PEMBANGUNAN APLIKASI MANAJEMEN RUANG *MEETING* PADA PT. TELKOMSEL INDONESIA Tbk. DENGAN MENGGUNAKAN *PLAY! FRAMEWORK*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Afdin Fadila Prima
NIM: 125150207111056

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

PEMBANGUNAN APLIKASI MANAJEMEN RUANG *MEETING* PADA PT. TELKOMSEL INDONESIA TBK. DENGAN MENGGUNAKAN *PLAY! FRAMEWORK*

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Afdin Fadila Prima
NIM: 125150207111056

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Denny Sagita R., S.Kom, M.Kom	Fajar Pradana, S.ST, M.Eng
NIP: 19851124 201504 1 001	NIP: 19871121 201504 1 004

Mengetahui
Ketua Program Studi Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 3 Agustus 2016

Afdin Fadila Prima

NIM:125150207111056

ABSTRAK

Afdin Fadila Prima, 2016. Pembangunan Aplikasi Mnajemen Ruang *Meeting* Pada PT. Telkomsel Indonesia Tbk. Dengan Menggunakan *PLAY! Framework*

Dosen Pembimbing : Denny Sagita R., S.Kom, M.Kom dan Fajar Pradana, S.ST, M.Eng

Manajemen penggunaan ruang *meeting* pada masa sekarang ini sudah menjadi bagian yang penting dalam proses perkembangan kemajuan bisnis suatu perusahaan dikarenakan sebagian besar kebijakan yang diambil oleh suatu perusahaan banyak yang ditentukan didalam ruang *meeting* tersebut. PT. Telkomsel Indonesia Tbk. yang merupakan salah satu operator telekomunikasi terbesar di indonesia dalam proses perkembangan perusahaannya belum melakukan manajemen terhadap penggunaan ruang *meeting* di dalam setiap gedung kerja yang dimilikinya. Prosedur penggunaan ruang *meeting*-nya juga terbilang konvensional, yaitu dengan menanyakan terlebih dahulu ketersediaan ruang *meeting* kepada resepsionis yang berada pada suatu gedung kerja. Prosedur ini tentu membuang-buang waktu ditengah kemajuan teknologi yang begitu pesat. Untuk mengatasi masalah tersebut diperlukan pembangunan sebuah aplikasi manajemen ruang *meeting* yang berbasis web yang khusus dibangun untuk PT. Telkomsel Indonesia Tbk. Dalam proses pengembangan aplikasinya akan mengacu pada SDLC *waterfall* dan dikombinasikan dengan kerangka kerja *PLAY!* pada tahap perancangan dan implementasi kode program. Di akhir penelitian akan dilakukan pengujian *usability* aplikasi dengan metode *System Usability Scale* untuk mengetahui kualitas aplikasi yang dikembangkan dari sisi *usability*-nya menurut *user* aplikasi.

Kata Kunci : Manajemen Ruang *Meeting, PLAY! Framework, System Usability Scale*

ABSTRACT

Meeting room management today is the important parts on development process of a company because, most of a company policy decided inside these meeting room. PT. Telkomsel Indonesia Tbk which is one of the largest telecommunications operator in indonesia in developing the company has not made use management concept in the use of meeting room inside its work office. The request of using its meeting room is conventional, the department who want to use these meeting room should ask to receptionist for the availability information of these meeting room. This procedure is wasting the time. To solve this problem, need to developed a meeting room management application based on web specifically for PT. Telkomsel Indonesia Tbk. In the application development process will refer to waterfall system development life cycle and combined with PLAY! Framework on the stage of design and implementation of program code. At the end of the study will be conducted usability testing of application with System Usability Scale method to determine the quality of the application developed in terms of usability by the user application.

Keywords : ***Meeting Room Management, PLAY! Framework, System Usability Scale***

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas berkah, rahmat serta hidayah-Nya sehingga penulis dapat menyelesaikan laporan Skripsi dengan judul "Pembangunan Aplikasi Manajemen Ruang *Meeting* Pada PT. Telkomsel Indonesia Tbk. Dengan Menggunakan *PLAY! Framework*" dengan baik dan tepat waktu. Keberhasilan tersebut tak lepas dari bantuan dan dukungan dari berbagai pihak baik secara moril maupun materil. Oleh karena itu dalam kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Bapak Denny Sagita R., S.Kom, M.Kom selaku dosen pembimbing 1.
2. Bapak Fajar Pradana, S.ST, M.Eng selaku dosen pembimbing 2.
3. Bapak Drs. Mardji, MT selaku Ketua Program Studi Informatika/Ilmu Komputer Universitas Brawijaya Malang.
4. Kedua orang tua yang selalu mendoakan kelancaran saya dalam menyelesaikan skripsi ini.
5. Pradhina Dewi Rumala Sari, A.md my soulmate
6. Kikit Maulana S.kom (calon), Fahmi Arief S.kom (calon), Genjah Amartha S.kom (calon), Randi Pratama S.kom (calon), Boni Saputra S.kom, Fadhil S.kom. thanks ma bro.
7. Imam Achmad Hambali S.kom ma bro terimakasih sudah sering minjemin printer.
8. I-Class Alpha 2012, Awal yang luar biasa.
9. Mas Sugeng D Winanjuar, selaku pembimbing KKN di PT. Telkomsel Indonesia Tbk. yang telah memberikan bimbingan selama masa KKN
10. Bapak Galih Hari, selaku penanggung jawab KKN di PT. Telkomsel Indonesia Tbk. yang telah memberikan bimbingan dan arahan selama masa KKN
11. Seluruh pihak yang telah membantu kelancaran skripsi saya yang tidak dapat kami sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih banyak kekurangan baik format laporan maupun isinya. Oleh karena itu, kami mengharapkan saran dan kritik yang membangun dari para pembaca guna perbaikan laporan selanjutnya. Semoga laporan ini dapat memberikan manfaat bagi semua pihak, Aamiin.

Malang, Agustus 2016

Afdin Fadila Prima
afdinfp@gmail.com

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

PT. Telkomsel Indonesia Tbk. adalah operator telekomunikasi seluler GSM di Indonesia dengan layanan pascabayar kartuHALO yang diluncurkan pada tanggal 26 Mei 1995. PT. Telkomsel Indonesia Tbk. mengklaim sebagai operator seluler terbesar di Indonesia yang tersebar dari Sabang sampai Merauke dan memiliki jaringan terluas yang mampu menjangkau lebih dari 95% populasi Indonesia di seluruh penjuru nusantara untuk melayani kebutuhan komunikasi berbagai lapisan masyarakat mulai dari kawasan perkotaan, ibukota kecamatan, daerah perintis, hingga desa perbatasan negeri, baik di gugusan pulau kecil ataupun di hutan pedalaman.

Untuk membahas setiap kebijakan yang akan dieksekusi oleh masing-masing departemen yang ada di PT. Telkomsel Indonesia, masing-masing departemen melakukan pembahasannya di dalam ruang *meeting* yang tersedia di setiap gedung kerja yang dimiliki oleh PT. Telkomsel Indonesia yang tersebar di seluruh Indonesia. Masalah yang sering muncul ketika suatu departemen akan melakukan rapat dengan memanfaatkan ruang *meeting* adalah menggunakan prosedur lama dan konvensional yakni dengan hanya menanyakan kepada *receptionist* apakah terdapat ruangan di suatu gedung kerja yang tidak terpakai. Jika terdapat ruangan yang tidak terpakai, maka dapat langsung digunakan oleh suatu departemen tersebut, tetapi jika tidak ada satupun ruangan yang kosong di gedung tersebut maka suatu departemen harus mencari ruangan lain yang tidak terpakai dengan prosedur yang sama di gedung lain, yaitu dengan menanyakannya lagi kepada *receptionist*. Prosedur tersebut sangat mengganggu kinerja dari setiap departemen pada PT. Telkomsel Indonesia karena banyaknya *effort* yang dikeluarkan dan waktu yang dihabiskan hanya untuk menggunakan satu ruang *meeting*.

Solusi yang akan diambil oleh penulis untuk menyelesaikan permasalahan diatas adalah mengembangkan sebuah perangkat lunak yang mampu melakukan manajemen terhadap penggunaan ruang *meeting* yang berbasis web. Solusi ini diambil untuk menekan banyaknya *effort* dan waktu yang terbuang bagi setiap departemen untuk menggunakan sebuah ruang *meeting* karena perangkat lunak yang dikembangkan berbasis web memiliki keunggulan yaitu dapat diakses dimana saja selama *user* dari perangkat lunak tersebut memiliki akses ke jaringan internet.

Pada proses implementasi kode program dari perangkat lunak yang akan dikembangkan penulis akan memanfaatkan sebuah *framework* pengembangan perangkat lunak berbasis *web* yang mendukung bahasa pemrograman *Java*. Bahasa pemrograman *Java* dipilih penulis karena disamping bahasanya yang bersifat *secure* yang memungkinkan penulis untuk mengunduh kode program yang *untrusted* dari internet, bahasa pemrograman *Java* juga bersifat OO(*Object Oriented*). Salah satu *framework* pengembangan perangkat lunak berbasis web

yang mendukung bahasa pemrograman *Java* adalah *PLAY! framework. PLAY! framework* dikembangkan pada tahun 2007 dan dibuka untuk komunitas pada tahun 2009. *PLAY! framework* mengusung konsep *convention over configuration,* yaitu merupakan paradigma desain pengembangan perangkat lunak untuk mengurangi jumlah keputusan yang harus diambil oleh *developers,* meningkatkan kesederhanaan tetapi tidak mengurangi fleksibilitas. Disamping itu *PLAY! framework* juga menganut pola arsitektural *Model-View-Controller*(MVC). Pada penelitian ini kerangka kerja *PLAY!* akan dikombinasikan dengan SDLC *waterfall* sebagai acuan dalam pengembangan aplikasi.

Di akhir penelitian, akan dilakukan pengujian terhadap *usability* aplikasi yang akan dikembangkan dengan menggunakan metode *System Usability Scale* (SUS) untuk mengetahui kualitas aplikasi dari sisi *usability*-nya menurut penilaian *user* aplikasi.

Berdasarkan pemaparan diatas, penulis mengambil judul penelitian "**Pembangunan Aplikasi Manajemen Ruang *Meeting* Pada PT. Telkomsel Indonesia Tbk. Dengan Menggunakan *PLAY! Framework***" dengan harapan perangkat lunak yang akan dibangun dapat menyelesaikan permasalahan manajemen ruang *meeting* yang sedang dihadapi oleh PT. Telkomsel Indonesia Tbk.

1.2 Rumusan masalah

Berdasarkan permasalahan yang diangkat pada bagian latar belakang, maka didapatkan rumusan masalah sebagai berikut :

1. Bagaimana implementasi SDLC *Waterfall* dalam pengembangan perangkat lunak?
2. Bagaimana hasil implementasi kerangka kerja *PLAY!* dalam proses pengembangan perangkat lunak?
3. Bagaimana hasil pengujian *usability* terhadap perangkat lunak yang akan dikembangkan dengan menggunakan metode *System Usability Scale*?

1.3 Tujuan

Menyelesaikan permasalahan manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. dengan mengembangkan sebuah aplikasi manajemen ruang *meeting* yang berbasis web yang dikembangkan dengan menggunakan kerangka kerja *PLAY!* dan hasil pengembangannya akan diuji dengan menggunakan metode pengujian *usability* SUS untuk mengetahui kualitas perangkat lunak yang dikembangkan dari sisi *usability*-nya menurut pendapat *user*.

1.4 Manfaat

Manfaat yang didapat dari penelitian ini adalah :

1. Permasalahan manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. dapat terselesaikan.
2. Penulis dapat mengetahui kualitas dari aplikasi manajemen ruang *meeting* yang akan dikembangkan dengan kerangka kerja *Play!* melalui pengujian *usability* aplikasi.

1.5 Batasan masalah

Agar pembahasan masalah pada penelitian ini tidak semakin meluas, penelitian ini dibatasi pada hal-hal berikut:

1. Studi kasus penelitian ini diangkat dari permasalahan manajemen ruang *meeting* yang ada pada PT. Telkomsel Indonesia Tbk.
2. Aplikasi yang dikembangkan adalah berbasis Web.
3. Aplikasi dikembangkan dalam lingkungan sistem operasi Windows 7.
4. Teknologi pengembangan yang digunakan adalah *PLAY! Framework*.
5. Penelitian ini tidak membahas tentang perbandingan kerangka kerja *PLAY!* dengan kerangka kerja yang lain dalam mendukung sebuah pengembangan perangkat lunak.
6. Penelitian ini tidak membahas tentang perbandingan metode pengujian *usability.*

1.6 Sistematika pembahasan

Sistematika pembahasan dalam skripsi ini adalah sebagai berikut:

BAB I Pendahuluan

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

BAB II Dasar Teori

Memaparkan teori dasar dan teori pendukung yang berhubungan dengan pengembangan aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk.

BAB III Metode Penelitian

Membahas metode penelitian yang digunakan dalam penelitian yang terdiri dari analisa permasalahan, studi literatur dan penyusunan dasar teori, analisis solusi, pengembangan perangkat lunak, analisa hasil penelitian, serta pengambilan kesimpulan dan saran.

BAB IV Analisis dan Perancangan

Melakukan analisis kebutuhan dan perancangan aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk.

BAB V Implementasi

Mengimplementasikan hasil analisis kebutuhan dan perancangan aplikasi manajemen ruang *meeting* pada PT Telkomsel Indonesia Tbk.

BAB VI Pengujian

Menguji aplikasi manajemen ruang *meeting* yang dikembangkan.

BAB VII Kesimpulan dan Saran

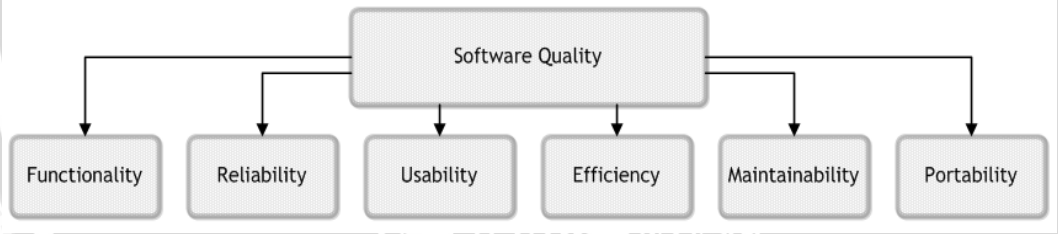
Menarik kesimpulan dari penelitian yang telah dilakukan dan memberikan saran untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Tinjauan pustaka diperoleh dari penelitian yang dilakukan oleh Vivy Kusuma Hertantri pada tahun 2015 yang berjudul "Pembangunan dan Analisis Kualitas Sistem Informasi Ekstrakurikuler Berbasis *Web* di SMA Negeri 1 Purbalingga". Penelitian tersebut membahas tentang bagaimana merancang, mengimplementasikan, dan menguji sistem informasi ekstrakurikuler yang akan dikembangkan.

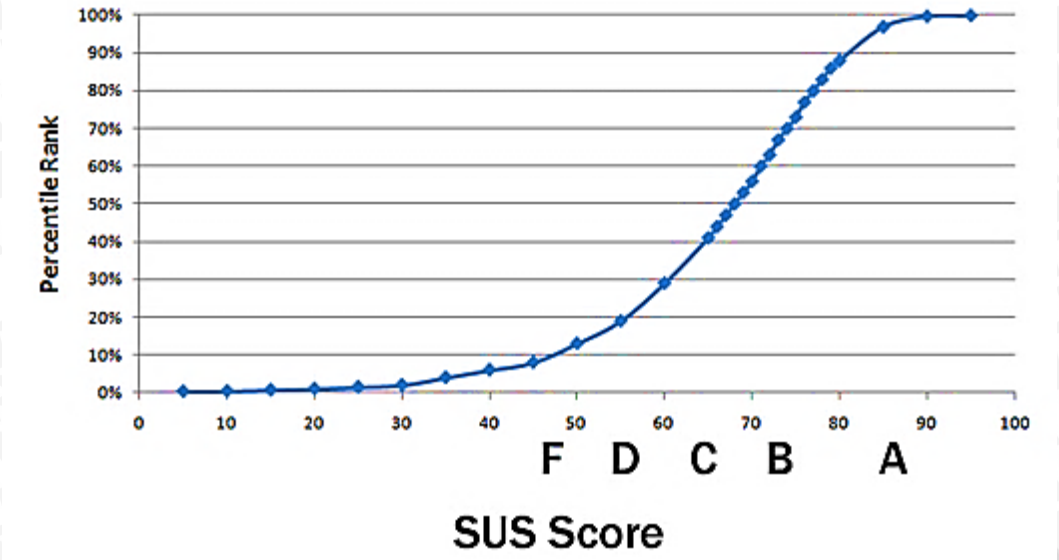
Khusus untuk bab pengujian, penelitian tersebut menitikberatkan kepada keseluruhan pengujian perangkat lunak sesuai dengan standar ISO 9126. Terdapat 6 ukuran kualitas standar ISO 9126 yaitu *Functionality* (kesesuaian perangkat lunak dengan kebutuhan yang dinyatakan maupun kebutuhan implisit), *Reliability* (kemampuan perangkat lunak dalam mempertahankan kinerja dalam suatu kondisi dan jangka waktu yang ditentukan), *Usability* (usaha yang diperlukan oleh *user* dalam menggunakan perangkat lunak), *Efficiency* (hubungan antara tingkat kinerja perangkat lunak dengan jumlah sumber daya yang digunakan), *Maintainability* (usaha yang diperlukan untuk melakukan perubahan tertentu pada perangkat lunak), *Portability* (kemampuan perangkat lunak ketika berada dalam kondisi lingkungan yang berbeda). Gambar 2.1 berikut menunjukkan aspek-aspek yang mempengaruhi kualitas suatu perangkat lunak berdasarkan ISO 9126:



Gambar 2.1 Kualitas Perangkat Lunak

Sumber : Vivy Kusuma Hertantry (2015)

Pada pengujian aspek *Usability*, penelitian tersebut memanfaatkan metode *System Usability Scale* (SUS). *System Usability Scale* (SUS) adalah suatu skala *Linkert* berbasis kuisioner yang digunakan untuk menilai *usability* dari suatu sistem yang dikembangkan oleh John Brooke pada tahun 80an. SUS memberikan hasil skor diantara 0-100 dimana 100 mengindikasikan kualitas *usability* terbaik (meiert, 2009). Pengambilan sampel penelitian mengacu pada teknik yang disampaikan oleh Nielson (2012) dengan melibatkan 15 calon *user* aplikasi. Setelah diketahui rata-rata skor SUS-nya, selanjutnya dibandingan dengan *range* yang dikemukakan oleh Souro (2011). Gambar 2.2 berikut ini menunjukkan *range* skor SUS yang dikemukakan oleh Souro (2011):



Gambar 2.2 *Range* Skor SUS

Sumber : Vivy Kusuma Hertantry (2015)

2.2 Definisi Manajemen

Definisi manajemen adalah proses pengelolaan dari tujuan yang efektif, diselenggarakan dan diawasi. Menurut George R. Terru fungsi manajemen adalah sebagai berikut :

1. *Planning* atau perencanaan.
2. *Organizing* atau pengorganisasian.
3. *Actuating* atau penggerakkan.
4. *Controlling* atau pengendalian.

2.3 Rekayasa Perangkat Lunak

Rekayasa Perangkat Lunak adalah pembuatan dan penggunaan prinsip-prinsip keahlian teknik untuk mendapatkan perangkat lunak yang ekonomis yang handal dan bekerja secara efisien pada mesin yang sesungguhnya (Roger S. Pressman, 2010). Rekayasa perangkat lunak mendirikan suatu pondasi untuk proses perangkat lunak yang lengkap dengan mengindentifikasi sejumlah aktifitas kerangka kerja yang berlaku untuk semua proyek perangkat lunak, terlepas dari ukuran kompleksitas. Beberapa pondasi dalam Rekayasa Perangkat Lunak:

1. Fokus pada kualitas (***A Quality Focus***)

   Pendekatan teknik apapun harus bersandar pada komitmen organisasi terhadap suatu mutu. Total kualitas manajemen dan filosofi yang sama mendorong budaya perbaikan proses yang berkesinambungan dan budaya inilah yang pada akhirnya mengarah pada pengembangan pendekatan yang semakin dewasa untuk Rekayasa Perangkat Lunak (RPL).

2. Proses (***Process***)

   Dasar untuk Rekayasa Perangkat Lunak (RPL) adalah lapisan proses. Proses pada Rekayasa Perangkat Lunak (RPL) adalah perekat yang memegang teknologi lapisan (*layer*) bersama-sama dan memungkinkan pengembangan perangkat lunak yang rasional dan tepat waktu. Proses mendefinisikan sebuah kerangka kerja untuk suatu set *key process areas* (KPAs) yang harus ditetapkan untuk penyampaian (*delivery*) yang efektif dari teknologi Rekayasa Perangkat Lunak (RPL). *Key process areas* membentuk dasar kontrol manajemen proyek perangkat lunak dan menetapkan konteks metode-metode teknis mana yang diterapkan, produk kerja (model dokumen, data, laporan, *form*, dll) yang diproduksi, *milestone* yang ditetapkan, kualitas yang terjamin dan perubahan yang dikelola dengan baik.

3. Metode (***Method***)

   Metode Rekayasa Perangkat Lunak (RPL) menyediakan teknis "bagaimana" untuk membangun perangkat lunak. Metode mencakup tugas yang meliputi analisis kebutuhan (*Requirement Analysis*), Perancangan (*Design*), Program konstruksi (*Construction Program*), pengujian (*Testing*), dan Pemeliharaan (*Maintenance*).

4. Alat Bantu (***Tools***)

   Alat bantu otomatis atau semi-otomatis menyediakan dukungan untuk proses dan metode. Ketika alat-alat diintegrasikan sehingga informasi yang dibuat oleh salah satu alat dapat digunakan oleh alat lainnya, sebuah sistem untuk mendukung perangkat lunak, yang disebut *Computer-Aided Software Engineering* (CASE), CASE menggabungkan *software*, *hardware*, dan *database* (sebuah *repository* berisi informasi penting tentang analisis, rancangan, program konstruksi, dan pengujian).

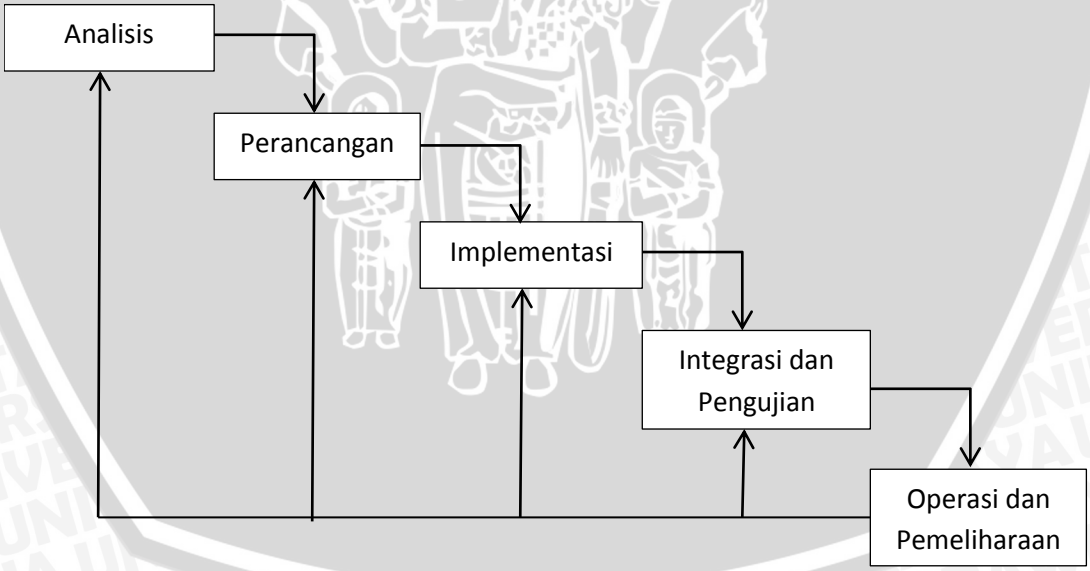
2.4 Model Proses Waterfall

Model proses *waterfall* adalah sebuah contoh dari proses perancangan dimana semua proses kegiatan harus terlebih dahulu direncanakan dan dijadwalkan sebelum dikerjakan (Sommerville,2011).

Model proses waterfall adalah proses pengembangan perangkat lunak dengan tahap-tahap utama dari model ini memetakan kegiatan-kegiatan pengembangan dasar (Sommerville,2003) yaitu:

1. Analisis dan definisi persyaratan. Pelayanan, batasan dan tujuan sistem ditentukan melalui konsultasi dengan pengguna sistem. Persyaratan ini kemudian didefinisikan secara rinci dan berfungsi sebagai spesifikasi sistem.
2. Perancangan sistem perangkat lunak. Proses perancangan sistem membagi persyaratan dalam sistem perangkat keras atau perangkat lunak. Kegiatan ini menentukan arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan deskripsi abstraksi sistem perangkat lunak yang mendasar dan hubungan-hubungannya.
3. Implementasi. Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program..
4. Integrasi dan pengujian sistem. Unit program atau program individual diintegrasikan dan diuji sebagai sistem yang lengkap untuk menjamin bahwa persyaratan sistem telah dipenuhi. Setelah pengujian sistem, perangkat lunak dikirim ke pelanggan.
5. Operasi dan pemeliharaan. Ini merupakan suatu fase siklus hidup yang paling lama. Sistem di-*install* dan dipakai. Pemeliharaan mencakup koreksi dari berbagai *error* yang tidak ditemukan pada tahap-tahap terdahulu, perbaikan atas implementasi unit sistem dan pengembangan pelayanan sistem, sementara persyaratan-persyaratan baru ditambahkan.

Gambar 2.3 berikut menunjukkan model proses *waterfall* menurut Ian Sommerville:



Gambar 2.3 Model Proses Waterfall

Sumber: Diadaptasi dari Ian Sommerville (2003)

2.5 *Unified Modelling Languange* (UML)

Menurut (Adi Nugroho,2005). "Dalam kerangka spesifikasi, *Unified Modelling Languange* (UML) menyediakan model-model yang tepat, tidak mendua arti *(ambigu)* serta lengkap. Secara khusus, *Unified Modelling Languange (*UML*)* menspesifikasikan langkah-langkah penting dalam pengambilan keputusan analisis, perancangan serta implementasi dalam sistem yang sangat bernuansa perangkat lunak *(software intensive system)*. Dalam hal ini, *Unified Modelling Languange* (UML) bukanlah merupakan bahasa pemrograman tetapi model-model yang tercipta berhubungan langsung dengan berbagai macam bahasa pemrograman, sehingga adalah mungkin melakukan pemetaan *(mapping)* langsung dari model-model yang dibuat dengan *Unified Modelling Languange* (UML) dengan bahasa-bahasa pemrograman berorientasi obyek, seperti *Java, Borland Delphi, Visual Basic, C++,* dan lain-lain. Pemetaan *(mapping) Unified Modelling Languange* (UML) bersifat dua arah yaitu:

a. Generasi kode bahasa pemrograman tertentu dari *Unified Modelling Languange* (UML) *forward engineering.*

b. Generasi kode belum sesuai dengan kebutuhan dan harapan pengguna, pengembang dapat melakukan langkah balik bersifat *iterative* dari implementasi ke *Unified Modelling Languange* (UML) hingga didapat sistem/piranti lunak yang sesuai dengan harapan pengguna dan pengembang".

Dalam membangun model visual dari sistem yang akan dikembangkan, terdapat beberapa diagram yang dibutuhkan seperti dibawah ini:

2.5.1 *Use Case Diagram*

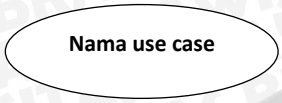
Use Case adalah teknik untuk menekan persyaratan fungsional sebuah sistem. *Use Case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memeberi sebuah narasi tentang bagaimana sistem tersebut digunakan. *Use Case Diagram* menampilkan aktor mana yang menggunakan *use case* mana, *use case* mana yang memasukkan *use case* lain dan hubungan antara aktor dan *use case* (Martin Fowler,2005).

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case.*

- Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tetapi aktor belum tentu merupakan orang.
- *Use case* merupakan fungsionalitas yang disediakan oleh sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Gambar 2.1 berikut menunjukkan simbol-simbol yang ada pada *use case diagram*:

Tabel 2.1 Simbol-simbol *Use case diagram*

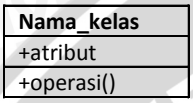
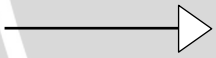
Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case.*
Aktor/*actor*  **Nama aktor**	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tetapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Asosiasi/*association* _____	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi/*extend* <<extend>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu; mirip dengan *inharitance* pada pemrograman berorientasi obyek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari yang lainnya.
Menggunakan/*include* <<include>> 	*Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.

Sumber: Diadaptasi dari Rosa A.S. dan M. Shalahuddin (2014)

2.5.2 Class Diagram

Class diagram (diagram kelas) merupakan himpunan dari objek-objek yang sejenis. Sebuah objek memiliki keadaan sesaat (*state*) dan perilaku (*behavior*). *State* sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam *attribute/properties.* Sedangkan perilaku suatu objek mendefinisikan bagaimana sebuah objek bertindak/beraksi dan memberikan reaksi (Munawar, 2005). Gambar 2.2 berikut menunjukkan simbol-simbol yang ada pada *class diagram*:

Tabel 2.2 Simbol-simbol *Class diagram*

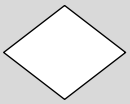
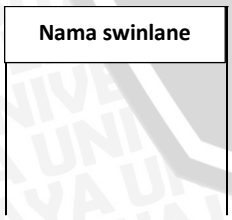
Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem.
Asosiasi/*association* 	Relasi antar kelas dengan makna umum, asosiasi biasanya disertai dengan *multiplicity*.
Asosiasi berarah/*directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan kelas yang lain, asosiasi biasanya disertai dengan *multiplicity*.
Generalisasi/*generalization* 	Relasi antar kelas dengan makna generalisasi-spesialisasi.
Kebergantungan/*dependency* 	Kebergantungan antar kelas.
Agregasi/*aggregation* 	Relasi antar kelas dengan makna semua-sebagian.

Sumber: Diadaptasi dari Rosa A.S. dan M. Shalahuddin (2014)

2.5.3 Activity Diagram

Activity diagram (diagram aktivitas) adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, *activity diagram* memainkan peran mirip diagram alir, tetapi perbedaan prinsip antara notasi diagram alir adalah *activity diagram* mendukung *behavior paralel. Node* pada sebuah *activity diagram* disebut sebagai *action,* sehingga diagram tersebut menampilkan sebuah *activity* yang tersusun dari *action* (Martin Fowler, 2005). Gambar 2.3 berikut menunjukkan simbol-simbol yang ada pada *activity diagram*:

Tabel 2.3 Simbol-simbol *Activity diagram*

Simbol	Deskripsi
Status awal 	Status awal aktifitas sistem, sebuah *activity diagram* memiliki sebuah status awal.
Aktifitas 	Aktifitas yang dilakukan sistem, aktifitas biasanya diawali dengan kata kerja.
Percabangan/*decision* 	Asosiasi percabangan dimana jika terdapat pilihan aktifitas lebih dari satu.
Penggabungan/*join* 	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir.
Swimlane 	Memisahkan organisasi bisnis yang bertanggungjawab terhadap aktifitas yang terjadi.

Sumber: Diadaptasi dari Rosa A.S. dan M. Shalahuddin (2014)

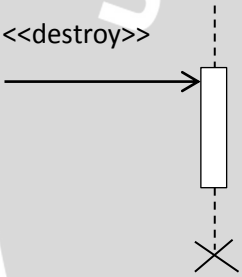
2.5.4 Sequences Diagram

Sequence diagram adalah grafik dua dimensi dimana obyek yang ditunjukkan dalam dimensi horizontal, sedangkan *lifeline* ditunjukkan dalam dimensi vertikal. Gambar 2.4 berikut menunjukkan simbol-simbol yang ada pada *sequence diagram*:

Tabel 2.4 Simbol-simbol *Sequence* diagram

Simbol	Deskripsi
Aktor  **Nama aktor** Atau Nama aktor	Orang, proses, atau sistem lainyang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol aktor adalah gambar orang, tetapi aktor belum tentu merupakan orang;biasanya dinyatakan dengan kata benda di awal frase nama aktor
Garis hidup/*lifeline* 	Menyatakan kehidupan suatu objek
Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi,semua yang berhubungan dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya
Pesan tipe *create* <<create>> 	Menyatakan suatu objek membuat objek yang lain; arah panah menuju ke objek yang dibuat

Tabel 2.4 Simbol-simbol *Sequence* diagram (lanjutan)

Simbol	Deskripsi
Pesan tipe *call* 1 : nama_metode() 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri
Pesan tipe *send* 1 : masukan 	Menyatakan suatu objek mengirimkan data/informasi ke objek lainnya
Pesan tipe *return* 1 : keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain; arah panah menuju ke objek yang diakhiri

Sumber: Diadaptasi dari Rosa A.S. dan M. Shalahuddin (2014)

2.6 Basis data (MySql)

MySql (baca: mai-se-kyu-el) merupakan *software* yang tergolong sebagai DBMS (*Database Management System*) yang bersifat *open source. Open source* menyatakan bahwa *software* ini di lengkapi dengan *source code* (kode yang dipakai untuk membuat MySql), selain tentu saja bentuk sistem operasi, dab bisa diperoleh dengan cara men-*download* (mengunduh) di internet secara gratis.

MySql awalnya dikembangkan oleh konsultan bernama TcX yang berlokasi di Swedia. Sekarang pengembangannya berada dibawah naungan perusahaan MySql AB. Adapun *software* dapat diunduh di www.mysql.com.

MySql dalam operasi klien-server memanfaatkan daemon MySql di sisi server dan berbagai macam program dan pustaka yang berjalan di sisi klien. MySql mampu menangani data yang cukup besar, perusahaan awal pengembang MySql yaitu TcX mengaku menyimpan data dalam 40 *database,* 10.000 tabel dan sekitar 7 juta baris, totalnya kurang lebih 100 *Gigabytes* data.

2.7 Metode *System Usability Scale* (SUS)

System Usability Scale (SUS) adalah sebuah *tools* pengujian *usability* dalam bentuk kuisioner yang dikembangkan oleh John Brooke pada tahun 1986. *Tools* ini terdiri dari 10 pertanyaan dengan masing-masing pertaanyan memiliki 5 opsi jawaban bagi responden. Kelima opsi jawaban tersebut dimulai dari skala yang menunjukkan *"Strongly Disagree"* (skala nomor 1) hingga skala yang menunjukkan "*Strongly Agree*" (skala nomor 5). Untuk pertanyaan yang sama sekali tidak dipahami oleh *user* aplikasi (responden), maka dapat dipilih skala nomor 3. Gambar 2.4 berikut ini menunjukkan *form* dari SUS yang berisi 10 pertanyaan tersebut:

Participant ID: _____ Site: _____ Date: ___/___/___

System Usability Scale

Instructions: For each of the following statements, mark one box that best describes your reactions to the website *today*.

		Strongly Disagree				Strongly Agree
1.	I think that I would like to use this website frequently.	☐	☐	☐	☐	☐
2.	I found this website unnecessarily complex.	☐	☐	☐	☐	☐
3.	I thought this website was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need assistance to be able to use this website.	☐	☐	☐	☐	☐
5.	I found the various functions in this website were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this website.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this website very quickly.	☐	☐	☐	☐	☐
8.	I found this website very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using this website.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this website.	☐	☐	☐	☐	☐

Please provide any comments about this website:

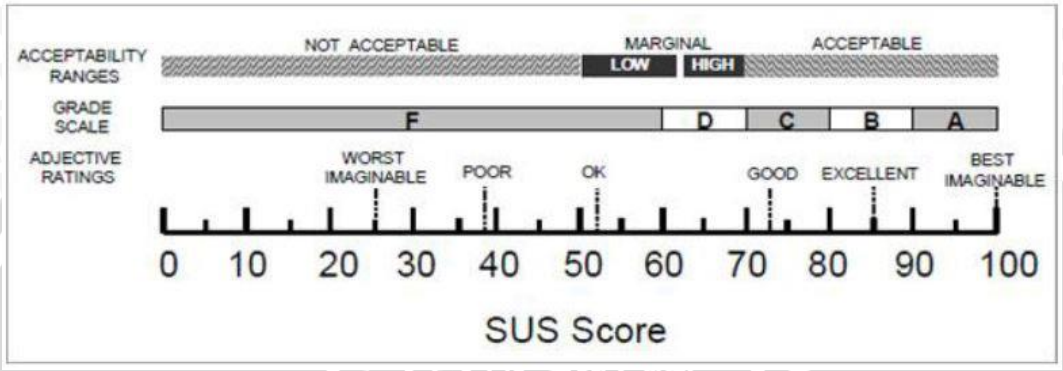
Gambar 2.4 *Form System Usability Scale*

Sumber : John Brooke(1986)

SUS memiliki strategi *scoring* yang rumit dimana kesepuluh item pertanyaan tersebut dibagi kedalam 2 kelompok yaitu kelompok pertanyaan positif (1,3,5,7,9) dan kelompok pertanyaan negatif (2,4,6,8,10). Kontribusi dari masing-masing item adalah 0-4 poin dimana, untuk item pertanyaan positif kontribusi skornya dihitung dengan mengurangi posisi skala pada skala *Linkert* dengan 1 poin sedangkan untuk item pertanyaan negatif kontribusi skornya dihitung

dengan mengurangi poin 5 dengan posisi dari skala yang ditunjuk oleh skala *Linkert*. Tidak hanya sampai disitu, setelah semua nilai diperoleh, kemudian dijumlahkan dan dikali dengan 2,5 untuk mendapatkan *range* skor antara 0-100.

Bangor, Kortum dan Milner (2008,2009) dalam "*Determining What Individual SUS Score Mean: Adding an Adjective Rating Scale*", telah mengumpulkan data dari penggunaan SUS selama lebih dari 1 dekade dengan variasi sistem dan teknologi yang berbeda sampai terkumpul hingga lebih dari 3500 hasil *scaling* dengan SUS. Pengumpulan data ini bertujuan untuk mengevaluasi hasil *scaling* untuk menentukan *adjective* seperti "Good", "Poor", "Excellent". Gambar 2.5 berikut menunjukkan hasil penelitian tersebut:



Gambar 2.5 Kriteria Nilai SUS

Sumber : Bangor, Kortum, dan Miller (2008,2009)

berdasarkan penelitian Sauro's (2011), Sauro's menyimpulkan bahwa SUS reliable (respon dari *client* konsisten berdasarkan item *scale,* dan SUS mampu menunjukkan perbedaan pada sampel pengujian yang lebih kecil dibandingkan kuisioner yang lain), SUS valid (SUS mengukur yang seharusnya diukur), SUS bukan *diagnostic* (SUS tidak menunjukkan apa yang membuat sistem yang dikembangkan *usable* atau tidak), Skor dari SUS bukan sebuah prosentase (meskipun mengembalikan nilai 0-100 untuk, tetapi penguji harus melihat rangking dari SUS untuk mengetahui *Adjective* dari nilai tersebut), SUS mengukur *learnability* dan *usability*, Skor dari SUS memeiliki korelasi sederhana dengan *task performance*

2.8 Java

Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Meskipun pada awal saat dirilis sekitar tahun 90-an, Java dirancang untuk digunakan pada sistem-sistem kecil seperti TV kabel atau *home theater,* sekarang sudah merambah keseluruh aplikasi pada komputer bahkan beberapa institusi pendidikan beralih dari pemrograman dengan *Pascal* dan C++ ke pemrograman Java (M. Shalahudin, 2009).

Dalam situs resminya (http://www.Java.com) diterangkan bahwa:

"*Java allows yout to play online games, chat with people around the world, calculate your mortgage interest, and view images in 3D, just to name a few. It's also integral to the internet applications and other e-business solutions that are the foundation of corporate computing*"

Java memungkinkan anda untuk bermain game-game online, *chatting* dengan orang diseluruh dunia, menghitung bunga hipotek anda, dan melihat gambar dalam 3D, hanya dalam beberapa langkah. Ini juga bagian integral dari aplikasi dan solusi *e-business* yang merupakan dasar dari komputasi perusahaan.

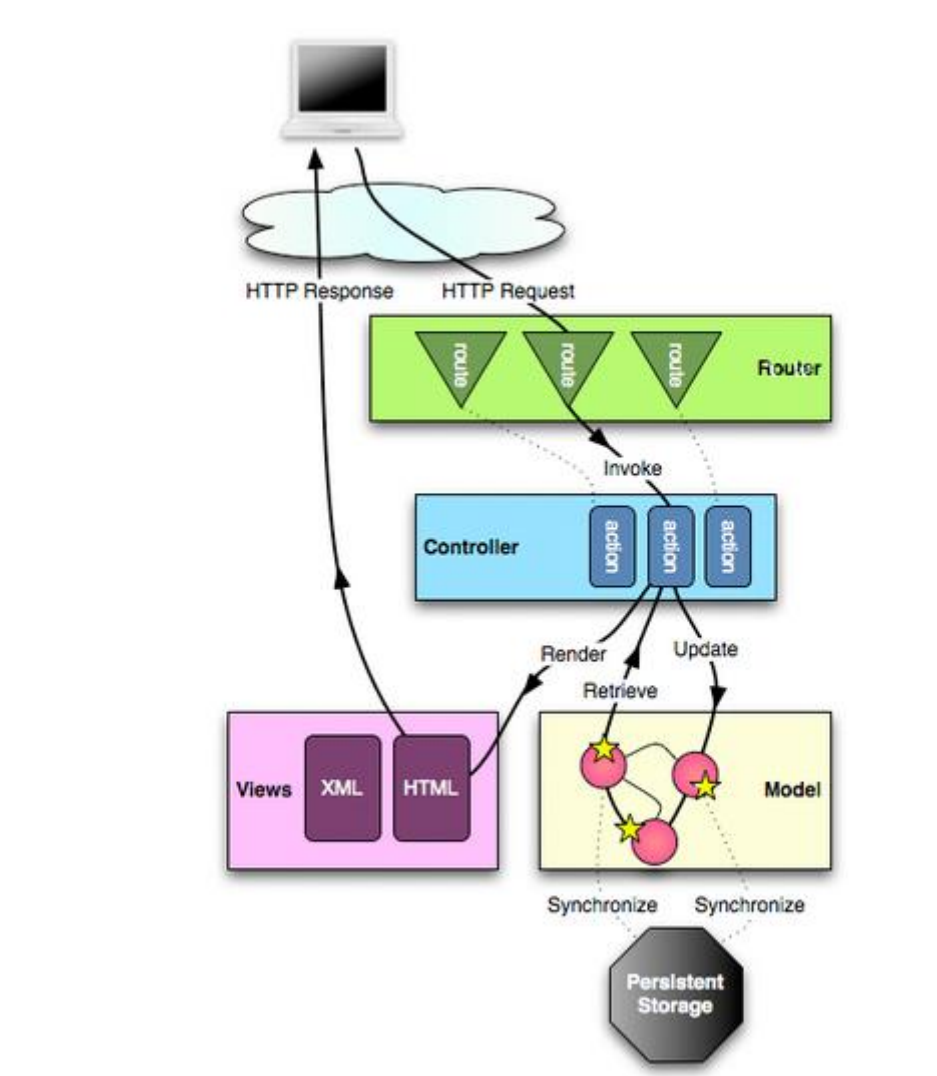
Alasan utama pembentukan pemrograman Java adalah untuk membuat aplikasi-aplikasi yang dapat diletakkan di berbagai macam perangkat elektronik, seperti *microwave oven* dan *remote control,* sehingga Java harus bersifat *portable* atau yang sering disebut dengan *platform independent* (tidak tergantung pada sistem operasi). Itulah yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah '*write once, run everywhere*', yang berarti kode program Java hanya ditulis sekali, namun dapat dijalankan di bawah *platform* manapun, tanpa harus melakukan perubahan kode program (E.M. Budi, 2003).

2.9 PLAY! Framework

Kerangka kerja *PLAY!* menggunakan bahasa pemrograman *Java* dan mengikuti pola arsitektural MVC *(Model View Controller)* yang diaplikasikan ke arsitektur Web. Pola tersebut membagi aplikasi kedalam 2 lapisan: lapisan Presentasi dan lapisan Model. Lapisan Presentasi dibagi menjadi lapisan *View* dan lapisan *Controller*. Lapisan Model adalah representasi dari *domain-specific* dari informasi dimana aplikasi dioperasikan. Logika domain berarti sebuah baris data (contohnya mengkalkulasi jika sekarang adalah hari ulang tahun *user,* atau total pajak, dan biaya pelayaran untuk sebuah barang belanja). Banyak aplikasi menggunakan mekanisme ruang penyimpanan yang *persistent* seperti *database* untuk menyimpan data. MVC tidak secara spesifik menyebut lapisan akses data karena itu dipahami berada di lapisan bawah, atau telah di enkapsulasi oleh lapisan Model. Lapisan *View* me-*render* model kedalam bentuk yang sesuai untuk interaksi, umumnya adalah *user interface.* Banyak *View* dapat muncul untuk sebuah model tunggal, demi tujuan yang berbeda. Dalam aplikasi web, *View* biasanya di-*render* dalam format web seperti HTML, XML, atau JSON. Namun terdapat beberapa kasus dimana *View* dapat diekspresikan kedalam bentuk biner, misalnya secara dinamis me-*render chart diagram*. Lapisan *Controller* bertugas merespon *events* (umumnya aksi *user*) dan memprosesnya, dan mungkin juga meminta perubahan pada model. Dalam aplikasi web, *event* biasanya adalah sebuah *request* HTTP: sebuah *Controller* mendengarkan *request* HTTP, meng-ekstrak data yang relevan dari sebuah *event*, seperti parameter query *string, request header..* dan mengaplikasikan perubahan pada objek model. Seluruh *request* HTTP melalui alur sebagai berikut (Zenexity,2007):

- Sebuah *request* HTTP diterima oleh kerangka kerja *Play!.*
- Komponen Router mencoba mencari rute yang paling spesifik yang dapat menerima *request* tersebut, kemudian *action method* yang sesuai dilibatkan.
- Kode aplikasi di eksekusi.
- Jika *View* yang kompleks perlu untuk di-*generated*, sebuah *file tamplate* akan di *render.*
- Hasil dari *action method* (kode respon HTTP, konten) kemudian di tulis sebagai respon HTTP.

Gambar 2.6 dibawah ini adalah rangkuman dari alur *request* HTTP:



Gambar 2.5 Alur Request HTTP

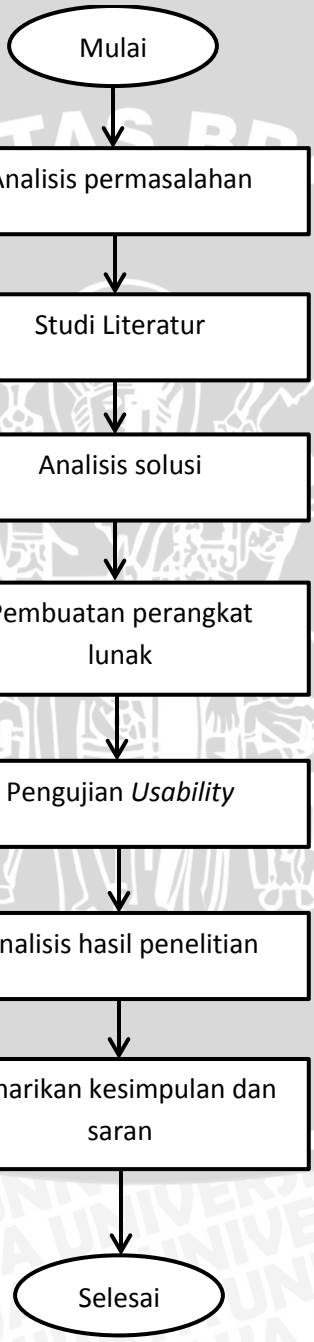
Sumber: Zenexity (2007)

Kerangka kerja *PLAY!* menggunakan bahasa pemrograman dasar *Java*, sehingga untuk implementasi kode program dapat dilakukan di dalam sebuah *IDE*. Kerangka kerja *PLAY!* mendukung beberapa *IDE* yang banyak digunakan saat ini diantaranya *Netbeans, Eclipse,* dan *IntelliJ IDEA.*

Pada lapisan *View* sebagai sarana komunikasi antara bahasa *HTML* dan *Java,* diperlukan bahasa lain yang dikenali oleh *browser* yang mampu menjadi jembatan penghubung antara kedua bahasa utama yang digunakan oleh kerangka kerja *Play!* tersebut. Bahasa yang dipilih oleh pengembang kerangka kerja *Play!* adalah bahasa *Groovy* (*grails*). Bahasa ini dipilih karena dikenali oleh *browser* didalam suatu *file .html* sebagai logika bukan sebagai *statement*, sehingga selain sebagai penghubung bahasa *Java* yang *server-side* dengan bahasa *HTML* yang *client-side*, Bahasa *Groovy* juga dapat dimanfaatkan untuk membangun logika di dalam sebuah *file .html* yang dikenali dan diperbolehkan oleh *browser.*

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan mangenai prosedur dan kegiatan-kegiatan yang dilakukan dalam pengerjaan skripsi. Diawali dengan analisa permasalahan, studi literatur, analisa solusi, pembuatan perangkat lunak, pengujian *usability*, analisa hasil penelitian, dan penarikan kesimpulan dan saran. Gambar 3.1 berikut adalah diagram metode penelitian:



Gambar 3.1 Diagram Metode Penelitian

3.1 Analisis Permasalahan

Analisis permasalahan adalah tahap melakukan analisis terhadap permasalahan yang diangkat dalam penelitian ini. Studi kasus yang diangkat oleh penulis adalah permasalahan yang terdapat pada PT. Telkomsel Indonesia Tbk. yakni terkait dengan pengelolaan penggunaan ruang *meeting* pada setiap gedung kerja yang dimiliki oleh PT. Telkomsel Indonesia Tbk. yang tersebar diseluruh Indonesia. Proses analisis permasalahan akan diawali dengan menyususun daftar pertanyaan seputar pengelolaan ruang *meeting* dan menjadwalkan sebuah wawancara dengan PT. Telkomsel Indonesia Tbk. untuk menggali lebih dalam mengenai permasalahan yang sedang dihadapi oleh PT. Telkomsel Indonesia Tbk. dalam melakukan pengelolaan terhadap penggunaan ruang *meeting.* Proses wawancara dijadwalkan selama 2 hari untuk mengantisipasi adanya kendala teknis yang menyebabkan tidak dapat terselesaikannya proses wawancara selama 1 hari. Dari hasil wawancara ini diharapkan akan dapat diperoleh informasi detil mengenai permasalahan pengelolaan ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Informasi inilah yang akan dijadikan dasar oleh penulis dalam mengumpulkan data kajian pustaka dan dasar teori sebagai dasar pengetahuan penulis dalam melaksanakan penelitian ini.

3.2 Studi Literatur

Studi literatur merupakan penelusuran pengetahuan dalam rangka menentukan data kajian pustaka dan menyusun dasar teori yang digunakan sebagai dasar pengetahuan dalam penelitian ini. Penelusuran pengetahuan dapat bersumber dari buku, internet, dan jurnal yang berkaitan dengan informasi hasil analisis permasalahan yang dilakukan sebelumnya. Teori yang nantinya akan digunakan oleh penulis sebagai dasar pengetahuan melaksanakan penelitian ini diantaranya adalah jurnal yang berkaitan dengan pengujian *usability* sebagai tinjauan pustaka dalam penelitian, definisi tentang manajemen, informasi mengenai ruang *meeting*, pengertian apa itu RPL, Informasi mengenai jenis *System Development Life* Cycle (SDLC) yang akan digunakan dalam penelitian, pengetahuan tentang diagram *Unified Modelling Language* (*Use case diagram*, *Class diagram*, *Activity diagram*, *Sequence diagram*), basis data (MySql), *Java*, dan tentu saja *PLAY! Framework* sebagai kerangka kerja pengembangan aplikasi manajemen ruang *meeting*.

3.3 Analisis Solusi

Analisis solusi adalah tahap analisis terhadap hasil analisis permasalahan dan hasil pembelajaran terhadap literatur yang telah dikumpulkan dalam rangka memperoleh solusi guna menyelesaikan permasalahan yang ada. Tahap analisis solusi terbagi kedalam 2 fase yaitu:

1. Fase pembelajaran

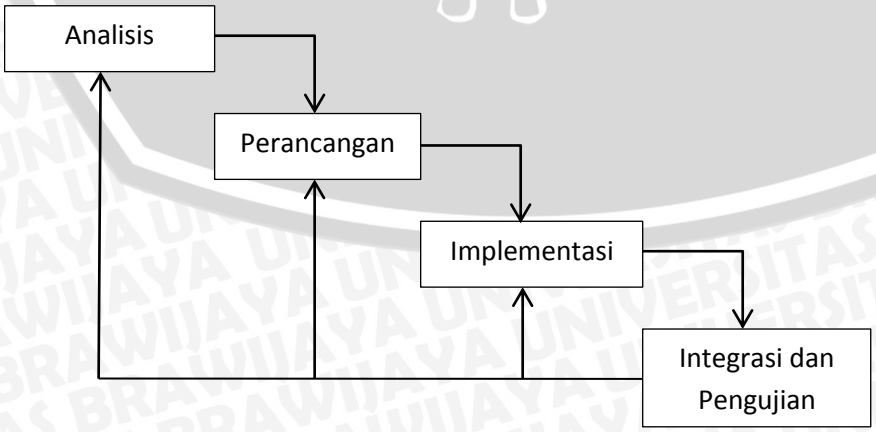
   Dalam fase ini akan dilakukan pengkajian ulang dari hasil analisis permasalahan dan studi literatur yang telah dikumpulkan sebagai dasar penulis menyampaikan pendapat kepada *client* dalam hal ini adalah PT. Telkomsel Indonesia Tbk. mengenai solusi permasalahan yang diambil penulis.

2. Fase pengambilan keputusan

   Dalam fase ini penulis akan melakukan presentasi untuk mengemukaan solusi yang diambil penulis dalam menyelesaikan permasalahan pengelolaan penggunaan ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Setelah selesai dilaksanakan presentasi, akan dilanjutkan dengan diskusi antara penulis dan *client* guna memperoleh kesepakatan mengenai teknologi dasar, teknologi pengembangan perangkat lunak, dan jadwal pelaksanaan kegiatan pengembangan perangkat lunak.

3.4 Pembuatan Perangkat Lunak

Tahap pembuatan perangkat lunak adalah tahap pelaksanaan kegiatan pengembangan perangkat lunak yang terbagi kedalam beberapa tahap sesuai dengan model pengembangan yang diacu oleh penulis yaitu model pengembangan *waterfall* berdasarkan hasil pada tahap analisa solusi. Model pengembangan *waterfall* adalah salah satu model pengembangan perangkat lunak yang proses eksekusinya adalah linier. Maksud dari linier disini adalah pengerjaanya runut dari satu tahap ke tahap selanjutnya, tidak dikerjaan secara paralel. Dalam proses eksekusinya, model pengembangan ini tidak mengijinkan seorang *developer* kembali ke tahap-tahap sebelumnya sebelum seluruh tahap dalam model pengembangan *waterfall* ini telah selesai dilaksanakan, jadi sebelum masuk ke pengerjaan tahap selanjutnya seorang *developer* harus dapat memastikan bahwa suatu tahap yang dikerjakan saat itu telah selesai sehingga tidak akan ada lagi perubahan informasi pada pengerjaan di tahap-tahap selanjutnya. Gambar 3.2 berikut merepresentasikan model pengemangan *waterfall* yang diacu oleh penulis:



Gambar 3.2 Model Proses *Waterfall* Penelitian

Berdasarkan gambar 3.2 diatas, terlihat bahwa model proses *waterfall* yang diacu penulis tersusun atas tahap-tahap yaitu Analisis, Perancangan, Implementasi, dan Integrasi dan pengujian. Dalam model proses tersebut tidak terdapat tahap operasi dan pemeliharaan. Tahap tersebut sengaja tidak penulis sertakan karena dalam peneltian ini tahap operasi dan pemeliharaan tersebut tidak dilakukan.

Berdasarkan gambar 3.2 pula, berikut detail kegiatan yang akan dilakukan penulis pada masing-masing tahap:

3.4.1 Analisis

Pada tahap ini dilakukan analisa terhadap kebutuhan sistem manajemen ruang *meeting* yang akan dikembangkan sebagai dasar pengembangan sistem tersebut. Proses analisa kebutuhan tersebut dilakukan melalui wawancara dengan departemen *General Affair* pada PT. Telkomsel Indonesia Tbk. yang berwenang langsung terhadap pengelolaan *ruang meeting* untuk mengumpulkan data-data yang dibutuhkan dalam pengembangan aplikasi manajemen ruang *meeting* dimana data-data tersebut akan digunakan untuk menyusun diagram analisis kebutuhan sistem yang terdiri dari 6 bagian yaitu:

1. Identifikasi aktor

   Pada tahap ini dilakukan pembahasan terkait siapa saja aktor yang nantinya akan terlibat di dalam aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. dan tugas dari masing-masing aktor yang tridentifikasi.

2. Analisis kebutuhan fungsional

   Pada tahap analisis kebutuhan fungsional sistem akan dilakukan analisis untuk menentukan apa saja yang dapat dilakukan oleh sistem. Dalam analisa ini diperlukan pengetahuan khusus di bidang pengembangan perangkat lunak agar hasil analisa yang diperoleh menjadi optimal.

3. Analisis kebutuhan non-fungsional

   Kebutuhan non-fungsional diantaranya adalah system *performance, system security,* dll. Dalam melakukan analisa kebutuhan non-fungsional akan diintegrasikan dengan hasil dari analisa kebutuhan fungsional agar nantinya tidak terjadi kesalahan yang menyebabkan harus digantinya daftar kebutuhan fungsional.

4. Pemodelan *use case*

   Pada tahap pemodelan *use case* akan dilakukan perancangan diagram *use case* untuk memperjelas hubungan antara aktor yang teridentifikasi dengan sistem aplikasi manajemen ruang *meeting* yang akan dikembangkan.

5. Pembuatan *use case* skenario

   Pada tahap ini akan dijabarkan skenario dari masing-masing *case* pada hasil pemodelan *use case* sebagai dasar perancangan diagram sekuensial.

3.4.2 Perancangan

Tahap perancangan akan dilakukan ketika semua kebutuhan sistem telah diperoleh melalui tahap analisis sistem. Pada tahap ini akan dibangun beberapa diagram perancangan sistem diantaranya:

1. Diagram sekuensial (*Sequence diagram*)

   Pada tahap ini akan dilakukan perancangan diagram sekuensial untuk merepresentasikan alur sebuah *case* pada diagram *use case* berdasarkan waktu yang dilihat dari sudut pandang aktor.

2. Diagram kelas (*Class diagram*)

   Pada tahap ini akan dilakukan perancangan diagram kelas sesuai dengan analisis kebutuhan yang telah dibuat. Dimana masing-masing kelas akan masuk kedalam 3 blok berbeda sesuai fungsinya yaitu blok *Model*, blok *View*, dan blok *Controller.*

3. Diagram Entitas (*Entity Relationship Diagram*)

   Pada tahap perancangan diagram entitas akan dilakukan perancangan/pemodelan Basis data menggunakan *Entity Relationship Diagram* (diagram hubungan entitas). Diagram tersebut mampu menjelaskan hubungan antar entitas yang terdapat didalam sistem.

Selain 3 diagram diatas, pada tahap ini juga akan disertakan perancangan antar muka dari aplikasi manajemen ruang *meeting* yang akan dikembangkan dimana pada tahap perancangan antar muka ini proses perancangannya dilakukan dengan memanfaatkan *tools* online yang banyak tersedia di internet. Hasil akhir perancangan antar muka adalah sebuah *wireframe.*

3.4.3 Implementasi

Pada tahap implementasi akan dilakukan implementasi terhadap hasil perancangan yang telah dilakukan untuk mengembangkan aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Penulis akan memanfaatkan kerangka kerja *PLAY!* sebagai *framework* pengembangan aplikasi manajemen ruang meeting ini. Tahap implementasi sistem ada tiga yaitu:

1. Lingkungan implementasi

   Lingkungan implementasi akan membahas tentang bagaimana kondisi lingkungan implementasi aplikasi manajemen ruang *meeting* dalam hal ini software dan hardwarenya.

2. Implementasi antar muka

   Pada tahap implementasi antar muka akan dikembangkan sebuah antar muka aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. sesuai dengan hasil perancangan antar muka sistem yang telah dilakukan.

3. Implementasi program

   Pada implementasi program akan ditampilkan beberapa *source code* dari blok *model* pada aplikasi manajemen ruang *meeting* yang mewakili fitur pada aplikasi manajemen ruang *meeting. Source code* tersebut akan dapat menjelaskan beberapa proses bisnis dalam aplikasi manajemen ruang *meeting* yang akan dikembangkan.

3.4.4 Pengujian

Pada tahap pengujian ini akan dilakukan pengujian terhadap aplikasi manajemen ruang *meeting* untuk menunjukkan bahwa aplikasi yang dikembangkan dapat bekerja sesuai dengan spesifikasi dari kebutuhan sistem yang telah dibuat yang menjadi dasar dari aplikasi manajemen ruang *meeting.* Pengujian sistem dilakukan dengan 2 metode yaitu *black box testing* dan *white box testing.*

3.5 Pengujian *Usability*

Pada tahap ini akan dilakukan pengujian *usability* dengan metode *System Usability Scale*(SUS) terhadap perangkat lunak yang telah selesai dikembangkan untuk mengetahui tingkat kepuasan dari *client* dan tingkat kualitas dari perangkat lunak yang dikembangkan.

3.6 Analisa Hasil Penelitian

Setelah seluruh proses pengembangan perangkat lunak selesai dilakukan, akan dilakukan Evaluasi terhadap hasil pengujian perangkat lunak yang dikembangkan berdasarkan pada hasil pengujian perangkat lunak menggunakan metode *black box* dan *white box* testing dan pengujian *usability.*

3.7 Penarikan Kesimpulan dan Saran

Penarikan kesimpulan dilakukan setelah seluruh tahapan pengembangan perangkat lunak selesai dilakukan yang meliputi perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak. Kesimpulan diambil dari hasil pengujian dan analisis sistem dan analisa hasil penelitian. Tahap akhir dari penelitian ini adalah penyertaan saran dari penulis agar dapat memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penelitian yang akan dilakukan selanjutnya.

BAB 4 ANALISIS DAN PERANCANGAN

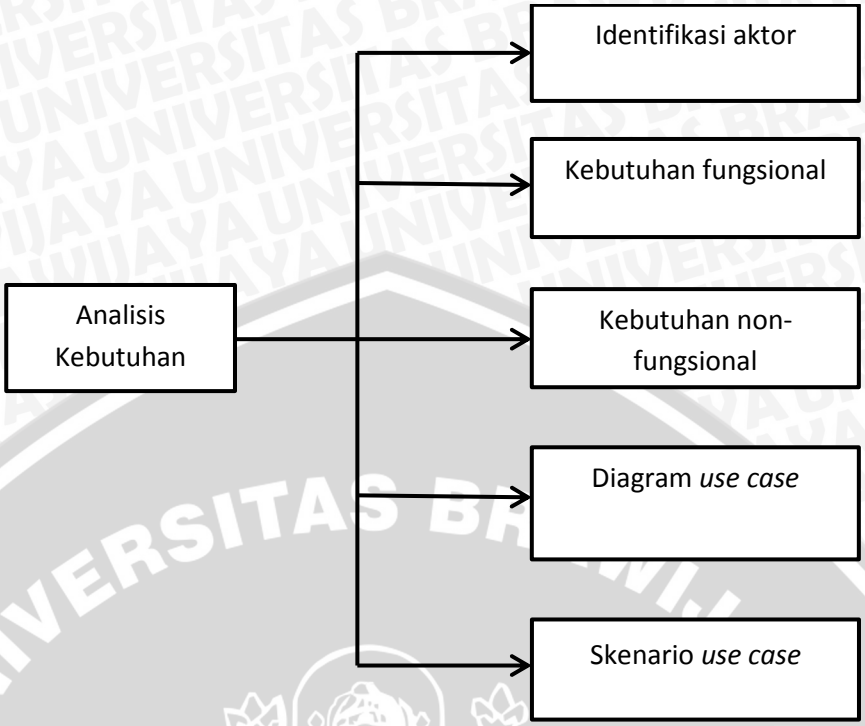
4.1 Deskripsi Umum Sistem

Aplikasi manajemen ruang *meeting* yang akan dikembangkan adalah aplikasi yang diharapkan dapat membantu dalam melakukan proses manajemen penggunaan ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Aplikasi manajemen ruang *meeting* yang akan dikembangkan ini khusus untuk menyelesaikan permasalahan pengelolaan ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Aplikasi manajemen yang akan dikembangkan ini berbasis web sehingga dapat diakses dari mana saja selama *user* aplikasi ini terhubung ke jaringan internet.

Dalam prosesnya setelah disesuaikan dengan kebutuhan dari PT. Telkomsel Indonesia Tbk., aplikasi manajemen ruang *meeting* ini akan membutuhkan 4 orang aktor yaitu *guest, administrator*, *requestor,* dan *receptionist*. *Guest* dalam aplikasi disematkan untuk *user* yang akan melakukan *login* sampai dengan *login* tersebut berhasil. *Administrator* dalam aplikasi ini adalah user yang memiliki kedudukan paling tinggi, user ini akan bertugas mengelola penggunaan ruang *meeting* yang terdaftar di dalam aplikasi mulai dari menerima request pemesanan ruangan, membatalkan request, melihat detail request, menambah ruangan baru, meng-*update* informasi ruangan, termasuk menambahkan user baru kedalam aplikasi manajemen ini juga merupakan kewenangan dari *administrator*. User *administrator* diperankan oleh perwakilan departemen *General Affair* dimana merupakan departemen yang bertugas mengelola ruang *meeting. Requestor* adalah user yang nantinya diberi kewenangan untuk melakukan request ruangan, user ini diperankan oleh perwakilan dari masing-masing departemen sehingga masing-masing departemen dapat melakukan pemesanan ruang *meeting* di dalam aplikasi. *Receptionist* adalah user yang bertugas menyiapkan segala kebutuhan request penggunaan ruang *meeting* yang terdaftar di dalam aplikasi mulai dari fasilitas seperti internet/intranet sampai dengan konsumsi yang dipesan oleh *requestor* ketika membuat sebuah *request* penggunaan ruang *meeting*. Bersama-sama komposisi *task* dari ketiga user tersebut membentuk sebuah sistem manajemen ruang *meeting* yang teraplikasikan pada PT. Telkomsel Indonesia Tbk.

4.2 Analisis Kebutuhan

Sesuai dengan deskripsi pada sub-bab pembuatan perangkat lunak, tahap analisis kebutuhan ini terdiri dari 6 tahap yaitu, identifikasi aktor untuk mengidentifikasi siapa saja aktor yang terlibat didalam sistem, analisis kebutuhan fungsional, analisis kebutuhan non-fungsional, perancangan diagram *use case*, dan pembuatan skenario *use case.* Gambar 4.1 berikut ini merepresentasikan tahap-tahap dalam analisis kebutuhan sistem:



Gambar 4.1 Diagram Analisis Kebutuhan

4.2.1 Identifikasi Aktor

Dalam tahap identifikasi aktor ini dilakukan identifikasi terhadap aktor-aktor yang nantinya akan terlibat didalam aplikasi manajemen ruang *meeting*. Proses identifikasi dilakukan melalui wawancara dengan pihak *General Affair* pada PT. Telkomsel Indonesia Tbk. dalam rangka memenuhi daftar kebutuhan sistem yang akan dikembangkan. Tabel 4.1 berikut mengidentifikasi aktor-aktor yang terlibat didalam sistem beserta peran dari masing-masing aktor tersebut:

Tabel 4.1 Tabel Daftar Aktor

Aktor	Deskripsi
Administrator	Adalah user dengan hak akses tertinggi di dalam sistem, user ini bertugas melakukan pengelolaan baik pengelolaan request, pengelolaan user, pengelolaan gedung, dan pengelolaan ruang *meeting* didalam aplikasi
Requestor	Adalah user yang memiliki wewenang untuk membuat sebuah request penggunaan ruang *meeting*
Receptionist	Adalah user yang bertugas menyiapkan segala kebutuhan terhadap sebuah ruang *meeting* yang telah dipesan
Guest	Adalah user yang merupakan *generalisasi* dari 3 user lain di dalam sistem seperti yang telah dijelaskan diatas

4.2.2 Kebutuhan Fungsional

Pada tahap ini akan didefinisikan daftar kebutuhan fungsional dari aplikasi manajemen ruang *meeting*. Tabel 4.2 berikut mendefinisikan daftar kebutuhan fungsional dari aplikasi manajemen ruang *meeting*:

Tabel 4.2 Kebutuhan Fungsional

No.	Kode Kebutuhan	Kebutuhan	Nama Kebutuhan	Prioritas	Peran
1.	MIRA-F-01.01	Sistem dapat memberikan akses ke user untuk masuk ke dalam sistem	*Login*	Tinggi	*Guest*
2.	MIRA-F-01.02	Sistem dapat me-*logout* user dari sistem	*Logout*	Tinggi	*Administrator, Requestor, Receptionist*
3.	MIRA-F-01.03	Sistem dapat menampilkan detail informasi dari suatu request ruangan	*See request detail*	Sedang	*Administrator, Requestor*
3.	MIRA-F-02.01	Sistem dapat menampilkan daftar request dari *requestor* yang telah di "Approve" oleh *administrator*	*See accepted request room list*	Sedang	*Administrator*
4.	MIRA-F-02.02	Sistem dapat menampilkan daftar request ruangan yang dibuat oleh *requestor*	*See request room list*	Sedang	*Administrator*

Tabel 4.2 Kebutuhan Fungsional (lanjutan)

No.	Kode Kebutuhan	Kebutuhan	Nama Kebutuhan	Prioritas	Peran
5.	MIRA-F-02.03	Sistem dapat menerima request penggunaan ruang meeting yang dibuat oleh *requestor*	*Accept request*	Tinggi	*Administrator*
6.	MIRA-F-02.04	Sistem dapat menampilkan daftar request pembatalan ruangan	*See request cancel room list*	Sedang	*Administrator*
7.	MIRA-F-02.05	Sistem dapat menerima permintaan pembatalan request ruangan	*Accept cancel request*	Tinggi	*Administrator*
8.	MIRA-F-02.06	Sistem dapat menolak permintaan pembatalan request ruangan	*Decline cancel request*	Tinggi	*Administrator*
9.	MIRA-F-02.07	Sistem dapat menampilkan daftar user dari aplikasi manajemen ruang *meeting*	*Manage user*	Sedang	*Administrator*
10.	MIRA-F-02.08	Sistem dapat menambahkan user baru melalui basis data LDAP	*Add new user LDAP*	Tinggi	*Administrator*

Tabel 4.2 Kebutuhan Fungsional (lanjutan)

No.	Kode Kebutuhan	Kebutuhan	Nama Kebutuhan	Prioritas	Peran
11.	MIRA-F-02.09	Sistem dapat menambahkan user baru melalui input manual	*Add new user manual*	Sedang	*Administrator*
12.	MIRA-F-02.10	Sistem dapat mengaktifkan user yang sebelumnya di-*deactivate*	*Activate user*	Sedang	*Administrator*
13.	MIRA-F-02.11	Sistem dapat men-*deactivate* user	*Deactivate user*	Sedang	*Administrator*
14.	MIRA-F-02.12	Sistem dapat menampilkan daftar informasi gedung	*Manage building*	Sedang	*Administrator*
15.	MIRA-F-02.13	Sistem dapat menambahkan informasi gedung baru	*Add new building*	Sedang	*Administrator*
16.	MIRA-F-02.14	Sistem dapat meng-*update* informasi gedung yang telah tersimpan di dalam basis data sistem	*Update building information*	Sedang	*Administrator*
17.	MIRA-F-02.15	Sistem dapat menampilkan daftar informasi ruangan	*Manage room*	Sedang	*Administrator*

Tabel 4.2 Kebutuhan Fungsional (lanjutan)

No.	Kode Kebutuhan	Kebutuhan	Nama Kebutuhan	Prioritas	Peran
18.	MIRA-F-02.16	Sistem dapat menambahkan informasi ruangan baru	*Add new room*	Sedang	*Administrator*
19.	MIRA-F-02.17	Sistem dapat meng-*update* informasi ruangan yang telah tersimpan di dalam basis data sistem	*Update room information*	Sedang	*Administrator*
20.	MIRA-F-02.18	Sistem dapat mengirimkan notifikasi *email* ke *requestor* dari suatu request	*Send Email Notification*	Tinggi	*Administrator*
21.	MIRA-F-03.01	Sistem dapat menampilkan daftar request dari *requestor* yang melakukan login	*See request list*	Sedang	*Requestor*
22.	MIRA-F-03.02	Sistem dapat melakukan pembatalan request	*Request cancel room*	Sedang	*Requestor*
23.	MIRA-F-03.03	Sistem dapat melakukan pemesanan penggunaan ruangan	*Standart Booking*	Tinggi	*Requestor*

Tabel 4.2 Kebutuhan Fungsional (lanjutan)

No.	Kode Kebutuhan	Kebutuhan	Nama Kebutuhan	Prioritas	Peran
24.	MIRA-F-03.04	Sistem dapat melakukan pemesanan penggunaan ruangan yang terjadwal lebih dari satu kali	*Scheduled Booking*	Tinggi	*Requestor*
25.	MIRA-F-04.01	Sistem dapat menampilkan daftar request yang diterima oleh *administrator*	*See accepted request room list receptionist*	Sedang	*Receptionist*
26.	MIRA-F-04.02	Sistem dapat menampilkan hasil filter berdasarkan tanggal	*Filter*	Sedang	*Receptionist*
27.	MIRA-F-04.03	Sistem dapat memberikan opsi untuk mengunduh atau tidak sebuah file PDF daftar hadir	*Download attendees list*	Tinggi	*Receptionist*
28.	MIRA-F-04.04	Sistem menutup suatu pemesanan ruang *meeting*	*Close room*	Tinggi	*Receptionist*

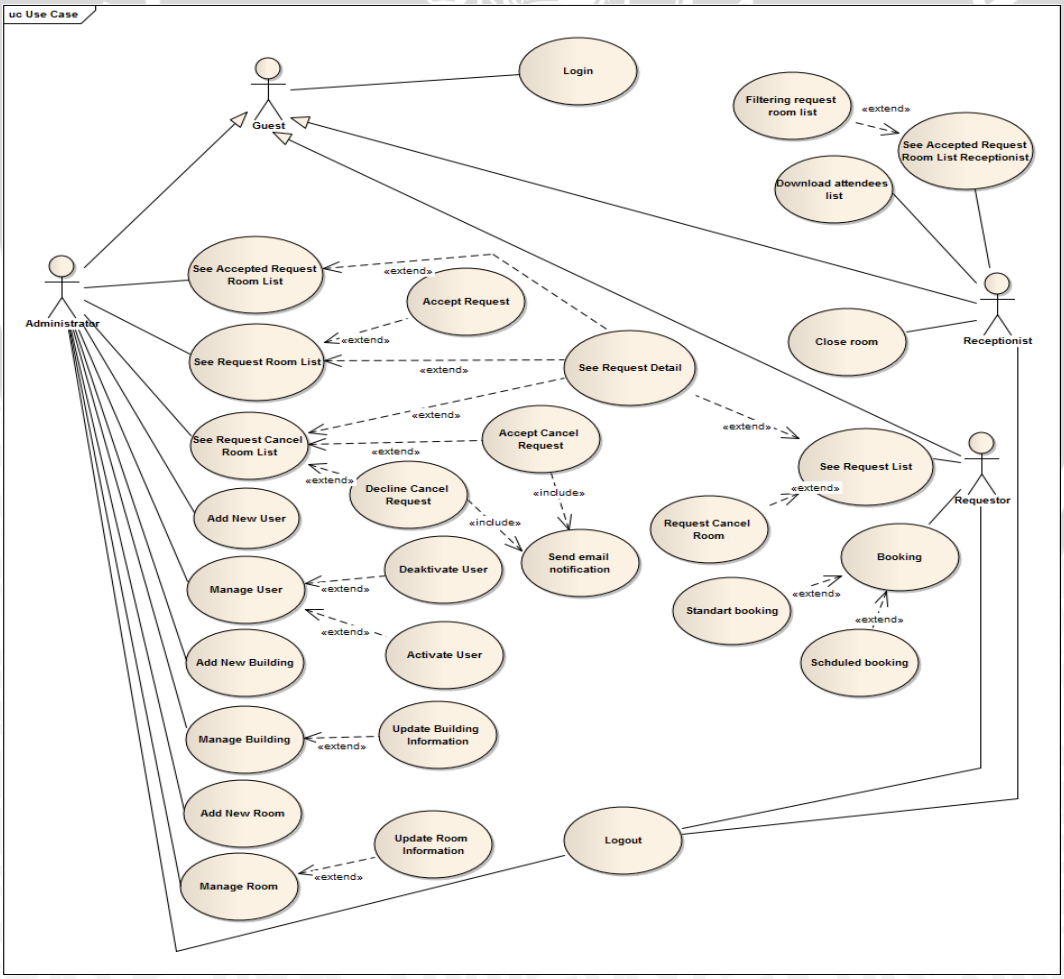
4.2.3 Kebutuhan Non-Fungsional

Pada prosesnya aplikasi manajemen ruang *meeting* yang akan dikembangkan ini akan diserahkan kepada PT. Telkomsel Indonesia Tbk. untuk mengatasi permasalahan pengelolaan penggunaan ruang *meeting*. Oleh karena itu diperlukan pengujian *usability* dari aplikasi yang akan dikembangkan untuk mengetahui tingkat kepuasan dari *client*. Tabel 4.3 berikut mengidentifikasi kebutuhan non-fungsional dari aplikasi manajemen ruang *meeting* yang akan dikembangkan:

Tabel 4.3 Kebutuhan Non-Fungsional

No	Kode Kebutuhan	Parameter	Deskripsi
1.	MIRA-NF-01	Usability	Hasil pengujian usability aplikasi dengan menggunakan metode SUS menghasilkan skor diatas 70

4.2.4 Diagram *Use Case*



Gambar 4.2 Diagram Use Case

Gambar 4.2 pada halaman 33 diatas menunjukkan diagram *use case* dari aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. yang terdiri dari 4 aktor dan masing-masing interaksinya dengan sistem. Aktor *guest* seperti terlihat pada gambar 4.2 diatas memiliki *case* yaitu *login. Guest* merupakan generalisasi dari 3 aktor lain yang ada di dalam sistem yaitu *administrator, requestor,* dan *receptonist.* Aktor *administrator* memiliki 18 *case* yaitu *see accepted request room list, see request room list, see request cancel room list, request detail* yang merupakan *extends* dari 3 *case* sebelumnya, *accept request* yang merupakan *extends* dari *case see request room list, accept cancel request* dan *decline cancel request* yang merupakan *extends* dari *case see request cancel room list, send email notification* include terhadap *case accept* dan *decline cancel request*, *deactivate* dan *activate user* yang merupakan *extends* dari *case manage user, manage user, add new user, update building information* yang merupakan *extends* dari *case manage building, manage building, add new building, update room information* yang merupakan *extends* dari *case manage room, manage room,* dan *add new room*. Aktor *requestor* memiliki 6 *case* yaitu *see request list, request cancel room list dan see request detail* yang merupakan *extends* dari *case see request list, standar* dan *scheduled booking* yang merupakan *extends* dari *case booking*. Aktor *receptionist* memiliki 4 *case* yaitu *see accepted request room list receptionist, filter, download pdf*, dan *close room*.

4.2.5 Skenario *Use case*

Tabel 4.4 Use Case Login

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-01.01
Nama kebutuhan	Login
Tujuan	Melakukan otentifikasi aktor yang akan mengakses sistem
Deskripsi	*Use case* ini menjelaskan bagaimana seorang aktor (*guest*) dapat masuk ke dalam sistem manajemen ruang meeting
Aktor	*Guest*
Skenario Utama	
Kondisi awal	Aktor mengakses halaman login dari sistem dan akan melakukan proses *login*
Aksi aktor	**Reaksi sistem**
1. Menginputkan *username* dan *password* kemudian menekan tombol “login”	2. Sistem mengecek ada atau tidaknya data *username* dan *password* tersebut di dalam basis data sistem

Tabel 4.4 Use Case Login (lanjutan)

	3. Sistem me-*render* halaman home berdasarkan informasi *privillage* yang dimiliki oleh *username* dan *password* yang di-*input*-kan oleh aktor
Skenario alternatif 1 : *username* dan *password* tidak dikenali sistem	
	Sistem me-*render* halaman login dengan menyertakan pesan eror bahwa *username* dan *password* yang di-*input*-kan oleh aktor tidak ditemukan oleh sistem
Kondisi akhir	Aktor dapat *login* ke dalam sistem

Tabel 4.5 Use Case Logout

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-01.02
Nama kebutuhan	Logout
Tujuan	Logout dari aplikasi manajemen ruang *meeting*
Deskripsi	*Use case* ini menjelaskan bagaimana seorang aktor (*Administrator,Requestor,Receptionist*) dapat keluar dari dalam sistem manajemen ruang meeting
Aktor	*Administrator, Requestor, Receptionist*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan *logout* dari sistem
Aksi aktor	**Reaksi sistem**
1. Menekan tombol "Logout"	2. Sistem me-*render* halaman login
Kondisi akhir	Aktor keluar dari sistem

Tabel 4.6 Use Case See Request Detail

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-01.03
Nama kebutuhan	See request detail
Tujuan	Menampilkan detail request ruangan yang dibuat oleh aktor *administrator* dan *requestor*

Tabel 4.6 Use Case See Request Detail (lanjutan)

Deskripsi	*Use case* ini menjelaskan bagaimana informasi detail request suatu ruangan dapat ditampilkan oleh sistem
Aktor	*Administrator, requestor*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melihat detail request ruangan yang dibuat oleh aktor *requestor*
Aksi aktor	**Reaksi sistem**
1. Memilih simbol "detail" yang terdapat pada setiap baris daftar request ruangan	2. Sistem menampilkan detail dari request ruangan tertentu yang dipilih oleh aktor
Kondisi akhir	Aktor dapat melihat detail request ruangan

Tabel 4.7 Use Case See Accepted Request Room List

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.01
Nama kebutuhan	See accepted request room list
Tujuan	Menampilkan daftar request ruangan yang telah diterima oleh aktor *administrator*
Deskripsi	*Use case* ini menjelaskan bagaimana informasi daftar request ruangan yang telah diterima oleh aktor *administrator* ditampilkan oleh sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melihat daftar request ruangan yang telah diterima oleh aktor
Aksi aktor	**Reaksi sistem**
1. Melakukan login	2. Sistem me-*render* halaman home aktor
Kondisi akhir	Aktor dapat melihat daftar request ruangan yang telah diterima

Tabel 4.8 Use Case See Request Room List

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.02
Nama kebutuhan	See request room list
Tujuan	Menampilkan daftar request ruangan yang belum diterima oleh aktor *administrator*
Deskripsi	*Use case* ini menjelaskan bagaimana informasi daftar request ruangan yang belum diterima oleh aktor *administrator* ditampilkan oleh sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melihat daftar request ruangan yang belum diterima oleh aktor
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Request room"	
2. Memilih sub-menu "Request room list"	3. Menampilkan informasi daftar request ruangan yang belum diterima oleh aktor
Kondisi akhir	Aktor dapat melihat daftar request ruangan yang belum diterima

Tabel 4.9 Use Case Accept Request

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.03
Nama kebutuhan	Accept request
Tujuan	Menerima request penggunaan ruang *meeting* yang diajukan oleh aktor *receptonist*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* menerima request penggunaan ruangan yang dibuat oleh aktor *receptionist*
Aktor	*Administrator*

Tabel 4.9 Use Case Accept Request (lanjutan)

Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan penerimaan terhadap request ruangan yang dibuat oleh aktor *reqeustor*
Aksi aktor	
1. Memilih menu "Request room"	
2. Memilih sub-menu "Request room list"	3. Sistem menampilkan daftar request ruangan yang belum diterima oleh aktor
4. Memilih simbol "accept" yang terdapat pada kolom *action* di setiap baris request	5. Sistem merubah status request menjadi "Approved"
Kondisi akhir	Aktor dapat menerima request ruangan yang dibuat oleh aktor *requestor,* status request ruangan menjadi "Approved"

Tabel 4.10 Use Case See Request Cancel Room List

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.04
Nama kebutuhan	See request cancel room list
Tujuan	Menampilkan daftar request pembatalan pemesanan ruangan
Deskripsi	*Use case* ini menjelaskan bagaimana informasi daftar request pembatalan ruangan ditampilkan oleh sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor *administrator* telah melakukan *login* dan dapat melihat daftar request pembatalan pemesanan ruangan yang telah diterima oleh aktor

Tabel 4.10 Use Case See Request Cancel Room List (lanjutan)

Aksi aktor	Reaksi sistem
1. Memilih menu "Request room"	
2. Memilih sub-menu "Request cancel room list"	3. Sistem menampilkan daftar request pembatalan pemesanan ruang *meeting*
Kondisi akhir	Aktor dapat melihat daftar request pembatalan ruangan yang telah diterima sebelumnya oleh aktor

Tabel 4.11 Use Case Accept Cancel Request

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.05
Nama kebutuhan	Accept cancel request
Tujuan	Menerima request pembatalan pemesanan ruangan yang diajukan oleh aktor *requstor*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* menerima request pembatalan pemesanan ruang *meeting*
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan penerimaan terhadap request pembatalan ruangan yang telah diterima sebelumnya oleh aktor
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Request Room"	
2. Memilih sub-menu "Request cancel room list"	3. Sistem menampilkan daftar request pembatalan pemesanan ruangan

Tabel 4.11 Use Case Accept Cancel Request (lanjutan)

4. Memilih simbol "Accept" pada kolom *action*	5. Sistem merubah status request tertentu yang disetujui oleh aktor menjadi "Cancelled" dan mengirimkan notifikasi kepada *requestor*
Kondisi akhir	Aktor dapat menerima request pembatalan ruangan, status request menjadi "Cancelled"

Tabel 4.12 Use Case Decline Cancel Request

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.06
Nama kebutuhan	Decline cancel request
Tujuan	Menolak request pembatalan pemesanan ruangan yang diajukan oleh aktor *requstor*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* menolak request pembatalan pemesanan ruang *meeting*
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan penolakan terhadap request pembatalan ruangan yang telah diterima sebelumnya oleh aktor
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Request Room"	
2. Memilih sub-menu "Request cancel room list"	3. Sistem menampilkan daftar request pembatalan pemesanan ruangan
4. Memilih simbol "Decline"	5. Sistem mengirimkan notifikasi kepada *requestor* yang bersangkutan yang berisi informasi request pembatalan ditolak oleh aktor
Kondisi akhir	Aktor dapat menolak request pembatalan ruangan, sistem mengirimkan notifikasi kepada aktor *requestor* yang bersangkutan

Tabel 4.13 Use Case Manage User

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.07
Nama kebutuhan	Manage user
Tujuan	Menampilkan daftar user yang tersimpan di dalam basis data sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat menampilkan daftar user yang ada di dalam sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melihat daftar user yang ada di dalam sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	
2. Memilih sub-menu *user* "Update"	3. Sistem menampilkan daftar user yang ada di dalam aplikasi
Kondisi akhir	Aktor dapat melihat daftar user yang ada di dalam aplikasi

Tabel 4.14 Use Case Add New User LDAP

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.08
Nama kebutuhan	Add new user (LDAP)
Tujuan	Menambahkan user baru ke dalam sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat menambahkan user baru ke dalam sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat menambahkan user baru ke dalam sistem

Tabel 4.14 Use Case Add New User LDAP (lanjutan)

Aksi aktor	Reaksi sistem
1. Memilih menu "Manage"	
2. Memilih sub-menu *user* "New"	3. Sistem menampilkan form pencarian user berdasarkan *username*
4. Meng-*input*-kan *username* dari user yang akan ditambahkan	5. Sistem melakukan pencarian data user berdasarkan *username* di basis data LDAP (Basis data yang menyimpan daftar pegawai PT. Telkomsel Indonesia Tbk.) dan menampilkan hasil pencariannya ke halaman aktor
6. Aktor memilih simbol "Add" pada salah satu dari daftar user yang ditampilkan oleh sistem	7. Sistem menyimpan informasi user baru ke dalam basis data sistem
Skenario alternatif 1 : tidak ada data user yang ditampilkan oleh sistem	
	Sistem menampilkan informasi bahwa *username* yang di-*input*-kan oleh aktor tidak sesuai dengan seluruh data dalam basis data LDAP
Kondisi akhir	Aktor dapat menambahkan user baru ke dalam sistem

Tabel 4.15 Use Case Add New User Manual

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.09
Nama kebutuhan	Add new user (Manual)
Tujuan	Menambahkan user baru ke dalam sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat menambahkan user baru ke dalam sistem
Aktor	*Administrator*

Tabel 4.15 Use Case Add New User Manual (lanjutan)

Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat menambahkan user baru ke dalam sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	
2. Memilih sub-menu *user* "New (Manual)"	3. Sistem menampilkan *form* tambah user
4. Meng-*input*-kan data kedalam *form* tersebut dan memilih *button* "save"	5. Sistem menyimpan informasi user baru ke dalam basis data sistem
Kondisi akhir	Aktor dapat menambahkan user baru ke dalam sistem

Tabel 4.16 Use Case Activate User

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.09
Nama kebutuhan	Activate user
Tujuan	Mengaktifasi kembali user sistem yang sebelumnya telah dideaktifasi
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* melakukan aktifasi terhadap user sistem yang sebelumnya telah dideaktifasi
Aktor	*Administrator*

Tabel 4.16 Use Case Activate User (lanjutan)

Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan aktifasi user
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	
2. Memilih sub-menu *user* "Update"	3. Sistem menampilkan daftar user yang ada di dalam sistem
4. Memilih simbol "Activate" pada salah satu baris data user	5. Sistem merubah status user di dalam basis data sistem menjadi "Active"
Kondisi akhir	Aktor dapat mengaktifasi user yang berstatus deaktif

Tabel 4.17 Use Case Deactivate User

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.10
Nama kebutuhan	Deactivate user
Tujuan	Mendeaktifasi user sistem yang memiliki status aktif
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat melakukan deaktifasi terhadap user sistem yang memiliki status "Active"
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan deaktifasi user sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	

Tabel 4.17 Use Case Deactivate User (lanjutan)

2. Memilih sub-menu *user* "Update"	3. Sistem menampilkan daftar user yang ada di dalam sistem
4. Memilih simbol "Deactivate" pada salah satu baris data user yang ditampilkan oleh sistem	5. Sistem merubah status user di dalam basis data sistem menjadi "Deactive"
Kondisi akhir	Aktor dapat melakukan deaktifasi user

Tabel 4.18 Use Case Manage Building

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.11
Nama kebutuhan	Manage building
Tujuan	Menampilkan daftar gedung yang tersimpan di dalam basis data sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat menampilkan daftar gedung yang tersimpan di dalam basis data sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat menampilkan daftar gedung
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	
2. Memilih sub-menu *building* "Update"	3. Sistem menampilkan daftar gedung yang tersimpan di dalam basis data sistem
Kondisi akhir	Aktor dapat melihat daftar gedung yang tersimpan di dalam basis data sistem

Tabel 4.19 Use Case Add New Building

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.12
Nama kebutuhan	Add new building
Tujuan	Menambahkan informasi gedung baru ke dalam sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat menambahkan informasi gedung baru ke dalam sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat menambahkan informasi gedung baru ke dalam sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	
2. Memilih sub-menu *building* "New"	3. Sistem menampilkan *form* penambahan gedung baru
4. Meng-*input*-kan informasi gedung baru dan memilih *button* "Add"	5. Sistem menyimpan informasi gedung baru ke dalam basis data sistem
Kondisi akhir	Aktor dapat menambahkan informasi gedung baru ke dalam basis data sistem

Tabel 4.20 Use Case Update Building Information

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.13
Nama kebutuhan	Update building information

Tabel 4.20 Use Case Update Building Information (lanjutan)

Tujuan	Meng-*update* informasi gedung yang tersimpan di dalam basis data sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat meng-*update* informasi gedung yang tersimpan di dalam sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan update informasi gedung yang tersimpan di dalam sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	
2. Memilih sub-menu *building* "Update"	3. Sistem menampilkan daftar gedung yang tersimpan di dalam basis data sistem
4. Memilih simbol "Update building"	5. Sistem menampilkan *form update* informasi gedung
6. Meng-*input*-kan informasi baru dari suatu gedung kerja dan memilih *button* "Add"	7. Sistem menyimpan *update* informasi gedung ke dalam basis data sistem
Kondisi akhir	Aktor dapat meng-*update* informasi gedung dan menyimpannya ke dalam basis data sistem

Tabel 4.21 Use Case Manage Room

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.14
Nama kebutuhan	Manage room
Tujuan	Menampilkan daftar ruang *meeting* yang tersimpan di dalam basis data sistem

Tabel 4.21 Use Case Manage Room (lanjutan)

Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat melihat daftar ruang *meeting* yang tersimpan di dalam sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melihat daftar ruang *meeting* yang ada di dalam sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	
2. Memilih sub-menu *room* "Update"	3. Sistem menampilkan daftar ruangan yang tersimpan di dalam basis data sistem
Kondisi akhir	Aktor dapat melihat daftar ruang *meeting* yang tersimpan di dalam basis data sistem

Tabel 4.22 Use Case Add New Room

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.15
Nama kebutuhan	Add new room
Tujuan	Menambahkan informasi ruangan baru ke dalam sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat menambahkan informasi ruangan baru ke dalam sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat menambahkan informasi ruangan baru ke dalam sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	

Tabel 4.22 Use Case Add New Room (lanjutan)

2. Memilih sub-menu *room* "New"	3. Sistem menampilkan *form* penambahan ruangan baru
4. Meng-*input*-kan informasi ruangan baru dan memilih *button* "Add"	5. Sistem menyimpan informasi ruangan baru ke dalam basis data sistem
Kondisi akhir	Aktor dapat menambahkan informasi ruangan baru ke dalam basis data sistem

Tabel 4.23 Use Case Update Room Information

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.16
Nama kebutuhan	Update room information
Tujuan	Meng-*update* informasi ruangan yang tersimpan di dalam basis data sistem
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat meng-*update* informasi ruangan yang tersimpan di dalam sistem
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan update informasi ruangan yang tersimpan di dalam sistem
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Manage"	

Tabel 4.23 Use Case Update Room Information (lanjutan)

2. Memilih sub-menu *room* "Update"	3. Sistem menampilkan daftar ruangan yang tersimpan di dalam basis data sistem
4. Memilih simbol "Update room"	5. Sistem menampilkan *form update* informasi ruangan
6. Meng-*input*-kan informasi baru dari suatu ruangan kerja dan memilih *button* "Add"	7. Sistem menyimpan *update* informasi ruangan ke dalam basis data sistem
Kondisi akhir	Aktor dapat meng-*update* informasi ruangan dan menyimpannya ke dalam basis data sistem

Tabel 4.24 Use Case Send Email Notification

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-02.17
Nama kebutuhan	Send email notification
Tujuan	Mengirimkan notifikasi email kepada *requstor* terkait request pembatalan pemesanan ruangan
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *administrator* dapat mengirimkan email notifikasi tindak lanjut dari request pembatalan pemesanan ruangan
Aktor	*Administrator*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat mengirimkan email notifikasi kepada *requestor*
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Request room"	

Tabel 4.24 Use Case Send Email Notification (lanjutan)

2. Memilih sub-menu *room* "Request cancel room"	3. Sistem menampilkan daftar request pembatalan pemesanan ruangan
4. Memilih simbol "Accept" atau "Decline"	5. Sistem mengirimkan email notifikasi kepada *requestor* sebagai tindak lanjut dari request pembatalan pemesanan ruangan
Kondisi akhir	Aktor dapat mengirim email notifikasi kepada *requestor*

Tabel 4.25 Use Case See Request List

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-03.01
Nama kebutuhan	See request list
Tujuan	Menampilkan daftar request ruangan yang telah dibuat oleh aktor *requestor*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *requestor* dapat menampilkan daftar request yang pernah dibuat
Aktor	*Requestor*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melihat daftar request yang pernah dibuat sebelumnya
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Home"	2. Sistem menampilkan daftar request yang pernah dibuat oleh aktor
Kondisi akhir	Aktor dapat melihat daftar request yang pernah dibuat sebelumnya

Tabel 4.26 Use Case Request Cancel Room

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-03.02
Nama kebutuhan	Request cancel room

Tabel 4.26 Use Case Request Cancel Room (lanjutan)

Tujuan	Melakukan request pembatalan pemesanan ruang *meeting*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *requestor* dapat melakukan request pembatalan ruang *meeting*
Aktor	*Requestor*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan request pembatalan ruang *meeting*
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Home"	2. Sistem menampilkan daftar request yang pernah dibuat oleh aktor
3. Memilih simbol "Cancel" pada salah satu baris data	4. Sistem merubah status request menjadi "Cancel request pending"
Kondisi akhir	Aktor dapat melakukan request pembatalan ruang *meeting*

Tabel 4.27 Use Case Standart Booking

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-03.03
Nama kebutuhan	Standart Booking
Tujuan	Melakukan request pemesanan ruang *meeting*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *requestor* dapat melakukan request penggunaan ruang *meeting*
Aktor	*Requestor*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan request penggunaan ruang *meeting*
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Request room"	

Tabel 4.27 Use Case Standart Booking (lanjutan)

2. Memilih sub-menu "Standart request"	3. Sistem me-*render* halaman request room yang berisi *form* pengecekan ketersediaan ruangan dan tabel daftar ruangan yang tersedia
4. Meng-*input*-kan informasi yang dibutuhkan sesuai dengan *form* pengecekan ketersediaan ruangan	5. Sistem melakukan pengecekan ketersediaan ruangan sesuai informasi dari aktor, sistem menampilkan daftar ruangan yang tersedia ke halaman request room bagian tabel daftar ruangan tersedia
6. Memilih salah satu ruangan dari daftar ruangan yang ditampilkan oleh sistem	7. Sistem menampilkan *form* detail request ruangan
8. Meng-*input*-kan kebutuhan informasi sesuai *form* detail request ruangan dan memilih *button* "Book"	9. Sistem menyimpan informasi request dari aktor
Kondisi akhir	Aktor dapat melakukan request penggunaan ruang *meeting*

Tabel 4.28 Use Case Scheduled Booking

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-03.04
Nama kebutuhan	Scheduled booking
Tujuan	Melakukan request pemesanan ruang *meeting* terjadwal lebih dari 1 kali

Tabel 4.28 Use Case Scheduled Booking (lanjutan)

Deskripsi	*Use case* ini menjelaskan bagaimana aktor *requestor* dapat melakukan request penggunaan ruang *meeting* terjadwal lebih dari 1 kali
Aktor	*Requestor*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan request penggunaan ruang *meeting* terjadwal lebih dari 1 kali
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Request room"	
2. Memilih sub-menu "Scheduled request"	3. Sistem me-*render* halaman request room yang berisi *form* pengecekan ketersediaan ruangan dan tabel daftar ruangan yang tersedia
4. Meng-*input*-kan informasi yang dibutuhkan sesuai dengan *form* pengecekan ketersediaan ruangan	5. Sistem melakukan pengecekan ketersediaan ruangan sesuai informasi dari aktor, sistem menampilkan daftar ruangan yang nantinya akan digunakan oleh aktor
6. Memilih *button* "Book"	7. Sistem menampilkan *form* detail request ruangan
8. Meng-*input*-kan kebutuhan informasi sesuai *form* detail request ruangan dan memilih *button* "Book"	9. Sistem menyimpan informasi request dari aktor
Kondisi akhir	Aktor dapat melakukan request penggunaan ruang *meeting* terjadwal lebih dari 1 kali

Tabel 4.29 Use Case See Accepted Request Room List Receptonist

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-04.01
Nama kebutuhan	See accepted request room list receptionist
Tujuan	Menampilkan daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *receptionist* dapat menampilkan daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*
Aktor	*Receptionist*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melihat daftar request ruangan yang telah diterima oleh *administrator*
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Home"	2. Sistem menampilkan daftar request ruangan yang telah diterima oleh *administrator*
Kondisi akhir	Aktor dapat melihat daftar request ruangan yang telah diterima oleh *administrator*

Tabel 4.30 Use Case Filter

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-04.02
Nama kebutuhan	Filter
Tujuan	Melakukan filter daftar request ruangan yang telah diterima oleh aktor *administrator*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *receptionist* dapat melakukan filter terhadap daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*
Aktor	*Receptionist*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat melakukan filter terhadap daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*

Tabel 4.30 Use Case Filter (lanjutan)

Aksi aktor	Reaksi sistem
1. Memilih menu "Home"	2. Sistem menampilkan daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*
3. Mengisi *form* filter ruangan	4. Sistem melakukan filter berdasarkan informasi yang diberikan aktor dan menampilkan hasil filternya ke layar
Kondisi akhir	Aktor dapat melihat daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*

Tabel 4.31 Use Case Download Attendees List

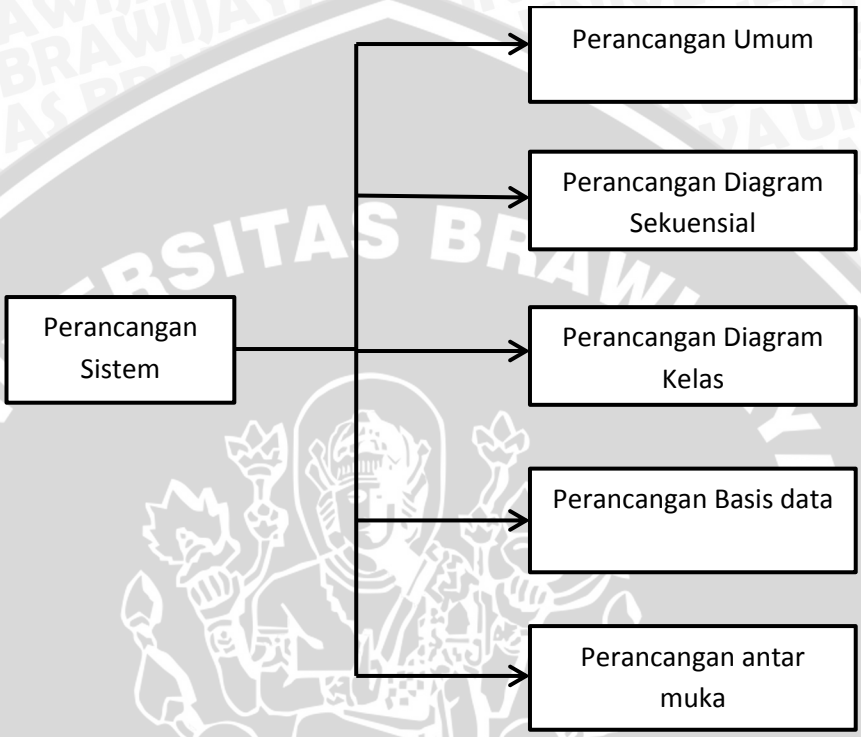
Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-04.03
Nama kebutuhan	Download attendees list
Tujuan	Mengunduh pdf daftar hadir dari suatu request ruangan yang telah diterima oleh aktor *administrator*
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *receptionist* dapat mengunduh pdf daftar hadir dari suatu request ruang *meeting* yang telah diterima oleh aktor *administrator*
Aktor	*Receptionist*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat mengunduh pdf daftar hadir dari suatu request ruang *meeting*
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Home"	2. Sistem menampilkan daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*
3. Memilih simbol "download" pada salah satu dari daftar request ruangan	4. Sistem mengunduhkan file pdf daftar hadir dari ruang *meeting* yang dipilih aktor
Kondisi akhir	Aktor dapat mengunduh file daftar hadir dari suatu ruang *meeting* yang telah dipesan

Tabel 4.32 Use Case Close Room

Skenario Kasus Pada Sistem	
Kode kebutuhan	MIRA-F-04.04
Nama kebutuhan	Close room
Tujuan	Menutup suatu pesanan penggunaan ruang *meeting* setelah selesai dilaksanakan
Deskripsi	*Use case* ini menjelaskan bagaimana aktor *receptionist* dapat menutup suatu pesanan penggunaan ruang *meeting* setelah selesai dilaksanakan
Aktor	*Receptionist*
Skenario Utama	
Kondisi awal	Aktor telah melakukan *login* dan dapat Menutup suatu pesanan penggunaan ruang *meeting* setelah selesai dilaksanakan
Aksi aktor	**Reaksi sistem**
1. Memilih menu "Home"	2. Sistem menampilkan daftar request ruang *meeting* yang telah diterima oleh aktor *administrator*
3. Memilih simbol "close" pada salah satu dari daftar request ruangan	4. Sistem menutup request ruang *meeting* dan membuka untuk request baru
Kondisi akhir	Aktor dapat menutup suatu pesanan penggunaan ruang *meeting* setelah selesai dilaksanakan

4.3 Perancangan Sistem

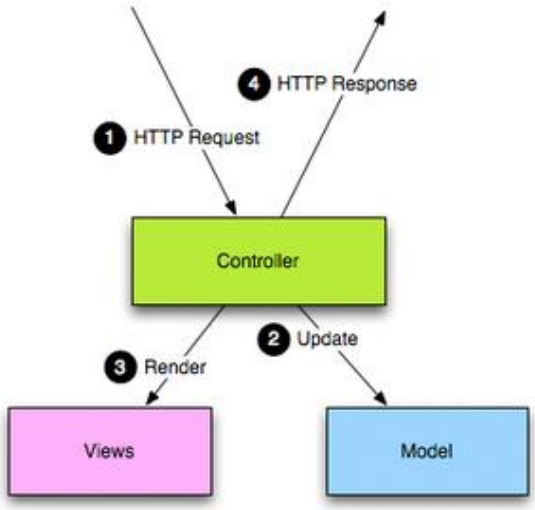
Sesuai dengan deskripsi pada sub-bab pembuatan perangkat lunak, tahap perancangan sistem ini terdiri dari 5 tahap yaitu, perancangan umum, perancangan diagram sekuensial, perancangan diagram kelas, perancangan diagram entitas, dan perancangan antar muka. Gambar 4.3 berikut merepresentasikan urutan pada tahap perancangan sistem:



Gambar 4.3 Diagram Perancangan Sistem

4.3.1 Perancangan Umum

Pada tahap perancangan umum akan digambarkan rancangan umum dari kerangka kerja *PLAY!* yang nantinya kan dijadikan dasar implementasi kode program dengan menggukanan kerangka kerja *PLAY!*. rancangan umum dari kerangka kerja *PLAY!* dapat dilihat pada gambar 4.4 berikut:



Gambar 4.4 Konsep Utama *PLAY! Framework*

Sumber: Zenexity (2007)

Seperti terlihat pada gambar 4.4 diatas, kerangka kerja *PLAY!* menganut pola arsitektural MVC (*Model View Controller*) yang terdiri dari 3 lapisan yaitu lapisan *controllers, models, views*. Ketiga lapisan tersebut berada pada direktori *app* yang berturut-turut *app/controllers, app/models, app/views*. Berikut penjelasan dari masing-masing lapisan:

1. *App/controllers*

   Sebuah *package controllers* dalam kerangka kerja *PLAY!* adalah sebuah kelas *Java public* dengan method static sebagai aksinya. Method atau aksi akan di-*invoke* ketika terdapat request HTTP yang diterima oleh kerangka kerja *PLAY!*. Aksi atau method tersebut akan mengekstrak data yang relevan dari request HTTP, kemudian melakukan *read* atau *update* pada objek model, dan mengirim kembali hasil pengolahan datanya ke *user* dengan membungkusnya kedalam sebuah response HTTP. Kelas *Java* pada kerangka kerja *PLAY!* mengharuskannya meng-*extends* kelas *controllers* sebagai syarat dikenalinya kelas tersebut sebagai lapisan *controllers* oleh kerangka kerja *PLAY!*.

2. *App/models*

   Sebuah *package models* dalam kerangka kerja *PLAY!* berisi sekumpulan kelas *Java* yang menggunakan semua fitur *object oriented* yang tersedia di dalam bahasa *Java*. Didalam setiap kelas tersebut mengandung struktur data dan operasi yang dapat dioperasikan oleh aplikasi.

3. *App/views*

Sebuah *package views* dalam kerangka kerja *PLAY!* berisi sekumpulan file *.html* yang dikenali oleh browser untuk menampilkan sebuah halaman website. *Views* dalam kerangka kerja *PLAY!* di-*generate* menggunakan sebuah sistem template yang effisien yang disediakan oleh *PLAY!. Package* ini dapat berisi HTML, XML, dan JSON.

Menarik untuk dibahas adalah, kerangka kerja *PLAY!* menggunakan bahasa lain yang digunakan untuk menghubungkan antara bahasa *Java* yang *server-side* dan bahasa *HTML* yang *client-side*. Bahasa yang digunakan oleh kerangka kerja *PLAY!* tersebut adalah bahasa *Groovy*. Bahasa *Groovy embedded* didalam file *.html* yang digunakan kerangka kerja *PLAY!* untuk me-*render* tampilan website dan dikenali oleh *web browser* untuk membuat sebuah struktur logika didalam sebuah file *.html*. Dengan menggunakan *template tag* tertentu bahasa *Groovy* ini dapat dikenali oleh *web browser*. Berikut beberapa *template tag* dalam bahasa *Groovy* yang dapat dikenali ketika diimplementasikan kedalam sebuah file *.html*:

1. *Template tag* untuk mengambil nilai judul halaman yang dikirim oleh lapisan *controllers* dilambangkan sebagai berikut:

   ```
   #{get 'title'}Used if title not set#{/get}
   ```

2. *Template tag* untuk menambahkan sebuah file *.html* kedalam suatu fragmen file *.html* yang lain dapat menggunakan *tag* sebagai berikut:

   ```
   #{include 'tree.html'/}
   ```

3. *Template tag* untuk membangun sebuah *script logic* didalam file *.html* dapat menggunakan *tag* berikut:

   ```
   %{ out.print("HelloWorld") }%
   ```

4. *Template tag* untuk membangun sebuah logika percabangan:

   ```
   #{if cond}...#{/if}#{elseif cond}...#{/elseif}#{else}...#{/else}
   ```

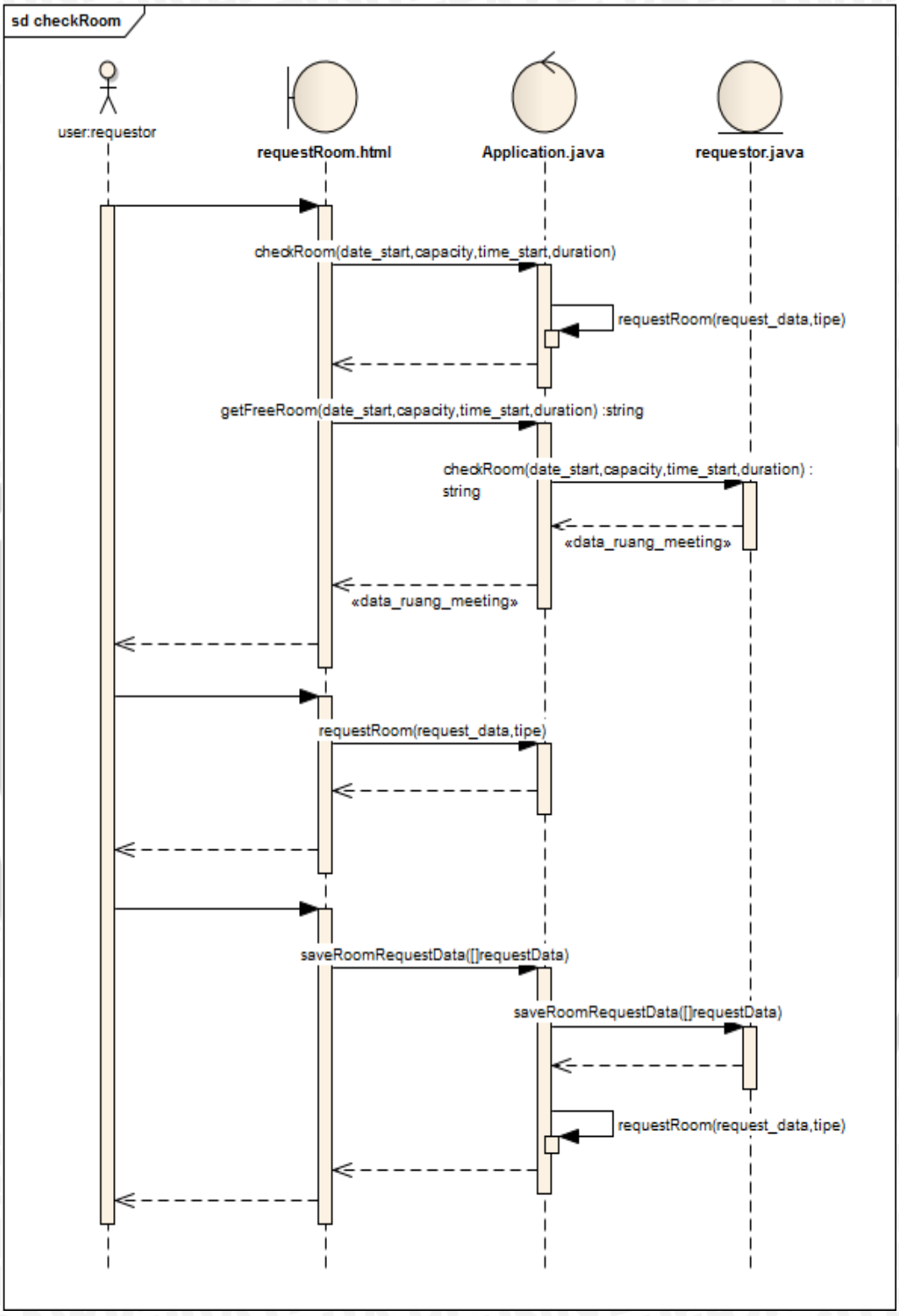
5. *Template tag* untuk membangun sebuah logika perulangan:

   ```
   #{list items:0..10, as:'i'}${i}#{/list}
   ```

4.3.2 Perancangan Diagram Sekuensial (*Sequence Diagram*)

Pada tahap perancangan ini akan dilakukan perancangan diagram sekuensial untuk merepresentasikan alur sebuah *case* pada diagram *use case* berdasarkan waktu yang dilihat dari sudut pandang aktor. Berikut beberapa contoh diagram sekuensial dari aplikasi manajemen ruang *meeting*:

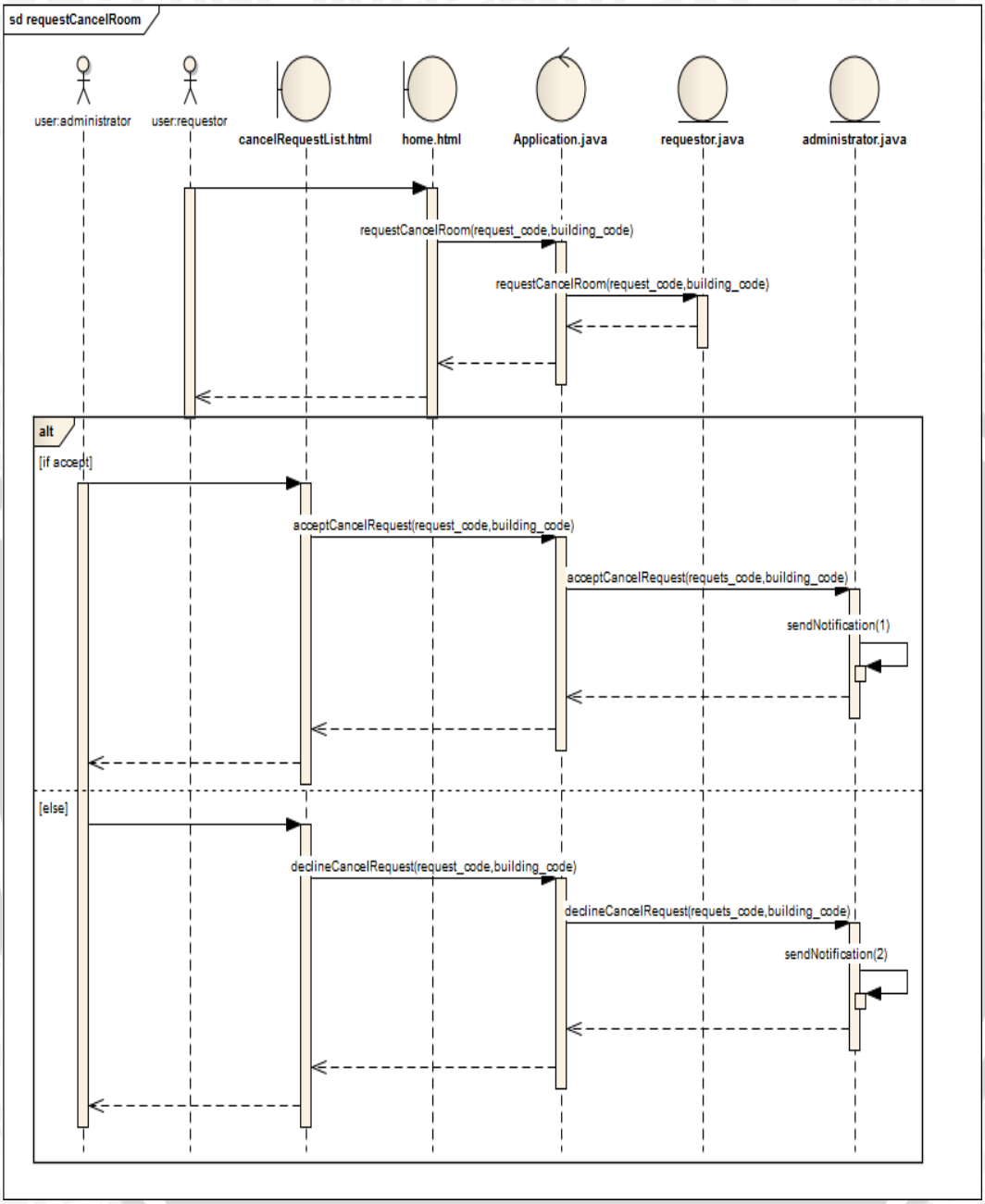
1. *Sequence Standart Booking*



Gambar 4.5 *Sequence Standart Booking*

Seperti terlihat pada gambar 4.5 pada halaman 60 diatas, proses pemesanan ruangan dilakukan oleh *requestor*. Proses pemesanan ruangan diawali dengan *requestor* membuka halaman *requestRoom.html,* kemudian *requestor* diminta untuk mengisi *form* pengecekan ketersediaan ruangan dengan meng-*input*-kan data tanggal *meeting*, kapasitas ruangan yang diinginkan, jam dimulainya *meeting,* dan durasi dari meeting tersebut. Setelah seluruh data *form* terisi user harus menekan *button* "Check", button ini akan meng-*invoke* method checkRoom yang ada di *Application.java* (*Controllers*). Kemudian dilanjutkan dengan *Application.java* meng-*invoke* method requestRoom dengan data *form* pengecekan yang sudah diisi oleh *requestor*(*request_data*) dimana data *request_data* ini mewakili 4 parameter yaitu *date_start, capacity, time_start,* dan *duration* dan parameter tipe. Data tipe disini akan memberi tahu *file* requestRoom.html bagian manakah dari *file* tersebut yang akan dieksekusi. Terdapat struktur percabangan dalam *file* requestRoom.html. Dengan dieksekusinya method requestRoom maka sistem akan me-*render* kembali halaman requestRoom.html dengan data tipe yang berbeda. Dari halaman requestRoom.html akan di *invoke* method getFreeRoom. Method ini akan mengembalikan data daftar ruang *meeting* yang tersedia untuk dipesan sesuai dengan data dalam *form* pengecekan ketersediaan ruangan yang diisi oleh *requestor* sebelumnya. Setelah seluruh daftar ruang *meeting* yang tersedia ditampilkan di layar monitor, *requestor* dapat memilih salah satu dari daftar ruangan tersebut untuk dapat melanjutkan ke step yang selanjutnya. Sekali *requestor* memilih salah satu dari daftar ruang *meeting* tersebut, sistem akan me-*render* halaman requestRoom.html sekali lagi dengan tipe yang berbeda untuk menampilkan *form* detail pemesanan ruangan. *Requestor* diwajibkan untuk mengisi *form* tersebut. Setelah *form* tersebut terisi, *requestor* dapat menekan *button* "Book" untuk memberitahu sistem bahwa *requestor* telah selesai mengisi data detail *meeting*. Setelah *button* "Book" tersebut ditekan oleh *requestor,* sistem akan meng-*invoke* method saveRequestRoomData untuk menyimpan data pemesanan ruang *meeting* dari *requestor* tersebut. Jika proses penyimpanan berhasil, *requestor* akan di *direct* kembali ke halaman requestRoom.html tanpa adanya pesan *error.* Setelah seluruh proses selesai dilakukan, *requestor* tinggal menunggu konfirmasi dari *administrator* terkait request yang telah ia buat.

2. *Sequence Request Cancel Room*

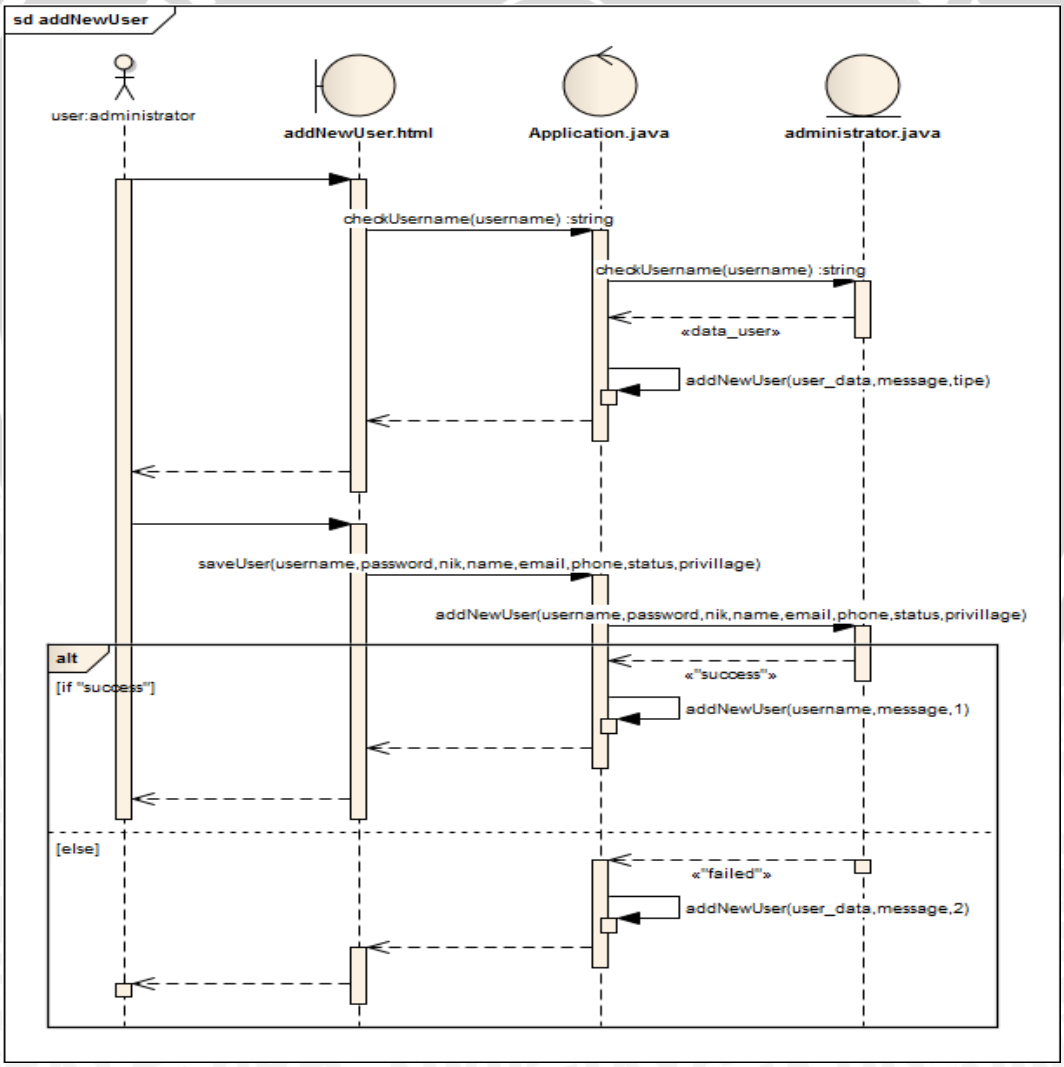


Gambar 4.6 *Sequence Request Cancel Room*

Seperti terlihat pada gambar 4.6 diatas, proses request pembatalan ruangan dapat dilakukan oleh user *requestor* dengan persetujuan dari *administrator*. Alurnya hampir mirip dengan proses booking, hanya saja pada *sequence "*request cancel room" ini penulis buat lebih detail dengan melibatkan *administrator* yang berwenang melakukan persetujuan request pembatalan ruangan. Proses request ini diawali dengan *requestor* yang membukan halaman home.html. Halaman home.html dari *requestor* berisi konten daftar request yang pernah dibuat oleh *requestor* tersebut beserta statusnya. Di halaman tersebut *requestor* diberikan

hak untuk melakukan request pembatalan ruangan dengan di munculkannya *button* "Cancel". Dengan *requestor* menekan *button* tersebut, maka sistem akan meng-*invoke* method requestCancelRoom dengan data *request_code* dan *building_code* yang ada di *Application.java*, kemudian method di *Application.java* ini akan meng-*invoke* method dengan nama yang sama yang terdapat pada kelas *requestor.java*. Method ini akan merubah status dari request yang dibuat oleh *requestor* tersebut. Perubahan status request ini dapat langsung dilihat oleh *administrator* dengan membuka file cancelRequestList.html. di halaman tersebut *administrator* disediakan 3 *button* yaitu "Accept", "Decline", "Detail". Dengan menekan *button* "Accept", maka *administrator* telah memberi tahu sistem untuk meng-*invoke* method acceptCancelRequest sedangkan jika *administrator* menekan *button* "Decline", maka *administrator* telah memberi tahu sistem untuk meng-*invoke* method declineCancelRequest. Kedua pilihan ini akan memberikan *trigger* bagi method sendNotification untuk mengirimkan notifikasi kepada *requestor* melalui email.

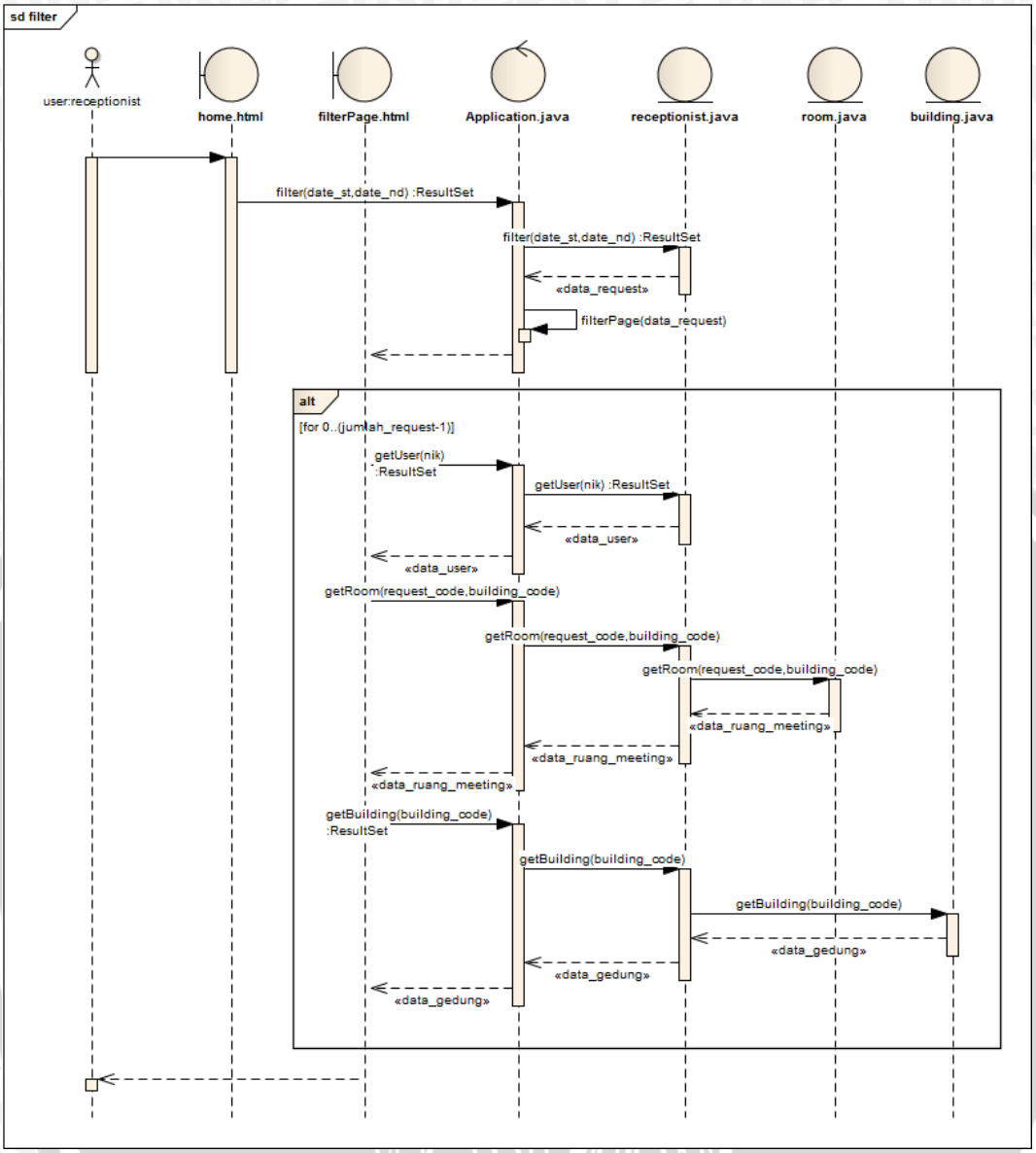
3. *Sequence Add New User*



Gambar 4.7 *Sequence Add New User*

Seperti terlihat pada gambar 4.7 pada halaman 63 diatas, proses menambahkan user baru hanya bisa dilakukan oleh user *administrator*. Proses penambahan user mirip dengan proses pemesanan ruangan dimana hanya satu halaman .html yang dilibatkan dalam proses ini. Penambahan user baru diawali dengan *administrator* yang membuka halaman addNewUser.html. ketika pertama kali mengakses halaman ini sistem akan meminta *administrator* untuk menginputkan username dari user baru yang akan ditambahkan. Setelah *administrator* selesai meng-*input*-kan username dan menekan *button* "Check", sistem akan meng-*invoke* method checkUsername yang ada di *Application.java*, kemudian dari *Application.java* akan diteruskan ke kelas *administrator.java* dengan meng-*Invoke* method dengan nama yang sama. Kelas *administrator.java* akan melakukan koneksi ke basis data LDAP (basis data daftar biodata pegawai PT. Telkomsel Indonesia Tbk.), tetapi dalam penelitian ini penulis akan memanfaatkan layanan *hosting* online yang banyak tersedia di internet untuk menyimpan data *dummy* pegawai PT. Telkomsel Indonesia Tbk. karena alasan keamanan. Setelah terkoneksi ke basis data LDAP tersebut, sistem akan melakukan *query* untuk mencari dan mencocokkan data username yang di-*input*-kan oleh *administrator* dengan daftar username pegawai yang ada di basis data LDAP. Hasil dari query ini akan dikembalikan ke *Application.java*. *Application.java* yang menerima data *return* dari kelas *administrator.java* akan me-*render* halaman addNewUser.html dengan tipe yang berbeda. Halaman addNewUser.html ini akan menampilkan daftar hasil pengecekan username oleh sistem. Pada masing-masing baris daftar tersebut disediakan *button* "Detail" untuk memeriksa data dari pemilik username yang akan ditambahkan untuk mencegah kekeliruan penambahan user baru. Setelah *administrator* menemukan data yang sesuai, maka *administrator* dapat menekan *button* "Add User" untuk memberi tahu sistem agar meng-*invoke* method saveUser pada *Application.java*. Method ini akan meneruskan proses penyimpanan data user baru dengan meng-*invoke* method saveUser yang ada pada kelas *administrator.java.* method ini akan melakukan penyimpanan data user baru ke dalam basis data aplikasi manajemen ruang *meeting*. Hasil dari proses penambahan user baru ini adalah sebuah *message* dari sistem untuk memberitahu *administrator* apakah proses penyimpanan data user baru tersebut berhasil atau gagal.

4. *Sequence Filter*

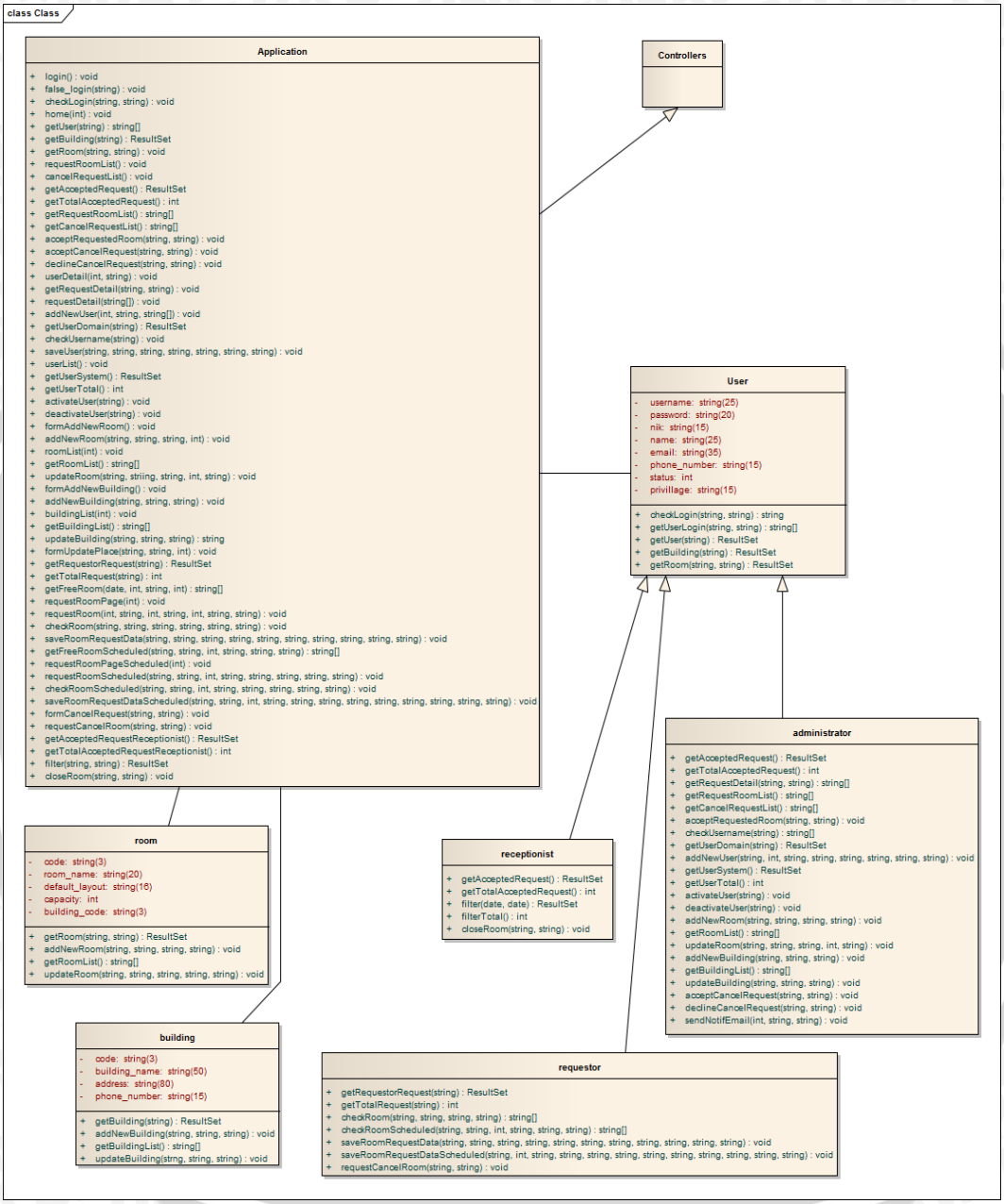


Gambar 4.8 *Sequence Filter*

Sesuai dengan gambar 4.8 diatas, proses filter dapat dilakukan oleh *receptionist*. Proses ini diawali dengan *receptionist* membuka halaman home.html yang berisi daftar request ruangan yang telah diterima oleh *administrator*. Pada halaman ini juga terdapat *form* filter berdasarkan tanggal pelaksanaan *meeting*. Untuk melakukan filter *receptionist* dapat mengisikan rentang tanggal pada *form* tersebut kemudian menekan *button* "Filter". Ketika *button* ini ditekan maka sistem akan langsung meng-*invoke* method filter pada *Application.java* dan diteruskan ke kelas *receptionist.java.* data hasil *query*-nya akan dikembalikan ke Controller, dan Controller akan me-*render* halaman filterPage.html untuk menampilkan daftar fiter yang dikembalikan oleh method filter.

4.3.3 Perancangan Diagram Kelas (*Class Diagram*)

Perancangan diagram kelas didasarkan pada hasil perancangan diagram sekuensial. Gambar 4.9 berikut merepresentasikan diagram kelas dari aplikasi manajemen ruang *meeting*:



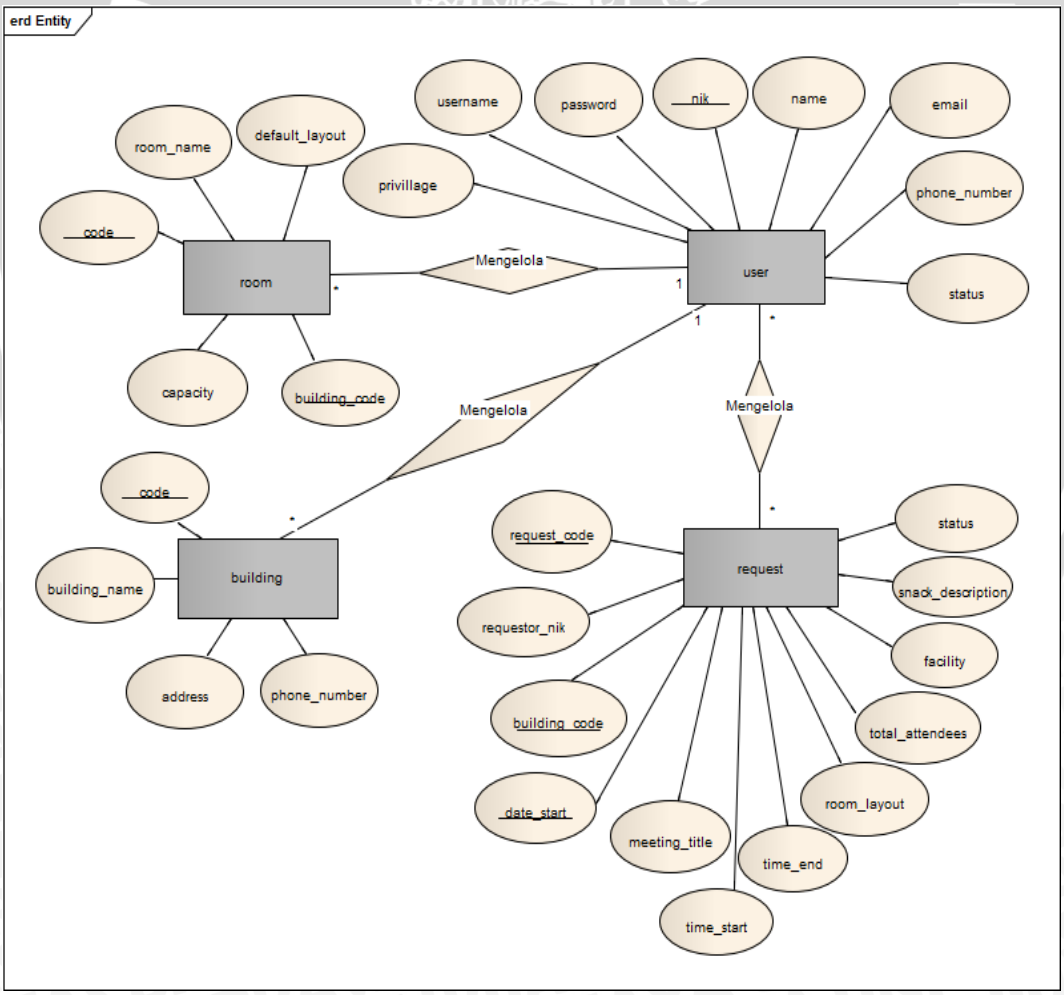
Gambar 4.9 Diagram Kelas

Seperti terlihat pada gambar 4.9 diatas, perancangan diagram kelas pada aplikasi manajemen ruang *meeting* terdiri 7 kelas dengan 1 kelas khusus yang dimiliki oleh kerangka kerja *PLAY!.* ketujuh kelas tersebut diantaranya adalah kelas *Appilcation.java*, kelas ini memiliki hubungan asosiasi dengan 3 kelas lain yaitu kelas *user.java, room.java,* dan *building.java*. kelas *user.java* terbagi kedalam 3 kelas lain yaitu kelas *administrator.java, requestor.java,* dan

receptionist.java. kelas *user.java* mewakili apa yang dapat dilakukan oleh user sistem secara umum beserta daftar atribut yang dimiliki oleh *user* sistem. Kelas *user.java* sesuai penjelasan sebelumnya di-*extends* oleh kelas *administrator.java* yang mendefinisikan apa saja yang dapat dilakukan oleh user *administrator, requestor.java* yang mendefinisikan apa saja yang dapat dilakukan oleh user *requestor,* dan *receptionist.java* yang medefinisikan apa saja yang dapat dilakukan oleh user *receptionist*. Kelas *Application.java* dalam implementasinya akan bertindak sebagai *Controllers* dalam format MVC yang dimiliki oleh kerangka kerja *PLAY!,* oleh karena itu kerangka kerja *PLAY!* mewajibkan kelas *Application.java* meng-*extends* 1 kelas khusus yaitu *Controllers.java* yang dimiliki oleh kerangka kerja *PLAY!* sebagai syarat dikenalinya kelas tersebut sebagai kelas *Controllers*.

4.3.4 Perancangan Diagram Entitas (*Entity Relationship Diagram*)

Perancangan diagram entitas ditujukan sebagai dasar membangun sebuah lingkungan basis data dari aplikasi manajemen ruang *meeting.* Gambar 4.10 berikut merepresentasikan rancangan dari diagram entitas aplikasi manajemen ruang *meeting*:



Gambar 4.10 Diagram Entitas

Seperti terlihat pada gambar 4.10 pada halaman 67 diatas, perancangan diagram entitas pada aplikasi manajemen ruang *meeting* menghasilkan 4 entitas yaitu *user, request, room,* dan *building*. Entitas *user* memiliki beberapa atribut yaitu *username, password, nik, name, email, phone_number, status,* dan *privillage*. Entitas *request* memiliki beberapa atribut yaitu *request_code, requestor_nik, building_code, date_start, meeting_title, time_start, time_end, room_layout, total_attendees, facility, snack_description,* dan *status*. Entitas *room* memiliki beberapa atribut yaitu *code, room_name, default_layout, capacity, building_code*. Entitas *building* memiliki beberapa atribut yaitu *code, building_name, address, phone_number*. Entitas *user* terhadap entitas-entitas lain dalam perancangan diagram entitas memiliki relasi "Mengelola".

4.3.5 Perancangan Antar Muka

Perancangan antar muka ditujukan untuk merancang antar muka aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Berikut beberapa hasil rancangan antar muka aplikasi manajemen ruang *meetig*:

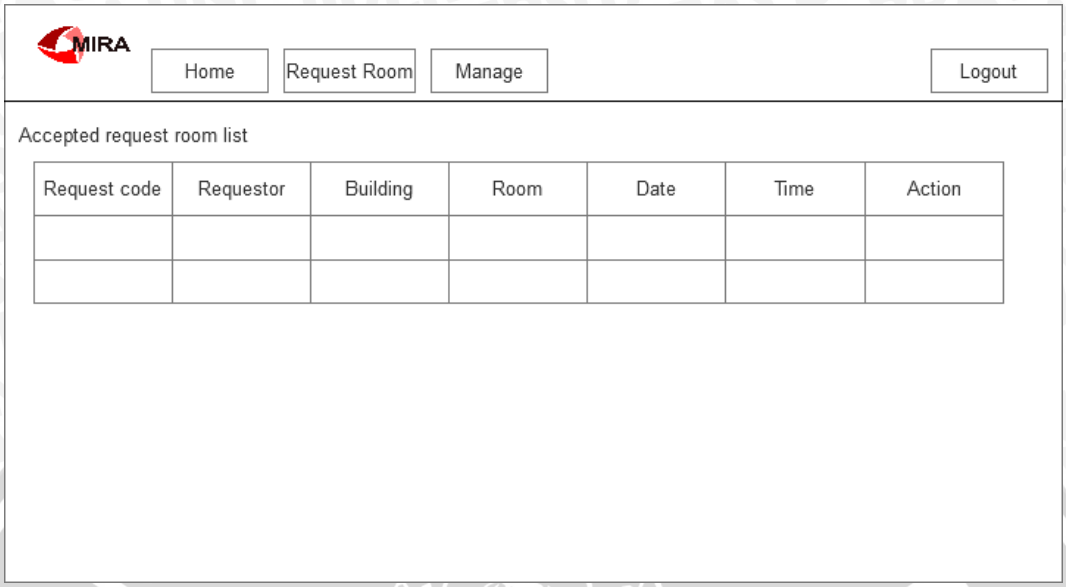
1. Login

Gambar 4.11 Halaman *Login*

Pada gambar 4.11 terlihat sebuah formulir login yang akan terdiri dari *username* dan *password* yang harus diisi oleh *guest* untuk mendapatkan hak akses kedalam aplikasi manajemen ruang *meeting* sesuai dengan *privillage*-nya yaitu *administrator, requestor,* atau *receptionist.*

2. Home

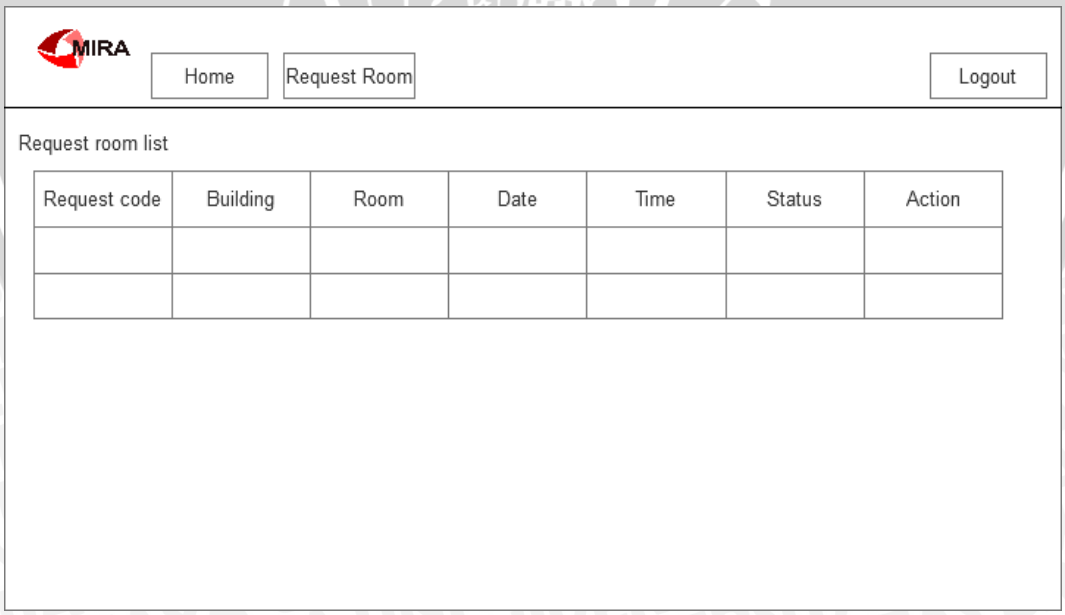
- *Administrator*



Gambar 4.12 Halaman *Home Administrator*

Pada gambar 4.12, halaman *home administrator* akan menampilkan daftar request dari *requestor* yang telah diterima oleh *administrator*. Daftar request tersebut akan ditampilkan dalam bentuk tabel sesuai dengan gambar 4.12.

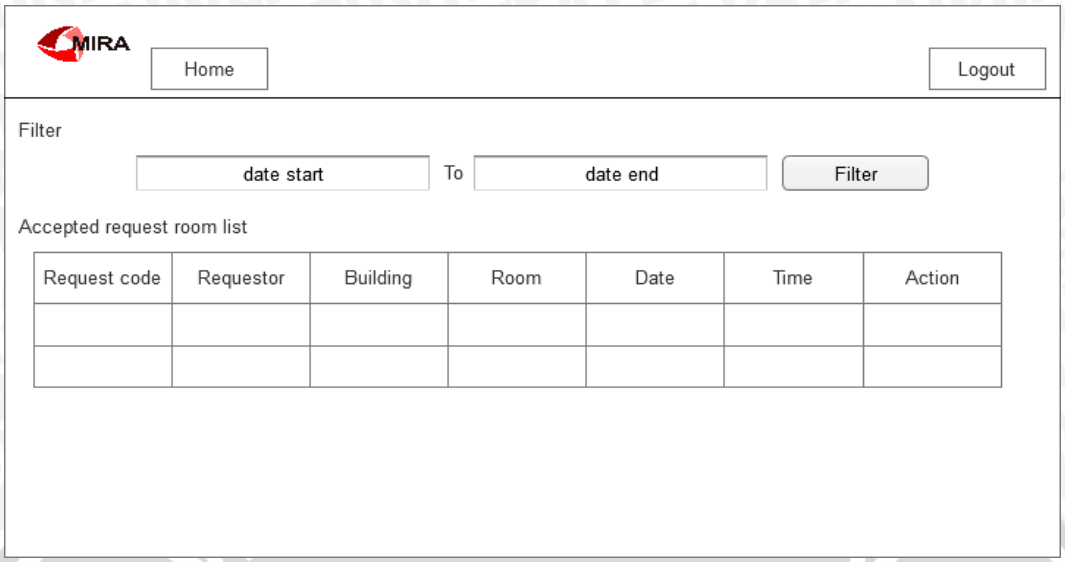
- *Requestor*



Gambar 4.13 Halaman *Home Requestor*

Gambar 4.13 diatas menunjukkan rancangan halaman *home requestor* akan menampilkan tabel daftar request yang telah dibuat oleh *requestor*. Daftar request tersebut akan ditampilkan dalam format tabel sesuai dengan gambar 4.13.

- *Receptionist*

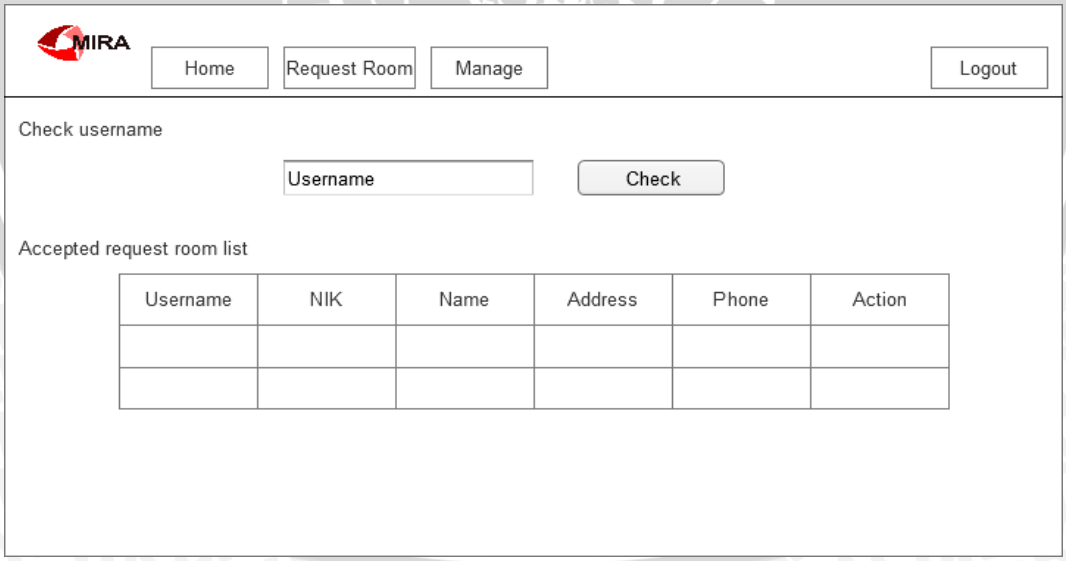


Gambar 4.14 Halaman *Home Receptionist*

Gambar 4.14 menunjukkan rancangan halaman *home receptionist* yang akan berisi informasi daftar *request* ruangan yang telah diterima oleh *administrator*. Informasi request tersebut akan ditampilkan dalam format tabel sesuai dengan gambar 4.14.

3. Add new user (*administrator*)

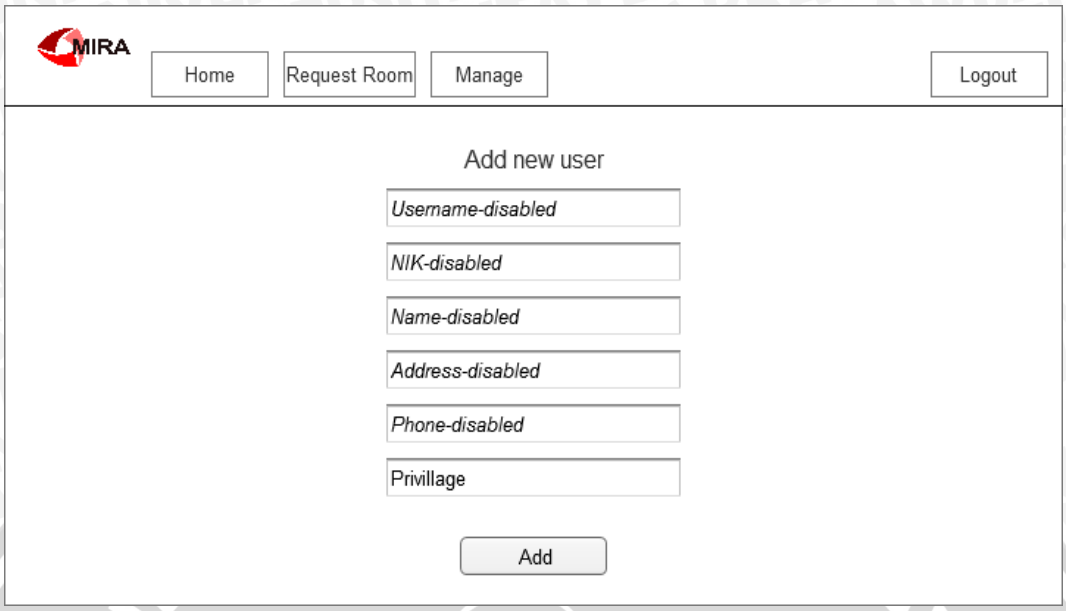
- Cek *username*



Gambar 4.15 Halaman Cek *Username* (*administrator*)

Gambar 4.15 diatas menunjukkan rancangan halaman cek username yang merupakan rangkaian dari fitur *Add New User* untuk mendapatkan data *user* baru yang akan ditambahkan kedalam sistem. *Administrator* diminta untuk meng-*input*-kan *username* yang akan ditambahkan kedalam sistem, dan hasilnya akan ditampilkan dalam tabel *Username list* sesuai gambar 4.15.

- Formulir tambah user baru

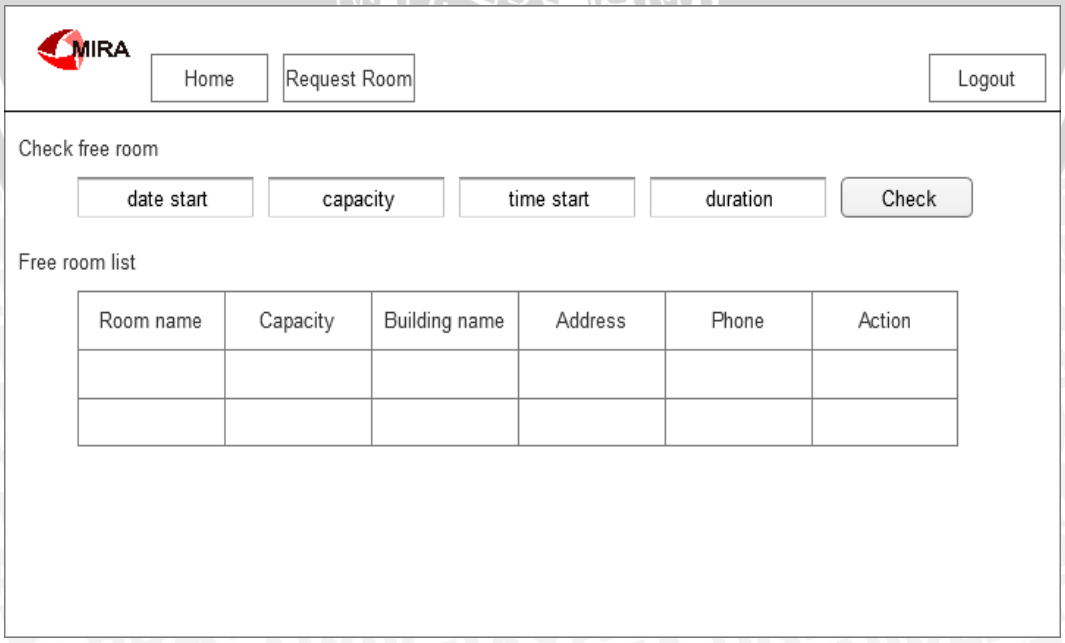


Gambar 4.16 Halaman Formulir Tambah *User* (*administrator*)

Gambar 4.16 menunjukkan rancangan halaman form tambah *user* yang merupakan rangkaian dari fitur *Add new user* yang merupakan kelanjutan dari halaman cek *username*. Pada halaman ini *administrator* hanya diminta untuk mendefinisikan *privillage* dari *user* yang dipilih pada tahap sebelumnya yaitu halaman cek *username*.

4. Booking (*requestor*)

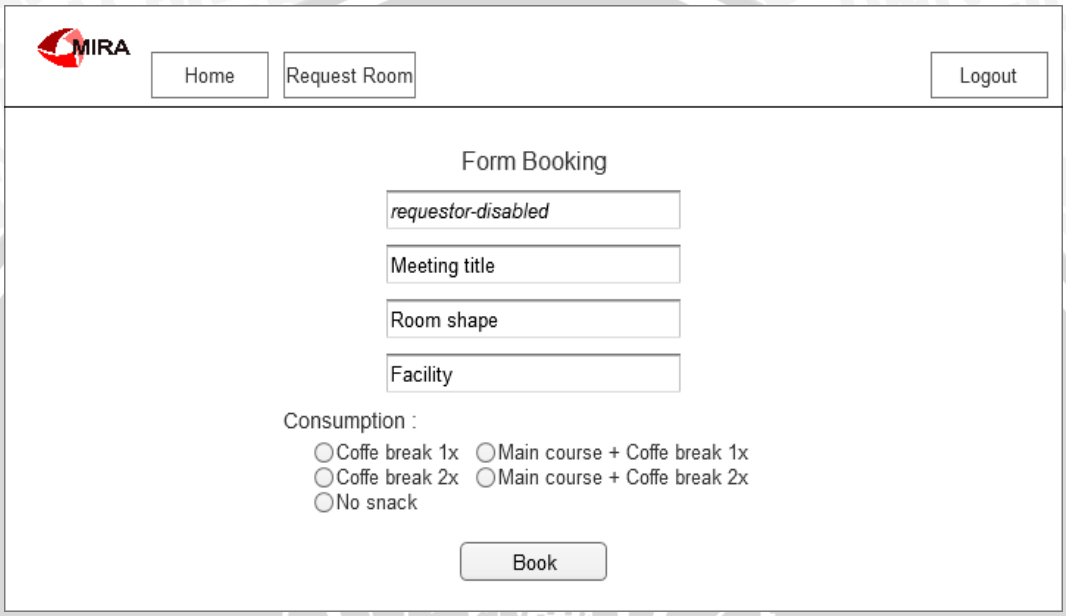
- Cek availabilitas ruangan



Gambar 4.17 Halaman cek ruangan (*requestor*)

Gambar 4.17 pada halaman 71 diatas menunjukkan rancangan halaman cek ruangan yang merupakan riangkaian dari fitur *Standart Booking* untuk mendapatkan daftar ruangan yang tersedia sesuai dengan input pada formulir *Check Free Room*. *Requestor* nantinya diminta untuk mendefinisikan kriteria dan teknis terkait pemesanan ruang *meeting*. Informasi yang didefinisikan oleh *requestor* tersebut akan ditampilkan pada tabel *Free room list* sesuai dengan gambar 4.17 diatas.

- Formulir pemesanan ruangan



Gambar 4.18 Halaman formulir pemesanan ruangan (*requestor*)

Gambar 4.18 diatas menunjukkan rancangan halaman formulir pemesanan ruangan yang merupakan rangkaian dari fitur *Booking* dan *Scheduled Booking* untuk menginputkan detail dari suatu request ruangan. Informasi detail yang diperlukan adalah judul *meeting*, layout ruangan, fasilitas, dan kriteria konsumsi yang diinginkan oleh *requestor*.

BAB 5 IMPLEMENTASI

Bab ini akan membahas mengenai implementasi aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Pembahasan pada bab ini meliputi lingkungan implementasi, implementasi antarmuka, dan implementasi kode program.

5.1 Lingkungan Implementasi

Sub bab lingkungan implementasi membahas tentang lingkungan implementasi aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. yang terdiri dari lingkungan perangkat keras dan lingkungan perangkat lunak sebagai berikut:

5.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam proses perancangan dan implementasi aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. ini adalah sebuah *Laptop* dengan spesifikasi sebagai berikut:

1. *Processor* Intel Core™ i7-3610QM CPU @ 2.3GHz
2. RAM : 4 GB
3. *System type* : 64-bit *Operating System*
4. HDD 1000 GB
5. *Monitor* 14''

5.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam proses perancangan dan implementasi aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. ini adalah sebagai berikut:

1. XAMPP v3.2.1
2. *Mozilla Firefox* v40.0.3
3. *Enterprise Architect*
4. *Axure RP Pro* v7.0
5. *Adobe Photoshop CS6*
6. *Play! Framework* v1.4.0
7. *Netbeans IDE* v8.0

5.2 Implementasi Antarmuka

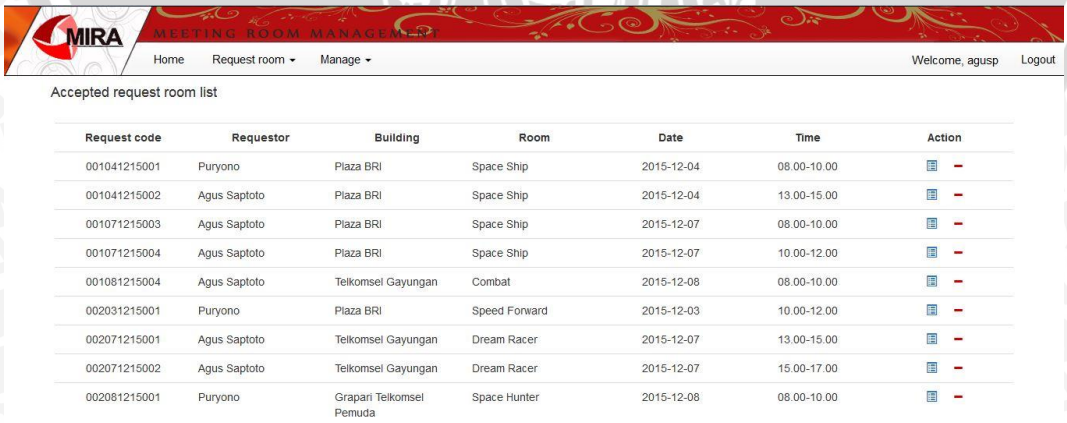
Sub bab implementasi antarmuka akan menampilkan dan membahas tentang hasil implementasi antar muka aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk. Berikut daftar tampilan antarmuka aplikasinya:

5.2.1 Tampilan Halaman *Login*

Gambar 5.1 Halaman *Login*

Halaman *login* adalah halaman yang pertama kali ditampilkan ketika *user*(*guest*) mengakses aplikasi. Sesuai dengan gambar 5.1 diatas pada halaman ini terlihat sebuah formulir login yang terdiri dari *username* dan *password* yang harus diisi oleh *guest* untuk mendapatkan hak akses kedalam aplikasi manajemen ruang *meeting* sesuai dengan *privillage*-nya yaitu *administrator, requestor,* atau *receptionist.*

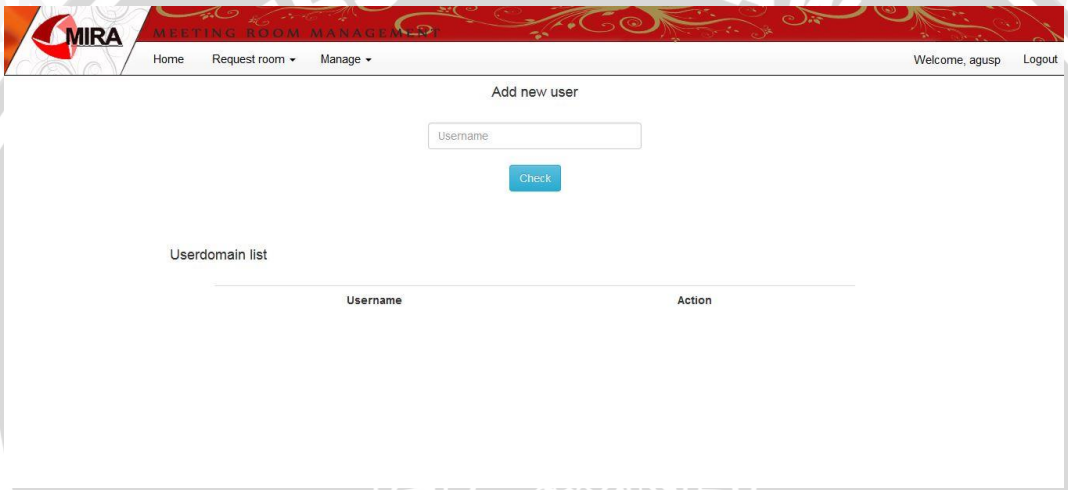
5.2.2 Tampilan Halaman *Home* (*administrator*)



Gambar 5.2 Halaman *Home Administrator*

Sesuai dengan gambar 5.2 pada halaman 74 diatas, halaman *home administrator* menampilkan daftar request dari *requestor* yang telah diterima oleh *administrator.* Daftar request tersebut ditampilkan dalam sebuah tabel yang terdiri dari 7 kolom sesuai gambar 5.2 diatas dengan kolom-kolom yaitu *Request code* yang berisi daftar kode request ruangan, *Requestor* berisi nama *requestor* yang melakukan request, *Building* berisi nama gedung dari masing-masing request, *Room* berisi nama ruangan dari masing-masing request, *Date* berisi tanggal dilaksanakannya *meeting* dari masing-masing request, *Time* berisi waktu pelaksanaan dari masing-masing request, dan *Action* yaitu kolom aksi yang dapat diaplikasikan terhadap masing-masing request yang terdiri dari aksi "detail" untuk menampilkan informasi yang lebih banyak dari suatu request dan aksi "remove" untuk membatalkan request.

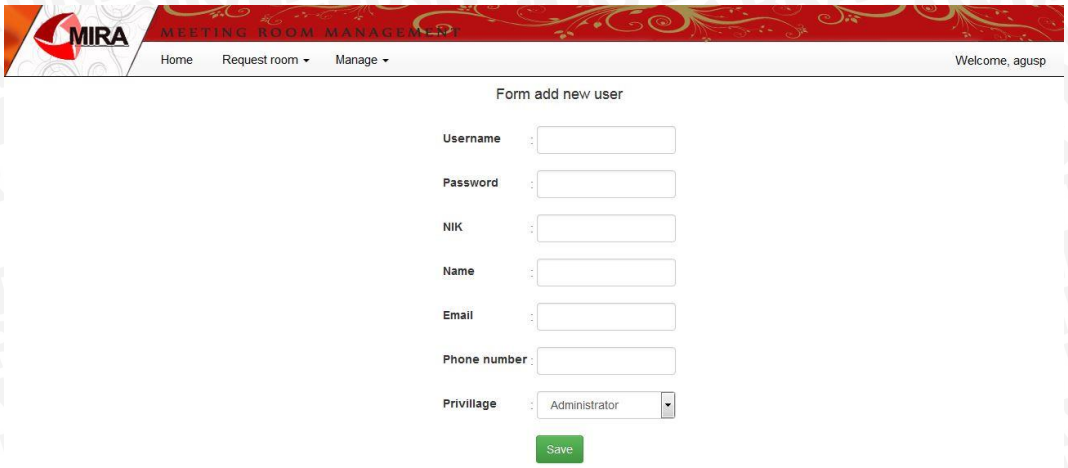
5.2.3 Tampilan Halaman Cek Username (*administrator*)



Gambar 5.3 Halaman Cek *Username* (*administrator*)

Halaman cek username adalah rangkaian dari fitur *Add New User LDAP* untuk mendapatkan data *user* baru yang akan ditambahkan kedalam sistem. Data *user* tersebut diambil dari *database* LDAP yang menyimpan data pegawai dari PT. Telkomsel Indonesia Tbk. *administrator* cukup mengisikan *username* dari *user* yang akan ditambahkan pada formulir input *username* yang terdapat pada halaman ini. Daftar *username yang* sesuai dengan input dari *administrator* akan ditampilkan pada tabel *Userdomain list* sesuai gambar 5.3 diatas.

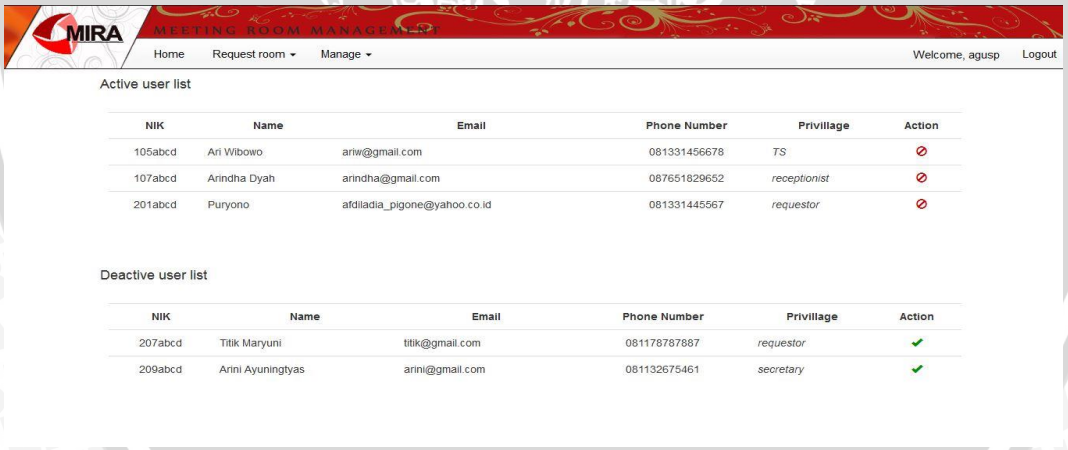
5.2.4 Tampilan Halaman Formulir Tambah *User* (*administrator*)



Gambar 5.4 Halaman Formulir Tambah *User Manual* (*administrator*)

Selain fitur *Add new user LDAP* yang mengambil data user dari basis data LDAP, terdapat pula fitur *Add new user manual* dimana fitur ini memungkinkan *administrator* untuk menambahkan user baru kedalam sistem secara manual. sesuai dengan gambar 5.4 diatas, *form add new user manual* terdiri dari 7 *field* yang harus diisi yaitu *username*, *password, NIK, name, email, phone number,* dan *privillage* dari *user* yang akan ditambahkan kedalam sistem.

5.2.5 Tampilan Halaman List *User* (*administrator*)



Gambar 5.5 Halaman List *User* (*administrator*)

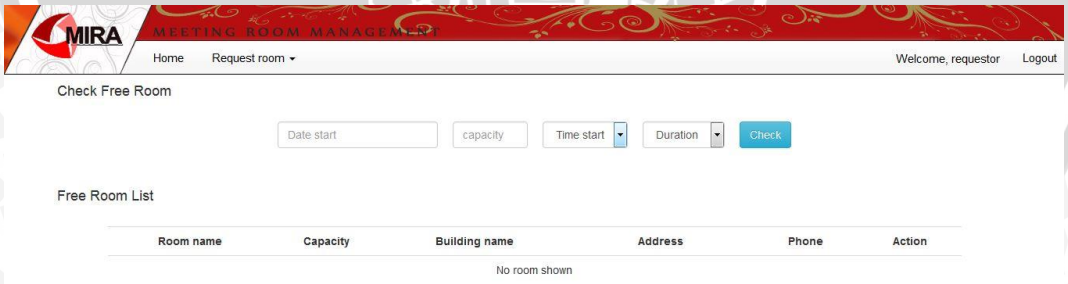
Halaman list *user* menampilkan daftar *user* yang tersimpan didalam *database* sistem. Sesuai gambar 5.5 diatas, tampilan halaman list *user* terbagi menjadi 2 tabel yaitu tabel *Active user list* dan tabel *Deactive user list.* Masing masing tabel tersusun atas 6 kolom yaitu *NIK, Name, Email, Phone number, Privillage,* dan *Action.* Kolom *Action* yaitu kolom yang berisi aksi yang dapat diaplikasikan terhadap masing-masing *user* yaitu aksi "Deactive" dengan simbol *danger* pada tabel *Active user list* untuk mendeaktifkan user sementara, dan aksi "Activate" dengan simbol centang untuk mengaktifkan user sementara pada tabel *Deactive user list.*

5.2.6 Tampilan Halaman *Home* (*requestor*)

Gambar 5.6 Halaman *Home Requestor*

Halaman *home requestor* menampilkan tabel daftar request yang telah dibuat oleh *requestor.* Sesuai dengan gambar 5.6 diatas, tabel tersebut terdiri dari 7 kolom dengan rincian kolom adalah *Request code* yang berisi daftar kode dari setiap request yang dibuat oleh *requestor, Building* berisi daftar nama gedung dari setiap request, *Room* berisi daftar nama ruangan dari setiap request, *Date* berisi daftar tanggal dilaksanakannya *meeting* dari setiap request, *Time* berisi daftar waktu dilaksanakannya *meeting* dari setiap request, *Status* berisi daftar status dari setiap request, dan *Action* yaitu kolom aksi yang dapat diaplikasikan terhadap masing-masing request yang terdiri dari aksi "detail" untuk menampilkan informasi yang lebih banyak dari suatu request dan aksi "remove" untuk membatalkan request.

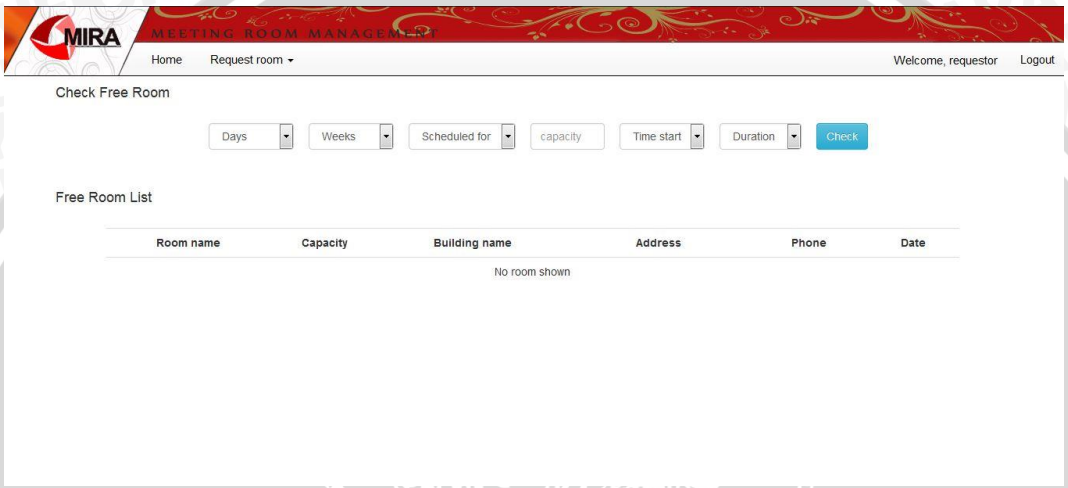
5.2.7 Tampilan Halaman Cek Ruangan (*requestor*)



Gambar 5.7 Halaman cek ruangan (*requestor*)

Halaman cek ruangan adalah riangkaian dari fitur *Standart Booking* untuk mendapatkan daftar ruangan yang tersedia sesuai dengan input pada formulir *Check Free Room*. Sesuai gambar 5.7 pada halaman 77 diatas, data input yang perlu diisikan oleh *requestor* adalah *Date start* atau tanggal akan dilaksanakannya *meeting, Capacity* atau kapasistas ruangan yang diinginkan, *Time start* atau waktu akan dimulainya *meeting*, dan *Duration* atau durasi waktu akan dilaksanakannya *meeting*. Setelah semua informasi tersebut terpenuhi, *requestor* dapat menekan *button* "Check" untuk mendapatkan daftar ruangan yang tersedia yang akan ditampilkan oleh sistem pada tabel *Free Room List.*

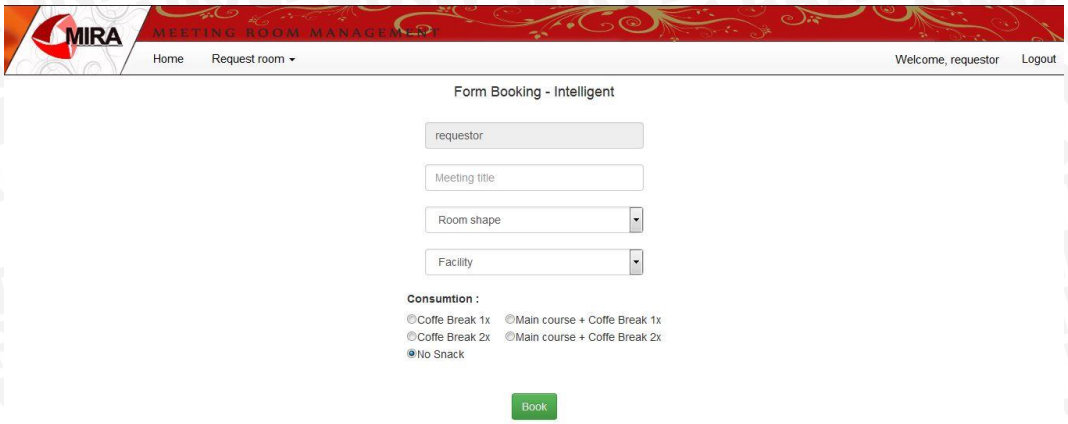
5.2.8 Tampilan Halaman Cek Ruangan Terjadwal (*requestor*)



Gambar 5.8 Halaman Cek Ruangan Terjadwal (*requestor*)

Halaman cek ruangan terjadwal adalah rangkaian dari fitur *Scheduled Booking* untuk mendapatkan daftar ruangan yang tersedia yang dipilihkan oleh sistem untuk digunakan *requestor* selama jangka waktu tertentu. tidak seperti fitur *Standart Booking* yang membebaskan *requestor* untuk memilih salah satu dari daftar ruangan yang ditampilkan sistem, fitur *Scheduled Booking* ini mewajibkan *requestor* untuk memilih semua ruangan yang ditampilkan oleh sistem karena ruangan-ruangan tersebut telah dijadwalkan oleh sistem sesuai dengan input dari *requestor* pada formulir *Check Free Room.* Sesuai dengan gambar 5.8 diatas formulir *Check Free Room* terdiri dari input opsi *Days* untuk menentukan nama hari akan dilaksanakannya *meeting, Weeks* berisi pilihan *"first", "middle", "last"* yaitu untuk menentukan kapan dalam 1 bulan akan dilaksanakan *meeting*, *Scheduled for* berisi pilihan 6 *month* dan 12 *month* yaitu untuk menentukan berapa lama *meeting* tersebut akan dijadwalkan apakah 6 bulan atau 12 bulan, *Capacity* atau kapasistas ruangan yang diinginkan, *Time start* atau waktu akan dimulainya *meeting*, dan *Duration* atau durasi waktu akan dilaksanakannya *meeting*. Setelah semua informasi tersebut terpenuhi, *requestor* dapat menekan *button* "Check" untuk mendapatkan daftar ruangan yang tersedia yang akan ditampilkan oleh sistem pada tabel *Free Room List* dimana masing-masing baris ruangan tersebut mewakili setiap bulan akan dilaksanakannya suatu *meeting.*

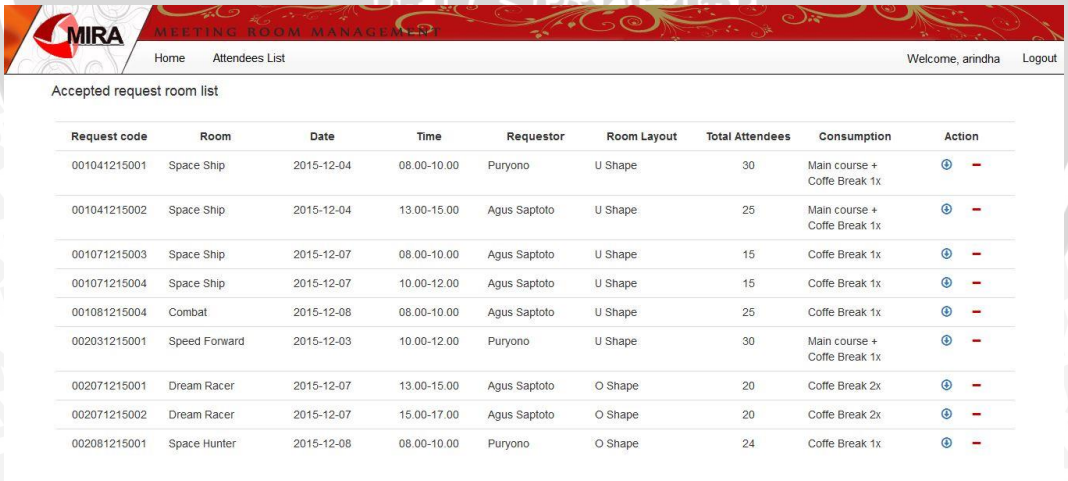
5.2.9 Tampilan Halaman Formulir Pemesanan Ruangan (*requestor*)



Gambar 5.9 Formulir Pemesanan Ruangan (*requestor*)

Halaman formulir pemesanan ruangan merupakan rangkaian dari fitur *Standart Booking* dan *Scheduled Booking* untuk menginputkan detail dari suatu request ruangan. Sesuai dengan gambar 5.9 diatas, *requestor* diminta untuk menginputkan data *Meeting title* atau judul dari suatu *meeting, Room shape* atau layout ruangan tempat akan dilaksanakannya *meeting, Facility* atau fasilitas yang disediakan didalam ruang *meeting,* dan *Snack description* atau pilihan jenis konsumsi yang akan disediakan didalam ruang *meeting*. Setelah semua terisi *requestor* dapat menyimpan data pemesanan ruang *meeting* yang telah lengkap dengan menekan *button "Book"*.

5.2.10 Tampilan Halaman *Home* (*receptionist*)



Gambar 5.10 Halaman *Home Receptionist*

Halaman *home receptionist* berisi informasi daftar *request* ruangan yang telah diterima oleh *administrator*. Sesuai dengan gambar 5.10 pada halaman 79 diatas, data *request* ruangan yang telah diterima ditampilkan dalam sebuah tabel *Accepted request room list* yang terdiri dari 9 kolom yaitu kolom *Request code* yang berisi daftar kode *request, Room* yang berisi nama ruangan dari masing-masing *request, Date* yang berisi daftar tanggal pelaksanaan *meeting, Requestor* yang berisi daftar nama *requestor* dari masing-masing *request, Room layout* yang berisi daftar layout ruangan yang akan digunakan pada masing-masing *request, Total attendees* yang berisi daftar total peserta dalam setiap *request, Consumption* yang berisi daftar konsumsi dari masing-masing *request,* dan *Action* yang berisi aksi yang dapat diaplikasikan pada masing-masing *request*. Terdapat 2 aksi yang dapat diaplikasikan pada masing-masing *request* yaitu "Download pdf" yang diwakili oleh simbol *download* untuk mengunduh daftar hadir dari masing-masing *request*, dan "Close room" yang diwakili oleh simbol minus untuk menutup ruangan yang telah selesai digunakan agar dapat tersedia kembali untuk dipesan.

5.3 Implementasi Kode Program

Pada tahap implementasi kode program akan ditampilkan dan dijelaskan 3 kode program yang masing-masing mewakili 1 fitur kompleks dari aplikasi manajemen ruang *meeting* ini. Ketiga fitur tersebut adalah *standart booking, scheduled booking, dan add new user*. Berikut hasil implementasinya

5.3.1 Kode Program Fitur *Standart Booking*

Tabel 5.1 Kode Program Fitur *Standart Booking*

```
1.  public static String[] checkRoom(String date_start,
2.  String capacity, String time_start, String duration)
3.          throws SQLException{
4.      int count = 0;
5.      String[]ruang;
6.      Connection conn = models.db_connect.connect();
7.      Statement st = conn.createStatement();
8.      Statement stm = conn.createStatement();
9.      Statement sts = conn.createStatement();
10.     ResultSet rs;
11.     String half_time = "";
12.     String [] temp_time = time_start.split("\\.");
13.     int time = Integer.parseInt(temp_time[0])+1;
14.     if(time<10){
15.         half_time = "0" + time + ".00";}
16.     else{
17.         half_time = time + ".00";}
18.
19.     ResultSet res = st.executeQuery("SELECT r.code as
20. "
21.             + "rcode,r.room_name as rname,r.capacity
22. "
23.             + "as rcapacity,r.building_code as
```

Tabel 5.1 Kode Program Fitur *Standart Booking* (lanjutan)

```
rbcode,"
                + "b.building_name as bbname,b.address as
"
                + "baddress,b.phone_number as bphone "
                + "FROM room r,building b WHERE "
                + "r.building_code=b.code AND capacity>="
                + "'"+capacity+"' ORDER BY "
                + "r.capacity,rname ASC");
      //percabangan if-else berdasarkan durasi meeting
      if(duration.equalsIgnoreCase("1")){
          while(res.next()){
              rs = stm.executeQuery("SELECT
request_code"
              + ",building_code FROM request WHERE"
              + "(MID(request_code,1,3)='"+
              res.getString("rcode")+"' AND "
              +
"building_code='"+res.getString("rbcode")+"')"
              + " AND date_start='"+date_start+"' "
              + "AND (time_start='"+time_start+"'"
              + " OR time_end='"+half_time+"') "
              + "AND status IN (0,1,2,3,5,8)");
             //menghitung total row hasil query
              if(!rs.next()){
                  count++;
              }
          }
          while(res.previous()){}
          ruang = new String[count];
          count=0;
          while(res.next()){
              rs = sts.executeQuery("SELECT
request_code"
              + ",building_code FROM request WHERE "
              + "(SUBSTRING(request_code,1,3)='"+
              res.getString("rcode")+"' AND "
              +
"building_code='"+res.getString("rbcode")+"') "
              + "AND date_start='"+date_start+"' "
              + "AND (time_start='"+time_start+"' "
              + "OR time_end='"+half_time+"') "
              + "AND status IN (0,1,2,3,5,8)");

              if(!rs.next()){
                  String r = res.getString("rname")+"-
"+
              res.getString("rcapacity")+"-"
              +res.getString("bbname")+"-"+
              res.getString("baddress")+"-
"+res.getString("bphone")
              +"-"+res.getString("rcode")
              +"-"+res.getString("rbcode");
```

Tabel 5.1 Kode Program Fitur *Standart Booking* (lanjutan)

```
76.                 ruang[count++] = r;
77.                 }
78.             }}
79.         else{
80.             while(res.next()){
81.                 rs = stm.executeQuery("SELECT
82.     request_code"
83.                 + ",building_code FROM request WHERE "
84.                 + "(MID(request_code,1,3)='"+
85.                 res.getString("rcode")+"' AND "
86.                 +
87.     "building_code='"+res.getString("rbcode")+"') "
88.                 + "AND date_start='"+date_start+"' "
89.                 + "AND (time_start='"+time_start+"' "
90.                 + "OR time_start='"+half_time+"' "
91.                 + "OR time_end='"+half_time+"')"
92.                 + "AND status IN (0,1,2,3,5,8)");
93.                 if(!rs.next()){
94.                     count++;
95.                 }
96.             }
97.             while(res.previous()){}
98.             ruang = new String[count];
99.             count=0;
100.            while(res.next()){
101.                rs = sts.executeQuery("SELECT
102.    request_code"
103.                + ",building_code FROM request WHERE "
104.                + "(SUBSTRING(request_code,1,3)='"
105.                +res.getString("rcode")+"' AND "
106.                +
107.    "building_code='"+res.getString("rbcode")+"') "
108.                + "AND date_start='"+date_start+"' "
109.                + "AND (time_start='"+time_start+"' "
110.                + "OR time_start='"+half_time+"' OR "
111.                + "time_end='"+half_time+"')"
112.                + "AND status IN (0,1,2,3,5,8)");
113.                if(!rs.next()){
114.                    String r = res.getString("rname")
115.                +"-"+res.getString("rcapacity")
116.                +"-"+res.getString("bbname")+"-"
117.                +res.getString("baddress")+"-"+
118.                res.getString("bphone")+
119.                "-"+res.getString("rcode")
120.                +"-"+res.getString("rbcode");
121.                    ruang[count++] = r;
122.                }}}
123.    return ruang;}
```

Seperti terlihat pada tabel 5.1 pada halaman 82 diatas, fitur *standart booking* diwakili oleh sebuah method dengan nama *checkRoom*. Method ini menerima parameter yaitu *date_start, capacity, time_start,* dan *duration.* Method ini mengembalikan nilai sebuah *Array* dengan tipe data *String* yang berisi informasi daftar ruangan kosong. Method ini diawali dengan menentukan nilai *half_time* pada baris 11-17 dimana nilai variabel *half_time* digunakan untuk mengecek ada tidaknya *meeting* yang dimulai pada saat *half_time* atau yang selesai pada saat *half_time*. Kemudian dilanjutkan dengan melakukan *query* untuk mendapatkan informasi seluruh ruangan yang memenuhi kualifikasi kapasitas ruangan>=*capacity* pada baris 19-31. Hasil *query* tersebut disimpan dalam variabel *res*. Setelah diperoleh daftar ruangan yang memenuhi kualifikasi, dilanjutkan dengan percabangan berdasarkan durasi waktu dari request yang dibuat. Percabangan ini perlu dilakukan untuk memastikan bahwa daftar ruangan yang nantinya akan ditampilkan ke *requestor* adalah benar-benar ruangan yang kosong dan siap untuk digunakan. Percabangan ini hanya terdiri dari 2 yaitu jika durasi request 1 jam dan jika durasi request 2 jam. Pada masing-masing percabangan jika memenuhi syarat *if* maka dilakukan *query* dengan memeriksa apakah untuk masing-masing nilai dalam variabel *res* terdapat ruangan yang masih aktif digunakan seperti terlihat pada baris 55-65 untuk durasi request 1 jam dan baris 101-112 untuk durasi request 2 jam, jika iya maka ruangan tersebut tidak akan disimpan dalam variabel *String[]ruang*. Jika tidak maka informasi ruang tersebut akan disimpan dalam variabel *String[]ruang* pada baris 76 untuk durasi request 1 jam dan 121 untuk durasi request 2 jam. Proses ini diulang sebanyak data dalam variabel *res*. Ketika semua proses telah selesai, variabel *ruang* yang berisi daftar ruangan yang kosong akan digunakan sebagai nilai *return.*

5.3.2 Kode Program Fitur *Scheduled Booking*

Tabel 5.2 Kode Program Fitur *Scheduled Booking*

```
1.  public String[] checkRoomScheduled(String days, String
2.  weeks, int schfor, String capacity, String time_start,
3.          String duration) throws SQLException {
4.      requestor req = new requestor();
5.      //menentukan nomor hari
6.      int d = 0;int count = 0;
7.      if(days.equalsIgnoreCase("Monday")){
8.          d = Calendar.MONDAY;
9.      }
10.     else if(days.equalsIgnoreCase("Tuesday")){
11.         d = Calendar.TUESDAY;
12.     }
13.     else if(days.equalsIgnoreCase("Wednesday")){
14.         d = Calendar.WEDNESDAY;
15.     }
16.     else if(days.equalsIgnoreCase("Thursday")){
17.         d = Calendar.THURSDAY;
18.     }
```

Tabel 5.2 Kode Program Fitur *Scheduled Booking* (lanjutan)

```
19.     else{
20.         d = Calendar.FRIDAY;
21.     }
22.     //-----------------------------------------------
23. ---------------------
24.     Calendar calendar = Calendar.getInstance();
25.     DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
26.     String dt = "";
27.     String [] room = new String[schfor];
28.
29.     while(schfor != 0){
30.         if(weeks.equalsIgnoreCase("first")){
31.             calendar.add(Calendar.MONTH, 1);
32.             calendar.set(Calendar.DATE, 1);
33.             while(calendar.getTime().getDay() != d-1){
34.                 calendar.add(Calendar.DATE, 1);
35.             }
36.         }
37.         else if(weeks.equalsIgnoreCase("middle")){
38.             calendar.add(Calendar.MONTH, 1);
39.             calendar.set(Calendar.DATE, 11);
40.             while(calendar.getTime().getDay() != d-1){
41.                 calendar.add(Calendar.DATE, 1);
42.             }
43.         }
44.         else{
45.             calendar.add(Calendar.MONTH, 1);
47.             calendar.set(Calendar.DATE, 21);
48.             while(calendar.getTime().getDay() != d-1){
49.                 calendar.add(Calendar.DATE, 1);
50.             }
51.         }
52.
53.         dt = df.format(calendar.getTime());
54.         String []rm = req.checkRoom(dt, capacity,
55. time_start, duration);
56.
57.         rm[0] += "-"+dt;
58.         room[count++] = rm[0];
59.         schfor--;
60.     }
61.
62.     return room;
63. }
```

Seperti terlihat pada tabel 5.2 diatas, fitur *scheduled booking* diwakili oleh method *checkRoomScheduled*. Method ini menerima parameter *days, weeks, schfor, capacity, time_start,* dan *duration*. Kembalian dari method ini adalah *Array* dengan tipe data *String* yang berisi daftar ruangan yang akan digunakan oleh *requestor* selama masa request (*schfor*) yang dibuat. Method ini diawali

dengan percabangan untuk menentukan nilai *days* dengan menggunakan modul *.text.DateFormat* seperti terlihat pada baris 7-21, nilai ini sebagai acuan hari dalam melaksanakan *meeting*. Selanjutnya dilakukan perulangan sebanyak masa request (*schfor*) yang dibuat untuk mendapatkan data ruang *meeting* yang tersedia dan akan dipilihkan oleh sistem untuk digunakan oleh *requestor*. Pada setiap perulangan akan di cek nilai dari variabel *weeks* apakah "first","middle", ataukah "last" seperti terlihat pada baris 30-51. Variabel *weeks* dengan nilai "first" mengidikasikan awal bulan yaitu tanggal 1-10 setiap bulannya, dimana pada rentang tanggal itu akan dicari manakah tanggal yang harinya sesuai dengan nilai *days*. Ketika telah diperoleh tanggal *meeting*-nya, maka dilanjutkan dengan mencari ruangan yang kosong dengan memanggil method *checkRoom* seperti terlihat pada baris 54-55. Hasil dari pemanggilan method *checkRoom* akan diambil hanya nilai pada kolom ke 0 pada setiap hasil pemanggilan method *checkRoom* dan disimpan data ruangnya kedalam variabel *String[]room* seperti terlihat pada baris 58. Proses ini diualang sebanyak nilai *schfor*. Ketika perulangan selesai method *checkRoomScheduled* akan mengembalikan nilai dari variabel *room* sebagai nilai *return*.

5.3.3 Kode Program Fitur *Add new user*

Tabel 5.3 Kode Program Fitur *Add new user*

```
1.  public static String [] checkUsername(String username)
2.  throws SQLException {
3.      Connection conn = models.db_connect.connect();
4.      Statement st =  conn.createStatement();
5.      ResultSet res = st.executeQuery("Select username
6.  FROM user WHERE username='"+username+"'");
7.
8.      if(res.next()){
9.          String [] user_data = new String[1];
10.         user_data[0] = "Identical username in system";
11.         return user_data;
12.     }
13.     else{
14.         Statement stm = conn.createStatement();
15.         ResultSet rs = stm.executeQuery("SELECT nik
16. FROM userdomain WHERE username LIKE '%"+username+"%'");
17.         if(rs.next()){
18.             rs.previous();
19.             int count=0;int total=0;
20.             while(rs.next()){
21.                 total++;
22.             }
23.             while(rs.previous()){}
24.             String [] user_data = new String[total];
25.             while(rs.next()){
26.                 user_data[count] = rs.getString("nik");
27.                 count++;
28.             }
29.             Statement sts =  conn.createStatement();
```

Tabel 5.3 Kode Program Fitur *Add new user* (lanjutan)

```
30.             ResultSet re = sts.executeQuery("Select
31. username FROM user WHERE username='"+username+"'");
32.             if(re.next()){
33.                 String [] user_dt = new String[1];
34.                 user_dt[0] = "Identical username in
35. system";
36.                 return user_dt;
37.             }
38.             else{
39.                 return user_data;
40.             }           }
41.         else{
42.             String [] user_data = new String[1];
43.             user_data[0] = "No username found";
44.             return user_data;}}}
```

seperti terlihat pada tabel 5.3 diatas, fitur *add new user* diwakili oleh method *checkUsername*. Method ini menerima parameter *username,* dan mengembalikan sebuah variabel *String[]*. Method ini diawali dengan *query* untuk mengecek apakah nilai dari variabel *username* sebagai parameter telah terdapat di dalam basis data sistem seperti terlihat pada baris 5-6, jika iya maka method akan mengembalikan nilai "Identical username in system" dimana prosesnya seperti terlihat pada baris 8-12. Jika tidak, akan dilanjutkan pengecekan *username* ke basis data LDAP (basis data user sistem PT. Telkomsel Indonesia) seperti terlihat pada baris 15-16. Untuk semua *username* yang identik akan dikembalikan oleh method melalui variabel *String[]user_data* dimana prosesnya dapat dilihat pada baris 17-40. Jika tidak terdapat satupun username yang identik, method akan mengembalikan nilai "No username found" seperti terlihat pada baris 41-44.

BAB 6 PENGUJIAN

Pada bab ini akan dibahas mengenai pengujian terhadap Aplikasi Manajemen Ruang *Meeting* pada PT. Telkomsel Indonesia Tbk. Terdapat 3 jenis pengujian yang dilakukan yaitu pengujian *Black Box,* pengujian *White Box,* dan pengujian *Usability* dengan menggunakan metode *System Usability Scale* (SUS).

6.1 Pengujian *Black Box*

Pengujian *black box* dilakukan untuk menguji apakah kebutuhan fungsional yang didefinisikan terpenuhi atau tidak. hasil pengujian *black box* dijelaskan pada tabel 6.1 berikut:

Tabel 6.1 Hasil Pengujian *Black Box*

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-01.01	*Login*	Kolom isian tidak diisi	Sistem dapat menampilkan peringatan bahwa kolom harus diisi	Sistem menampilk-an peringatan "*Please fill out this field*"	Valid
		Username dan/atau *Password* tidak sesuai	Sistem dapat menampilkan pesan yang menjelaskan bahwa *username* dan/atau *password* salah	Sistem menampilk-an pesan "*username or password is wrong*"	Valid
		Username dan *password* sesuai	Sistem akan dapat menampilkan halaman utama *user* sesuai dengan *privillage*-nya	Sistem menampilk-an halaman utama *user* sesuai dengan *privillage*-nya	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-01.02	*Logout*	Menekan *button* Logout	Sistem dapat menghapus *session user* dan men-*direct user* ke halaman login	Sistem menghapus *session* dan men-*direct user* ke halaman login	Valid
MIRA-F-01.03	*See request detail*	Memilih simbol "detail" pada salah satu baris dari daftar request ruangan	Sistem dapat menampilkan informasi lengkap dari suatu request yang dipilih	Sistem menampilkan informasi lengkap dari suatu request yang dipilih	Valid
MIRA-F-02.01	*See accepted request room list*	Sudah *login* sebagai *administrator*	Sistem dapat menampilkan halaman utama *administrator*	Sistem menampilkan halaman utama *administrator*	Valid
		Belum *login*/*login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid
MIRA-F-02.02	*See request room list*	Sudah *login* sebagai *administrator* dan memilih menu *Request room list*	Sistem dapat menampilkan halaman list request ruangan yang dibuat oleh *requestor*	Sistem menampilkan halaman list request ruangan yang dibuat oleh *requestor*	Valid
		Belum *login*/*login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-02.03	*Accept request*	Menekan simbol centang pada salah satu dari daftar request ruangan pada halaman *See request room list*	Sistem dapat merubah status request sehingga menghilang dari daftar permintaan request dan dapat tampil pada halaman utama *administrator*	Sistem merubah status request sehingga menghilang dari daftar permintaan request dan dapat tampil pada halaman utama *administrator*	Valid
MIRA-F-02.04	*See request cancel room*	Sudah *login* sebagai *administra-tor* dan memilih menu *request cancel room*	Sistem dapat menampilkan halaman request pembatalan pemesanan ruangan	Sistem menampilkan halaman request pembatalan pemesanan ruangan	Valid
		Belum *login*/*login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid
MIRA-F-02.05	*Accept cancel request*	Menekan simbol centang pada salah satu request pada halaman *cancel request list*	Sistem dapat merubah status request dan menghapus-nya dari daftar request yang diterima	Sistem merubah status request dan menghapus-nya dari daftar request yang diterima	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-02.06	*Decline cancel request*	Menekan simbol silang pada salah satu request pada halaman *cancel request list*	Sistem dapat merubah status request menjadi "remains | approved"	Sistem dapat merubah status request menjadi "remains | approved"	Valid
MIRA-F-02.07	*Manage user*	Sudah *login* sebagai *administrator* dan memilih menu Update user	Sistem dapat menampilkan halaman list user	Sistem menampilkan halaman list user	Valid
		Belum *login*/*login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid
MIRA-F-02.08	*Add new user LDAP*	Menginput-kan *username* yang tidak terdapat pada *userdomain*	Sistem dapat menampilkan pesan yang menjelaskan bahwa hasil pencarian adalah kosong	Sistem menampilkan pesan "*Username not found*"	Valid
		Menginput-kan *username* yang terdapat pada user domain	Sistem akan dapat menampilkan daftar *username* yang mirip dengan input dari *administrator*	Sistem akan menampilkan daftar *username* yang mirip dengan input dari *administrator*	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-02.09	*Add new user manual*	Tidak mengisi seluruh *field* dan memilih *button* "Save"	Sistem menampilkan pesan *error*	Sistem menampilkan pesan "*Please fill out this field*"	Valid
MIRA-F-02.10	*Activate user*	Memilih simbol *danger* pada salah satu *user* pada halaman list user, tabel *Active user*	Sistem dapat merubah status *user* dan memindah-kannya ke tabel *Deactive user*	Sistem merubah status *user* dan memindah-kannya ke tabel *Deactive user*	Valid
MIRA-F-02.11	*Deactivate user*	Memilih simbol centang pada salah satu *user* pada halaman list user, tabel *Deactive user*	Sistem dapat merubah status *user* dan memindah-kannya ke tabel *Active user*	Sistem merubah status *user* dan memindah-kannya ke tabel *Active user*	Valid
MIRA-F-02.12	*Manage building*	Sudah *login* sebagai *administra-tor* dan memilih menu *Update building*	Sistem dapat menampilkan halaman list gedung	Sistem menampilkan halaman list gedung	Valid
		Belum *login/login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-02.13	*Add new building*	Kolom isian tidak diisi	Sistem dapat menampilkan peringatan bahwa kolom harus diisi	Sistem menampilkan pesan "Please fill out this field"	Valid
		Mengisi semua kolom dan menekan *button* "Save"	Sistem dapat menampilkan pesan yang menyatakan bahwa penyimpanan berhasil	Sistem menampilkan pesan "New building has successfully saved"	Valid
MIRA-F-02.14	*Update building informati-on*	Memilih simbol *update* pada salah satu dari daftar gedung yang akan di-*update* dan melakukan perubahan data yang diinginkan	Sistem dapat menyimpan data perubahan yang dibuat oleh *administrator*	Sistem menyimpan data perubahan yang dibuat oleh *administrator*	Valid
MIRA-F-02.15	*Manage room*	Sudah *login* sebagai *administra-tor* dan memilih menu *Update room*	Sistem dapat menampilkan halaman list ruangan	Sistem menampilkan halaman list ruangan	Valid
		Belum *login*/*login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-02.16	*Add new room*	Kolom isian tidak diisi	Sistem dapat menampilkan peringatan bahwa kolom harus diisi	Sistem menampilkan pesan "Please fill out this field"	Valid
		Mengisi semua kolom dan menekan *button* "Save"	Sistem dapat menampilkan pesan yang menyatakan bahwa penyimpanan berhasil	Sistem menampilkan pesan "New room has successfully saved"	Valid
MIRA-F-02.17	*Update room information*	Memilih salah satu dari daftar ruangan yang akan di-*update* dan melakukan perubahan data yang diinginkan	Sistem dapat menyimpan data perubahan yang dibuat oleh *administrator*	Sistem menyimpan data perubahan yang dibuat oleh *administrator*	Valid
MIRA-F-02.18	*Send email notification*	Memilih simbol centang pada salah satu baris request pembatalan ruangan	Sistem dapat mengirimkan email notifikasi penerimaan pembatalan request kepada requestor	Sistem mengirimkan email notifikasi penerimaan pembatalan request kepada requestor	Valid
		Memilih simbol silang pada salah satu baris	Sistem dapat mengirimkan email notifikasi kepada requestor	Sistem mengirimkan email notifikasi kepada requestor	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-03.01	*See request list*	Sudah *login* sebagai *requestor*	Sistem dapat menampilkan halaman utama *requestor*	Sistem menampilkan halaman utama *requestor*	Valid
		Belum *login/login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid
MIRA-F-03.02	*Request cancel room*	Memilih simbol minus pada salah satu dari daftar request yang ingin diatalkan	Sistem dapat merubah status request dan menampilkan informasi batal request ke *administrator*	Sistem merubah status request dan menampilkan informasi batal request ke *administrator*	-
MIRA-F-03.03	*Standart Booking*	Kolom isian tidak diisi	Sistem dapat menampilkan peringatan bahwa kolom harus diisi	Sistem menampilkan pesan "*Please fill out this field*"	Valid
		Semua kolom diisi dan memilih *button* "Check"	Sistem dapat menampilkan daftar ruangan yang tersedia untuk dipesan	Sistem menampilkan daftar ruangan yang tersedia untuk dipesan	Valid
		Memilih simbol centang pada salah satu dari daftar ruangan kosong	Sistem dapat men-*direct requestor* ke halaman formulir detail *meeting*	Sistem men-*direct requestor* ke halaman formulir detail *meeting*	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-03.03	*Standart Booking*	Kolom isian pada halaman formulir pemesanan tidak diisi	Sistem dapat menampilkan peringatan bahwa kolom harus diisi	Sistem menampilkan pesan "*Please fill out this field*"	Valid
		Mengisi semua kolom pada halaman formulir pemesanan ruangan dan menekan *button* "Book"	Sistem dapat menyimpan request pemesanan ruang *meeting* kedalam *database* dan menampilkan pesan bahwa penyimpanan berhasil	Sistem menyimpan request pemesanan ruang *meeting* kedalam *database* dan menampilkan pesan "Your request has successfully saved"	Valid
MIRA-F-03.04	*Scheduled Booking*	Kolom isian tidak diisi	Sistem dapat menampilkan peringatan bahwa kolom harus diisi	Sistem menampilkan pesan "*Please fill out this field*"	Valid
		Semua kolom diisi dan memilih *button* "Check"	Sistem dapat menampilkan daftar ruangan yang tersedia untuk digunakan oleh *requestor* selama masa pemesanan	Sistem menampilkan daftar ruangan yang tersedia untuk digunakan oleh *requestor* selama masa pemesanan	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-03.04	*Scheduled Booking*	Memilih *button* "Book"	Sistem dapat men-*direct requestor* ke halaman formulir detail *meeting*	Sistem men-*direct requestor* ke halaman formulir detail *meeting*	Valid
		Kolom isian pada halaman formulir pemesanan tidak diisi	Sistem dapat menampilkan peringatan bahwa kolom harus diisi	Sistem menampilkan pesan "*Please fill out this field*"	Valid
		Mengisi semua kolom pada halaman formulir pemesanan ruangan dan menekan *button* "Book"	Sistem dapat menyimpan request pemesanan ruang *meeting* kedalam *database* dan menampilkan pesan bahwa penyimpanan berhasil	Sistem menyimpan request pemesanan ruang *meeting* kedalam *database* dan menampilkan pesan "Your request has successfully saved"	Valid
MIRA-F-04.01	*See accepted request room list receptionist*	Sudah *login* sebagai *receptionist*	Sistem dapat menampilkan halaman utama *receptionist*	Sistem menampilkan halaman utama *receptionist*	Valid
		Belum *login/login* gagal	Sistem dapat menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid

Tabel 6.1 Hasil Pengujian *Black Box* (lanjutan)

Kode Kebutuhan	Nama Pengujian	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat	Status
MIRA-F-04.02	*Filter*	Mengisi salah satu dari 2 *field* dalam form filter request	Sistem dapat menampilkan pesan error	Sistem menampilkan pesan "Please fill out this field"	Valid
		Mengisi kedua *field* dalam form filter request dengan tanggal diluar tanggal request aktif	Sistem dapat menampilkan pesan yang menyatakan tidak ada request yang aktif untuk tanggal yang dipilih	Sistem menampilkan pesan "No request to show"	Valid
		Mengisi kedua *field* dalam form filter request dengan rentang tanggal request aktif	Sistem dapat menampilkan daftar request ruangan yang aktif pada rentang tanggal yang dipilih	Sistem menampilkan daftar request ruangan yang aktif pada rentang tanggal yang dipilih	Valid
MIRA-F-04.03	*Donwload pdf*	Memilih simbol "download" pada salah satu request	Sistem dapat menampilkan file daftar hadir suatu ruangan dalam format pdf ke layar	Sistem menampilkan file daftar hadir suatu ruangan dalam format pdf ke layar	Valid
MIRA-F-04.04	*Close room*	Memilih simbol minus pada salah satu request	Sistem dapat menutup pemesanan ruangan	Sistem menutup pemesanan ruangan	Valid

6.2 Pengujian *White Box*

Pengujian *white box* dilakukan dengan pengujian *basis path.* Dalam sub bab ini akan dilakukan pengujian terhadap beberapa fitur utama Aplikasi Manajemen Ruang *Meeting* pada PT. Telkomsel Indonesia Tbk. diantaranya adalah fitur *Standart Booking, Scheduled Booking,* dan *Add new user*. Pengujian *basis path* dapat dilihat pada gambar 6.1 sampai 6.3 berikut:

6.2.1 Pengujian *White Box* Fitur Standart Booking

Pseudocode fitur *standart booking* dapat dilihat pada tabel 6.2 berikut:

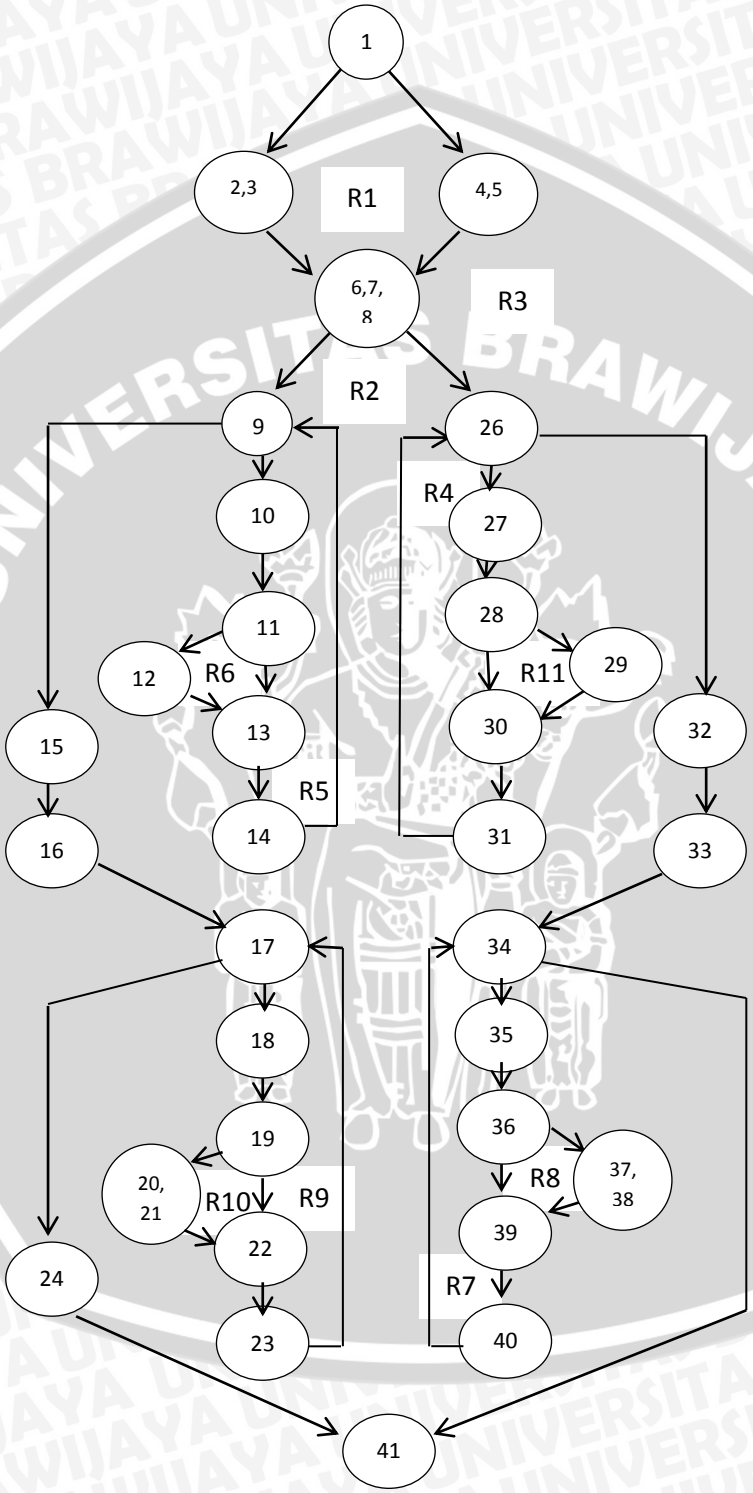
Tabel 6.2 Tabel *Pseudocode* Fitur *Standart Booking*

```
1.  IF time<10 THEN
2.          Set half_time
3.  END
4.  ELSE
5.          Set half_time
6.  END IF
7.  Res = Select clause untuk mendapatkan capacity>=capacity_defined
8.  IF duration==1 THEN
9.          WHILE res.next
10.                 Rs = Select clause untuk mendapatkan list ruangan yang terpesan
11.                 IF !Rs.next
12.                         Count++
13.                 END IF
14.         END WHILE
15.         Ruang  = new String[count]
16.         Count = 0
17.         WHILE res.next
18.                 Rs = Select clause untuk mendapatkan list ruangan yang terpesan
19.                 IF !Rs.next
20.                         String r = data ruangan
21.                         Ruang[count++] =  r
22.                 END IF
23.         END WHILE
24. END
25. ELSE
26.         WHILE res.next
27.                 Rs = Select clause untuk mendapatkan ruangan yang terpesan, durasi>1
28.                 IF !Rs.next
29.                         Count++
30                  END IF
31.         END WHILE
32.         Ruang  = new String[count]
33.         Count = 0
34.         WHILE res.next
35.                 Rs = Select clause untuk mendapatkan ruangan yang terpesan, durasi>1
36.                 IF !Rs.next
37.                         String r = data ruangan
38.                         Ruang[count++] =  r
39.                 END IF
40.         END WHILE
```

Tabel 6.2 *Pseudocode* Fitur *Standart Booking* (lanjutan)

41.	END IF

Flow graph fitur *standart booking* dapat dilihat pada gambar 6.1 berikut:



Gambar 6.1 *Flow Graph* Fitur *Standart Booking*

Perhitungan *Cyclomatic Complexity* untuk gambar 6.1 adalah sebagai berikut:

V(G) = E – N + 2 = 43 – 34 + 2 = 11

V(G) = P + 1 = 10 + 1 =11

V(G) = Jumlah Region = 11

Independent Path:

1. 1-2-3-6-7-8-9-15-16-17-24-41
2. 1-2-3-6-7-8-26-32-33-34-41
3. 1-4-5-6-7-8-9-15-16-17-24-41
4. 1-4-5-6-7-8-26-32-33-34-41
5. 1-2-3-6-7-8-9-10-11-13-14-9-15-16-17-18-19-22-23-17-24-41
6. 1-2-3-6-7-8-9-10-11-12-13-14-9-15-16-17-18-19-22-23-17-24-41
7. 1-2-3-6-7-8-26-27-28-30-31-26-32-33-34-35-36-39-40-34-41
8. 1-2-3-6-7-8-26-27-28-29-30-31-26-32-33-34-35-36-37-38-39-40-34-41
9. 1-4-5-6-7-8-9-10-11-13-14-9-15-16-17-18-19-22-23-17-24-41
10. 1-4-5-6-7-8-9-10-11-12-13-14-9-15-16-17-18-19-22-23-17-24-41
11. 1-4-5-6-7-8-26-27-28-29-30-31-26-32-33-34-35-36-37-38-39-40-34-41

Tabel 6.3 Unit Testing Fitur *Standart Booking*

No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
1	1, 2, 3, 6, 7, 8, 9, 15, 16, 17, 24, 41	Tanggal=14/07/2016,kapasitas ruang=150, jam rapat=08.00, durasi rapat=1 jam	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Valid
2	1, 2, 3, 6, 7, 8, 26, 32, 33, 34, 41	Tanggal=14/07/2016,kapasitas ruang=150, jam rapat=08.00, durasi rapat=2 jam	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Valid
3	1, 4, 5, 6, 7, 8, 9, 15, 16, 17, 24, 41	Tanggal=14/07/2016,kapasitas ruang=150, jam rapat=10.00, durasi rapat=1 jam	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Valid

Tabel 6.3 Unit Testing Fitur *Standart Booking* (lanjutan)

No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
4	1, 4, 5, 6, 7, 8, 26, 32, 33, 34, 41	Tanggal=14/07/2016,kapasitas ruang =150, jam rapat=10:00, durasi rapat=2 jam	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Valid
5	1, 2, 3, 6, 7, 8, 9, 10, 11, 13, 14, 9, 15, 16, 17, 18, 19, 22, 23, 17, 24, 41	Tanggal=14/07/2016,kapasitas ruang = 50, jam rapat=08:00, durasi rapat=1 jam	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Valid
6	1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 9, 15, 16, 17, 18, 19, 22, 23, 17, 24, 41	Tanggal=14/07/2016,kapasitas ruang 50, jam rapat=09:00, durasi rapat=1 jam	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Valid
7	1, 2, 3, 6, 7, 8, 26, 27, 28, 30, 31, 26, 32, 33, 34, 35, 36, 39, 40, 34, 41	Tanggal=14/07/2016,kapasitas ruang=50, jam rapat=08:00, durasi rapat=2 jam	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Valid
8	1, 2, 3, 6, 7, 8, 26, 27, 28, 29, 30, 31, 26, 32, 33, 34, 35, 36, 37, 38, 39, 40, 34, 41	Tanggal=14/07/2016,kapasitas ruang=50, jam rapat=09:00, durasi rapat=2 jam	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Valid
9	1, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 9, 15, 16, 17, 18, 19, 22, 23, 17, 24, 41	Tanggal=14/07/2016,kapasitas ruang=50, jam rapat=13:00, durasi rapat=1 jam	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Sistem tidak menampilkan daftar ruangan *meeting* yang tersedia	Valid

Tabel 6.3 Unit Testing Fitur *Standart Booking* (lanjutan)

No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
10	1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 9, 15, 16, 17, 18, 19, 22, 23, 17, 24, 41	Tanggal=14/07/2016,kapasitas ruan=50, jam rapat=15:00, durasi rapat=1 jam	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Valid
11	1, 4, 5, 6, 7, 8, 26, 27, 28, 29, 30, 31, 26, 32, 33, 34, 35, 36, 37, 38, 39, 40, 34, 41	Tanggal=14/07/2016,kapasitas ruang=50, jam rapat=15:00, durasi rapat=2 jam	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Sistem menampilkan daftar ruangan *meeting* yang tersedia	Valid

6.2.2 Pengujian *White Box* Fitur *Scheduled Booking*

Pseudocode fitur *scheduled booking* dapat dilihat pada tabel 6.4 berikut:

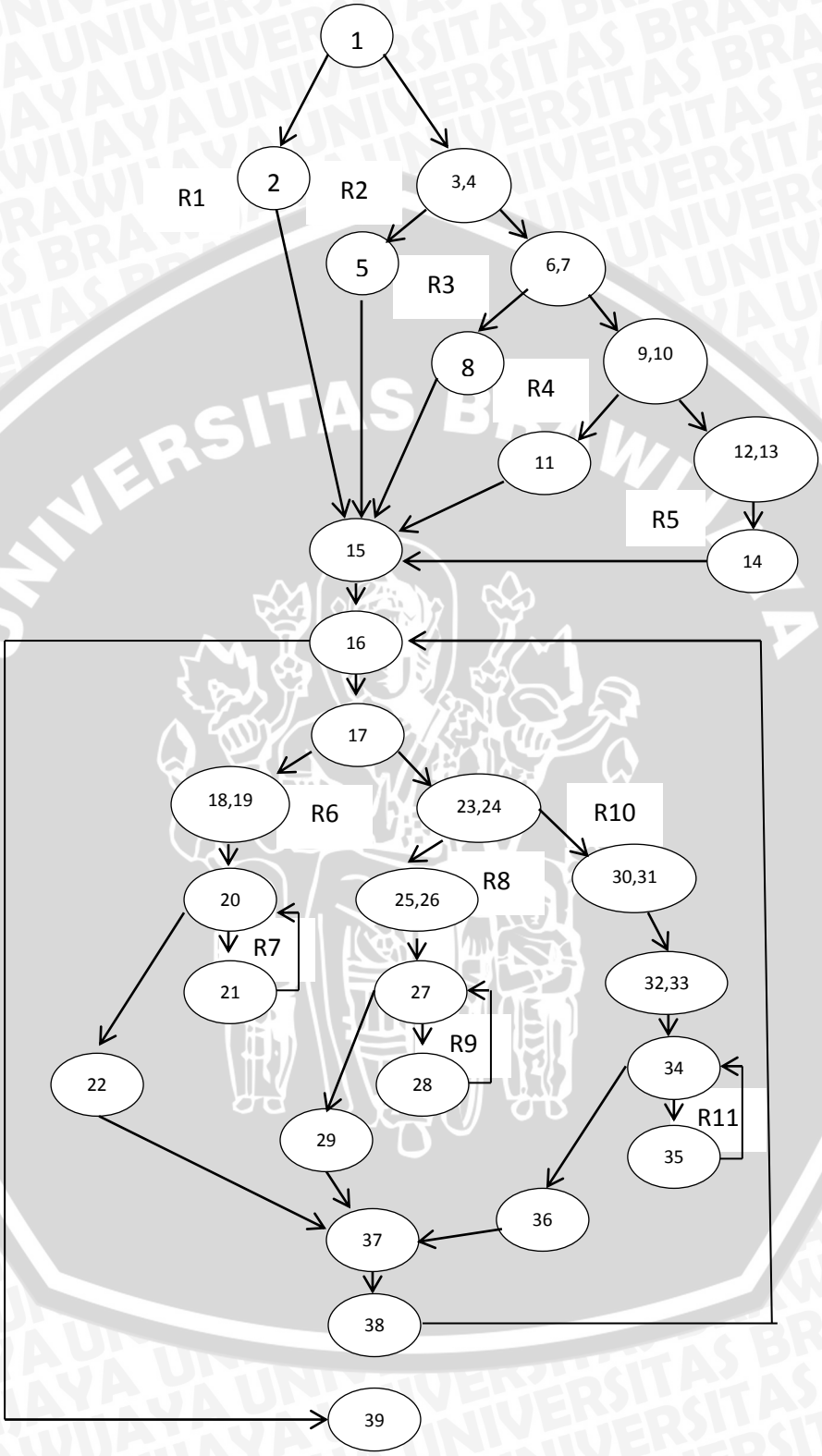
Tabel 6.4 *Pseudocode* Fitur *Scheduled Booking*

```
1.  IF days==”Monday”
2.          Set calendar d ke Monday
3.  END
4.  ELSE IF days==”Tuesday”
5.          Set calendar d ke Tuesday
6.  END
7.  ELSE IF days==”Wednesday”
8.          Set calendar d ke Wednesday
9.  END
10. ELSE IF days==”Thursday”
11.         Set calendar d ke Thursday
12. END
13. ELSE
14.         Set calendar d ke Friday
15. END IF
16. WHILE schfor !=0
17.         IF weeks==”first”
18.                 Tambah calendar d 1 bulan kedepan
19.                 Set tanggal calendar d ke 1
20.                 WHILE calendar date.day != d
21.                         Tambah calendar d 1 hari
22.                 END WHILE
```

Tabel 6.4 *Pseudocode* Fitur *Scheduled Booking* (lanjutan)

```
23.         END
24.         ELSE IF weeks==”middle”
25.                 Tambah calendar d 1 bulan kedepan
26.                 Set tanggal calendar d ke 11
27.                 WHILE calendar date.day != d
28.                         Tambah calendar d 1 hari
29.                 END WHILE
30.         END
31.         ELSE
32.                 Tambah calendar d 1 bulan kedepan
33.                 Set tanggal calendar d ke 21
34.                 WHILE calendar date.day != d
35.                         Tambah calendar d 1 hari
36.                 END WHILE
37.         END IF
38.         Panggil Method checkRoom(date, capacity,time_start,duration)
39. END WHILE
```

Flow graph fitur *Scheduled booking* dapat dilihat pada gambar 6.2 berikut:



Gambar 6.2 *Flow Graph* Fitur *Scheduled Booking*

Perhitungan *Cyclomatic Complexity* untuk gambar 6.2 adalah sebagai berikut:

V(G) = E – N + 2 = 39 – 30 + 2 = 11

V(G) = P + 1 = 10 + 1 =11

V(G) = Jumlah Region = 11

Independent Path:

1. 1-2-15-16-17-18-19-20-21-20-22-37-38-16-39
2. 1-3-4-5-15-16-17-18-19-20-21-20-22-37-38-16-39
3. 1-3-4-6-7-8-15-16-17-18-19-20-21-20-22-37-38-16-39
4. 1-3-4-6-7-9-10-11-15-16-17-18-19-20-21-20-22-37-38-16-39
5. 1-3-4-6-7-9-10-12-13-14-15-16-17-18-19-20-21-20-22-37-38-16-39
6. 1-2-15-16-17-23-24-25-26-27-28-27-29-37-38-16-39
7. 1-3-4-5-15-16-17-23-24-25-26-27-28-27-29-37-38-16-39
8. 1-3-4-6-7-8-15-16-17-23-24-25-26-27-28-27-29-37-38-16-39
9. 1-3-4-6-7-8-9-10-11-15-16-17-23-24-25-26-27-28-27-29-37-38-16-39
10. 1-3-4-6-7-8-9-10-11-12-13-14-15-16-17-23-24-25-26-27-28-27-29-37-38-16-39
11. 1-2-15-16-17-23-24-30-31-32-33-34-35-34-36-37-38-16-39

Tabel 6.5 Unit Testing Fitur *Scheduled Booking*

No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
1	1, 2, 15, 16, 17, 18, 19, 20, 21, 20, 22, 37, 38, 16, 39	Days=”*Monday*” , weeks=”*first*”, schfor=”6”/”12”	Sistem menampilkan daftar ruangan yang tersedia untuk hari senin pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari senin pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Valid

Tabel 6.5 Unit Testing Fitur *Scheduled Booking* (lanjutan)

No	Jalur	Data input	Hasil yang diharapkan	Hasil yang diperoleh	Status
2	1, 3, 4, 5, 15, 16, 17, 18, 19, 20, 21, 20, 22, 37, 38, 16, 39	Days="*Tuesday*", weeks="*first*", schfor="6"/"12"	Sistem menampilkan daftar ruangan yang tersedia untuk hari selasa pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari selasa pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Valid
3	1, 3, 4, 6, 7, 8, 15, 16, 17, 18, 19, 20, 21, 20, 22, 37, 38, 16, 39	Days="*Wednesday*", weeks="*first*", schfor="6"/"12"	Sistem menampilkan daftar ruangan yang tersedia untuk hari rabu pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari rabu pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Valid
4	1, 3, 4, 6, 7, 9, 10, 11, 15, 16, 17, 18, 19, 20, 21, 20, 22, 37, 38, 16, 39	Days="*Thursday*", weeks="*first*", schfor="6"/"12"	Sistem menampilkan daftar ruangan yang tersedia untuk hari kamis pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari kamis pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Valid
5	1, 3, 4, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 20, 22, 37, 38, 16, 39	Days="*Friday*", weeks="*first*", schfor="6"/"12"	Sistem menampilkan daftar ruangan yang tersedia untuk hari jum'at pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari jum'at pada rentang tanggal 1-10 setiap bulannya sebanyak nilai schfor	Valid

Tabel 6.5 Unit Testing Fitur *Scheduled Booking* (lanjutan)

No	Jalur	Data input	Hasil yang diharapkan	Hasil yang diperoleh	Status
6	1, 2, 15, 16, 17, 23, 24, 25, 26, 27, 28, 27, 29, 37, 38, 16, 39	Days=”*Monday”,* weeks=”*middle*”, schfor=”6”/”12”	Sistem menampilkan daftar ruangan yang tersedia untuk hari senin pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari senin pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Valid
7	1, 3, 4, 5, 15, 16, 17, 23, 24, 25, 26, 27, 28, 27, 29, 37, 38, 16, 39	Days=”*Tuesday”,* weeks=”*middle*”, schfor=”6”/”12”	Sistem menampilkan daftar ruangan yang tersedia untuk hari selasa pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari selasa pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Valid
8	1, 3, 4, 6, 7, 8, 15, 16, 17, 23, 24, 25, 26, 27, 28, 27, 29, 37, 38, 16, 39	Days=”*Wednesday”,* weeks=”*middle*”, schfor=”6”/”12”	Sistem menampilkan daftar ruangan yang tersedia untuk hari rabu pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari rabu pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Valid

Tabel 6.5 Unit Testing Fitur *Scheduled Booking* (lanjutan)

No	Jalur	Data input	Hasil yang diharapkan	Hasil yang diperoleh	Status
9	1, 3, 4, 6, 7, 9, 10, 11, 15, 16, 17, 23, 24, 25, 26, 27, 28, 27, 29, 37, 38, 16, 39	Days="*Thursday*", weeks="*middle*", schfor="6"/"12"	Sistem menampilkan daftar ruangan yang tersedia untuk hari kamis pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari kamis pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Valid
10	1, 3, 4, 6, 7, 9, 10, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27, 28, 27, 29, 37, 38, 16, 39	Days="*Friday*", weeks="*middle*", schfor="6"/"12"	Sistem menampilkan daftar ruangan yang tersedia untuk hari jum'at pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari jum'at pada rentang tanggal 11-20 setiap bulannya sebanyak nilai schfor	Valid
11	1, 2, 15, 16, 17, 23, 24, 30, 31, 32, 33, 34, 35, 34, 36, 37, 38, 16, 39	Days="*Monday*", weeks="*last*", schfor="6"/"12"	Sistem menampilkan daftar ruangan yang tersedia untuk hari senin pada rentang tanggal 21-30 setiap bulannya sebanyak nilai schfor	Sistem menampilkan daftar ruangan yang tersedia untuk hari jum'at pada rentang tanggal 21-30 setiap bulannya sebanyak nilai schfor	Valid

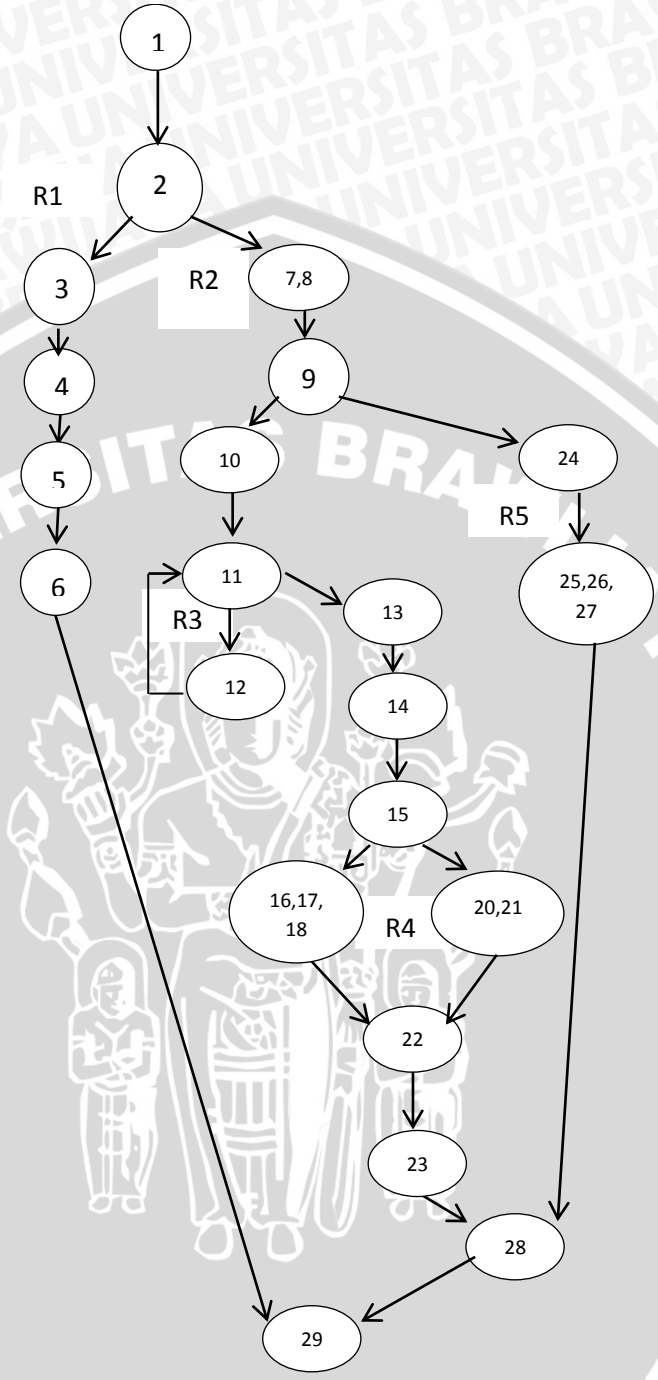
6.2.3 Pengujian *White Box* Fitur *Add New User*

Pseudocode fitur *Add new user* dapat dilihat pada tabel 6.6 berikut:

Tabel 6.6 *Pseudocode* Fitur *Add New User*

```
1.  Res = get username dari database sistem
2.  IF Res.next
3.          User_data = new String[1]
4.          User_data[0]= pesan
5.          Return User_data
6.  END
7.  ELSE
8.          Rs = get nik dari database LDAP
9.          IF Rs.next
10.                 User_data = new String[count]
11.                 WHILE Rs.next
12.                         User_data[count++] =Rs.nik
13.                 END WHILE
14.                 Re = get username dari database sistem
15.                 IF Re.next
16.                         User_dt = new String[1]
17.                         User_dt[0]= pesan
18.                         Return User_dt
19.                 END
20.                 ELSE
21.                         Return User_data
22.                 END IF
23.         END
24.         ELSE
25.                 User_data = new String[1]
26.                 User_data[0]=pesan
27.                 Return Usert
28.         END IF
29. END IF
```

Flow graph fitur *Add new user* dapat dilihap pada gambar 6.3 berikut:



Gambar 6.3 *Flow Graph* Fitur *Add new user*

Perhitungan *Cyclomatic Complexity* untuk gambar 6.3 adalah sebagai berikut:

V(G) = E – N + 2 = 25 – 22 + 2 = 5

V(G) = P + 1 = 4 + 1 = 5

V(G) = Jumlah Region = 5

Independent Path:

1. 1-2-3-4-5-6-29
2. 1-2-7-8-9-10-11-12-13-14-15-16-17-18-22-23-28-29
3. 1-2-7-8-9-10-11-12-13-14-15-20-21-22-23-28-29
4. 1-2-7-8-9-10-11-13-14-15-20-21-22-23-28-29
5. 1-2-7-8-9-24-25-26-27-28-29

Tabel 6.7 Unit Testing Fitur *Add New User*

No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
1	1, 2, 3, 4, 5, 6, 29	*Username* = "agusp"	Sistem menampilkan pesan "*Identical username in the system*"	Sistem menampilkan pesan "*Identical username in the system*"	Valid
2	1, 2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 22, 23, 28, 29	*Username* = "diana"	Sistem menampilkan pesan "*Identical username in the system*"	Sistem menampilkan pesan "*Identical username in the system*"	Valid
3	1, 2, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29	*Username* = "diana"	Sistem menampilkan daftar *username* didalam *database* LDAP yang sesuai	Sistem menampilkan daftar *username* didalam *database* LDAP yang sesuai	Valid
4	1, 2, 7, 8, 9, 10, 11, 13, 14, 15, 20, 21, 22, 23, 28, 29	*Username* = "diana"	Sistem menampilkan data kosong	Sistem menampilkan data kosong	Valid
5	1, 2, 7, 8, 9, 24, 25, 26, 27, 28, 29	*Username* = "afdinfadila"	Sistem menampilkan pesan "*no username found*"	Sistem menampilkan pesan "*no username found*"	Valid

6.3 Pengujian *Usability*

Pada sub-bab pengujian *usability*, dilakukan sebuah pengujian *usability* aplikasi manajemen ruang *meeting* dengan menggunakan metode *System Usability Scale* (SUS) dengan hasil akhir sebuah *adjective* yang mewakili sebuah *range* skor tertentu.

Pengujian *usability* aplikasi manajemen ruang *meeting* dilaksanakan pada tanggal 20 Juni 2016 di Gedung BRI Tower Lt. 16 JL. Basuki Rahmat No.122, Surabaya. Pengujian dimulai pada pukul 14.20 dan selesai pada pukul 17.05. Pengujian *usability* dengan menggunakan metode SUS ini melibatkan 18 responden dari 20 responden yang dijadwalkan hadir. 20 responden dipilih karena telah melebihi batas minimal pengujian dengan SUS yaitu 8-12 orang (Tullis dan stetson,2004). Tabel 6.8 berikut adalah rincian dari ke-18 responden:

Tabel 6.8 Data Responden Pengujian *Usability*

No.	NIP	Nama	Departemen
1.	68268	Aji Hermawan	IT Operation Support
2.	82091	Galih Hari Wibowo	IT Operation Support
3.	84120	Sugeng D Winanjuar	IT Operation Support
4.	84350	Hidayat Purnama	IT Operation Support
5.	88331	Ahmad Sugeng S W	IT Operation Support
6.	76401	Rina Yulianti	IT Operation Support
7.	82662	Andika Setya Purnama	IT Operation Support
8.	65440	Puryono	General Affair
9.	85501	Winda Ayu Ningrum	General Affair
10.	85522	Cintya Pangestika	General Affair
11.	85515	Andreas Nugraha	General Affair
12.	75052	Ruben Pius Jordy	Human Resource Dev.
13.	82803	Adrian Pradipto	Human Resource Dev.
14.	84605	Bayu Setyawan	Human Resource Dev.
15.	80625	Agus Nugroho	Human Resource Dev.
16.	88858	Brendha Agnes P	Human Resource Dev.
17.	78890	Yayuk Apriliani	Finance Affair
18.	75021	Ariyanto	Finance Affair

Prosedur pengujian *usability* adalah, responden diminta untuk menyelesaikan beberapa tugas menggunakan aplikasi manajemen ruang *meeting* yang telah di *deploy* kedalam server milik PT. Telkomsel Indonesia Tbk Regional Jawa Timur. Daftar kegiatan yang harus diselesaikan oleh responden sesuai dengan tabel 6.9 berikut:

Tabel 6.9 Kegiatan Pengujian *Usability*

No.	Kegiatan	Peserta/Departemen	Pemetaan AKtor
1.	Melakukan Standart Booking	Seluruh Departemen	General Affair=*Administrator*, IT Support=*Requestor,* Finance Affair=*Requestor*, Human Resource Dev.=*Requestor*, Human Resource Dev.=*Receptionist*
2.	Melakukan Scheduled Booking	Seluruh Departemen	General Affair=*Administrator*, IT Support=*Requestor,* Finance Affair=*Requestor*, Human Resource Dev.=*Requestor*, Human Resource Dev.=*Receptionist*
3.	Menambahkan user baru kedalam aplikasi	General Affair	General Affair=*Administrator*
4.	Menambahkan gedung dan ruangan baru kedalam aplikasi	General Affair	General Affair=*Administrator*
5.	Melakukan request pembatalan ruangan	Seluruh Departemen	General Affair=*Administrator*, IT Support=*Requestor,* Finance Affair=*Requestor*, Human Resource Dev.=*Requestor*

Setelah semua kegiatan diselesaikan, responden diminta untuk mengisi kuisioner pengujian *usability* dengan metode *System Usability Scale* (SUS) melalui *form* kuisioner dari metode pengujian tersebut dengan ketentuan penilaian sebagai berikut:

- Responden hanya boleh mengisi kolom skala 3 dalam kuisioner jika tidak memahami sama sekali tentang pertanyaan pada setiap nomornya.

Dari proses pengujian diperoleh hasil penilaian sesuai dengan tabel 6.10 pada halaman 115 dibawah ini:

Tabel 6.10 Perhitungan Skor SUS

Responden	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	Skor SUS
Aji Hermawan	4	1	5	2	5	1	4	1	4	2	**87,5**
Galih Hari Wibowo	5	1	5	2	4	2	5	1	5	2	**90,0**
Sugeng D Winanjuar	5	2	5	2	4	2	4	2	4	2	**80,0**
Hidayat Purnama	5	2	4	2	4	1	4	2	4	2	**80,0**
Ahmad Sugeng S W	5	1	5	2	4	2	4	2	4	2	**82,5**
Rina Yulianti	4	2	4	2	4	1	5	1	4	1	**85,0**
Andika Setya Purnama	5	2	4	2	4	2	4	2	5	2	**80,0**
Puryono	4	2	4	5	5	1	4	2	4	4	**67,5**
Winda Ayu Ningrum	4	2	4	2	5	2	4	2	4	2	**77,5**
Cintya Pangestika	4	2	4	2	5	1	4	2	4	1	**82,5**
Andreas Nugraha	4	1	4	2	4	2	4	2	4	2	**77,5**
Ruben Pius Jordy	4	2	4	4	5	2	4	2	4	1	**75,0**

Tabel 6.10 Perhitungan Skor SUS (lanjutan)

Responden	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	Skor SUS
Adrian Pradipto	5	2	4	1	5	2	4	2	4	1	**85,0**
Bayu Setyawan	5	2	4	2	5	2	4	2	5	2	**82,5**
Agus Nugroho	5	2	4	2	5	2	4	2	4	1	**82,5**
Brendha Agnes P	5	2	4	2	4	2	4	2	4	2	**77,5**
Yayuk Apriliani	4	2	4	5	4	2	4	2	4	1	**70,0**
Ariyanto	4	2	4	5	4	1	4	2	4	2	**70,0**

Seperti terlihat pada tabel 6.10 diatas, jawaban mentah kuisioner dari masing-masing responden terdapat pada kolom q1 sampai dengan q10. Kolom q1 sampai dengan q10 kemudian diolah untuk mendapatkan Skor SUS seperti yang terdapat pada kolom Skor SUS pada tabel 6.10 diatas. Untuk proses pengolahannya adalah seperti penjelasan penulis pada Bab 2 Landasan Kepustakaan Sub-bab Metode *System Usability Scale* yaitu untuk item-item kuisioner yang terdapat pada nomor ganjil nilai akhirnya adalah nilai yang tertera pada kolom dikurangi 1, kemudian untuk item-item kuisioner yang terdapat pada nomor genap nilai akhirnya adalah 5 dikurangi nilai yang tertera pada kolom. Kemudian seluruh nilai akhir dijumlah untuk masing-masing responden kemudian dikali dengan 2.5 sehingga diperoleh Skor SUS seperti yang tertera pada tabel 6.10 diatas pada kolom SUS Skor.

Berdasarkan tabel 6.10 pada halaman 115 diatas, diperoleh rata-rata skor SUS dari 18 responden adalah 79,6. Berdasarkan penelitian Bangor, Kortum dan Milner (2008,2009) dalam "*Determining What Individual SUS Score Mean: Adding an Adjective Rating Scale*". Range skor antara 70-80 termasuk kategori "Good" dan bernilai *adjective* "C".

6.4 Analisis Hasil Penelitian

Penelitian ini ditujukan untuk membangun sebuah aplikasi manajemen ruang *meeting* pada PT. Telkomsel Indonesia Tbk dan melihat bagaimana respon dari calon *user* aplikasi terkait aplikasi yang dikembangkan tersebut. Berdasarkan penelitian yang telah dilakukan berikut daftar analisisnya:

6.4.1 Analisis Pengujian *Black Box*

Pada tahap pengujian *black box* dilakukan pengujian terhadap ke-28 fitur aplikasi yang terbagi kedalam total 53 kasus uji. Kasus uji tersebut diantaranya adalah mencoba memberikan masukan dengan data yang benar, data yang salah, atau tidak memberikan masukan apapun kedalam sistem. Hasil dari pengujian *black box* untuk seluruh kasus uji adalah menghasilkan keluaran atau hasil yang sesuai dengan yang diharapkan atau valid. Sehingga dapat dikatakan seluruh fitur aplikasi manajemen ruang *meeting* yang dikembangkan telah teraplikasikan dan dapat bekerja dengan baik.

6.4.2 Analisis Pengujian *White Box*

Pada tahap pengujian *white box* ditampilkan pengujian terhadap 3 fitur utama aplikasi yaitu *standart booking, scheduled booking,* dan *add new user*. Hasil dari pengujian *white box* diperoleh data untuk fitur *standart booking* memiliki nilai kompleksitas 11 dengan jumlah *edge* adalah 43 dan *nodes* adalah 34 yang berarti fitur ini cukup sulit untuk di *maintain*, fitur *scheduled booking* memiliki nilai kompleksitas 11 dengan jumlah *edge* 39 dan *nodes* 30 yang berarti cukup sulit untuk di *maintain*, dan fitur *add new user* yang memiliki nilai kompleksitas 5 dengan jumlah *edge* 25 dan *nodes* 22 yang berarti mudah untuk di *maintain*. Ketiga fitur kompleks aplikasi memiliki kompleksitas dibawah 21 dimana kompleksitas 21-50 mengindikasikan bahwa fitur tersebut merupakan kandidat untuk dilakukan refaktor/desain ulang dikarenakan menurut Michael Bray dalam bukunya "*C4 Software Technology Reference Guide – A Prototype*" menyatakan bahwa nilai komleksitas 21-50 merupakan suatu program yang memiliki resiko tinggi.

6.4.3 Analisis Pengujian *Usability*

Pada tahap pengujian *usability* dilakukan pengujian dengan menggunakan metode *System Usability Scale* (SUS) yang melibatkan 18 responden calon pengguna aplikasi dan 5 *task* pengujian yang harus diselesaikan. Hasil dari pengujian *usability* dengan menggunakan metode SUS adalah rata-rata skor dari hasil pengujian menunjuk angka 79,6 yang mengindikasikan bahwa aplikasi yang dibuat termasuk kategori "Good" dan bernilai *adjective* "C".

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil penelitian dan pengujian yang dilakukan, dapat disimpulkan bahwa:

1. Perangkat lunak yang dikembangkan memanfaatkan SDLC *waterfall* yang dimulai dari tahap analisis kebutuhan sesuai dengan yang dijelaskan pada bab 4 penelitian sub-bab analisis kebutuhan sistem dimana aplikasi yang dikembangkan memiliki 4 orang aktor dan total 28 fitur aplikasi, dilanjutkan dengan tahap desain/perancangan sesuai dengan yang dijelaskan pada bab 4 penelitian sub-bab perancangan sistem dimana terdapat 3 desain diagram pengembangan aplikasi yaitu diagram *sequence,* diagram kelas, dan diagram entitas serta *wireframe* rancangan antarmuka aplikasi, dilanjutkan dengan tahap implementasi sesuai dengan pambahasan pada bab 5 penelitian dimana ditampilkan dan dijelaskan beberapa hasil implementasi antarmuka aplikasi dan 3 kode program dari fitur kompleks aplikasi, dan diakhiri dengan tahap integrasi dan pengujian sistem sesuai dengan penjelasan pada bab 6 penelitian dilakukan 3 jenis pengujian aplikasi yaitu *black box testing, white box testing,* dan pengujian *usability*.
2. Setelah dilakukan pengembangan aplikasi dengan memanfaatkan kerangka kerja *PLAY!* yang telah disesuaikan dengan hasil rancangan aplikasi pada bab 4, dimana struktur rancangannya telah disesuaikan dengan struktur kerangka kerja *PLAY!* tersebut, penulis berpendapat proses implementasi kode programnya menjadi lebih mudah dan cepat dikarenakan struktur kerangka kerja *PLAY!* yang jelas (terdapat modul *Model-View-Controller*) sehingga alur *requestHTTP* dan *responseHTTP* dapat dibuat dan dimodifikasi dengan mudah, proses *rendering* halaman HTML dapat dilakukan dengan mudah sesuai dengan rancangan diagram *sequence* pada bab 4, *serta* penelusuran terhadap *error* yang terjadi pada proses implementasi juga dapat dilakukan dengan cepat dan mudah.
3. Pengujian *usability* dilakukan terhadap 18 responden calon pengguna aplikasi dengan menyelesaikan 5 kegiatan dengan hasil rata-rata skor SUS dari 18 responden tersebut adalah 79,6 yang mengindikasikan bahwa aplikasi yang dibuat termasuk kategori “Good” dan bernilai *adjective* “C”.

7.2 Saran

Berdasarkan hasil analisis pengujian *white box*, maka penulis menyarankan untuk pengembangan penelitian di masa yang akan datang yaitu dengan melakukan *enhancement* terhadap aplikasi manajemen ruang *meeting* yang telah dikembangkan untuk memperbaiki atau menurunkan kompleksitas beberapa fitur didalam aplikasi sehingga aplikasi yang dikembangkan ulang dapat lebih mudah dikelola dengan kompleksitas yang lebih kecil.

DAFTAR PUSTAKA

Hanson, 2000. *Pengertian Web*. Tersedia di: <http:// http://sir.stikom.edu/262/6/BAB%20II.pdf> [Diakses 4 Februari 2016]

Zenexity, Typesafe, 2007. *The Main Concept.* Tersedia di: < https://www.playframework.com/documentation/1.0/main> [Diakses 4 Februari 2016]

Rajendra, Dr. A.R. Dani. 2012. Comparative Study of Prototype Model For Software Engineering With System Development Life Cycle*. IOSR Journal of Engineering (IOSRJEN)*, Vol. 2, Issue 7, 2012

Pressman, R.S., 2011. *Pengertian Rekayasa Perangkat Lunak.* Tersedia di: <http://library.binus.ac.id/eColls/eThesisdoc/Bab2/2011-2-00058-IF%20Bab2001.pdf> [Diakses 8 Februari 2016]

Pressman, R.S., 2002. *Metode Pendekatan Sistem*. Tersedia di: <http://elib.unikom.ac.id/download.php?id=143737> [Diakses 9 Februari 2016]

Nugroho, Adi, 2005. *Focus Unified Modelling Languange (UML).* Tersedia di: <http://elib.unikom.ac.id/files/disk1/545/jbptunikompp-gdl-budisuhend-27218-8-babiil-i.pdf> [Diakses 9 Februari 2016]

Fowler, Martin, 2005. *Use Case Diagram.* Tersedia di: <http://elib.unikom.ac.id/download.php?id=40183> [Diakses 9 Februari 2016]

Munawar, 2005, *Class Diagram.* Tersedia di: < http://elib.unikom.ac.id/download.php?id=40183> [Diakses 9 Februari 2016]

Fowler, Martin, 2005. *Activity Diagram.* Tersedia di: <http://elib.unikom.ac.id/download.php?id=40183> [Diakses 9 Februari 2016]

A.S., Rosa, Shalahuddin, M., 2014. *Rekayasa Perangkat Lunak.* Bandung: Informatika

Kadir, Abdul, 2008. *MySql*. Tersedia di: <http://elib.unikom.ac.id/files/disk1/544/jbptunikompp-gdl-milalaenin-27172-6-unikom_m-i.pdf> [Diakses 9 Februari 2016]

Qcollege, 2004. *Hyper Text Markup Language (HTML).* Tersedia di: < http://elib.unikom.ac.id/download.php?id=40183> [Diakses 9 Februari 2016]

Tullis, and Stetson, *Februari 2013*, "SUS: A Retrospective". *Engineering Journal*. *Volume 8, No.2,* 22 Agustus 2016

Shalahudin, M., 2009. *Pengertian Java.* Tersedia di: < http://eprints.uny.ac.id/8036/4/Chapter_ii.pdf> [Diakses 9 Februari 2016]

Budi, E.M., 2003. *Pengertian Java.* Tersedia di: < http://eprints.uny.ac.id/8036/4/Chapter_ii.pdf> [Diakses 9 Februari 2016]

Romer, Toomas, 2011. *My Top 5 Play Framework Features.* Tersedia di: <http://zeroturnaround.com/rebellabs/my-top-5-play-framework-features/> [Diakses 21 Februari 2016]

Kurniawan, Tri A., 2014. *SDLC: Prototyping Model.* [gambar] (Malang, Materi Rekayasa Perangkat Lunak)

Sommerville, Ian, 2011. *Waterfall System Development Life Cycle.* Tersedia di: <http://meiryani.net/bahan-kuliah/analisa-perancangan-sistem/items/16-1-6-pendekatan-waterfall-dalam-pengembangan-sistem/16-1-6-pendekatan-waterfall-dalam-pengembangan-sistem> [Diakses 12 April 2016]

Meiert, 2009. *SUS: How to Easily Grade Your Site's Usability.* Tersedia di: <http://meiert.com/en/blog/20091127/sus-how-to-grade/> [Diakses 6 Juni 2016]

.Bray, Michael, 1997. *C4 Software Technology Reference Guide – A Prototype.* Pittsburgh:Carnegie Mellon University

Brooke, John, *Februari 2013*, "SUS: A Retrospective". *Engineering Journal*. *Volume 8, No.2,* 20 Mei 2016.

Bangor, Kortum, and Miller, *Februari 2013*, "SUS: A Retrospective". *Engineering Journal*. *Volume 8, No.2,* 20 Mei 2016.