knowledge management* adalah suatu disiplin ilmu yang mempromosikan pendekatan menyeluruh untuk mengidentifikasi, mengelola, dan membagi seluruh aset informasi yang dimiliki oleh suatu perusahaan. Aset informasi ini bisa berupa *database*, dokumen, kebijakan, prosedur, dan keahlian serta pengalaman yang dimiliki oleh karyawan. *Knowledge management* termasuk membangun, mengimplementasikan, dan memelihara infrastruktur teknis dan organisasi untuk memungkinkan saling berbagi *knowledge* serta memilih vendor dan teknologi tertentu yang dapat mendukungnya.



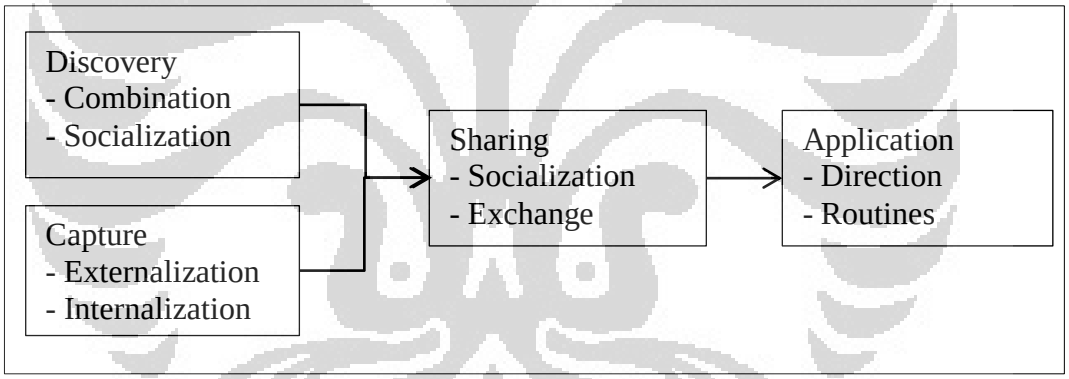

Gambar 2.2. *Overview* dari *knowledge management solution* [1]

*Knowledge management* adalah proses bisnis untuk memperoleh, mengelola, dan mengomunikasikan *tacit knowledge* maupun *explicit knowledge* sehingga memungkinkan orang lain untuk menggunakannya secara lebih efektif dan produktif. Dalam bukunya, Becerra-Fernandez menuliskan bahwa dalam membangun sebuah *knowledge management* ada dua poin yang perlu diperhatikan yaitu *knowledge management solution* (seperti yang diperlihatkan pada Gambar 2.2) dan *knowledge management system*. *knowledgem management solution* merujuk pada langkah untuk memfasilitasi suatu *knowledge management*. Langkah tersebut dapat dibagi dalam empat level, yaitu (1) *knowledge management processes*, (2) *knowledge management Systems*, (3) *knowledge management Mechanisms and Technologies*, dan (4) *knowledge management Infrastructure*.

*Knowledge management process* merupakan sebuah proses yang didalamnya berisi proses mendapatkan (*discovering*), proses menangkap *knowledge* (*capturing*), proses berbagi *knowledge* (*sharing*), dan proses menerapkan *knowledge* (*applying*). Empat *knowledge management process* ini didukung oleh *Knowledge Management System* .

## 2.3 *Knowledge Management System*

Bahasan mengenai *knowledge* dan *management* sudah dijabarkan pada penjelasan di atas. *System* adalah perpaduan dari beberapa bagian suatu aksi yang dikerjakan bersama-sama. *Knowledge Management System* dapat diartikan sebagai kumpulan proses yang mengoordinasi penggunaan informasi, pengetahuan, dan pengalaman yang berjalan dan bekerja bersama-sama. *Knowledge management system* juga sudah digambarkan dalam bentuk piramida *knowledge management solution* dimana *knowledge management systems* merupakan integrasi dari teknologi dan mekanisme yang dibangun untuk mendukung *knowledge management process*. Proses-proses yang terjadi merupakan dasar dari konsep *knowledge management system* dimana poin penting dari *knowledge management system* ini terletak pada *discovering*, *capturing*, *sharing*, dan *applying knowledge* seperti yang diperlihatkan pada Gambar 2.3.




Gambar 2.3. *Knowledge Management Process*[1]

*Knowledge Discovery* merupakan suatu konsep yang menggambarkan suatu proses pencarian data. *Knowledge discovery* dapat didefinisikan sebagai pengembangan pengetahuan *tacit* maupun eksplisit baru dari data dan informasi atau dari sintesis pengetahuan sebelumnya. Penemuan pengetahuan eksplisit baru bergantung pada kombinasi paling langsung, sedangkan penemuan pengetahuan tacit yang paling baru bergantung langsung pada sosialisasi. Pada *knowledge discovery* ini, intinya terdapat pada kombinasi dan sosialisasi terhadap *knowledge* yang *tacit* maupun yang eksplisit.

*Knowledge Capture* dapat didefinisikan sebagai proses mengambil pengetahuan eksplisit ataupun *tacit* yang berada dalam diri seseorang, artefak, atau entitas organisasi. Proses *knowledge capture* secara langsung memanfaatkan dua proses yaitu eksternalisasi dan internalisasi. Eksternalisasi melibatkan konversi *tacit knowledge* ke dalam bentuk eksplisit seperti kata-kata, konsep, visual, atau bahasa kiasan. Eksternalisasi juga membantu menerjemahkan *tacit knowledge* individu ke dalam bentuk-bentuk eksplisit yang dapat lebih mudah dipahami oleh anggota kelompok. Contoh dari eksternalisasi dapat digambarkan dalam sebuah tim konsultan, dimana konsultan menuliskan sebuah dokumen yang menjelaskan suatu pembelajaran bahwa tim telah belajar tentang organisasi klien, eksekutif klien, dan pendekatan mengenai pekerjaan tersebut dalam suatu *assignment*. Hal tersebut menangkap *tacit knowledge* yang diperoleh dari anggota tim. Internalisasi merupakan konversi dari eksplisit ke *tacit knowledge*. Hal tersebut merepresentasikan gagasan tradisional dari sebuah pembelajaran. Pengetahuan eksplisit dapat diwujudkan dalam tindakan dan praktek sehingga individu yang memperoleh pengetahuan dapat mengalami kembali apa yang orang lain telah alami. Sebuah contoh dari internalisasi adalah konsultan baru yang membaca buku tentang pengembangan perangkat lunak yang inovatif dan mempelajarinya. Pembelajaran ini membantu konsultan dan organisasi menangkap pengetahuan yang terkandung dalam buku tersebut.

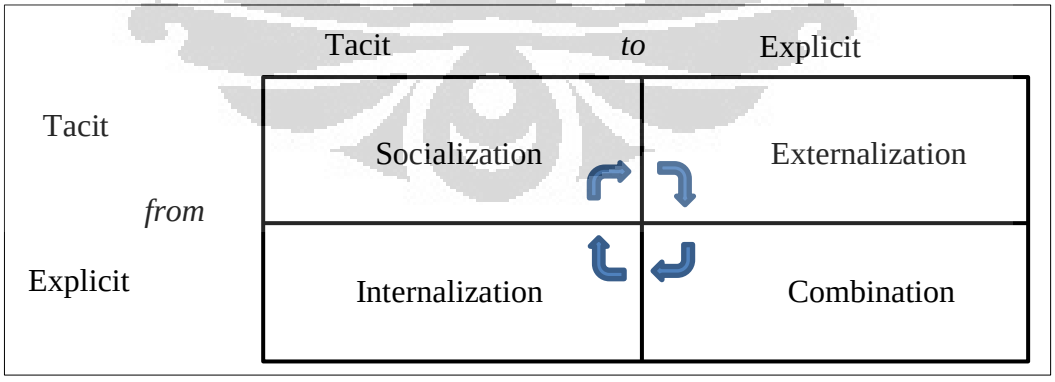
*Knowledge Sharing* merupakan proses dimana pengetahuan eksplisit atau *tacit* dikomunikasikan dan dibagi kepada orang lain. Ada tiga hal penting yang perlu diperhatikan dalam *knowledge sharing*. Pertama, berbagi pengetahuan memiliki arti transfer secara efektif sehingga penerima pengetahuan dapat mengerti dengan cukup baik. Kedua, apa yang dibagi adalah pengetahuan, bukan rekomendasi berdasarkan pengetahuan; yang tadinya melibatkan penerima memperoleh pengetahuan bersama serta memiliki kemampuan untuk mengambil tindakan berdasarkan pengetahuan tersebut, sedangkan yang terakhir hanya melibatkan pemanfaatan pengetahuan tanpa penerima internalisasi pengetahuan bersama. Ketiga, berbagi pengetahuan dapat terjadi di seluruh individu maupun seluruh kelompok, departemen, atau organisasi. *Knowledge sharing* memiliki dua aspek yang terdapat di dalamnya, yaitu aspek *exchange* dan aspek sosialisasi.

Aspek *exchange* lebih fokus kepada *explicit knowledge*, sedangkan aspek sosialisasi lebih cenderung kepada *tacit knowledge*.

*Knowledge Application* merupakan proses penerapan *knowledge* setelah melalui tahap *discovery*, *capture*, dan *storage*. *Knowledge* berkontribusi langsung terhadap kinerja organisasi bila digunakan untuk membuat keputusan dan melakukan tugas-tugas. Tentu saja proses penerapan *knowledge* tergantung pada *knowledge* yang tersedia dan bergantung pada *knowledge discovery*, *capture*, dan *storage*. Semakin baik proses *discovery*, *capture*, dan *storage*, semakin besar kemungkinan bahwa *knowledge* yang dibutuhkan untuk membuat keputusan yang efektif tersedia. Pada prosesnya terdapat dua komponen yaitu *direction* dan *routine*. *Direction* mengacu pada proses melalui mana individu-individu yang memiliki pengetahuan mengarahkan tindakan individu lain tanpa mentransfer ke orang yang menekankan arah *knowledge*, sedangkan *Routines* melibatkan pemanfaatan pengetahuan tertanam dalam prosedur, aturan, dan norma-norma yang menuntun perilaku masa depan.

### 2.3.1 *Knowledge Management Life Cycle*

Ada dua macam pengetahuan, yaitu *explicit knowledge* dan *tacit knowledge*. *Explicit knowledge* dan *tacit knowledge* sudah banyak diterangkan di atas. Dalam konsepnya, *knowledge life cycle* dapat dilihat melalui Gambar 2.4.




Gambar 2.4. Konversi antara *tacit* dan *explicit knowledge* [9]

Dimensi pertama dari penciptaan *knowledge* adalah *tacit-explicit dimension*. Pada Gambar 2.4, kedua mode tersebut muncul dua kali. Matrik

tersebut menjelaskan tentang empat kemungkinan konversi yang dapat terjadi dari dua *knowledge* tersebut yaitu *tacit* dan *explicit knowledge*.

2.3.1.1 *Tacit knowledge* ke *tacit knowledge* (*Socialization*)

*Tacit knowledge* berkaitan erat dengan pengalaman dan intuisi seseorang sehingga sangat sulit untuk dikomunikasikan. Melalui proses sosialisasi, seseorang dapat berbagi pengalaman dan intuisi melalui cerita, pengalaman, dan juga mencontoh. Proses transfer *knowledge* yang terjadi pada sosialisasi juga terjadi secara langsung dari satu orang ke orang lain.

2.3.1.2 *Tacit knowledge* ke *explicit knowledge* (*Externalization*)

Eksternalisasi adalah suatu proses mengolah *tacit knowledge* menjadi suatu konsep yang eksplisit. Hal ini bisa dilakukan diantaranya dengan dokumentasi. Dengan dokumentasi, *knowledge* yang bersifat *tacit* / implisit dapat ditangkap dengan mudah.

2.3.1.3 *Explicit knowledge* ke *explicit knowledge* (*Combination*)

Kombinasi adalah suatu bentuk penggabungan antara *explicit knowledge* dengan *explicit knowledge* yang lain. Sebagai contoh, seorang manager di suatu perusahaan ingin menggunakan dokumen mengenai aturan perusahaan dan dokumen mengenai *project budget* kemudian digunakan untuk membuat suatu dokumen baru mengenai *quality assurance* plan untuk sebuah *project*. Jika kapasitas suatu manager lebih sedikit dibandingkan dengan dokumen yang digunakannya, ini disebut kombinasi.

2.3.1.4 *Explicit knowledge* ke *tacit knowledge* (*Internalization*)

Internalisasi adalah suatu bentuk penyerapan *explicit knowledge* menjadi *tacit knowledge*. Secara singkat, internalisasi dapat dilakukan dengan *learning by doing*. Sebagai contoh, seseorang yang ingin belajar memasak akan melihat buku resep masakan untuk dicobanya. Proses ini yang disebut internalisasi.

2.3.2 *Knowledge Audit*

*Knowledge audit* merupakan sebuah proses yang terdapat di dalam sebuah *knowedge management system* [6]. *Knowledge audit* merupakan suatu bentuk pemeriksaan dan evaluasi yang sistematik terhadap kesehatan *knowledge* suatu

organisasi, dimana pemeriksaan itu meliputi kebutuhan *knowledge* suatu organisasi, *knowledge* aset yang sudah ada, *knowledge flow*, kebutuhan *knowledge* dimasa mendatang, serta analisis celah *knowledge*, baik pada kelakuan seseorang dalam *sharing*, maupun *creating knowledge* [10]. Di satu sisi, *knowledge audit* dapat menyatakan sebuah SWOTR (*Strengths, Weaknesses, Opportunities, Threats*, dan *Risks*) dari sebuah perusahaan. Sebuah *knowledge audit* juga meliputi pemeriksaan terhadap strategi, *leadership*, *collaborative*, *learning culture*, dan infrastruktur teknologi dari suatu organisasi dalam berbagai macam proses *knowledge*.

Dalam rangka mengubah suatu organisasi menjadi sebuah *learning organization* dan menjamin sebuah strategi *knowledge management* yang efektif, sebuah *knowledge audit* harus dilakukan, yang kemudian akan menyediakan sebuah keadaan saat itu dari kemampuan *knowledge* dari suatu organisasi dan sebuah arahan terhadap di mana dan bagaimana meningkatkan kemampuan tersebut dalam rangka menjadikan persaingan di era *knowledge* yang perubahannya begitu cepat ini.

*Knowledge audit* dapat dipandang sebagai *review* dari aset pengetahuan perusahaan dan sistem manajemen pengetahuan yang terkait. Hal ini perlu dilakukan untuk dapat memperoleh informasi mengenai apa pengetahuan yang sudah dimiliki oleh perusahaan, apa pengetahuan yang belum dimiliki oleh perusahaan padahal pengetahuan tersebut dibutuhkan, apa pengetahuan yang tidak tertata, siapa yang membutuhkan pengetahuan ini, serta bagaimana menggunakan pengetahuan tersebut. Informasi-informasi inilah yang dibutuhkan untuk bisa dibuat acuan sehingga dapat dibayangkan seperti apa *knowledge management* yang akan dibuat.

Melalui *knowledge audit* ini dapat memperoleh nilai-nilai yang bisa dijadikan pertimbangan untuk pembangunan sistem *knowledge management* yang akan dibuat. Nilai-nilai tersebut dapat dijabarkan sebagai berikut.

- *Knowledge audit* menunjukkan secara pasti nilai mana yang dibuat melalui sumber-sumber yang ada.
- Melihat mana pengaruh terbaik yang dapat diterapkan melalui berbagi peningkatan pengetahuan dan pembelajaran yang terorganisasi.

- Membantu memprioritaskan proyek-proyek untuk meningkatkan praktik manajemen pengetahuan.
- Merupakan komponen kunci sebagai perencanaan strategis untuk perusahaan yang berbasis pengetahuan.

Komponen-komponen yang terdapat pada *knowledge audit* yaitu *knowledge need analysis, knowledge inventory analysis, knowledge flow analysis*, dan terakhir *knowledge mapping* seperti terlihat pada Gambar 2.5.


Gambar 2.5. Komponen Knowledge Audit [10]

Dalam perancangan *knowledge managemen* yang akan dibuat, proses *knowledge audit* sangat berpengaruh pada sistem secara keseluruhan karena dari *knowledge audit* tersebut akan menghasilkan informasi sejauh mana perusahaan tersebut mengenal *knowledge management*, seberapa besar *knowledge management process* yang sudah dijalankan secara tidak sadar, serta *knowledge* apa saja yang dibutuhkan dalam *knowledge management system* yang akan diterapkan.

## 2.4 *System Development Life Cycle* (SDLC) [11]

SDLC merupakan proses pembuatan sistem dengan menggunakan model dan metodologi SDLC untuk mengembangkan sistem-sistem tersebut. Metodologi-metodologi ini membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi.

### 2.4.1 Tahapan SDLC

SDLC merupakan serangkaian tahapan yang dibutuhkan dalam pengembangan suatu sistem, yang dimulai dari investigasi dan analisis kebutuhan dasar, desain, implementasi, dan pemeliharaan.

#### *2.4.1.1 User requirement*

Tahap ini menekankan pada kebutuhan apa saja yang diberikan oleh *user* kepada programer. Kebutuhan tersebut kemudian di klasifikasikan satu per satu untuk kemudian dianalisis dan dibuat dalam bentuk *use case* diagram.

#### 2.4.1.2 Desain dan Analisis Sistem

Tahap ini menekankan desain sistem per kebutuhan. Desain sistem dilihat dari seberapa kompleks kebutuhan yang diberikan *user*. Proses desain sistem disesuaikan berdasarkan spesifikasi pada tahap sebelumnya.

#### 2.4.1.3 Implementasi Sistem

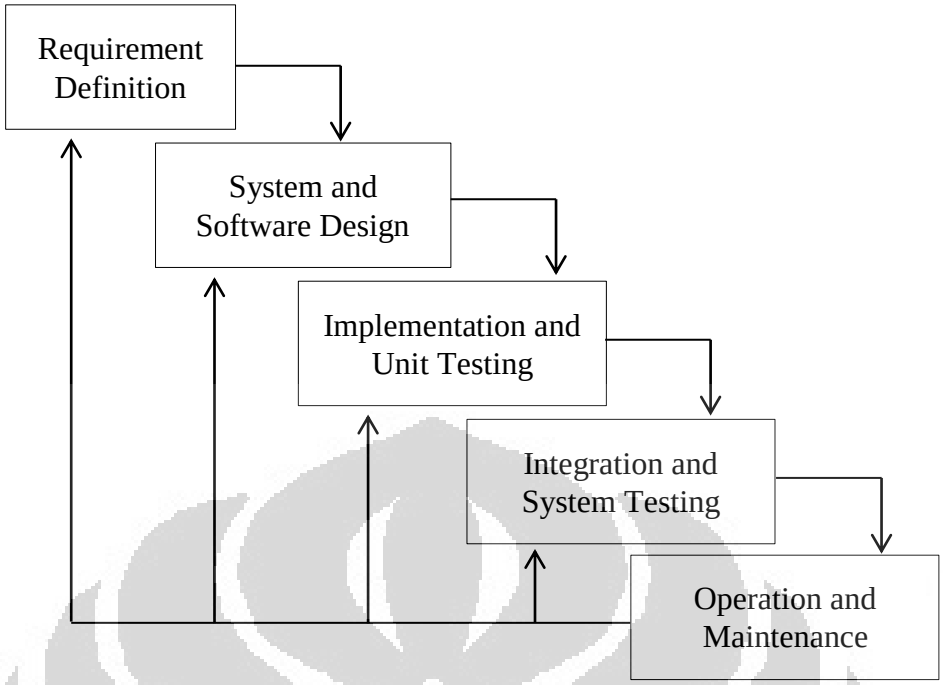
Tahap ini berfokus pada implementasi sistem. Prosesnya bisa berupa *prototyping*. Implementasi sistem ini merupakan tahap *finishing*, karena pada tahap ini juga disertai dengan *system testing*.

#### 2.4.1.4 Pemeliharaan Sistem

Tahapan ini berfokus pada pengoperasian dan pemeliharaan terhadap sistem yang baru agar dapat bekerja dengan baik.

### 2.4.2 Model SDLC

SDLC terdiri dari berbagai jenis model, di antaranya adalah *waterfall model, spiral model, build and fix model, prototyping model*, dan yang lainnya. Dalam pengerjaan *knowledge management system* ini, menggunakan *waterfall model* seperti terlihat pada Gambar 2.6.




Gambar 2.6. *Waterfall Model* [11]

Model *waterfall* memiliki beberapa kelebihan, di antaranya adalah :

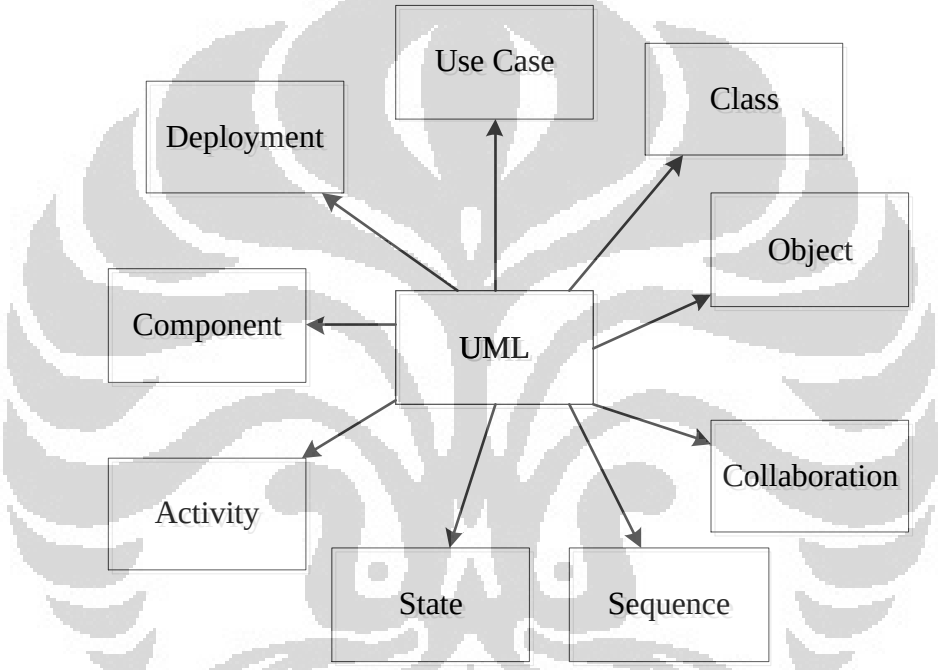
- Proses-prosesnya mudah dipahami dan jelas
- Mudah dalam pengelolaan proyek
- Dokumen dihasilkan setiap akhir fase
- Sebuah fase dijalankan setelah fase sebelumnya selesai
- Struktur sistem jelas
- Kebutuhan *user* sangat dipahami sehingga kecil terjadinya perubahan kebutuhan *user*.

Namun dari sekian kelebihan yang dimiliki oleh *waterfall model*, model ini masih memiliki beberapa kekurangan, di antaranya adalah:

- Proyek dunia nyata jarang mengikuti alur proses
- Kesulitan jika terjadi perubahan kebutuhan, sehingga
- Waktu pengerjaan bertambah
- Ada anggota tim yang harus menunggu pekerjaan pekerja lain
- Kesabaran customer

### 2.4.3 *Unified Modeling Language*(UML)

UML adalah suatu bahasa standar untuk menuliskan *blueprint* perangkat lunak. UML dapat digunakan untuk memvisualisasikan, membuat spesifikasi, mengonstruksikan dan mendokumentasikan *the artifact of a software-intensive system*. UML sesuai untuk pemodelan berbagai macam sistem, mulai dari sistem informasi *enterprise*, aplikasi, *web-based* yang terdistribusi, hingga sistem yang *realtime*. Terdapat 9 diagram UML yang merepresentasikan setiap tahapan proses pembuatan system seperti terlihat pada Gambar 2.7.




Gambar 2.7. UML Diagram [8]

#### 2.4.3.1 Use Case Diagram

*Use case diagram* merupakan diagram UML yang menggambarkan *requirement view*. Diagram ini menunjukkan *user* dan fungsi apa yang disediakan oleh sistem untuk *user* tersebut. Sebuah fungsi dideskripsikan sebagai sebuah transaksi yang memiliki nilai.

#### 2.4.3.2 Class Diagram

*Class diagram* menunjukkan struktur logika dari sebuah sistem. *Class diagram* merupakan inti dari notasi UML dan sebagai *object-oriented design*.

Sebuah *class diagram* memberikan gambaran umum mengenai sebuah sistem dengan menunjukkan kelas-kelasnya dan membuat hubungan di antara kelas-kelas tersebut. *Class diagram* bersifat statis; menunjukkan interaksi antarkelas tapi tidak menunjukkan apa yang terjadi ketika kelas-kelas tersebut melakukan interaksi. Selain dinamakan statis, *class diagram* juga disebut *logical view*.

#### 2.4.3.3 Object Diagram

*Object diagram* hampir sama dengan *class diagram*, hanya saja pada object diagram hanya menunjukkan objek tunggal dan relasinya. *Object diagram* biasa digunakan untuk menunjukkan suatu objek dan ingin menggambarkan objek tersebut dengan cukup jelas dengan memberi beberapa properti di dalamnya.

#### 2.4.3.4 Collaboration Diagram

*Collaboration diagram* merupakan object diagram yang ditambahkan dengan *message arrow* untuk menunjukkan suatu pesan. Diagram ini memperlihatkan rangkaian pesan yang dikirim di antara objek yang dikolaborasikan untuk skenario yang terpisah-pisah.

#### 2.4.3.5 Sequence Diagram

Diagram ini menunjukkan rangkaian pesan yang dikirim antara objek yang terkolaborasi. Diagram ini mirip dengan *collaboration diagram*, yang membedakan adalah bentuk penggambaran diagramnya. *Sequence diagram* menyoroti *flow control* di antara *object diagram*.

#### 2.4.3.6 State Diagram

*State diagram*, atau lebih dikenal sebagai *statechart*, berfungsi untuk mengilustrasikan state dari sebuah objek yang dapat berubah-ubah dan transisi yang menggerakkan objek antar *state*.

#### 2.4.3.7 Activity Diagram

*Activity diagram* menjelaskan tentang aliran dari sebuah aktivitas atau tugas. Diagram ini mirip dengan state diagram, hanya saja pada *activity diagram* memiliki *decision point* dan *synchronization bar*. *Synchronization bar* menunjukkan aktivitas yang dapat berjalan bersamaan. *Activity diagram* juga biasa disebut dengan diagram alir atau *flowchart diagram*.

#### 2.4.3.8 Component Diagram

*Component diagram* mengilustrasikan struktur fisik dari sistem yang ingin diterapkan. Diagram ini menunjukkan komponen dari *software* dan keterkaitan dari komponen-komponen tersebut.

#### 2.4.3.9 Deployment Diagram

*Deployment diagram* menggambarkan sebuah koneksi fisik yang menghubungkan komponen-komponen diagram tersebut. Arsitektur fisik dari sistem meliputi *hardware* yang digunakan hingga komunikasi yang digunakan untuk menghubungkan antar *hardware* tersebut.

## 2.5 ***Knowledge Management Tools*** **[2]**

Dalam merancang sebuah *knowledge management system*, tidak bisa terlepas dari *framework* apa yang akan digunakan. *Knowledge management tools* merupakan *tools* atau alat yang digunakan sebagai *framework* untuk mengimplementasikan sistem *knowledge management* yang akan dibuat.

.

### 2.5.1 Pemilihan *Knowledge Management Tools*

Wikipedia merupakan cikal bakalnya *tools* WikiMedia yang disediakan oleh organisasi Wikipedia sebagai *software opensource*. Salah satu kegunaan WikiMedia selain untuk *content management system*, juga digunakan untuk *framework knowledge management system*. Tentunya ada banyak *knowledge management tools* yang tersedia yang dapat digunakan secara bebas bahkan berbayar. *knowledge management tools* tersebut di antaranya adalah ILIAS, Intellexerr Categorizer, Moodle, dan WikiMedia. Namun selain dari empat yang sudah disebutkan tersebut masih banyak lagi *knowledge management tools* yang ada.

#### 2.5.1.1 ILIAS

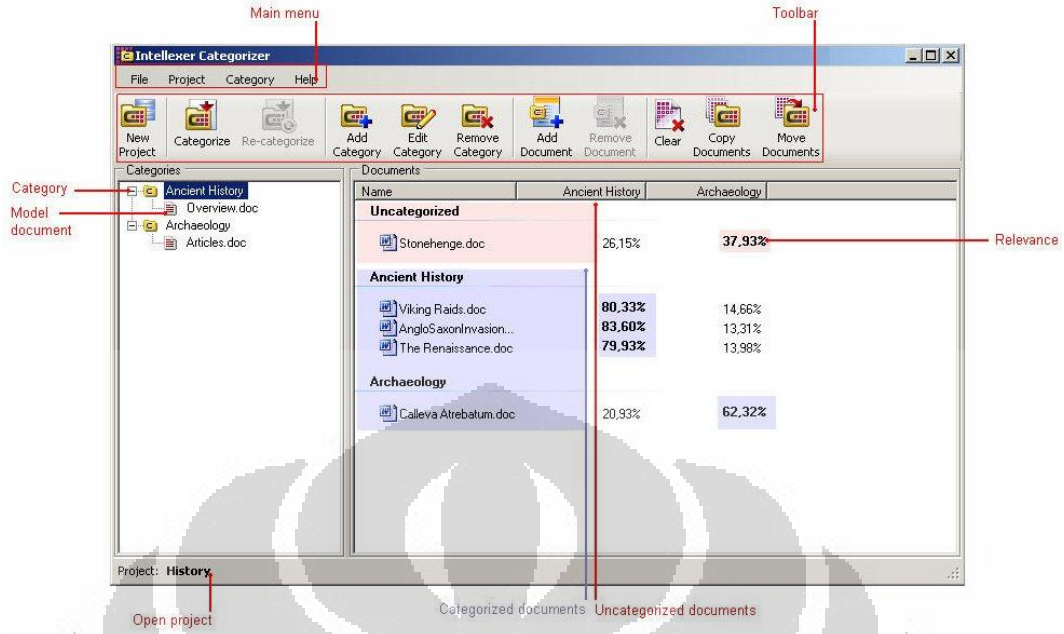
**ILIAS** (*Integriertes Lern-, Informations- und Arbeitskooperations-System* [bahasa Jerman untuk *"Integrated Learning, Information and Work Cooperation System"*]) adalah sebuah *Knowledge Management System* yang digunakan untuk

sarana pembelajaran *online* dan berbasis web dan *open source* atau biasa disebut dengan istilah *open source web-based learning management system* (LMS). Mendukung *learning content management* (termasuk SCORM 2004) dan alat untuk kolaborasi, komunikasi, evaluasi dan penilaian. Perangkat lunak ini diterbitkan di bawah *GNU General Public License* dan dapat berjalan pada server yang mendukung PHP dan MySQL [13]. Tampilan ILIAS dapat dilihat pada Gambar 2.8.


Gambar 2.8. *Home page* ILIAS [12]

### *2.5.1.2 Intellexer Categorizer*

*Intellexer categorizer* merupakan salah satu *knowledge management tools* yang ada dan bersifat komersil. *Software* ini dirilis oleh perusahaan asal Belarus bernama EffectiveSoft. *Intellexer categorizer* secara unik menyediakan fasilitas kepada *user* untuk membuat kategori-kategori berdasarkan definisi-definisi sendiri, selanjutnya kategori itu akan dijadikan acuan untuk penempatan dokumen-dokumen yang menjadi tempat berkumpulnya *knowledge* yang bersifat *tacit* atau *explicit knowledge*. Tampilan *intellexer categorizer* dapat dilihat pada Gambar 2.9.

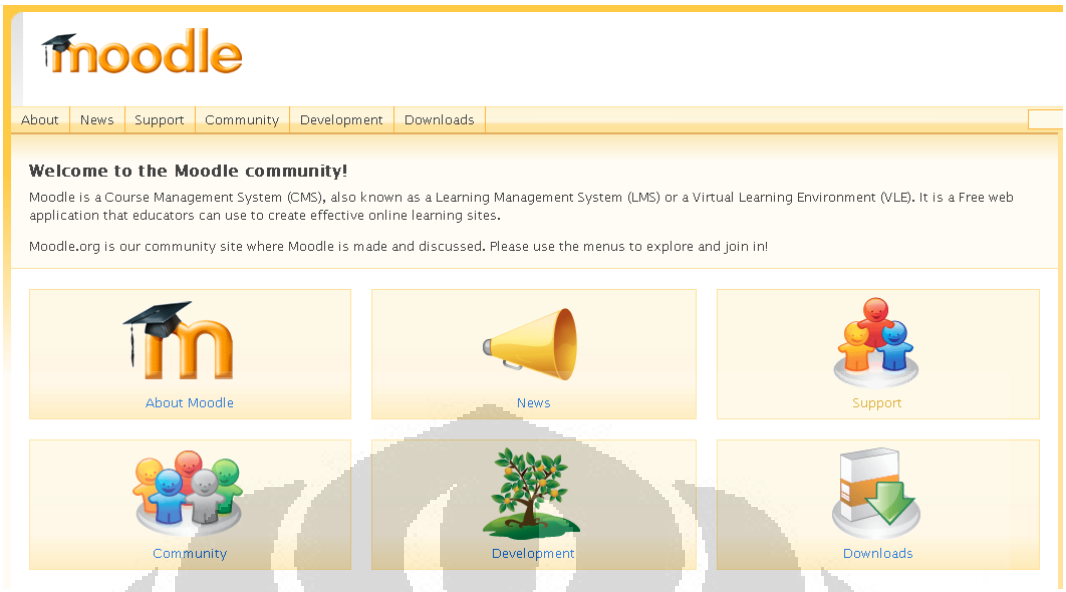



Gambar 2.9. Tampilan Window Intellexer Categorizer [14]

### 2.5.1.3 Moodle

Merupakan singkatan dari "*Modular Object-Oriented Dynamic Learning Environment*". Moodle merupakan *open-source e-learning software platform*, yang bias juga disebut sebagai *Course Management System, Learning Management system, Virtual Learning Environment*. Moodle dikenal pada tahun 1999. Sampai saat ini sebanyak 49,952 telah ter-*register* dalam moodle dan digunakan oleh 37 juta *user* dalam 3,7 juta *course*.

Dalam penggunaannya, moodle banyak digunakan sebagai *Learning Management System*. Namun, moodle juga dapat digunakan sebagai *framework Knowledge Management System*. Akan tetapi ,akan sulit untuk mengembangkan *Knowledge Management System* yang bersifat *semantic web* di sini karena pada dasarnya moodle ini sudah terpaket sebagai *tools* untuk *learning system* sehingga *knowledge management tools* yang dibuat tidak bisa efektif. Tampilan Moodle dapat dilihat pada Gambar 2.10.

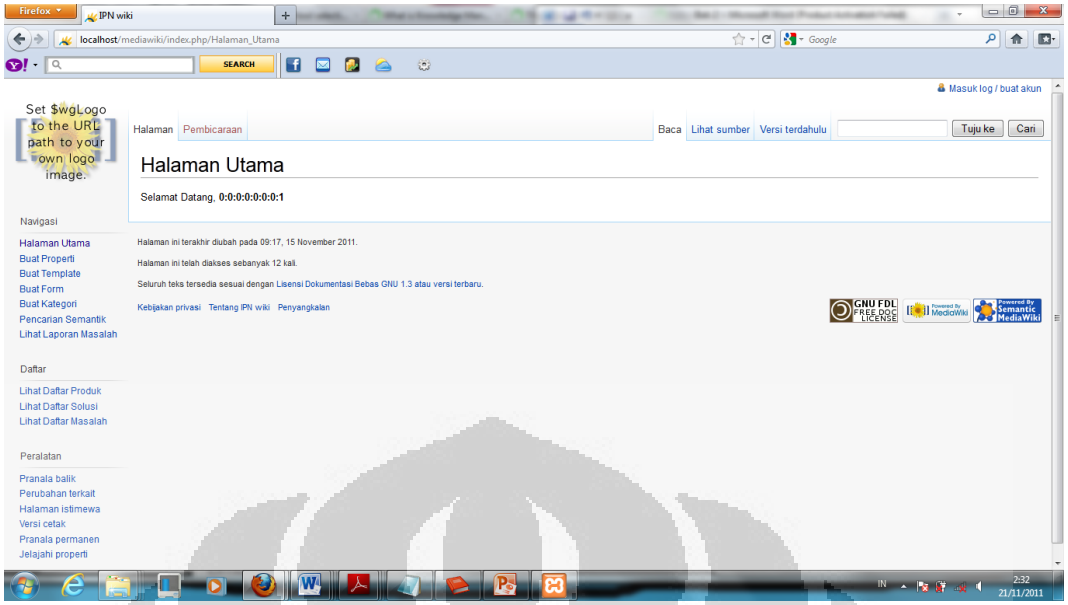



Gambar 2.10. *Home page* moodle [15]

### 2.5.1.4 Mediawiki

WikiWikiWeb merupakan nama pertama dari *software* wiki di dunia maya. WikiWikiWeb pertama kali diperkenalkan oleh Ward Cunningham pada tanggal 25 Maret 1995 sebagai persembahan untuk Portland Pattern Reposity, sebuah perusahaan pengoleksi pola-pola desain *software*.

Cunningham merupakan penemu ide dan pencipta aplikasi Wiki *engine* yang pertama. Ia menggunakan nama Wiki setelah terinspirasi oleh sebuah istilah yang ia dengar di *airport* di kota Honolulu Hawai, yaitu istilah untuk menyebut *shuttle bus*, WikiWiki *bus*. Halaman WikiWikiWeb-nya yang pertama dibuat menjadi berformat HTML dengan menggunakan *script perl* pada tahun 1994-1995. Wiki ditemui dalam banyak bentuk dan aplikasi karena wiki di bawah lisensi GNU maka siapa saja dapat menggunakannya dan memodifikasinya. Oleh karena itu, perkembangan wiki meluas hingga saat ini salah satu aplikasinya yang bernama Mediawiki dapat digunakan. Tampilan Mediawiki dapat dilihat pada Gambar 2.11.




Gambar 2.11. Homepage Mediawiki [16]

Jika dibandingkan dengan ketiga *knowledge management tools* sebelumnya, dapat dikatakan bahwa Mediawiki memiliki poin penting yang tidak dimiliki *knowledge management tools* lainnya yaitu fleksibilitas dan banyaknya *extension* yang tersedia untuk Mediawiki ini. Dalam penerapannya, ILIAS dan moodle lebih cocok digunakan sebagai *Learning Management System*. Sedangkan *Intellexer categorizer* merupakan *knowledge management tools* yang digunakan sebagai *Knowledge-Based management* di mana *knowledge* yang dikelola berupa artefak-artefak yang terdokumentasi. Sedangkan dengan Mediawiki, *knowledge management system* yang dikembangkan dapat diintegrasikan dengan teknologi *semantic web* yang di terapkan dalam salah satu *extension* Mediawiki yaitu *semantic* Mediawiki [4].

### 2.5.2 *Semantic Web*

Dewasa ini, penelitian mengenai *semantic web* sudah banyak dilakukan. *Semantic web* sendiri merujuk kepada teknik yang memungkinkan konten pada web untuk dapat dimengerti oleh mesin [7]. Istilah *semantic web* itu sendiri dicetuskan oleh Tim Berners – Lee, penemu *World Wide Web*. Disadari atau tidak, konten yang saat ini terletak pada *web server* sebenarnya hanya dimengerti oleh manusia atau *user* yang mengaksesnya. Sedangkan mesin sama sekali tidak tahu

sama sekali mengenai konten yang terdapat pada *web server* tersebut. *Semantic wiki* muncul dengan prinsip bahwa setiap konten memiliki properti atau atributnya sendiri sehingga konten tersebut memiliki entitas yang dapat dimengerti oleh mesin. Prinsip tersebut yang dewasa ini disebut-sebut akan muncul pada Web 3.0, generasi ketiga dari *World Wide Web*. Bahkan web 3.0 itu sendiri sering disamakan dengan *semantic web*. Teknologi *semantic web* yang dikembangkan oleh Tim Berners – Lee antara lain adalah RDF, OWL, dan SPARQL.

#### 2.5.2.1 Semantic Mediawiki [3]

Sebelum membahas mengenai *semantic* Mediawiki, akan dibahas mengenai awal mula dibentuknya *semantic* Mediawiki. Bermula dari adanya masalah yang terdapat pada sistem wiki. Dengan adanya sistem wiki, pengguna dapat membuat halaman wiki tersendiri dan dapat mengisi halaman tersebut secara kolaboratif bersama dengan pengguna lainnya. Apabila berkunjung ke alamat situs wikipedia, maka akan ditemukan banyak sekali artikel yang ditulis secara kolaboratif oleh para pengguna Wikipedia. Mulai dari atikel berita, *event*, ataupun artikel tentang *knowledge* lainnya dan umumnya kualitas dari artikel-artikel yang dimiliki Wikipedia tergolong baik dan rapih (harus ada kutipan, format harus konsisten, dan juga adanya mekanisme validasi referensi). Prinsip yang digunakan oleh Wikipedia dalam menjaga konsistensi artikelnya sangat menarik yaitu dengan mempercayakan kepada pengguna untuk menilai dan menyunting tulisan tersebut. Namun seiring meningkatnya jumlah pengguna Wikipedia, semakin banyak juga artikel yang disimpan oleh Wikipedia yang pada akhirnya menimbulkan permasalahan tersendiri. Beberapa permasalahan yang menjadi isu saat ini adalah konsistensi konten dan efisiensi dalam mengakses *knowledge*.

Dengan semakin banyaknya jumlah artikel yang masuk ke *database* Wikipedia, semakin banyak pula informasi yang disimpan dalam *database* Wikipedia dan tentunya sangat mungkin informasi yang sama tersebar di berbagai halaman wiki yang berbeda. Masalah yang timbul adalah ketika ada informasi yang diganti di dalam sebuah artikel, maka idealnya seluruh informasi yang sama dan tersebar di berbagai halaman wiki tersebut juga harus berubah. Konsep yang ingin diterapkan sama dengan konsep *Database Management System* (DBMS)

yang memiliki fitur "*ON UPDATE CASCADE*" dimana DBMS akan menghapus atau memutakhirkan data yang berada di tabel utama dan relasinya [5]. Apabila konsep ini dapat diterapkan, konsistensi dari setiap informasi yang ada di dalam wikipedia dapat dipertahankan. Kemudian dengan banyaknya jumlah artikel yang dimiliki oleh Wikipedia ternyata menimbukan masalah dalam melakukan akses informasi yang tersimpan di dalam Wikipedia. Dengan semakin banyaknya artikel, pengguna akan semakin sulit dalam mencari dan membandingkan setiap artikel yang dianggap relevan oleh pengguna karena terlalu banyaknya halaman yang harus diakses.

Melihat permasalahan tersebut, beberapa orang peneliti dari institut *Applied Informatics and Formal Description Method* (AIFB) yang berasal dari Jerman mengusulkan sebuah konsep yang bernama "*Semantic* Wikipedia" untuk menyelesaikan permasalah-permasalahan tersebut. Usulan *Semantic* Wikipedia tersebut ditindak lanjuti dengan usulan pembuatan subsistem (*extension*) yang bernama *Semantic* Mediawiki.

*Semantic* Mediawiki merupakan teknologi yang digunakan untuk membangun sebuah *semantic web* pada Mediawiki. Awalnya setiap informasi yang terdapat pada Mediawiki hanya dapat dimengerti oleh manusia. Dengan menggunakan konsep *semantic web* yaitu dengan menggunakan semantic Mediawiki, informasi yang sudah diberi atribut dan properti akan dapat dimengerti oleh mesin. Dampak yang timbul adalah, setiap informasi yang sama akan dapat dikenali oleh mesin sehingga ketika ada perubahan yang dilakukan oleh *user*, mesin dapat mengetahuinya. Konsep *semantic* Mediawiki lebih ditekankan pada proses *query* yang lebih baik. Jika mesin mampu mengenali setiap informasi lebih dalam, dalam melakukan *query* pun dapat lebih baik lagi. Ini yang menjadi keunggulan utama mengapa digunakan Mediawiki sebagai *knowledge management tools* dan *semantic* Mediawiki sebagai *extension* di dalamnya. Untuk menampilkan *semantic* Mediawiki, ada *extension* khusus sebagai pasangan *semantic* Mediawiki yaitu *semantic form*.

#### 2.5.2.2 Semantic Form

*Semantic form* merupakan *extension* dari Mediawiki yang fungsinya adalah untuk membuat dan menampilkan *form* yang terintegrasi dengan *semantic*

Mediawiki. *Semantic form* mengizinkan *user* untuk menambah, mengubah, dan melakukan *query* data menggunakan *form*. *Semantic form* hanya dapat digunakan jika terdapat *semantic* Mediawiki di mana struktur datanya sudah di *markup* oleh *semantic* Mediawiki. Tanpa adanya *semantic* Mediawiki maka *semantic form* tidak akan bekerja sama sekali. Komponen utama dari fungsionalitas *semantic form* adalah *form definition page*, yang terdapat pada *namespace* baru yaitu 'Form:'.

Di dalam *semantic form model*, kategori Mediawiki ekuivalen dengan tabel pada *database* dan halaman Mediawiki pada kategori tersebut ekuivalen dengan *row* di dalam tabel *database*. Kemudian, properti-properti *semantic* Mediawiki pada halaman merupakan kolom pada tabel *database*, dengan kata lain setiap halaman di dalam kategori harus mempunyai set properti yang sama. Semua ini dicapai dengan menggunakan *template* dengan dikaitkan dengan kategori. Untuk setiap tabel (*template*+kategori) memiliki keterkaitan dengan *form*. Penjelasan tersebut merupakan prinsip kerja dari *semantic form* pada *semantic* Mediawiki.

Alur kerja dalam membuat sebuah form adalah sebagai berikut [18].

a) Menentukan properti yang akan merepresentasikan struktur data dari "*class*". Untuk setiap properti sebuah halaman di dalam properti, *namespace* akan tercipta. Jadi properti juga dapat didokumentasikan.
b) Membuat Mediawiki *templates*, satu untuk setiap *class*. Objek sebagai satu set dari properti yang ingin ditetapkan sebagai *page* menggunakan *form*. Sebuah *template* mendefinisikan sebuah *shortcut* untuk mendefinisikan properti dan nilainya dan akan menetapkan *layout visual* dari sebuah *class*.
c) Membuat *form* untuk membuat objek yang memiliki nilai. *Form* ini akan menetapkan apa yang *user* dapat masukkan melalui input *form*. Sebuah *form* untuk satu atau lebih *template*.
d) Membuat kategori, biasanya satu untuk setiap *form* / objek.
e) Meng-*enable link* menuju *form*.
f) Menambah data melalui *form*.

## 2.6 **Decision Support System**

*Decision support system* merupakan sebuah sistem komputer yang digunakan untuk mengolah data menjadi informasi yang dapat digunakan untuk mengambil keputusan. Dengan menggunakan *decision support system* dapat diambil beberapa manfaat di antaranya adalah pengambilan keputusan yang rasional, sesuai dengan jenis keputusan yang diperlukan, membuat peramalan, membandingkan alternatif tindakan, membuat analisis dampak, membuat model.

*Decision support system* merupakan bahasan yang cukup luas dan kompleks yang di dalamnya dibahas algoritma rumit untuk menciptakan suatu keputusan yang valid. Namun dalam penulisan ini hanya akan dibahas mengenai *decision support system* pada sebuah artikel kemudian artikel yang memiliki paling banyak reputasi *review* akan diurutkan berdasarkan *rank* yang paling tinggi. Dari *query* tersebut, dapat diambil suatu keputusan yaitu penulis yang memiliki artikel terbaik akan diberi *reward* sesuai dengan aturan yang ada. *Review* yang ada dibuat menggunakan *extension* Mediawiki yaitu *rating bar*. Melalui *extension* ini, dapat dibuat bermacam *query* untuk menampilkan artikel berdasarkan *rating*.

# BAB 3
# PERANCANGAN KNOWLEDGE MANAGEMENT SYSTEM DENGAN SEMANTIC MEDIAWIKI

Bab ini menerangkan perancangan sistem secara bertahap mengikuti metodologi *System Development Life Cycle* (SDLC) yang sudah dijelaskan pada bab sebelumnya. Tahapan perancangan ini meliputi *user requirement*, desain dan analisis sistem, dan implementasi. Dari tahapan yang sudah dibahas sebelumnya, testing tidak termasuk dalam bab ini karena pada tahap ini hanya sebatas perancangan sistem saja. *Testing* akan dibahas pada bab selanjutnya yang menerangkan tentang pengujian dan analisis sistem. Dalam mendokumentasikan setiap tahapan SDLC digunakan sebuah diagram standar yaitu dengan metode *Unified Modelling Language* (UML) yang sudah dibahas pada bab sebelumnya. Dengan UML, rancangan perangkat lunak dapat direpresentasikan ke dalam diagram-diagram yang memiliki fungsi masing-masing. Berikut adalah diagram-diagram yang merepresentasikan rancangan dari sistem yang akan dibuat, meliputi *use case diagram, object diagram, class diagram,* dan *deployment diagram*.

## 3.1 **User Requirement**

### 3.1.1 Deskripsi

Dalam membangun sebuah sistem, tahapan perumusan kebutuhan dari *user* atau sering disebut dengan *user requirement* merupakan tahapan yang sangat penting karena menyangkut tentang fungsionalitas sistem yang akan dibangun. Dalam hal ini ada dua *requirement* yang harus dipenuhi. Pertama adalah *requirement* mengenai sistem umum yang diinginkan oleh *user* dan yang kedua adalah *system requirement* yang bisa didapat dengan menggunakan *knowledge audit* yang sudah dijelaskan sebelumnya. Jadi, pada tahapan *user requirement* ini ada dua hal yang akan menjadi *output* yaitu *functional requirement* dan *knowledge mapping*.

### 3.1.2 Functional Requirement

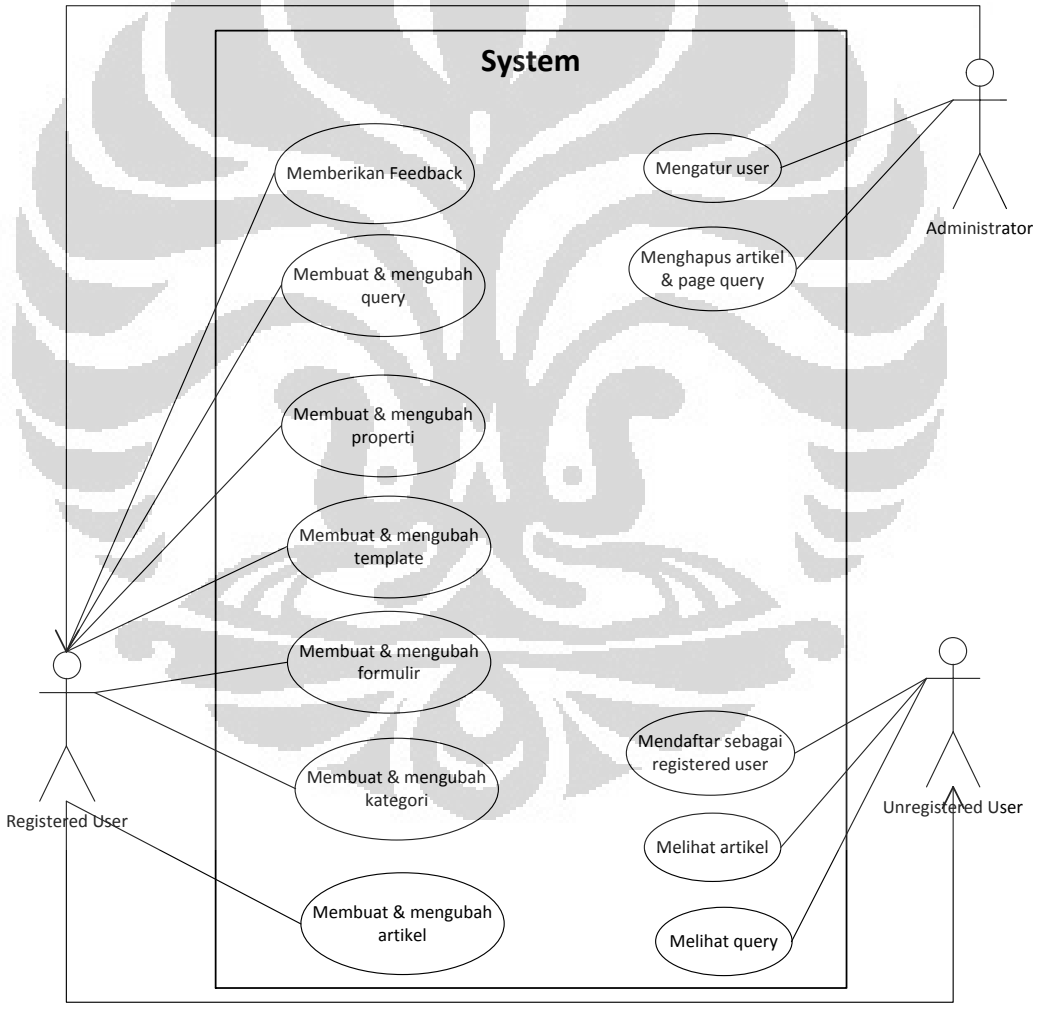
*Functional requirement* merupakan suatu kebutuhan terhadap fungsionalitas sistem yang akan dibuat. Kebutuhan fungsional ini diminta langsung oleh *user* sebagai pihak yang akan menggunakan sistem secara langsung. Penentuan *functional requirement* ini dilakukan dengan cara wawancara dan tukar wawasan. Dari hasil wawancara kepada *user*, didapatkan hasil *functional requirement* sebagai berikut.

a. Setiap orang dapat mendaftarkan diri sebagai *user* di dalam sistem.
b. Hanya *user* yang terdaftar yang mampu membuat dan mengubah tulisan.
c. Setiap tulisan dapat di*review* / dinilai oleh semua *user* yang terdaftar.
d. Setiap tulisan tercatat siapa penulisnya, waktu penulisannya, dan kategori tulisan.
e. Setiap perubahan tulisan tercatat rincian *history*.
f. Jenis pengkategorian artikel berdasarkan standar FCAPS yaitu *Fault, Configuration, Accounting, Performance, Security*.
g. Terdapat subkategori problem dan *solution* yang subkategori itu terelasi secara langsung.
h. Terdapat *query* tentang artikel berdasarkan kategori.
i. Terdapat *query* tentang artikel yang memiliki *rank review* tertinggi.
j. Terdapat *query* tentang *user* yang paling banyak membuat artikel.
k. Terdapat *query* tentang artikel-artikel terbaru.
l. Terdapat *query* tentang *user* yang paing sering memberikan *review*.
m. Setiap artikel yang ditulis dikirimkan melalui *email* kepada semua *user*.

*Functional requierement* di atas merupakan fungsi-fungsi yang harus ada pada sebuah sistem untuk memenuhi kebutuhan dari *user* yang akan menggunakannya, dalam hal ini operator pada perusahaan telekomunikasi. Requirement yang diberikan akan diintegrasikan kembali dengan *knowledge management tools* yang akan digunakan sehingga dari sisi pengembang harus mengintegrasikan *functional requirement* yang ada dengan sistem yang akan dibangun.

Dari sisi pembuatan artikel sampai dengan pengkategorian, ada beberapa fungsi yang harus didefinisikan, di antaranya adalah pembuatan properti, *template*, formulir, dan kategori, kemudian setelah tercipta keempat aspek tersebut baru halaman dapat tercipta.

Dari sisi pengembang, *requirement* di atas merupakan *requirement* dasar yang masih dapat dikembangkan lagi menjadi sebuah *system requirement*. Fungsi-fungsi yang harus ada dikategorikan lagi sehingga terbentuk fungsi sistem dasar yang bisa merepresentasikan sistem secara keseluruhan. Untuk menggambarkan fungionalitas sistem dapat digunakan dengan *use case diagram*. Berikut hasil pengkategorian fungsi yang dirangkum dalam *use case diagram*.




Gambar 3.1. *Use case diagram*

Dari diagram pada Gambar 3.1, terdapat tiga aktor yang memiliki fungsionalitas masing-masing. Aktor pertama adalah *unregistered user*, yaitu *user* yang belum teregistrasi. Semua pengguna yang belum teregistrasi mampu menjalankan tiga fungsi seperti yang digambarkan, yaitu mendaftar sebagai *user* yang terdaftar, melihat artikel yang ada, dan melihat berbagai *query* yang ditampilkan. Aktor kedua adalah *registered user* yaitu *user* yang sudah terdaftar dalam sistem. Aktor ini dapat melakukan hal-hal yang berkaitan dangan *input* data dan *edit data*. Ada beberapa fungsi yang dapat dijalankan oleh aktor ini, di antaranya adalah membuat dan mengubah properti, membuat dan mengubah *template*, membuat dan mengubah formulir, membuat dan mengubah kategori, membuat dan mengubah halaman artikel. Selain melakukan empat fungsi utama, aktor ini juga dapat membuat dan mengubah halaman *query*. Halaman *query* digunakan untuk memanggil informasi semantik yang terdapat dalam database sehingga memunculkan informasi yang diinginkan. Aktor ini juga dapat membuat *review* terhadap artikel yang sudah dibuat sebelumnya. *Review* ini sebatas memberikan penilaian terhadap artikel yang ada. Aktor terakhir merupakan *administrator*. Aktor ini memiliki *priviledge* yang paling tinggi. Jika *registered user* dapat melakukan fungsi-fungsi yang dilakukan oleh *unregistered user*, *administrator* dapat melakukan semua fungsi yang ada. Fungsi khusus yang hanya bisa dilakukan oleh *administrator* adalah mengatur *user*, menghapus halaman artikel dan halaman *query*.

### 3.1.3 *Knowledge Audit*

Seperti yang telah dijelaskan pada Bab 2 mengenai *knowledge audit*, bahwa *knowledge audit* merupakan tahapan pada *requirement* yang pada tahap ini akan dianalisis mengenai *knowledge needs*, *knowledge inventory* yang sudah ada, serta bagaimana pemetaan *knowledge* pada perusahaan tersebut. *Knowledge audit* dapat dilakukan berdasarkan beberapa cara, salah satunya adalah dengan wawancara terhadap orang-orang yang akan menggunakan sistem tersebut. Dari hasil wawancara yang telah dilakukan, didapatkan hasil sebagai berikut.

### *3.1.3.1 Knowledge Inventory*

Berdasarkan hasil wawancara yang telah dilakukan, didapatkan hasil identifikasi *inventory* yang terdapat pada perusahaan tersebut sebagai berikut. Hasil wawancara *user* tersebut juga merupakan tahap untuk mengetahui *Knowledge needs*, yaitu *knowledge* apa saja yang biasa digunakan dan dibutuhkan ke depannya untuk digunakan dalam *knowledge management system* yang akan dibuat. Dari hasil analisis *knowledge needs* tersebut didapat sebuah tabel *knowledge inventory*.

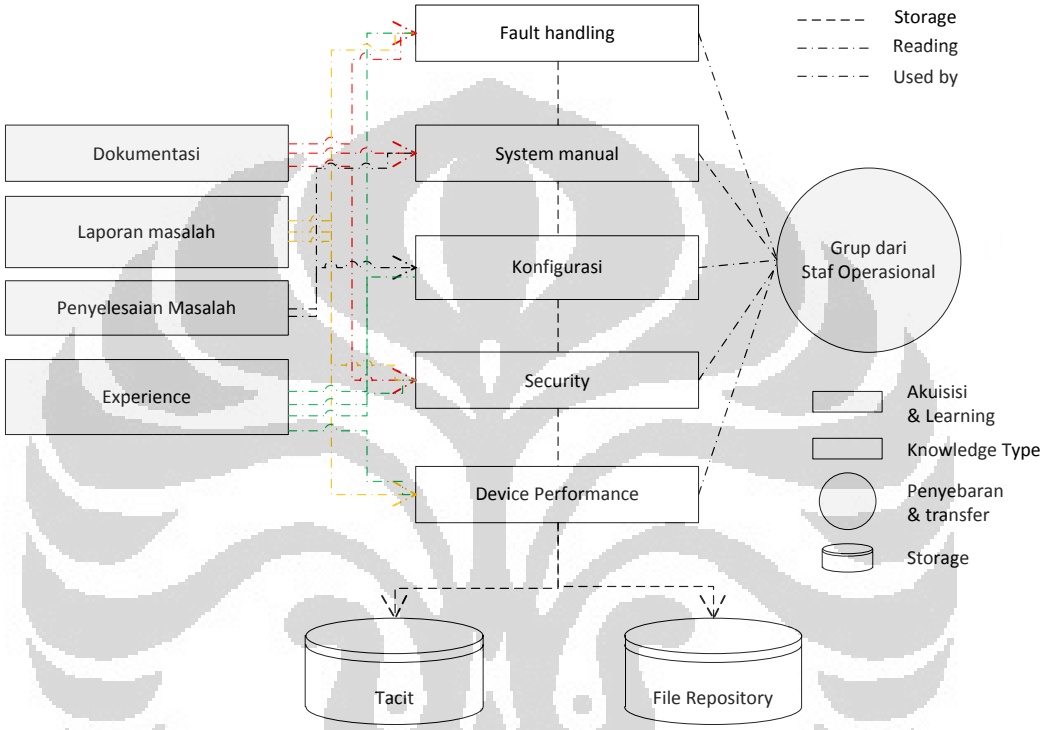
Tabel 3.1. Tabel *knowledge inventory*

| No | Aset Pengetahuan | Format | Pemilik | Lokasi | Jenis |
|---|---|---|---|---|---|
| 1 | *Fault Management* | Dokumen | Divisi Operasional | Server Dokumen | *Know – when, know – where, know - why* |
| 2 | Konfigurasi | *Configuration Code* | Divisi Operasional | Personal | *Know - how* |
| 3 | *Performance Management* | Dokumen | Divisi Operasional | Server Dokumen | *Know – when, know – where, know - why* |
| 4 | *Security Management* | Dokumen | Divisi Operasional | Server Dokumen | *Know – when, know – where, know - why* |

Dari Tabel 3.1, diperlihatkan bahwa sebenarnya ada empat aset pengetahuan yang ada pada divisi Operasional pada perusahaan yang bersangkutan. Divisi Operasional sendiri merupakan divisi IP *Network*, sehingga pekerjaannya berada pada *operation area*. Pada area operasional, *fault management*, konfigurasi, *performance management*, dan *security management* merupakan aset pengetahuan yang selalu digunakan.

### *3.1.3.2 Knowledge Mapping*

*Knowledge mapping* merupakan tahap akhir dari proses *knowledge audit*. Hasil wawancara yang sudah diolah, akan menghasilkan beberapa informasi *knowledge* yang bisa diklasifikasikan sehingga informasi *knowledge* tersebut bisa dipetakan berdasarkan klasifikasinya. Peta *knowledge* dibentuk berdasarkan tabel inventaris *knowledge* yang telah dibuat sebelumnya.



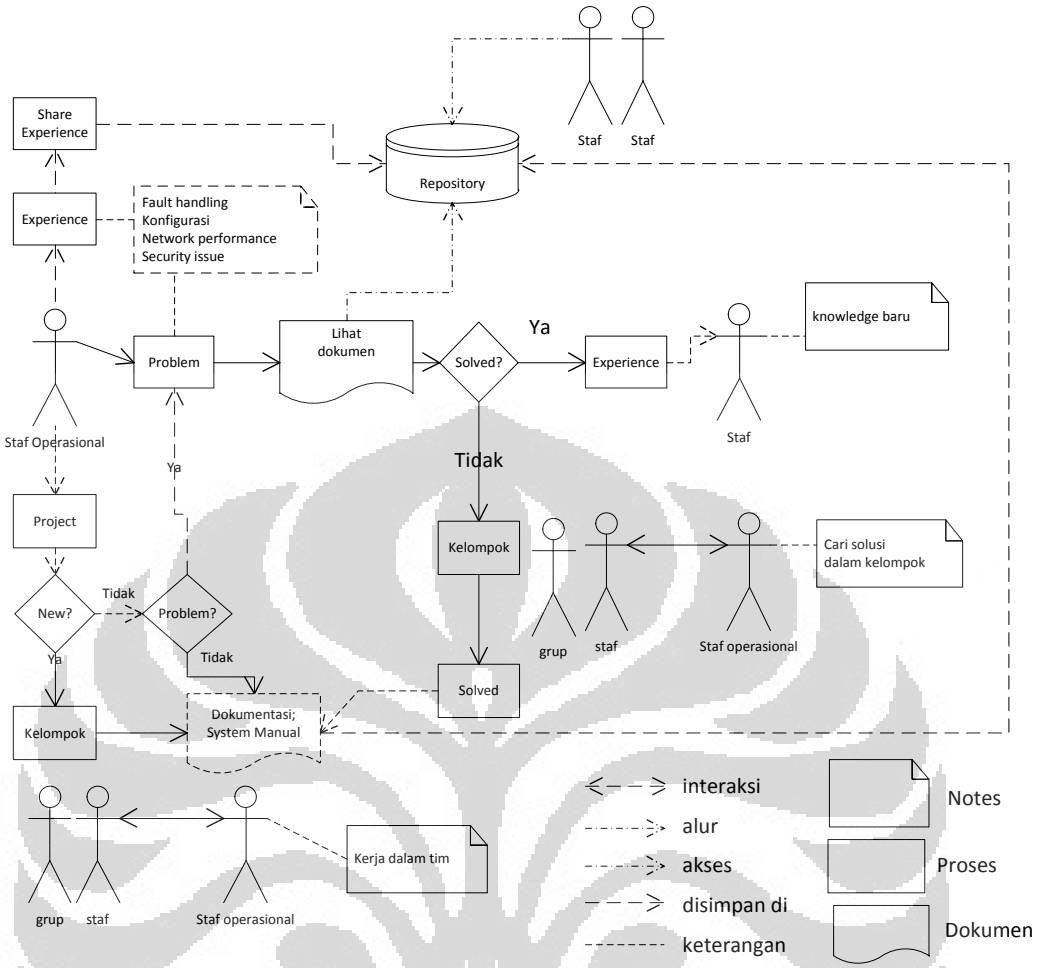

Gambar 3.2. Peta *knowledge*

Dari Gambar 3.2, dapat dilihat bahwa penyebaran *knowledge* tersebar ke seluruh staf operasional kemudian *knowledge type* memperlihatkan tipe *knowledge* yang disebarkan. Tipe-tipe *knowledge* ini didapatkan dari tabel *knowledge inventory*. Pada sisi akuisisi dan *learning knowleddge* terdapat empat poin yaitu dokumentasi, laporan masalah, penyelesaian masalah, *case study* dan *experience*. Empat bentuk di atas merupakan bentuk *capturing knowledge* yang bisa dilakukan pada divisi operasional yang terdapat proses internalisasi dan eksternalisasi. Proses internalisasi ditunjukkan melalui *eksplisit knowledge* dikonversi menjadi *tacit knowledge* yaitu terdapat pada penyelesaian masalah ketika staf operasional bisa menggunakan system manual untuk menyelesaikan

masalah tersebut. Kemudian proses eksternalisasi terjadi pada konversi dari *tacit knowledge* menuju *explicit knowledge* yang terlihat pada *experience* yang dituangkan pada konfigurasi dan sistem manual dalam bentuk dokumentasi. Dari semua *knowledge* yang ada disimpan dalam dua bentuk yaitu *tacit repository* dan *file repository*. *Tacit repository* merupakan *repository* yang tidak nyata yang terdapat pada pengalaman masing-masing staf setelah proses *transfer*. *File repository* merupakan *storage* yang digunakan untuk menyimpan hasil dokumentasi.

### 3.1.4 Diagram Alir *Knowledge*

Pada *Knowledge Management System* terdapat *knowledge flow* atau aliran *knowledge*. Tujuan dari dibentuknya diagram alir *knowledge* ini adalah untuk mengetahui dari mana saja sumber pengetahuan tersebut berasal, di manakah *knowledge* tersebut disimpan, serta siapa sajakah yang mengetahui *knowledge* tersebut. Berikut merupakan gambar dari diagram alir *knowledge* yang terjadi pada divisi operasional suatu perusahaan.




Gambar 3.3. Diagram alir *knowledge*

Gambar 3.3 di atas menggambarkan tentang alur *knowledge* yang terjadi pada setiap staf di mana pada divisi operasional terdapat beberapa *capture knowledge*. Setiap staf yang menjalankan tugasnya akan dihadapkan pada suatu pilihan masalah yang mana staf tersebut dapat langsung menyelesaikannya atau tidak. Ketika staf tersebut membutuhkan *resource*, maka staf akan mengakses dokumentasi pada *repository*. Jika terdapat permasalahan yang tidak ditemukan, permasalahan tersebut diselesaikan dalam kelompok. Setelah permasalahan terselesaikan, staf tersebut telah mendapatkan *knowledge*-nya. Untuk dapat di *transfer* ke staf yang lain, harus ada *knowledge sharing* dengan melakukan proses dokumentasi. Dokumentasi tersebut kemudian diletakkan pada *repository*.

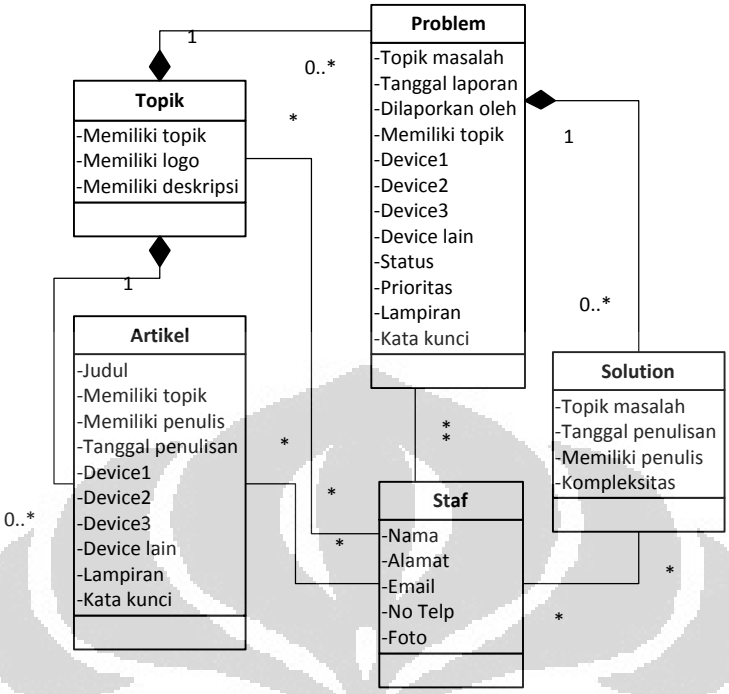
Walaupun demikian, setiap staf dapat menuangkan pengalamannya ke dalam *repository* tanpa harus melalui tahap menemui masalah. Pengalaman yang pernah dialaminya dapat dituliskan atau didokumentasikan dalam bentuk *case study*. Pengalaman tersebut kemudian diletakkan dalam *repository* dan akhirnya dapat diakses oleh orang yang membutuhkan.

Sedangkan untuk penanganan *project*, hal ini sedikit berbeda karena penanganan *project* berarti apakah *project* yang ditangani merupakan *project* baru ataukah *project* lama. Jika itu merupakan *project* lama, harusnya tidak ada masalah dalam pengerjaannya. Namun, jika dalam pengerjaannnya terdapat masalah, staf tersebut harus melalui tahap penyelesaian masalah. Jika *project* tersebut merupakan *project* baru, penangan *project* dilakukan bersama dengan tim dan pengembangannya harus didokumentasikan dalam bentuk *system manual* agar *project* tersebut terekam dengan baik.

## 3.2 **Desain dan Analisis Sistem**

### 3.2.1 Pemodelan *Knowledge Management System*

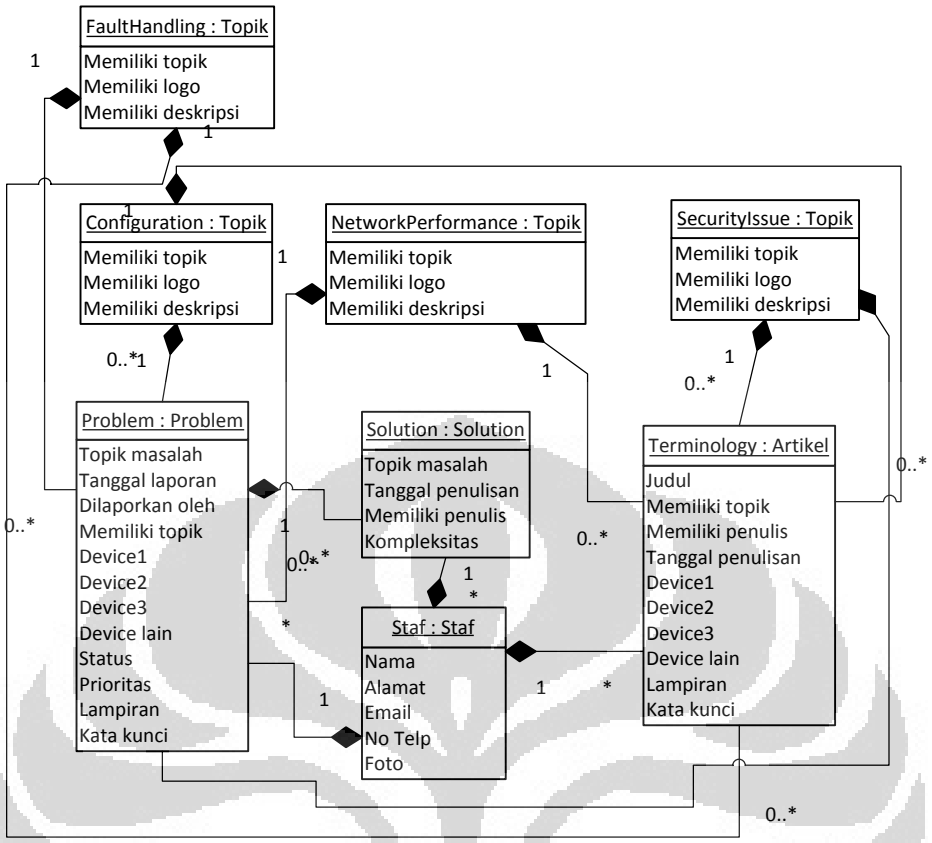
Desain dan analisis sistem merupakan tahap paling penting karena pada tahap ini dilakukan pembentukan *class* dan *objek* sebagai tiang penyangga sistem. Jadi, dari *class* dan *object diagram* ini dapat dilihat sistem secara utuh. *Class diagram* menggambarkan entitas yang terdapat pada sistem yang akan dibangun. Entitas tersebut nantinya akan dimasukkan dalam *object diagram*. Pada sistem yang dibuat terdapat enam kelas yang merepresentasikan fungsinya masing-masing. Class diagram dapat dilihat pada Gambar 3.4.




Gambar 3.4. *Class diagram*

Kelas topik digunakan untuk mengklasifikasikan empat topik dasar yaitu *fault handling*, konfigurasi, *network performance*, dan *security issue*. Dari empat topik tersebut, masing-masing memiliki tiga kelas yang terkait yaitu problem, artikel, dan staf. Isi dari kelas problem adalah problem-problem yang berkaitan dengan salah satu kategori yang sudah disebutkan sebelumnya. Setiap kategori boleh memiliki banyak problem, tapi setiap problem hanya diperbolehkan memiliki satu kategori. Setiap problem boleh memiliki beberapa solusi, tapi setiap solusi hanya dimiliki oleh satu problem. Artikel hampir sama dengan problem. Artikel berdiri di bawah kategori sehingga posisi artikel dan problem sejajar, sedang solusi berada di bawah problem.

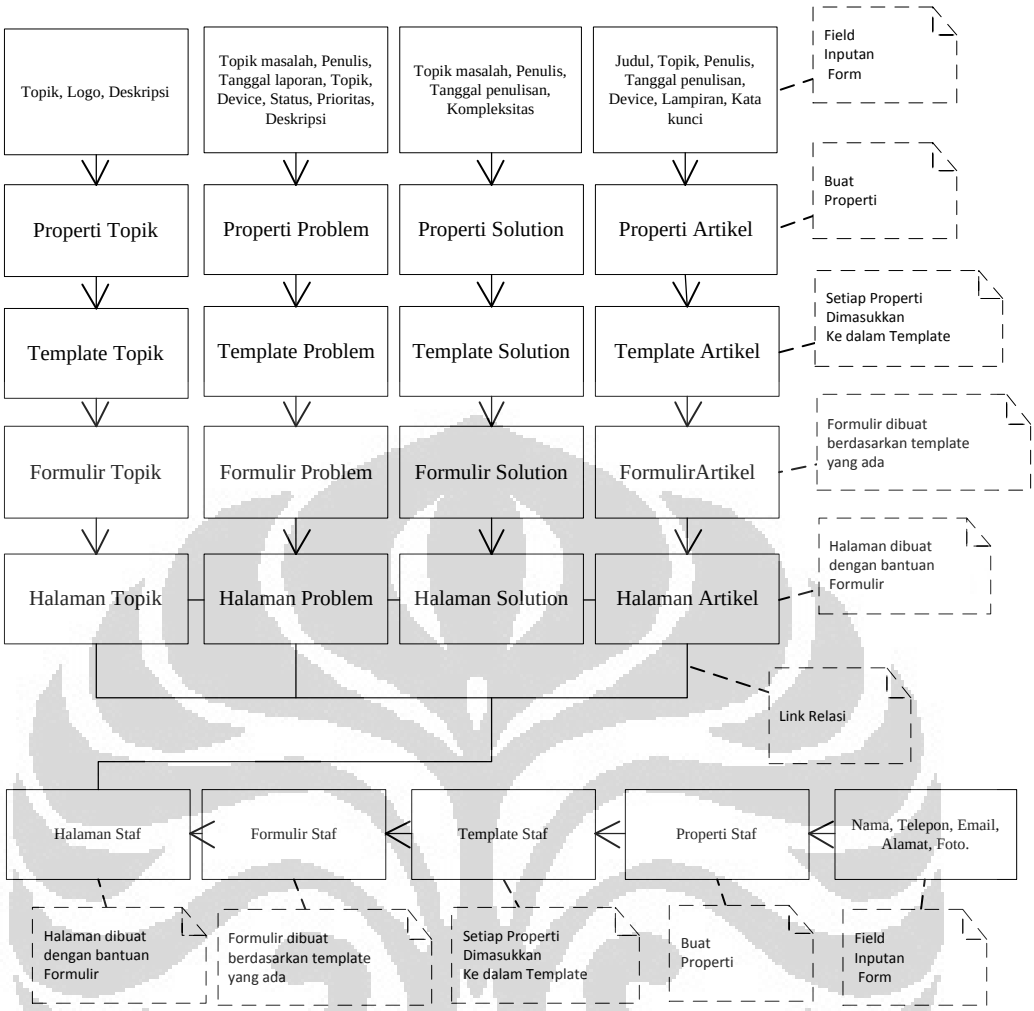
Setiap topik memiliki entitas sendiri sehingga bisa disebut masing-masing topik sebagai *object*. Selain topik, problem, *solution*, dan artikel juga merupakan *object* yang terelasi dengan masing-masing topik. *Object diagram* dapat dilihat pada Gambar 3.5.




Gambar 3.5. *Object diagram*

### 3.2.2 Integrasi *Class Diagram* dengan *Semantic Form* pada Mediawiki

Konsep *knowledge management system* yang dibuat diterapkan menggunakan *framework* Mediawiki. Dalam Mediawiki terdapat *extionsion semantic* Mediawiki dan *semantic form*, penjelasan tentang keduanya sudah ada pada bab sebelumnya. *Semantic form* merupakan *extionsion* penting dalam membangun sistem ini. Pada penerapannya, *class diagram* yang sudah dibangun harus diimplementasikan pada Mediawiki sebagai *framework* untuk membangun *knowledge management system* ini.

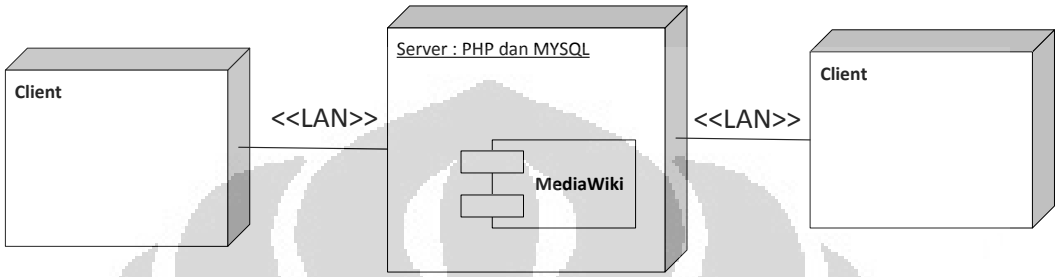



Gambar 3.6, Integrasi Konsep KMS pada Mediawiki

Dari Gambar 3.6 dapat dilihat bahwa *class diagram* yang sudah diklasifikasikan pada subbab sebelumnya didefinisikan sebagai *object* yang akan dibuat halaman berdasarkan tahapan *semantic form*. Pertama, entitas yang terdapat pada setiap kelas dibuat menjadi properti-properti. Pada "buat property" terjadi proses membuat entitas yang terdapat pada masing-masing kelas menjadi properti pada *semantic form*. Properti-properti tersebut digunakan untuk membuat *template*. *Template* yang sudah dibuat kemudian digunakan untuk membuat *form*. Jadi, *form* tersebut memiliki *template* masing-masing berdasarkan karakteristik kelasnya. Setelah formulir dibuat, halaman sudah bisa dibuat. Setiap *object* dibuat berdasarkan formulir yang sudah dibuat tersebut. Terakhir, untuk membuat relasi antar *object*, setiap entitas pada objek harus saling direlasikan.

### 3.2.3 *Deployment Diagram*

*Deployment diagram* digunakan untuk merepresentasikan rancangan tata letak fisik dari sistem yang menunjukkan perangkat keras dan perangkat lunak dalam sistem tersebut. Pada implementasi *knowledge management system*, akan dibangun seperti pada *deployment diagram* pada Gambar 3.7.




Gambar 3.7. *Deployment diagram*

# BAB 4
# IMPLEMENTASI DAN INTEGRASI *KNOWLEDGE MANAGEMENT SYSTEM* BERBASIS SEMANTIC MEDIAWIKI

Pada bab sebelumnya sudah dipaparkan mengenai perancangan *Knowledge Management System* dengan tahapan yang sudah dilewati yaitu *requirement definition* serta *system and software design*. Ada lima tahapan yang harus dilewati, tiga sisanya yaitu *implementation and unit testing*, *integration and system testing*, *operation and maintenance*. Dari tahapan tersebut, bab ini akan membahas seputar implementasi dan integrasi sistem, sedangkan *operation and maintenance* tidak akan dibahas karena merupakan tahapan perawatan sistem.

## 4.1 Implementasi Mediawiki

Pada Bab 2 sudah dijelaskan mengenai *framework* Mediawiki yang digunakan dalam mengembangkan *Knowledge Management System* ini. Pada implementasi Mediawiki, diterapkan beberapa *extension* yang berupa modul-modul yang diterapkan untuk memenuhi *requirement definition* yang sudah dibuat sebelumnya. Untuk itu, ada beberapa modul utama yang harus dipasang dalam implementasi *Knowledge Management System* ini, di antaranya adalah modul *semantic Mediawiki*, *semantic form*, *article feedback*, *rating bar*, dan beberapa modul pendukung lain yang sifatnya tidak utama tetapi cukup berperan dalam sistem ini.

### 4.1.1 Modul *Semantic* Mediawiki

Modul ini merupakan ekstensi Mediawiki yang harus dipasang untuk dapat menggunakan fitur *semantic* yang ada. Dengan ekstensi ini, basis data harus di *upgrade* sehingga mendukung fitur *semantic* yang ada. Dalam pemasangan dibutuhkan file ekstensi *semantic* Mediawiki, pada sistem diterapkan *semantic* Mediawiki versi 1.5 RC1. Untuk dapat digunakan, *semantic* Mediawiki harus didefinisikan pada localsetting.php di Mediawiki yaitu dengan menambahkan sintaks seperti pada Gambar 4.1.

```
include_once("$IP/extensions/SemanticMediawiki/SemanticMediawik
i.php" );

enableSemantics('localhost');
```

Gambar 4.1. *Syntax* untuk mengaktifkan *semantic* Mediawiki [17]

Setelah mendefinisikan *semantic* Mediawiki pada Mediawiki yang ada, basis data harus diperbarui untuk dapat mendukung semantic Mediawiki tersebut. Untuk mengelola *semantic* Mediawiki dapat diakses melalui $IP/Istimewa:AdminSMW.

### 4.1.2 Modul *Semantic Form*

Modul ini merupakan ekstensi yang harus ada untuk mendukung *semantic* Mediawiki. Dengan adanya *semantic form*, pengguna dapat membuat properti sebagai atribut semantik kemudian menggunakan properti tersebut untuk membuat *template* dari halaman semantik yang dibuat. Selanjutnya *template* tersebut dijadikan rujukan untuk membuat form input yang mengandung properti semantik di dalamnya. Setelah itu, setiap *form* dapat dikategorikan dalam satu kategori.

Dalam implementasinya, pemasangan *semantic form* membutuhkan file *semantic form* itu sendiri. Dalam sistem ini digunakan *semantic form* 2.3 yang sudah mendukung *semantic* Mediawiki 1.6. Untuk pemasangannya, *semantic form* harus didefinisikan dahulu ke dalam file localsetting.php pada Mediawiki yaitu dengan menambahkan sintaks seperti pada Gambar 4.2.

```
include_once("$IP/extensions/SemanticForms/SemanticForms.php");
```

Gambar 4.2. *Syntax* untuk menginstal *semantic* form [17]

Setelah didefinisikan, akan muncul fungsi-fungsi untuk membuat properti, *template*, formulir, dan kategori pada halaman istimewa.

### 4.1.3 Modul *Article Feedback*

Modul ini merupakan ekstensi yang digunakan untuk menilai setiap artikel berdasarkan tingkat dapat dipercaya, kejelasan, kelengkapan, dan baik/buruk penulisan. Dengan *article feedback* ini, semua artikel dapat diberikan nilai secara spesifik untuk setiap komponen tersebut.

Dalam implementasinya, ekstensi ini membutuhkan ekstensi pendukung di antaranya adalah *click tracking, email capture*, *pref switch*, dan *simple survey*. Dari keempat ekstensi tersebut hanya *pref switch* dan *simple survey* yang mampu dijalankan dalam Mediawiki ini. Untuk pemasangan *article feedback* dan ekstensi pendukungnya, semua ekstensi tersebut harus didefinisikan dalam localsetting.php dan menambahkan sintax seperti pada Gambar 4.3.

```
#Article Feedback

require_once( "$IP/extensions/ClickTracking/ClickTracking.php"
);

require_once( "$IP/extensions/EmailCapture/EmailCapture.php" );

require_once( "$IP/extensions/PrefSwitch/PrefSwitch.php" );

require_once( "$IP/extensions/SimpleSurvey/SimpleSurvey.php" );

require_once(
"$IP/extensions/ArticleFeedback/ArticleFeedback.php" );
```

Gambar 4.3. *Syntax* untuk menginstal *article feedback* [17]

Setelah didefinisikan, setiap halaman akan diberikan modul *article feedback*. Untuk konfigurasi agar hanya halaman tertentu saja yang menggunakan modul *article feedback* ini, file articlefeedback.php harus dikonfigurasi sebagai berikut seperti pada Gambar 4.4.

```
$wgArticleFeedbackCategories=array('Artikel','Problem','Solutio
n');

$wgArticleFeedbackBlacklistCategories = array('Main');

$wgArticleFeedbackLotteryOdds = 0;
```

Gambar 4.4. Konfigurasi modul *article feedback* [17]

Komponen $wgArticleFeedbackLotteryOdds merupakan komponen untuk mengatur apakah modul tersebut dapat digunakan pada halaman yang tidak didefinisikan kategorinya pada konfigurasi *article feedback* di atas. Jika $wgArticleFeedbackLotteryOdds bernilai 0, tidak ada halaman selain yang didefinisikan yang dapat menggunakan modul ini. Jika $wgArticleFeedbackLotteryOdds bernilai 100, semua halaman di luar kategori yang didefinisikan akan dapat menggunakan modul tersebut. Nilai tersebut bersifat persentase.

Setelah semua ekstensi didefinisikan, database juga harus diperbarui dengan menambah tabel yang sudah disediakan masing-masing ekstensi. Dalam penambahan tabel tersebut harus diperhatikan prefiksnya sehingga tidak menimbulkan *error* ketika pemasangan.

### 4.1.4 Modul *Rating Bar*

Modul ini merupakan modul yang digunakan untuk memberikan nilai secara keseluruhan terhadap halaman yang dinilai sehingga setiap halaman bisa dibuat peringkat berdasarkan hasil *vote* tersebut. Selain itu, halaman yang baru di *vote* juga bisa dipantau serta *user* yang melakukan *vote* dapat ditampilkan.

Dalam implementasinya, ekstensi ini menggunakan w4g *rating bar* 2.1.2. Untuk dapat menggunakannya, ekstensi ini harus didefinisikan terlebih dahulu ke dalam *file* localsetting.php pada Mediawiki yaitu dengan menambahkan *sintax* seperti pada Gambar 4.5.

```
# RatingBar
require_once("$IP/extensions/RatingBar/W4g_rb.php");
```

Gambar 4.5. *Syntax* untuk menginstal *rating bar* [17]

Setelah didefinisikan, halaman yang ingin diberikan *rating bar* didefinisikan pada *template* yang ada. Pada sistem ini, rating bar ditempatkan pada kategori artikel sehingga ekstensi tersebut diletakkan di index.php/template:artikel yaitu dengan menambahkan sintax seperti pada Gambar 4.6.

```
...
!Rating
|-
!colspan="3"|{{#w4grb_rate:{{PAGENAME}}}}
|-
!Rating saat ini
|:
|{{#w4grb_rawrating:{{PAGENAME}}}}
```

Gambar 4.6. *Syntax* untuk menampilkan *rating bar* [17]

Hasil *vote* tersebut bisa dimonitor dengan membuat halaman baru yaitu daftar peringkat yang berisi artikel terbaik, *vote* terakhir, artikel terbaik dengan minimal *vote*, serta daftar *top voter*. Untuk konfigurasi halaman tersebut dapat menambahkan *parser* <w4grb_ratinglist> diikuti dengan komponen yang ingin dilihat. Untuk halaman daftar peringkat, konfigurasinya seperti pada Gambar 4.7.

```
==Daftar artikel 20 terbaik==
<w4grb_ratinglist toppages items="20" category:"Artikel notitle
nosort "/>

==Daftar 20 vote terakhir==
<w4grb_ratinglist latestvotes items="20" category:"Artikel"
nosort "/>

==Daftar artikel terbaik dengan minimal 5 vote==
<w4grb_ratinglist toppages items="5" category:"Artikel"
minvotecount="5" notitle/>

==Daftar Top Voters==
<w4grb_ratinglist topvoters items="15"/>
```

Gambar 4.7. *Syntax* untuk menampilkan *rating* [17]

### 4.1.5 Modul Pendukung Lain

Selain empat modul utama di atas, ada beberapa modul pendukung yang dipasang dalam Mediawiki ini, di antaranya adalah Get*User*Name untuk mengambil *username* atau *user* yang sedang aktif, ParserFunction yaitu ekstensi untuk mendukung penulisan fungsi *parser* pada Mediawiki seperti pada *rating bar* menggunakan parser <w4grb_ratinglist>, Rename*User* digunakan untuk merubah

nama *user*, SyntaxHighlight digunakan untuk membuat tulisan seperti di *highlight*, TitleKey digunakan untuk *search* yang lebih *advance*, TreeAndMenu digunakan untuk membuat navigasi *sidebar* berbentuk *tree*, dan yang terakhir *wikieditor* digunakan untuk editor saat proses sunting artikel dengan cara mengaktifkan *toolbar editor* pada menu *preferences*.

## 4.2 Integrasi Desain Sistem pada *Semantic* Mediawiki

Desain sistem yang sudah dibuat dan dijelaskan pada bab sebelumnya diintegrasikan dengan *semantic* Mediawiki. Dalam proses integrasinya, ada beberapa hal yang harus diperhatikan yaitu menyesuaikan *object diagram* dengan penerapan properti dan kategori pada *semanctic form*. Sudah dijelaskan sebelumnya bahwa *semantic form* digunakan untuk membuat properti sebagai atribut *semantic* pada halaman Mediawiki. Atribut tersebut harus sesuai dengan atribut pada *object diagram* desain sistem sehingga didapat diagram integrasi *semantic* Mediawiki pada bab sebelumnya Gambar 3.5, kemudian dari diagram yang sudah dibuat diimplementasikan menjadi bentuk properti, *template*, formulir, dan kategori pada *semantic form* dan juga integrasi properti *semantic form* tersebut untuk digunakan di halaman utama serta untuk halaman *query*.

### 4.2.1 Membuat Komponen Properti, *Template*, Formulir, dan Kategori.

Dalam proses integrasi sistem, atribut *object diagram* harus bisa dikonversi ke dalam properti *semantic form*. Untuk itu pengerjaannya dibagi ke dalam beberapa kategori.

#### 4.2.1.1 Kategori Topik

Pada kategori ini terdapat empat halaman yang dibuat yaitu *fault handling*, *configuration*, *network performance*, dan *security issue*. Untuk membuat halaman tersebut dibutuhkan properti: memiliki topik (string), memiliki logo (string), dan memiliki deskripsi (teks). Properti tersebut digunakan untuk membuat *template* dengan nama topik. Isi dari *template* tersebut nantinya digunakan untuk memanggil properti yang ada yang kemudian digunakan pada setiap halaman. Isi dari properti dimasukkan dengan menggunakan formulir. Formulir tersebut dibuat berdasarkan atribut yang sudah didefinisikan pada *template* yang sudah dibuat.

Kemudian dari formulir tersebut dapat dibuat keempat halaman dengan kategoti topik tersebut.

Dari halaman yang sudah dibuat, halaman tersebut dijadikan sebuah halaman utama untuk masing-masing topik. Di halaman tersebut berisi daftar artikel yang sudah dibuat yang mengandung properti memiliki topik yang sama. Contoh, halaman *fault handling* berisi tentang daftar artikel yang mengandung properti `{{memiliki artikel::fault handling}}`. Begitu pula pada halaman dengan topik yang lain. Untuk membuat daftar tersebut menggunakan *parser* `#ask` dari semantik Mediawiki. Ada tiga daftar yang ditampilkan yaitu daftar artikel yang mengandung topik yang bersangkutan, problem yang mengandung topik yang bersangkutan dengan status dibuka (artinya problem tersebut belum terselesaikan), dan problem yang mengandung topik yang bersangkutan dengan status ditutup (artinya problem tersebut sudah terselesaikan).

Membuat suatu daftar berarti melakukan *query* terhadap properti *semantic* Mediawiki. Untuk membuat daftar artikel yang mengandung topik tertentu menggunakan *syntax* seperti pada gambar 4.8.

```
==Daftar Artikel dengan Topik Fault Handling==
{{#ifeq:
{{#ask:[[Kategori:Artikel]][[Memiliki topik::Fault
Handling]]|format=count}} | 0 | Belum Ada artikel
| {{#ask: [[Kategori:Artikel]][[Memiliki topik::Fault
Handling]]
|format=table
|columns=5
|limit=20
```

Gambar 4.8. *Syntax* untuk menampilkan *query* daftar artikel

Untuk membuat daftar problem dengan topik tertentu dengan status dibuka menggunakan *syntax* seperti pada Gambar 4.9.

```
==Daftar Masalah yang berkaitan dengan Topik Fault Handling==
<div class="info">Daftar Masalah yang dibuka:</div>
{{#ifeq: {{#ask:[[Kategori:Problem]][[Memiliki topik::Fault
Handling]]|format=count}} | 0
| Belum Ada Problem Yang Dilaporkan
| {{#ask: [[Kategori:Problem]][[Status::Dibuka]][[Memiliki
topik::Fault Handling]]
|format=table
|columns=5
|limit=20
```

Gambar 4.9. *Syntax* untuk menampilkan *query* daftar problem yang berkaitan dengan topik *fault handling*

Untuk membuat daftar problem dengan topik tertentu dengan status ditutup, sama dengan *syntax* pada status dibuka hanya saja pada properti status menjadi `[[Status::Ditutup]]`. Untuk membuat halaman dengan topik yang lain, properti "memiliki topik" diganti dengan properti halaman yang bersangkutan.

#### 4.2.1.2 Kategori Artikel

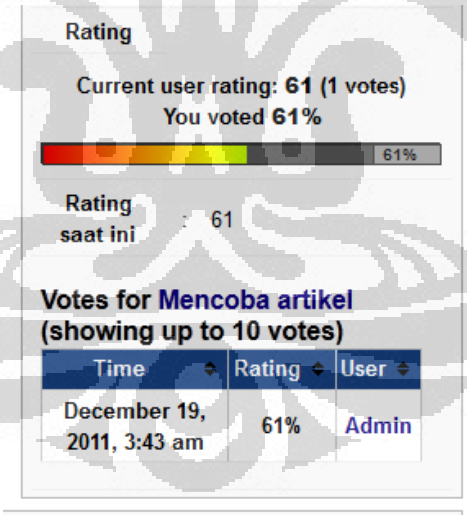
Kategori ini memiliki sembilan properti yaitu properti judul(string), memiliki topik(string), memiliki penulis(halaman), tanggal penulisan (tanggal), device1(string), device2(string), device3(string), device lain(string), lampiran(string), kata kunci(string). Properti tersebut digunakan untuk mendefinisikan *template* yang akan dibuat dengan nama `{{templat:artikel}}`. Templat tersebut digunakan untuk memanggil isi properti yang terdapat pada halaman yang sudah dibuat. Untuk memasukkan properti tersebut menggunakan formulir artikel.

Formulir tersebut dibuat berdasarkan atribut yang terdapat pada *template* artikel. Pada halaman dengan kategori artikel ini diimplementasi dengan modul *rating bar* yang dimasukkan pada *template* artikel. *Rating bar* ini digunakan untuk menilai halaman ini berdasarkan *user* yang menilainya. Untuk menampilkan *rating bar* menggunakan *syntax* seperti pada Gambar 4.10.

```
!Rating
|-
!colspan="3"|{{#w4grb_rate:{{PAGENAME}}}}
|-
!Rating saat ini
|:
|{{#w4grb_rawrating:{{PAGENAME}}}}
```

Gambar 4.10. *Syntax* untuk memanggil *rating list* tiap halaman artikel

Pada Gambar 4.10 terdapat fungsi `{{#w4grb_rate:{{PAGENAME}}}}` yang berfungsi untuk menampilkan *rating bar* pada halaman yang sedang diakses. Sedangkan `|{{#w4grb_rawrating:{{PAGENAME}}}}` digunakan untuk memanggil berapa nilai rata-rata rating pada halaman tersebut. `<w4grb_ratinglist pagevotes idpages={{PAGENAME}} item="5"/>` digunakan untuk menunjukkan kapan terakhir kali rating pada halaman ini diakses dan siapa yang mengaksesnya. Berikut gambar rating bar pada halaman dengan kategori artikel seperti terlihat pada Gambar 4.11.




Gambar 4.11. *Rating bar* dan *rating list*

Pada halaman dengan kategori artikel ini mengandung properti “memiliki topik” dan “device” yang dapat digunakan sebagai relasi untuk menjalankan *query* yang akan dijelaskan pada bahasan selanjutnya.

#### 4.2.1.3 Kategori Problem

Kategori ini mengandung 12 properti yaitu topik masalah(string), tanggal laporan(tanggal), pelapor(halaman), memiliki topik(string), device1(string), device2(string), device3(string), device lain(string), status(string), prioritas(string), lampiran(string), kata kunci(string). Setiap properti pada kategori ini digunakan untuk membuat *template* halaman problem yang artinya di halaman problem yang menggunakan *template* problem akan memiliki properti seperti yang sudah disebutkan di atas. Formulir yang digunakan untuk memasukkan nilai properti dibuat dengan menggunakan *template* problem.

Halaman yang yang mengandung kategori problem langsung terelasi dengan halaman yang mengadung kategori solusi. Untuk merelasikannya, pada *template* problem ditambahkan *syntax* seperti pada Gambar 4.12.

```
== Solusi ==
{|style="background:#eeffbb; width:100%; padding:0; margin:0;"
|{{#forminput:form=Solution|size=30|button text=Tulis
Solusi|autocomplete on category=Solution}}
|}<br />

{{#ask:[[Kategori:Solution]][[Topik masalah::{{{Topik
```

Gambar 4.12. *Syntax* untuk menampilkan *query solution* berdasarkn topik problem

Pada halaman dengan kategori problem ini juga mengandung properti “memiliki topik” dan “device” yang dapat digunakan sebagai relasi untuk menjalankan *query* untuk membuat daftar problem berdasarkan topik dan *device* yang terkait.

#### 4.2.1.4 Kategori *Solution*

Kategori ini mengandung empat properti yaitu topik masalah(string), tanggal penulisan(tanggal), memiliki penulis(string), dan kompleksitas(string). Setiap properti pada kategori ini digunakan pada *template* untuk medifinisikan atribut pada *template*. Untuk memasukkan nilai properti, digunakan formulir yang dibuat dengan *template solution* sehingga nilai yang dimasukkan dengan formulir *solution* tersebut dapat didefinisi sebagai properti pada *template solution*.

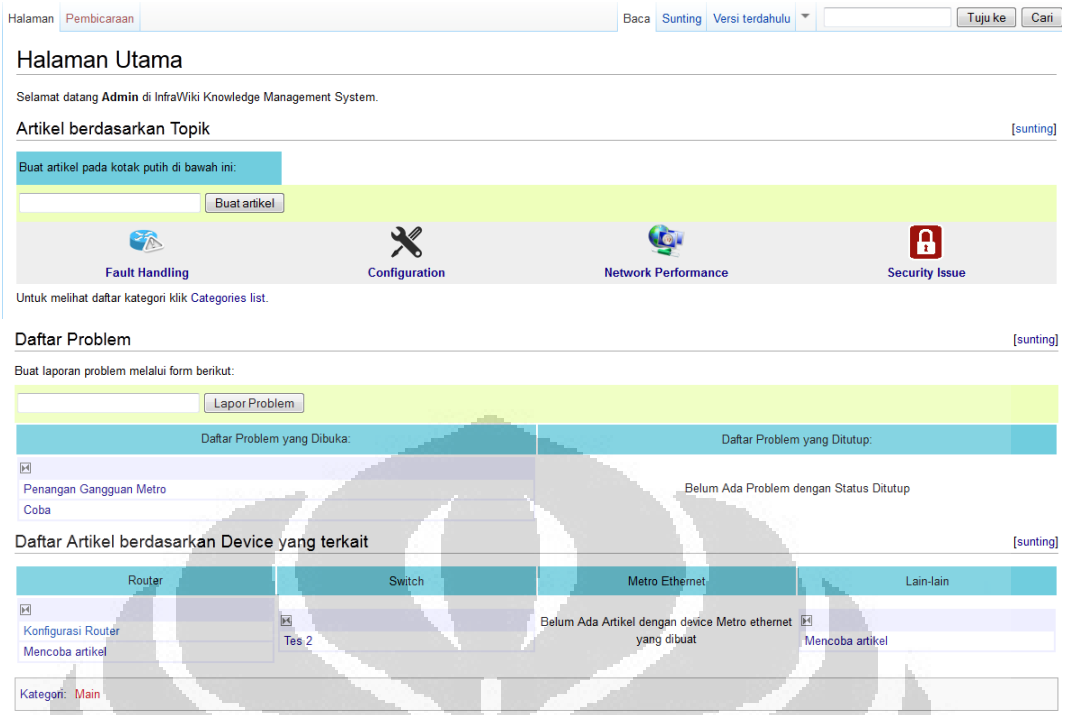
Seperti yang dijelaskan sebelumnya, halaman *solution* merupakan halaman yang langsung terelasi dengan halaman dengan kategori problem. Halaman *solution* akan berada pada halaman dengan kategori problem hanya dengan atribut properti yang sama saja yaitu atribut properti "topik masalah", artinya atribut "topik masalah" menjadi *primary key* untuk menghubungkan antara problem dan *solution*.

### 4.2.2 Membuat Halaman Utama

Setelah mengkonversi semua atribut yang terdapat pada *object diagram* ke dalam bentuk *semantic form*, atribut tersebut sekarang sudah didefinisi sebagai properti. Properti tersebut akan dimiliki oleh setiap halaman sehingga setiap halaman dapat dimengerti oleh mesin sebagai properti yang sudah didefinisi.

Halaman utama dibuat berdasarkan kebutuhan pengguna. Dalam hal ini pengguna membutuhkan akses kepada setiap topik sehingga pengguna dapat melihat artikel apa saja yang terdapat pada topik tersebut. Tidak hanya artikel problem yang berkaitan dengan topik tersebut juga ditampilkan di sana. Untuk itu pada halaman utama haruslah terdapat akses cepat untuk menuju ke halaman topik dengan semenarik mungkin.

Selain akses cepat menuju halaman topik, pengguna juga membutuhkan informasi mengenai problem terbaru yang ada pada sistem, sehingga pengguna yang memiliki kapasitas untuk menjawabnya dapat membuat solusi terhadap problem tersebut. Selain itu problem juga dibedakan berdasarkan status, apakah statusnya dibuka atau ditutup. Selain menampilkan daftar problem terbaru, di halaman utama juga harus dapat menampilkan artikel terbaru, hanya saja untuk kebutuhan dan efektivitas, daftar tersebut disesuaikan berdasarkan perangkat (*device*) yang terkait. Pengelompokan berdasarkan *device* ini perlu karena kebutuhan pengguna berbeda-beda berdasarkan jenis perangkat yang pengguna tangani sehingga dengan mengelompokkannya berdasarkan perangkat, pengguna dapat langsung melihat daftar artikel terbaru berdasarkan perangkat tersebut. Tentu saja daftar ini bersifat terbatas untuk beberapa artikel saja karena pada halaman utama dituntut untuk menampilkan informasi seefisien mungkin. Tampilan halaman dapat dilihat pada Gambar 4.13.




Gambar 4.13. Halaman utama *knowledge management system*

### 4.2.3 Membuat Halaman *Query*

Halaman *query* merupakan sekumpulan perintah *query* yang digunakan untuk mengumpulkan informasi atau *knowledge* yang dibutuhkan untuk keperluan pengguna. Dalam hal ini halaman *query* dibagi menjadi tujuh halaman yaitu halaman daftar artikel, daftar problem, daftar peringkat, daftar artikel dan problem berdasarkan *device router*, daftar artikel dan problem berdasarkan *device switch*, daftar artikel dan problem berdasarkan *device metro ethernet*, dan daftar artikel dan problem berdasarkan *device* yang lain.

Halaman *query* pada prinsipnya memanggil semua properti yang dibutuhkan dan diambil berdasarkan kriteria tertentu. Pada halaman daftar artikel, berarti halaman tersebut harus dapat menampilkan semua halaman dengan kategori artikel dengan menampilkan semua kategori yang dimilikinya. Begitu pula dengan halaman problem harus dapat menampilkan semua problem yang ada dengan mengeluarkan semua properti yang dimilikinya. Berikut contoh halaman daftar artikel seperti pada Gambar 4.14.


Halaman Pembicaraan

Daftar Artikel

Berikut ini adalah daftar artikel yang tersimpan di dalam sistem:
(Disusun berdasarkan Topik)

| Judul | Memiliki topik | Memiliki penulis | Tanggal penulisan | Kata kunci |
|---|---|---|---|---|
| Mencoba artikel | Fault Handling | Admin | 17 Desember 2011 | artikel coba |
| Tes 2 | Fault Handling | Admin | 17 Desember 2011 | Tes 2 |
| Konfigurasi Router | Configuration | Irvan | 20 Desember 2011 | OSPF |

Halaman ini terakhir diubah pada 06.11, 20 Desember 2011.

Halaman ini telah diakses sebanyak 15 kali.

Gambar 4.14. Halaman daftar artikel

Halaman daftar peringkat merupakan halaman yang di dalamnya memanfaatkan modul *rating bar*. Halaman ini merupakan hasil *vote* yang dimonitor dengan menampilkan daftar peringkat yang berisi artikel terbaik, *vote* terakhir, artikel terbaik dengan minimal vote, serta daftar *top voter*. Untuk *syntax* memanggil *parser* yang dibutuhkan sudah dijelaskan di bagian modul *rating bar*. Nilai yang diambil merupakan nilai *vote* yang sudah diberikan oleh setiap pengguna terhadap halaman dengan kategori artikel. Berikut contoh halaman daftar peringkat seperti pada Gambar 4.15.


Daftar artikel 20 terbaik

Berikut adalah daftar peringkat 20 artikel terbaik berdasarkan penilaian vote:

| Page | Rating | Vote count |
|---|---|---|
| Tes 2 | 68% | 1 |
| Mencoba artikel | 61% | 1 |
| Konfigurasi Router | 19% | 1 |

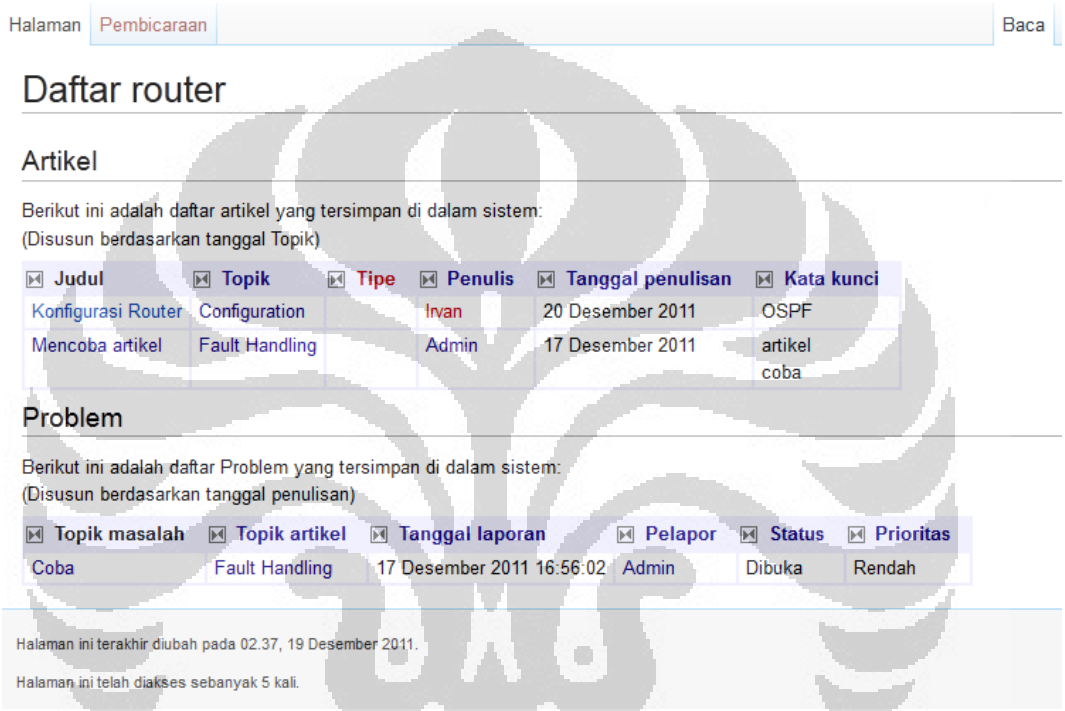
Daftar 20 vote terakhir

Latest votes (showing up to 20 votes)

| Time | Page | Rating | User |
|---|---|---|---|
| December 20, 2011, 11:28 am | Konfigurasi Router | 19% | Irvan |
| December 19, 2011, 3:47 am | Tes 2 | 68% | Admin |
| December 19, 2011, 3:43 am | Mencoba artikel | 61% | Admin |

Gambar 4.15. Halaman daftar peringkat

Halaman *query* yang terakhir adalah halaman daftar artikel berdasarkan perangkat yang bersangkutan, bisa berupa *router*, *switch*, *metro ethernet*, atau bisa perangkat yang lain. Pada dasarnya daftar ini merupakan versi lengkap dari daftar yang terdapat pada halaman utama. Pada halaman daftar ini, bukan hanya artikel yang ditampilkan, tapi juga problem yang berkaitan dengan perangkat yang bersangkutan akan ditampilkan bersama properti yang dimilikinya. Berikut contoh halaman tersebut seperti pada Gambar 4.16.




Gambar 4.16. Halaman daftar artikel dan problem berdasarkan *device*

## 4.3 Implementasi Sistem pada *Server*

Pada proses *system deployment*, sistem diimplementasikan pada salah satu *server* yang ada di divisi operasional pada perusahaan telekomunikasi. Proses implementasi sistem juga tidak lepas dari *software* pendukung dan topologi jaringan yang ada.

### 4.3.1 Spesifikasi *Server*

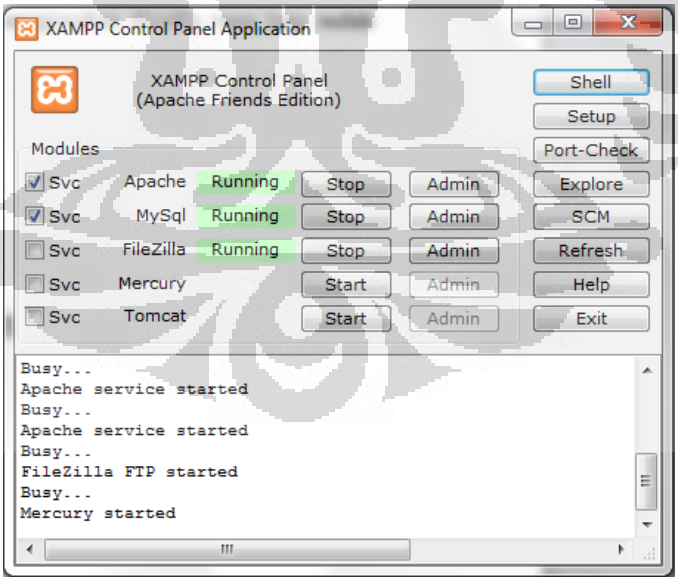
Berikut spesifikasi *server* yang digunakan untuk *system deployment*.

- Jenis : IBM PC

- *Processor* : Intel (R) Core(TM)2 Duo CPU E4400 @2.00GHz 1.00GHz
- *Memory* : 2083 MB
- OS : Windows Vista™ *Business*
- *System type* : 32-bit *Operating System*

### 4.3.2 Software Pendukung

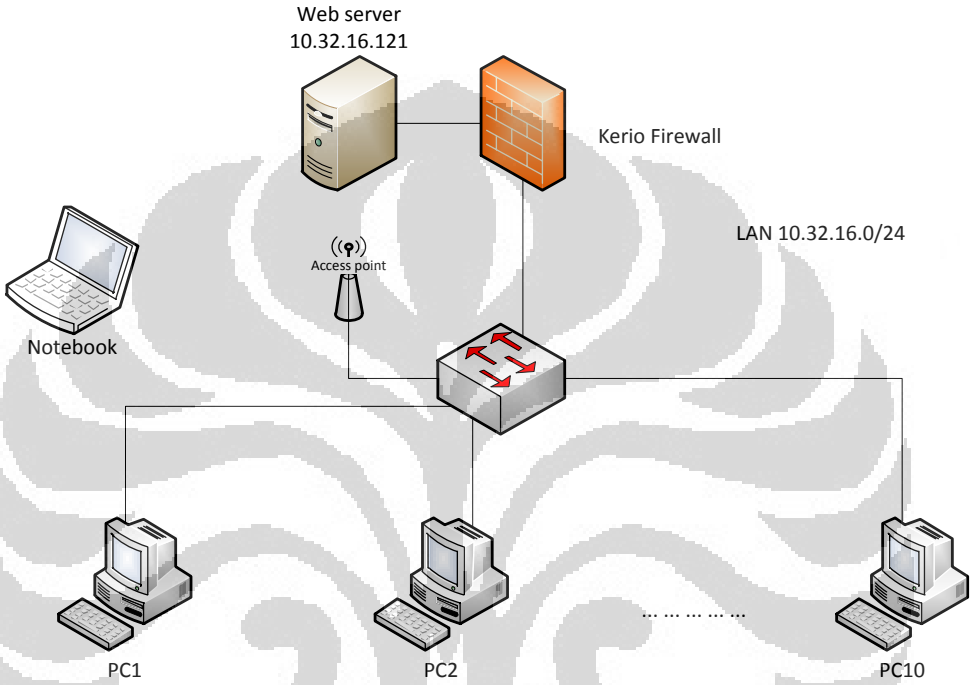
Dalam implementasi sistem yang dibangun dalam *server* tersebut, digunakan beberapa *software* pendukung di antaranya adalah apache2.20 dan mysql yang terbungkus dalam satu *software* bernama xampp. Xampp yang digunakan pada *server* tersebut adalah xampp 1.73 yang sudah dimodifikasi untuk bisa diakses di luar localhost (127.0.0.1). Php yang terdapat di dalam xampp tersebut juga sudah di*upgrade* menjadi php versi 5.3.6 yang sudah *support* untuk Mediawiki. Apache2.20 digunakan sebagai *webserver*, sedangkan mysql digunakan sebagai database sistem. Selain itu, untuk mengakses file yang terdapat pada server menggunakan *service* FTP menggunakan filezilla yang juga sudah termasuk dalam xampp 1.73. Tampilan control panel xampp dapat dilihat pada Gambar 4.17.




Gambar 4.17. *Xampp Control Panel*

### 4.3.3 Topologi Jaringan

Dalam membangun sebuah *Knowledge Management System* pada *server*, pasti dibutuhkan koneksi dari *client* ke *server*. *Client* merupakan perangkat pengguna yang digunakaan untuk mengakses *Knowledge Management System*. Berikut topologi LAN yang terdapat pada perusahaan telekomunikasi divisi operasional seperti pada Gambar 4.18.




Gambar 4.18. Topologi jaringan LAN perusahaan pengguna

# BAB 5
# PENGUJIAN DAN ANALISIS

Pada bab ini akan dibahas mengenai pengujian sistem yang dibagi menjadi tiga tahap dan analisis hasil pengujian. Selain analisis hasil pengujian juga dilakukan analisis terhadap *Knowledge Management System* yang dibangun.

## 5.1 Skenario Pengujian

Pada tahap ini terdapat tiga pengujian yang dilakukan yaitu *unit testing*, *load testing*, dan *usability testing*. Masing-masing pengujian dilakukan dengan metode yang berbeda.

### 5.1.1 *Unit Testing*

*Unit testing* merupakan pengujian sistem berdasarkan fungsi sistem tiap unit. Artinya skenario pengujian yang dibuat harus bisa mencakup seluruh fungsionalitas sistem, dalam hal ini harus dapat mencakup fungsionalitas *Knowledge Management System*. Tujuan dari *unit testing* adalah untuk mengukur tingkat keberhasilan fungsi yang dibuat yang dilakukan oleh pengguna. Target dari *unit testing* adalah mengeluarkan hasil aktual penggunaan fungsi sistem sehingga menghasilkan apakah fungsi tersebut berjalan dengan baik atau tidak. *Unit testing* dilakukan oleh responden acak yang tidak mengetahui tentang sistem yang dikembangkan dan *background* penerapan sistem tersebut. Latar belakang pengetahuan responden tidak mempengaruhi hasil *unit testing* sehingga pemilihan acak responden dapat dilakukan. Bentuk questioner untuk unit testing dapat dilihat pada halaman Lampiran 1. Berikut adalah skenario pengujian untuk *unit testing*.

1. *User* tak teregistrasi dapat melihat semua daftar dan artikel.
2. *User* tak teregistrasi dapat melakukan registrasi.
3. *User* tak teregistrasi tidak dapat membuat, mengedit, dan menghapus artikel.
4. *User* yang teregistrasi dapat membuat artikel melalui “buat artikel” di halaman utama.

5. Fungsi *search* berjalan dengan baik ditandai dengan adanya *suggestion*.
6. *User* yang teregistrasi dapat membuat artikel dengan Topik *Fault Handling*.
7. *User* yang teregistrasi dapat membuat artikel dengan Topik *Configuration*.
8. *User* yang teregistrasi dapat membuat artikel dengan Topik *Network Performance*.
9. *User* yang teregistrasi dapat membuat artikel dengan Topik *Security Issue*.
10. *User* yang teregistrasi dapat membuat artikel dengan menambahkan tag *device* terkait (*Router, switch, metro ethernet*, dan *device* lain).
11. *User* yang teregistrasi dapat mengunggah lampiran saat membuat artikel dan lampiran tersebut dapat terelasi dengan baik.
12. *User* yang teregistrasi dapat menambahkan kata kunci saat membuat artikel.
13. *Query* untuk artikel dengan topik *Fault Handling* berjalan dengan baik dilihat dari halaman *Fault handling*.
14. *Query* untuk artikel dengan topik *Configuration* berjalan dengan baik dilihat dari halaman *Configuration*.
15. *Query* untuk artikel dengan topik *Network Performance* berjalan dengan baik dilihat dari halaman *Network Performance*.
16. *Query* untuk artikel dengan topik *Security Issue* berjalan dengan baik dilihat dari halaman *Security Issue*.
17. Navigasi pada *sidebar* berfungsi semua dengan baik.
18. *Query* untuk artikel dengan perangkat *router* berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar *router* di *sidebar*.
19. *Query* untuk artikel dengan perangkat *switch* berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar *switch* di *sidebar*.
20. *Query* untuk artikel dengan perangkat *metro ethernet* berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar *metro ethernet* di *sidebar*.

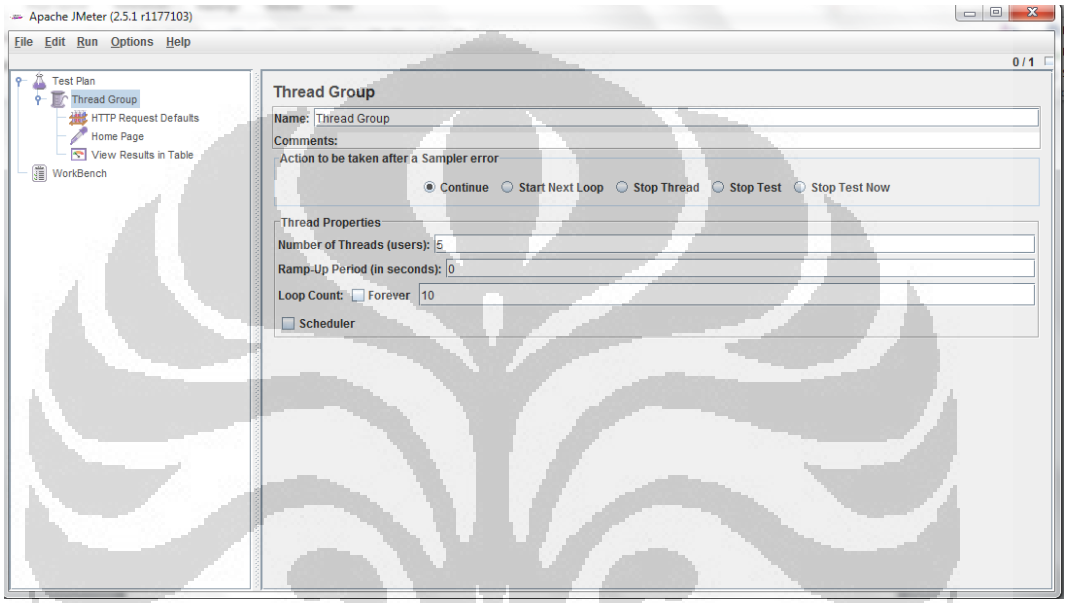
21. *Query* untuk artikel dengan perangkat selain ketiga perangkat tersebut berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar lain-lain di *sidebar*.
22. *User* teregistrasi dapat membuat tulisan problem dengan menggunakan *form* "buat problem" di halaman utama.
23. *Query* untuk halaman problem dapat dilakukan dengan melihat di halaman utama daftar problem berdasarkan status.
24. *Query* terhadap keseluruhan problem dapat dilihat di *sidebar* daftar problem.
25. *User* teregistrasi dapat membuat solusi terhadap problem yang ada dengan memasukkan topik masalah yang ada.
26. Solusi yang sudah dibuat akan terelasi ke dalam halaman problem pada setiap topik masalah.
27. *User* yang teregistrasi dapat melakukan *feedback* terhadap halaman artikel, problem, dan *solution* yang ada dengan kriteria dapat dipercaya, tidak bias, lengkap, dan ditulis dengan baik.
28. Artikel yang sudah diberi *feedback* akan terhitung secara akumulasi.
29. *User* yang sudah teregistrasi dapat melakukan *vote* terhadap artikel.
30. Hasil *vote* dapat dilihat di halaman "Daftar peringkat", yang berisi 20 artikel terbaik, 20 *vote* terakhir, artikel terbaik dengan minimal 5 *vote*, dan daftar *top voter*.
31. Pada setiap halaman artikel dapat dipantau 10 *vote* terakhir.
32. Admin dapat mengelola *user*, memblokir *user*, dan mengganti nama *user*.
33. Admin dapat menghapus semua halaman dan berkas.
34. Admin dapat mengelola halaman istimewa *semantic* Mediawiki.

### 5.1.2 *Load Testing*

*Load testing* merupakan pengujian yang dilakukan pada sistem untuk mengukur performa sistem. Seberapa besar beban yang dapat diterima oleh *user* untuk digunakan secara bersamaan dan juga berapa waktu respon yang diberikan oleh sistem kepada pengguna saat sistem tersebut diakses dalam sebuah jaringan. Pengujian ini menggunakan suatu *software* pendukung bernama *Jmeter*. *Software*

ini digunakan untuk menguji performa *webserver* untuk berbagai *service*. Salah satu servis yang digunakan dalam pengujian ini adalah *http request*.

Skenario pengujian ini yaitu memvariasikan jumlah *user virtual* yang dibuat dengan menggunakan *jmeter* untuk mengakses sistem sehingga didapatkan hasil *traffic*. Dalam pengujian ini, *jmeter* akan membuka salah satu *sample page* pada *webserver* yang sudah di*deploy* pada suatu perusahaan. Tampilan *Jmeter* dapat dilihat pada Gambar 5.1.




Gambar 5.1. *Control panel Jmeter*

Pada *load tesing* dilakukan tiga skenario. Skenario pertama adalah menguji *load* dengan akses tunggal pada halaman utama dan halaman artikel. Karena akses tunggal berarti hanya membutuhkan 1 *virtual user*. Kemudian diberikan *rump-up period* sebesar 1 *second*. Untuk menguji halaman tersebut digunakan pengulangan 10 kali pada setiap *thread*.

Skenario kedua adalah menguji *load* pada halaman utama dan artikel dengan memvariasikan *thread* menjadi 10, 20, 30, 40, dan 50 *virtual user*. 10 virtual *user* digunakan untuk menguji *load* saat *webserver* diakses dengan kondisi yang masih relatif ringan dan 50 virtual *user* menunjukkan kondisi ekstrim terhadap *server* yang diakses. Kemudian diberikan *rump-up period* sebesar 0 *second*, artinya *webserver* diakses oleh beberapa *user* dalam waktu yang

beramaan. Untuk menguji halaman tersebut digunakan pengulangan 10 kali pada setiap *thread*.

Skenario yang ketiga adalah menguji *load* pada halaman utama dan artikel dengan *virtual user* tetap yaitu 50 dan memvariasikan *rump-up* period antara 0, 2, 4, 6, 8, dan 10 detik. *Rump-up* 2 detik maksudnya adalah sebanyak beberapa *virtual user* mengakses dalam *periode* waktu 2 detik dalam waktu yang tidak bersamaan. Untuk menguji halaman tersebut digunakan pengulangan 10 kali pada setiap *thread*.

### 5.1.3 *Usability Testing*

*Usability testing* merupakan pengujian yang dilakukan dengan cara memberikan *questioner* mengenai kegunaan dari *Knowledge Management System* yang dibangun kepada pengguna dan responden acak. Terdapat dua skenario yang dibuat. Skenario pertama dikhususkan untuk pengguna yang akan menggunakan sistem tersebut dan skenario kedua diberikan kepada responden acak yang ditugaskan untuk menilai sistem dari beberapa aspek berdasarkan sudut pandang responden. Tujuan dari *usability testing* ini adalah untuk mengukur seberapa relevan *knowledge management system* yang dibuat terhadap penggunaan di lapangan, juga untuk menilai tampilan sistem serta fungsi sistem terhadap kemudahan penggunaan. Bentuk questioner untk usabiliti testing dapat dilihat pada Lampiran 3 dan Lampiran 5.

Berikut adalah skenario *usability testing* untuk pengguna.

1. Apakah *Knowledge Management System* yang dibuat menarik untuk dikembangkan?
2. Apakah sistem yang dikembangkan sudah merepresentasikan kebutuhan *knowledge* terhadap perusahaan?
3. Apakah sistem yang dikembangkan sudah relevan terhadap perusahaan?
4. Apakah tampilan sistem menarik?
5. Apakah navigasi sistem efisien?
6. Apakah halaman utama sistem merepresentasikan sistem secara keseluruhan?
7. Apakah sistem dapat berjalan dengan baik? (Ditandai dengan tidak adanya *bug*)

8. Apakah *query* yang dihasilkan merepresentasikan *knowledge* yang dibutuhkan?
9. Apakah fitur pencarian menarik dan membantu dengan dukungan *ajax suggestion*?
10. Apakah fitur *article feedback* merepresentasikan kebutuhan perusahaan terhadap *knowledge* yang ada di dalam sistem?
11. Apakah fitur *rating* yang disediakan merepresentasikan *decision support* terhadap kebutuhan perusahaan?
12. Apakah konsep *semantic* cukup relevan terhadap pengembangan *knowledge management system* pada perusahaan ini?
13. Apakah penerapan *Semantic* Mediawiki yang pengembang kembangkan dapat menambah nilai dari *Knowledge Management System* yang ada?
14. Apakah penggunaan Mediawiki sebagai *framework* sesuai dengan kebutuhan?

Skenario tersebut menghasilkan jawaban angka dengan kisaran 1-6 dimana 6 berarti "sangat setuju", 5 berarti "setuju", 4 berarti "cukup setuju", 3 berarti "agak setuju", 2 berarti "kurang setuju", 1 berarti "tidak setuju". Kemudian skenario untuk responden acak dapat sebagai berikut.

1. Apakah sistem yang dibuat menarik untuk dikembangkan?
2. Apakah tampilan halaman utama sistem menarik?
3. Apakah navigasi sistem representatif?
4. Apakah halaman daftar artikel yang dibuat informatif?
5. Apakah halaman daftar problem yang dibuat cukup informatif?
6. Apakah halaman daftar artikel yang dibuat cukup merepresentasikan peringkat halaman artikel?
7. Apakah sistem dapat berjalan dengan baik? (ditandai dengan tidak adanya bug)
8. Apakah fitur pencarian cukup menarik dan cukup membantu dengan dukungan *ajax suggestion*?

Skenario untuk responden acak juga menghasilkan jawaban dengan parameter yang sama yaitu 1-6.

## 5.2 Hasil Pengujian dan Analisa

Berikut adalah hasil pengujian yang telah dilakukan berdasarkan skenario yang sudah dibuat.

### 5.2.1 Hasil Pengujian *Unit Testing*

Dari hasil *questioner* yang dibagikan kepada 10 resonden acak yang ditugaskan untuk menilai fungsi sistem per unit, dihasilkan data berikut seperti pada Tabel 5.1.

Tabel 5.1. Skenrio *unit testing*

| No | Pertanyaan | Hasil |
|---|---|---|
| 1 | *User* tak teregistrasi dapat melihat semua daftar dan artikel. | 100% |
| 2 | *User* tak teregistrasi dapat melakukan registrasi. | 100% |
| 3 | *User* tak teregistrasi tidak dapat membuat, mengedit, dan menghapus artikel. | 100% |
| 4 | *User* yang teregistrasi dapat membuat artikel melalui buat artikel di halaman utama. | 100% |
| 5 | Fungsi *search* berjalan dengan baik di tandai dengan adanya *suggestion*. | 60% |
| 6 | *User* yang teregistrasi dapat membuat artikel dengan Topik *Fault Handling*. | 100% |
| 7 | *User* yang teregistrasi dapat membuat artikel dengan Topik *Configuration*. | 100% |
| 8 | *User* yang teregistrasi dapat membuat artikel dengan Topik *Network Performance*. | 100% |

| | | |
|---|---|---|
| 9 | *User* yang teregistrasi dapat membuat artikel dengan Topik *Security Issue*. | 100% |
| 10 | *User* yang teregistrasi dapat membuat artikel dengan menambahkan tag *device* terkait (Router, switch, metro ethernet, dan device lain). | 100% |
| 11 | *User* yang teregistrasi dapat mengunggah lampiran saat membuat artikel dan lampiran tersebut dapat terelasi dengan baik. | 100% |
| 12 | *User* yang teregistrasi dapat menambahkan kata kunci saat membuat artikel. | 100% |
| 13 | *Query* untuk artikel dengan topik *Fault Handling* berjalan dengan baik dilihat dari halaman *Fault handling*. | 90% |
| 14 | *Query* untuk artikel dengan topik *Configuration* berjalan dengan baik dilihat dari halaman *Configuration*. | 90% |
| 15 | *Query* untuk artikel dengan topik *Network Performance* berjalan dengan baik dilihat dari halaman *Network Performance*. | 90% |
| 16 | *Query* untuk artikel dengan topik *Security Issue* berjalan dengan baik dilihat dari halaman *Security Issue*. | 90% |
| 17 | Navigasi pada sidebar berfungsi semua dengan baik. | 100% |
| 18 | *Query* untuk artikel dengan perangkat router berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar router di *sidebar*. | 100% |

| | | |
|---|---|---|
| 19 | *Query* untuk artikel dengan perangkat switch berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar switch di *sidebar*. | 100% |
| 20 | *Query* untuk artikel dengan perangkat metro ethernet berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar metro ethernet di *sidebar*. | 100% |
| 21 | *Query* untuk artikel dengan perangkat selain ketiga perangkat tersebut berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar lain-lain di *sidebar*. | 100% |
| 22 | *User* teregistrasi dapat membuat tulisan problem dengan menggunakan *form* buat problem di halaman utama. | 100% |
| 23 | *Query* untuk halaman problem dapat dilakukan dengan melihat di halaman utama daftar problem berdasatkan status. | 100% |
| 24 | *Query* terhadap kesuluruhan problem dapat dilihat di sidebar daftar masalah. | 100% |
| 25 | *User* teregistrasi dapat membuat solusi terhadap problem yang ada dengan memasukkan topik masalah yang ada. | 100% |
| 26 | Solusi yang sudah dibuat akan terelasi ke dalam halaman problem pada setiap topik masalah. | 80% |
| 27 | *User* yang teregistrasi dapat melakukan *feedback* terhadap halaman artikel, problem, dan *solution* yang ada dengan kriteria dapat dipercaya, tidak bias, | 100% |

| | | |
|---|---|---|
| | lengkap, dan ditulis dengan baik. | |
| 28 | Artikel yang sudah diberi *feedback* akan terhitung secara akumulasi. | 100% |
| 29 | *User* yang sudah teregistrasi dapat melakukan *vote* terhadap artikel. | 100% |
| 30 | Hasil *vote* dapat dilihat di halaman "Daftar peringkat", yang berisi 20 artikel terbaik, 20 *vote* terakhir, artikel terbaik dengan minimal 5 *vote*, dan daftar *top voter*. | 90% |
| 31 | Pada setiap halaman artikel dapat dipantau 10 *vote* terakhir. | 100% |
| 32 | Admin dapat mengelola *user*, memblokir *user*, dan mengganti nama *user*. | 100% |
| 33 | Admin dapat menghapus semua halaman dan berkas. | 90% |
| 34 | Admin dapat mengelola halaman istimewa semantic Mediawiki. | 100% |

Dari hasil *unit testing* tersebut dapat dilihat bahwa hampir keseluruhan sistem dapat berjalan dengan baik. Untuk lebih lengkapnya, hasil dilampirkan pada halaman Lampiran 2. Adapun yang tidak sempurna terdapat pada poin 5, 13, 14, 15, 16, 26, 30, 33. Pada poin 5, hal ini terjadi karena fungsi *suggestion* yang diterapkan belum dapat beroperasi, tetapi untuk fungsi *search* itu sendiri sudah berjalan dengan baik. Kemudian untuk poin 13 sampai dengan 15 merupakan fungsi *query*. Pada satu penguji, fungi query tidak berjalan dengan baik, hal tersebut bukan karena fungsi yang tidak beroperasi, tetapi karena *delay uptade database* pada halaman yang melakukan *query* tidak secara spontan sehingga butuh waktu jeda dari suatu artikel dibuat sampai bisa ditampilkan pada halaman daftar. Untuk poin 26, terdapat satu responden yang menjawab "tidak" dan satu responden menjawab "tidak tahu", hal ini terjadi karena responden tidak cermat

ketika mencoba sistem yang diuji. Secara fungsi, relasi solusi terhadap problem sudah berjalan dengan baik. Untuk poin 30, terdapat *delay* untuk menampilkan peringkat pada halaman peringkat sehingga halaman yang sudah diperingkat tidak secara spontan muncul pada halaman peringkat. Untuk poin 33, secara fungsi sudah berjalan dengan baik, admin dapat menghapus semua halaman dan berkas, tetapi satu responden kurang cermat dalam menguji sistem yang ada.

Dari hasil pengujian unit ini, dapat dinilai bahwa *knowledge management system* secara fungsi sudah berjalan dengan baik dengan diambil rata-rata keberhasilan dalam pengujian sebesar 96,47%. Dari pengujian unit ini, beberapa responden mengeluhkan akses yang cukup lama pada halaman utama. Oleh karena itu akan dilakukan pengujian terhadap akses *http* dengan melihat *response time* pada pengujian performa yang dibahas selanjutnya pada hasil pengujian *load testing*. Pengujian unit ini juga bertujuan untuk memberikan pengalaman kepada responden terhadap penggunaan sistem yang dipandu dengan skenario *unit testing*. Pengalaman tersebut yang nantinya digunakan untuk menguji kegunaan sistem dengan *usability testing*.

### 5.2.2 Hasil Pengujian *Load Testing*

Pengujian *load* ini dibagi menjadi tiga skenario yaitu untuk membandingkan *load* pada halaman utama dan halaman artikel, untuk menguji ketahanan *load* sistem terhadap banyaknya *user* yang mengakses, serta untuk menguji ketahanan sistem untuk kondisi ekstrim jika *user* mengaksesnya tidak dalam satu waktu tapi pada periode tertentu.

Pada pengujian skenario pertama untuk menguji dua halaman yang berbeda yaitu halaman utama dan halaman artikel, didapatkan data sebagai berikut seperti pada Tabel 5.2.

Tabel 5.2. Hasil pengujian *load testing*

| *Sample#* | *Thread* | *Label* | *Response Time (ms)* | *Status* |
|---|---|---|---|---|
| 1 | *Thread Group 1-1* | Home Page | 14809 | *Success* |
| 2 | *Thread Group 1-1* | Halaman Artikel | 2219 | *Success* |
| 3 | *Thread Group 1-1* | Home Page | 4807 | *Success* |
| 4 | *Thread Group 1-1* | Halaman Artikel | 2183 | *Success* |

| | | | | |
|---|---|---|---|---|
| 5 | *Thread Group 1-1* | Home Page | 4693 | *Success* |
| 6 | *Thread Group 1-1* | Halaman Artikel | 2211 | *Success* |
| 7 | *Thread Group 1-1* | Home Page | 4650 | *Success* |
| 8 | *Thread Group 1-1* | Halaman Artikel | 2160 | *Success* |
| 9 | *Thread Group 1-1* | Home Page | 4719 | *Success* |
| 10 | *Thread Group 1-1* | Halaman Artikel | 2183 | *Success* |
| 11 | *Thread Group 1-1* | Home Page | 4696 | *Success* |
| 12 | *Thread Group 1-1* | Halaman Artikel | 2154 | *Success* |
| 13 | *Thread Group 1-1* | Home Page | 4669 | *Success* |
| 14 | *Thread Group 1-1* | Halaman Artikel | 2157 | *Success* |
| 15 | *Thread Group 1-1* | Home Page | 4717 | *Success* |
| 16 | *Thread Group 1-1* | Halaman Artikel | 2165 | *Success* |
| 17 | *Thread Group 1-1* | Home Page | 4688 | *Success* |
| 18 | *Thread Group 1-1* | Halaman Artikel | 2169 | *Success* |
| 19 | *Thread Group 1-1* | Home Page | 4708 | *Success* |
| 20 | *Thread Group 1-1* | Halaman Artikel | 2165 | *Success* |
| 21 | *Thread Group 1-1* | Home Page | 4704 | *Success* |
| 22 | *Thread Group 1-1* | Halaman Artikel | 2188 | *Success* |

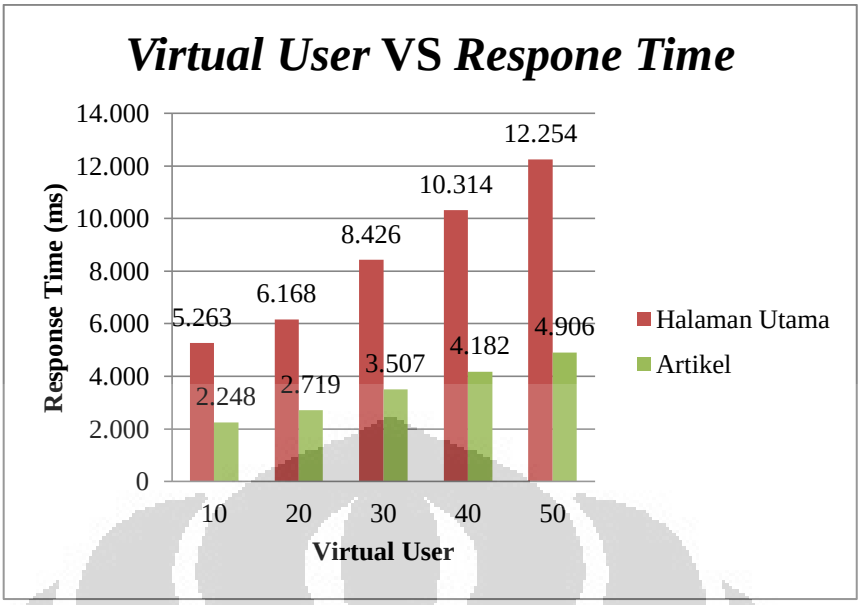
Pada pengujian *load* untuk halaman utama dan halaman artikel dengan mengambil sample artikel *Network_Security*, dapat dilihat hasilnya menunjukkan bahwa halaman utama menghasilkan *response time* yang lebih lama dibandingkan dengan halaman artikel yaitu dengan rata-rata 4705,1 ms yaitu sekitar 5 detik, sedangkan *response time* untuk halaman artikel rata-ratanya hanya sekita 2173,5 ms atau sekitar 2 detik. Pada sample 1 dapat dilihat bahwa *response time* yang diterima untuk mengakases halaman utama sebesar 14809 ms atau sekitar 15 detik. Hal ini terjadi karena digunakan untuk *load* pada saat awal mengakses halaman utama dimana pada halaman utama terdapat cukup banyak *query* yang dilakukan terhadap properti *semantic* Mediawiki. Oleh karena itu, load menjadi cukup berat untuk mengakses halaman utama di awal akses.

Untuk pengujian dengan skenario pertama dapat berjalan dengan baik. Namun bagaimana jika halaman tersebut diakses lebih dari satu pengguna dalam satu waktu. Apakah sistem tersebut masih dapat merespon permintaan *user* yang banyak tersebut? Untuk itu dilakukan pengujian terhadap beban sistem dari akses beberapa pengguna sekaligus berdasarkan skenario pengujian kedua. Berikut hasil dari pengujian dengan skenario kedua seperti terlihat pada Tabel 5.3.

Tabel 5.3. Hasil pengujian load testing dengan variasi vurtual *user*

| *Virtual User* | *Label* | Rata-rata waktu respon (ms) | Jumlah *sample* | *Min* respon (ms) | *Max* respon (ms) | *Error%* |
|---|---|---|---|---|---|---|
| 10 | Halaman Utama | 5.263 | 100 | 4.839 | 6.729 | 0% |
| | Artikel | 2.248 | 100 | 2.144 | 2.739 | 0% |
| 20 | Halaman Utama | 6.168 | 200 | 4.862 | 9.100 | 0% |
| | Artikel | 2.719 | 200 | 2.190 | 4.714 | 0% |
| 30 | Halaman Utama | 8.426 | 300 | 5.470 | 11.835 | 0% |
| | Artikel | 3.507 | 300 | 2.152 | 7.049 | 0% |
| 40 | Halaman Utama | 10.314 | 400 | 4.736 | 16.495 | 0% |
| | Artikel | 4.182 | 400 | 2.166 | 8.864 | 0% |
| 50 | Halaman Utama | 12.254 | 500 | 4.795 | 20.922 | 0% |
| | Artiel | 4.906 | 500 | 2.137 | 12.822 | 0% |

Dari data pada tabel diatas didapat bahwa untuk pengujian *load* dengan memvariasikan jumlah *user* yang mengakses server sekaligus dalam satu waktu, didapat bahwa setiap penambahan 10 *user*, terjadi penambahan *response time*. Untuk variasi 10 *user*, didapat bahwa rata-rata waktu respon untuk mengakses halaman utama mencapai sekitar 5 detik, sedangkan untuk mengakses *server* dengan variasi *user* 50 menghasilkan waktu respon rata-rata sebesar sekitar 12 detik. Untuk lebih jelasnya dapat dilihat melalui grafik pada Gambar 5.2.




Gambar 5.2. Grafik virtual *user* terhadap waktu respon

Dari grafik diatas didapat bahwa kenaikan waktu respon yang dihasilkan relatif sama, artinya dengan variasi yang diberikan 10 *user* setiap pengujian dihasilkan peningkatan waktu respon yang cukup konsisten. Dengan demikian bisa disimpulkan bahwa dengan semakin banyak *user* yang mencoba untuk mengakses *server* dalam satu waktu maka akan menambah beban *server* untuk bisa melayani *user* sehingga waktu respon yang dihasilkan menjadi besar.

Kemudian pada pengujian ketiga dilakukan dengan memvariasikan periode akses (*rump-up*) terhadap keadaan ekstrim untuk 50 *virtual user*. Dari hasil variasi antara 0, 2, 4, 6, 8, dan 10 detik didapatkan hasil sebagai berikut seperti pada Tabel 5.4.

Tabel 5.4. Hasil pengujian load testing dengan memvariasikan *rump-up periode*

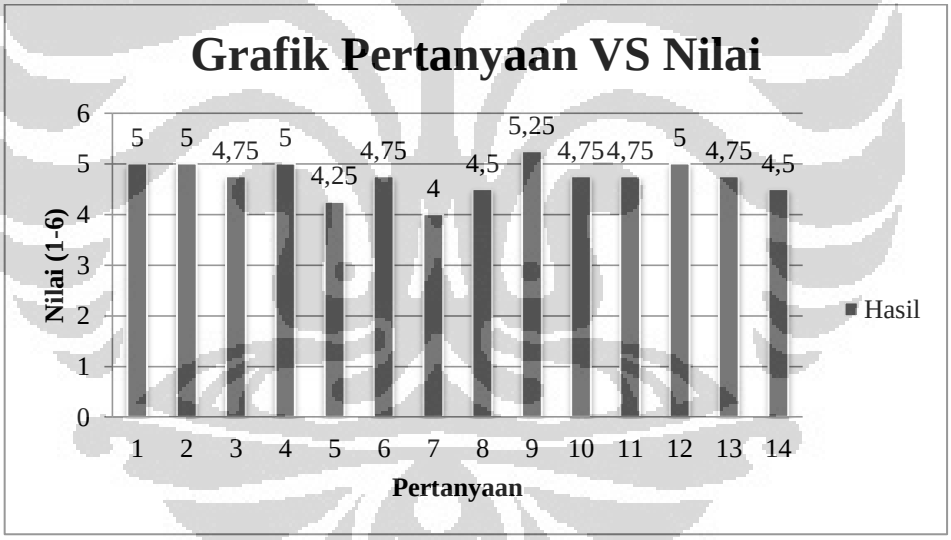
| *Rump-up* (detik) | *Label* | *Sample#* | Rata-rata respon (ms) | *Min* renspon | *Max* Respon | *Error%* |
|---|---|---|---|---|---|---|
| 0 | Halaman utama | 500 | 12254 | 4795 | 20922 | 0% |
| 10 | Halaman utama | 500 | 11694 | 5828 | 19473 | 0% |
| 20 | Halaman utama | 500 | 12352 | 4896 | 33662 | 0% |
| 30 | Halaman utama | 500 | 11196 | 5403 | 18953 | 0% |
| 40 | Halaman utama | 500 | 11382 | 4808 | 21224 | 0% |
| 50 | Halaman utama | 500 | 11772 | 4887 | 18050 | 0% |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | Artikel | 500 | 4906 | 2137 | 12822 | 0% |
| 10 | Artikel | 500 | 5014 | 2266 | 43202 | 0% |
| 20 | Artikel | 500 | 5023 | 2205 | 13294 | 0% |
| 30 | Artikel | 500 | 4578 | 2165 | 8703 | 0% |
| 40 | Artikel | 500 | 4526 | 2141 | 8999 | 0% |
| 50 | Artikel | 500 | 4819 | 2150 | 9556 | 0% |

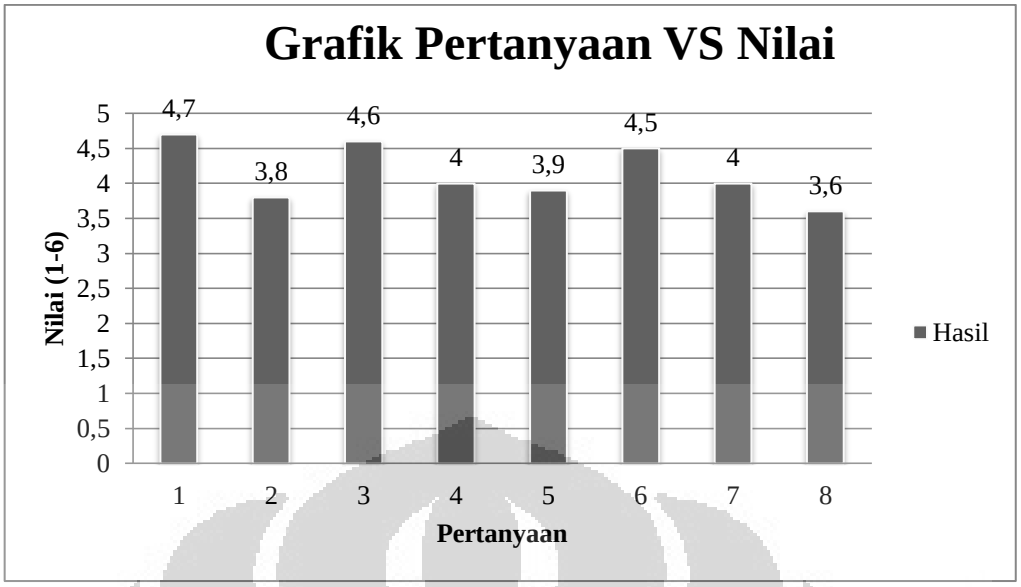
Dari hasil pengujian skenario tiga didapatkan data seperti yang dapat dilihat pada tabel di atas. Dari data hasil pengujian tersebut didapatkan bahwa rata-rata waktu respon yang dihasilkan untuk variasi periode *rump-up* tidak menunjukkan perbedaan yang signifikan.

### 5.2.3 *Usability Testing*

Dari hasil uji kegunaan dengan menggunakan dua skenario, didapatkan hasil sebagai berikut seperti pada Gambar 5.3 dan 5.4.




Gambar 5.3. Grafik pertanyaan terhadap nilai untuk pengguna




Gambar 5.4. Grafik pertanyaan terhadap nilai untuk responden acak

Grafik pertama menjelaskan hasil pengujian untuk pengguna. Dari grafik tersebut didapat bahwa rata-rata nilai dari semua pertanyaan didapatkan hasil 4,73. Hal tersebut berarti dari fungsi kegunaan sudah di atas cukup, dan sesuai terhadap kebutuhan perusahaan. Sedangkan pada grafik hasil pengujian terhadap responden acak, didapatkan hasil dengan rata-rata jawaban sekitar 4,14. Hal tersebut berarti bahwa sistem yang dikembangkaan cukup menarik menurut para responden. Untuk melihat hasil lengkapnya dapat dilihat pada halaman Lampiran 4 dan Lampiran 6.

## 5.3 **Analisis Representasi *Knowledge* pada KMS**

*Knowledge Management System* yang diterapkan merupakan sebuah sistem yang mengatur sumber-sumber *knowledge* menjadi dokumen-dokumen yang terekam dalam system. Dokumen tersebut berbentuk halaman-halaman wiki yang dapat diakses kembali oleh pengguna yang lain.

Seperti halnya konsep *knowledge managament process*, terdapat tahap *discovery*, *capture*, *sharing*, dan *application*. Tahap *discovery* disini direpresentasikan oleh halaman problem dimana problem yang ada disampaikan dalam bentuk halaman wiki dan ditanggapi dalam bentuk halaman solusi yang

terelasi terhadap halaman problem tersebut. Hal ini menunjukkan proses *sharing knowledge* berdasarkan *learning by* problem. Semakin banyak problem yang dapat di *share*, semakin banyak *knowledge* yang di rekam dalam sistem dalam bentuk solusi-solusi.

Kemudian pada tahap *knowledge capture* direpresentasikan pada halaman artikel dimana halaman ini berisi tentang *experience* atau *knowledge* individu yang di masukkan dalam bentuk halaman wiki pada sistem. Pada halaman artikel, *knowledge* yang di cantumkan murni dari pengetahuan pribadi sehingga setiap *knowledge* yang di share bisa sangat beragam untuk topik artikel yang sama. Hal ini memungkinkan proses *transfer knowledge* yang lebih cepat dan terdokumentasi jika dibandingkan proses *transfer knowledge* tradisional dari mulut ke mulut (diskusi) dengan proses *tacit to tacit*.

Kemudian tahap akhir merupakan proses aplikasi *knowledge* yang ada. Dengan banyaknya *knowledge* yang terekam dalam sistem, maka *knowledge* tersebut siap untuk digunakan sebagai *inventory knowledge* yang siap untuk dilengkapi dan dimaksimalkan.

# BAB 6
# KESIMPULAN

Dari pembahasan hasil dan analisa *knowledge management system* yang sudah dipaparkan sebelumnya, maka dapat diambil kesimpulan:

1. *Knowledge management system* yang diterapkan menggunakan *Semantic* Mediawiki dengan *knowledge* audit berdasarkan kebutuhan *knowledge* pada perusahaan telekomunikasi divisi operasional sudah sesuai dengan yang diharapkan ditandai dengan nilai pada *usability testing* dengan rata-rata sebesar 4,73 dari skala 1 sampai 6 dan cukup efektif dengan hasil *usability testing* dengan responden acak sebesar 4,14 dari skala 1 sampai 6.
2. *Knowledge management system* yang diterapkan menggunakan *Semantic* Mediawiki berhasil diterapkan dengan baik ditandai dari hasil *unit testing* yang hasilnya mencapai 96,47% dari 34 skenario yang diberikan.
3. *Knowledge management system* yang diterapkan menggunakan *semantic* Mediawiki dapat diakses dengan baik ditandai dari hasil load testing untuk mengakses halaman utama dengan rata-rata *response time* sebesar 4705,1 ms dan halaman artikel dengan rata-rata *response time* sebesar 2173,5 ms.
4. *Knowledge management system* yang diterapkan menggunakan *semantic* Mediawiki dapat menahan beban *user* hingga 50 *user* ditandai dengan *response time* yang dihasilkan pada kondisi 50 *user* mencapai rata-rata 12 detik dan tidak ada *error* yang terjadi pada pengujian *load*.
5. Implementasi *knowledge management system* telah berhasil dilakukan pada perusahaan telekomunikasi divisi operasional yang diakses menggunakan jaringan *Local Area Network*.
6. Transfer *knowledge* dilakukan menggunakan *knowledge management system* berdasarkan diagram alir *knowledge* pada perusahaan telekomunikasi divisi operasional.

## DAFTAR REFERENSI


[1] Bacerra-Fernandez, Irma, Avelino Gonzalez, and Rajiv Shaberwal. (2003). *Knowledge Management: Challanges, Solution and Technologies*. New Jersey: Pearson Education.

[2] Tiwana, Amrit. (2000). *The Knowledge Management Toolkit*. USA: Prentice Hall.

[3] El Farisi, Salman. (2010). *Prototype Knowledge Management System Berbasis Wiki Untuk Keperluan Proyek Rekayasa Perangkat Lunak: Studi Kasus Pusat Ilmu Komputer Universitas Indonesia*. Depok: Universitas Indonesia.

[4] Alshahrani, Mohammed, & Elhag, Taha. (2008). Developing a Framework for Knowledge Management System. *Journal of Knowledge Management System*.

[5] Chia-Han Yang, Ming-Yin Wu, Chien-Min Lin, & Don-Lin Yang. (2008). Implementation of Wiki-based Knowledge Management System for Small Research Groups. Journal of Knowledge Management System.

[6] Ying-Liang Wu, & Yi-Hua Li. (2008). Research on The Model of Knowledge audit. Journal of Knowledge Management System.

[7] Tomasev, Nenad, & Mladenic, Dunja. (2009). Semantic Web Wiki: Social Network Analysis of Page Editing.

[8] Riri, Fitri Sari. (2010). *Introduction to Unified Modelling Language (UML) lecture 7*. October 4, 2010. Dalam kuliah Software Engineering.

[9] Kurt, Schneider. (2009). *Experience and Knowledge Management in Software Engineering*. Germany: Springer-Verlag Berlin Heidelberg.

[10] Chowdury, Naguib. (2006). Knowledge Audit Module. *Overview and sample questionare from naguibch@yahoo.com*

[11] Sommerville, Ian. (2004). *Software Engineering* (7th ed.). England: Pearson Education Limited

[12] ILIAS. (2011). *ILIAS Features.* November 27, 2011. http://www.ilias.de/docu/goto_docu_lm_392.html.

[13] Wikipedia. (2011). *ILIAS.* November 27, 2011.http://en.wikipedia.org/wiki/ILIAS.

[14] Intellexer Categorizer. (2011). November 28, 2011. www.intellexer.categorizer.com

[15] Moodle LMS. (2011). November 28, 2011. http://moodle.org

[16] Mediawiki. (2011). November 4, 2011. http://www.Mediawiki.org/wiki/Mediawiki

[17] Semantic Mediawiki. (2011). November 10, 2011. http://edutechwiki.unige.ch/en/Main_Page

# LAMPIRAN

Lampiran 1. Skenario *unit testing*.

| | | | |
|---|---|---|---|
| **Nama system** | Knowledge Management System berbasis Semantic Mediawiki | | |
| **Pengembang** | Irvanda Kurniadi V. | **Nama Tester** | |
| **Tanggal testing** | | **Email** | |

| No | Pertanyaan | Hasil Aktual (ya/tidak) |
|---|---|---|
| 1 | *User* tak teregistrasi dapat melihat semua daftar dan artikel. | |
| 2 | *User* tak teregistrasi dapat melakukan registrasi. | |
| 3 | *User* tak teregistrasi tidak dapat membuat, mengedit, dan menghapus artikel. | |
| 4 | *User* yang teregistrasi dapat membuat artikel melalui buat artikel di halaman utama. | |
| 5 | Fungsi search berjalan dengan baik di tandai dengan adanya suggestion. | |
| 6 | *User* yang teregistrasi dapat membuat artikel dengan Topik Fault Handling. | |
| 7 | *User* yang teregistrasi dapat membuat artikel dengan Topik Configuration. | |
| 8 | *User* yang teregistrasi dapat membuat artikel dengan Topik Network Performance. | |

| | | |
|---|---|---|
| 9 | *User* yang teregistrasi dapat membuat artikel dengan Topik Security Issue. | |
| 10 | *User* yang teregistrasi dapat membuat artikel dengan menambahkan tag device terkait (Router, switch, metro ethernet, dan device lain). | |
| 11 | *User* yang teregistrasi dapat mengunggah lampiran saat membuat artikel dan lampiran tersebut dapat terelasi dengan baik. | |
| 12 | *User* yang teregistrasi dapat menambahkan kata kunci saat membuat artikel. | |
| 13 | Query untuk artikel dengan topik Fault Handling berjalan dengan baik dilihat dari halaman Fault handling. | |
| 14 | Query untuk artikel dengan topik Configuration berjalan dengan baik dilihat dari halaman Configuration. | |
| 15 | Query untuk artikel dengan topik Network Performance berjalan dengan baik dilihat dari halaman Network Performance. | |
| 16 | Query untuk artikel dengan topik Security Issue berjalan dengan baik dilihat dari halaman Security Issue. | |
| 17 | Navigasi pada sidebar berfungsi semua dengan baik. | |
| 18 | Query untuk artikel dengan perangkat router berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar router di sidebar. | |
| 19 | Query untuk artikel dengan perangkat switch berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar switch di sidebar. | |

| | | |
|---|---|---|
| 20 | Query untuk artikel dengan perangkat metro ethernet berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar metro ethernet di sidebar. | |
| 21 | Query untuk artikel dengan perangkat selain ketiga perangkat tersebut berjalan dengan baik, artikel yang terelasi muncul pada daftar di halaman utama dan di daftar lain-lain di sidebar. | |
| 22 | *User* teregistrasi dapat membuat tulisan problem dengan menggunakan form buat problem di halaman utama. | |
| 23 | Query untuk halaman problem dapat dilakukan dengan melihat di halaman utama daftar problem berdasatkan status. | |
| 24 | Query terhadap kesuluruhan problem dapat dilihat di sidebar daftar masalah. | |
| 25 | *User* teregistrasi dapat membuat solusi terhadap problem yang ada dengan memasukkan topik masalah yang ada. | |
| 26 | Solusi yang sudah dibuat akan terelasi ke dalam halaman problem pada setiap topik masalah. | |
| 27 | *User* yang teregistrasi dapat melakukan feedback terhadap halaman artikel, problem, dan solution yang ada dengan kriteria Dapat dipercaya, tidak bias, lengkap, dan ditulis dengan baik. | |
| 28 | Artikel yang sudah diberi feedback akan terhitung secara akumulasi. | |
| 29 | *User* yang sudah teregistrasi dapat melakukan vote terhadap artikel. | |

| | | |
|---|---|---|
| 30 | Hasil vote dapat dilihat di halaman “Daftar peringkat”, yang berisi 20 artikel terbaik, 20 vote terakhir, artikel terbaik dengan minimal 5 vote, dan daftar top voter. | |
| 31 | Pada setiap halaman artikel dapat dipantau 10 vote terakhir. | |
| 32 | Admin dapat mengelola *user*, memblokir *user*, dan mengganti nama *user*. | |
| 33 | Admin dapat menghapus semua halaman dan berkas. | |
| 34 | Admin dapat mengelola halaman istimewa semantic Mediawiki. | |
| Masukan: | | |

Lampiran 2. Hasil *unit testing*.

| Nama Responden | Pertanyaan | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| Rian C. | √ | √ | √ | √ | **x** | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Ian H. | √ | √ | √ | √ | **x** | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Suwega | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Valdo | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Dimas | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Helmi | √ | √ | x | √ | **x** | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Slamet | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Akbar | √ | √ | √ | √ | **x** | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Wawan | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Ari V. | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | **x** | **x** | **x** | **x** | √ |

| | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rian C. | √ | √ | √ | √ | √ | √ | √ | √ | **x** | √ | √ | √ | √ | √ | √ | √ | √ |
| Ian H. | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Suwega | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Valdo | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dimas | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Helmi | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | **x** | √ |
| Slamet | √ | √ | √ | √ | √ | √ | x | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| Akbar | √ | √ | √ | √ | √ | √ | √ | √ | | √ | √ | √ | √ | √ | √ | √ | √ |
| Wawan | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | **x** | √ | √ | √ | √ |
| Ari V. | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

Lampiran 3. Skenario Usability Testing untuk Pengguna

| **Nama system** | Knowledge Management System berbasis Semantic Mediawiki | | |
|---|---|---|---|
| **Pengembang** | Irvanda Kurniadi V. | **Nama Tester** | |
| **Tanggal testing** | | **Email** | |

| **No** | **Pertanyaan** | **Hasil Aktual** |
|---|---|---|
| 1 | Apakah knowledge management system yang dibuat menarik untuk dikembangkan? | |
| 2 | Apakah sistem yang dikembangkan sudah merepresentasikan kebutuhan knowledge terhadap perusahaan? | |
| 3 | Apakah sistem yang dikembangkan sudah relevan terhadap perusahaan? | |
| 4 | Apakah tampilan sistem menarik? | |
| 5 | Apakah navigasi sistem efisien? | |
| 6 | Apakah halaman utama sistem merepresentasikan sistem secara keseluruhan? | |
| 7 | Apakah sistem dapat berjalan dengan baik? (Ditandai dengan tidak adanya bug) | |
| 8 | Apakah query yang dihasilkan merepresentasikan | |

| | | |
|---|---|---|
| | knowledge yang dibutuhkan?membantu dengan dukungan ajax suggestion? (1-6) | |
| 9 | Apakah fitur pencarian menarik dan membantu dengan dukungan ajax suggestion? | |
| 10 | Apakah fitur article feedback merepresentasikan kebutuhan perusahaan terhadap knowledge yang ada di dalam sistem? | |
| 11 | Apakah fitur rating yang disediakan merepresentasikan decision support terhadap kebutuhan perusahaan? | |
| 12 | Apakah konsep semantic cukup relevan terhadap pengembangan knowledge management system pada perusahaan ini? | |
| 13 | Apakah penerapan Semantic Mediawiki yang pengembang kembangkan dapat menambah nilai dari KMS yang ada? | |
| 14 | Apakah penggunaan Mediawiki sebagai framework sesuai dengan kebutuhan? | |
| Masukan: | | |

Lampiran 4. Hasil Usability Testing untuk Pengguna

| Pertanyaan | Pak Taufiq | Pak I Wayan | Pak Johan | Pak Ismail | Rata-rata |
|---|---|---|---|---|---|
| 1 | 5 | 5 | 6 | 4 | 5 |
| 2 | 5 | 4 | 6 | 5 | 5 |
| 3 | 5 | 4 | 5 | 5 | 4,75 |
| 4 | 6 | 5 | 5 | 4 | 5 |
| 5 | 5 | 4 | 4 | 4 | 4,25 |
| 6 | 4 | 5 | 5 | 5 | 4,75 |

| | | | | | |
|---|---|---|---|---|---|
| 7 | 4 | 5 | 3 | 4 | 4 |
| 8 | 5 | 4 | 5 | 4 | 4,5 |
| 9 | 6 | 5 | 6 | 4 | 5,25 |
| 10 | 4 | 5 | 5 | 5 | 4,75 |
| 11 | 5 | 4 | 5 | 5 | 4,75 |
| 12 | 5 | 4 | 6 | 5 | 5 |
| 13 | 4 | 5 | 6 | 4 | 4,75 |
| 14 | 5 | 4 | 5 | 4 | 4,5 |

Lampiran 5. Skenario Usability Testing untuk Responden Acak

| | | | |
|---|---|---|---|
| **Nama system** | Knowledge Management System berbasis Semantic Mediawiki | | |
| **Pengembang** | Irvanda Kurniadi V. | **Nama Tester** | |
| **Tanggal testing** | | **Email** | |

| **No** | **Pertanyaan** | **Hasil Aktual** |
|---|---|---|
| 1 | Apakah sistem yang pengembang buat menarik untuk dikembangkan? (1-6) | |
| 2 | Apakah tampilan Halaman Utama sistem menarik? (1-6) | |
| 3 | Apakah navigasi sistem representatif? (1-6) | |
| 4 | Apakah daftar artikel yang dibuat informatif?(1-6) | |
| 5 | Apakah daftar problem yang dibuat cukup informatif?(1-6) | |
| 6 | Apakah daftar peringkat cukup merepresentasikan peringat halaman artikel?(1-6) | |
| 7 | Apakah sistem dapat berjalan dengan baik? Ditandai dengan tidak adanya bug (1-6) | |
| 8 | Apakah fitur pencarian cukup menarik dan cukup membantu dengan dukungan ajax suggestion? (1-6) | |

| Masukan: |
|---|
| |

Lampiran 6. Hasil Unit Testing untuk Responden Acak

| Nama Responden | Pertanyaan | | | | | | | | Rata-rata |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| Rian C. | 5 | 4 | 5 | 4 | 4 | 5 | 3 | 4 | 4,3 |
| Ian H. | 6 | 4 | 6 | 5 | 5 | 4 | 6 | 4 | 5 |
| Suwega | 4 | 4 | 5 | 4 | 4 | 5 | 4 | 4 | 4,3 |
| Valdo | 4 | 4 | 4 | 3 | 3 | 4 | 4 | 5 | 3,9 |
| Dimas | 4 | 3 | 4 | 4 | 4 | 4 | 2 | 2 | 3,4 |
| Helmi | 5 | 4 | 4 | 4 | 3 | 5 | 5 | 3 | 4,1 |
| Slamet | 5 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 3,9 |
| Akbar | 5 | 5 | 5 | 4 | 4 | 5 | 5 | 4 | 4,6 |
| Wawan | 5 | 3 | 4 | 5 | 5 | 5 | 3 | 2 | 4 |
| Ari V. | 4 | 3 | 5 | 4 | 4 | 4 | 4 | 4 | 4 |