PENGEMBANGAN APLIKASI MANAJEMEN RUANG *MEETING* DENGAN MENERAPKAN *TOOL* YSLOW SEBAGAI METODE PENGUJIAN PERANGKAT LUNAK

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Imam Achmad Hambali
NIM: 125150207111050

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

PENGESAHAN

PENGEMBANGAN APLIKASI MANAJEMEN RUANG *MEETING* DENGAN MENERAPKAN *TOOL* YSLOW SEBAGAI METODE PENGUJIAN PERANGKAT LUNAK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Imam Achmad Hambali
NIM: 125150207111050

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	Aditya Rachmadi., S.ST, M.TI
NIP: 19851124 201504 1 001	NIK: 201201 860421 1 001

Mengetahui
Ketua Jurusan 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, 10 Agustus 2016

Imam Achmad Hambali

NIM: 125150207111050

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* dengan Menerapkan *Tool* Yslow Sebagai Metode Pengujian Perangkat Lunak" 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 Aditya Rachmadi., S.ST, M.TI selaku dosen pembimbing 2.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Program Studi Informatika/Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak Agus Puryono beserta staff selaku narasumber utama yaitu kepala divisi *General Affair* PT.Telkomsel Indonesia Tbk regional Jawa Timur.
5. Bapak Teguh Budihartoyo dan Ibu Choirun Nissa, selaku orang tua penulis dan seluruh keluarga atas dukungan dan do'a yang telah diberikan.
6. Mega Manfaati yang telah membantu dan memberikan *support* dalam pengerjaan skripsi dari awal hingga akhir.
7. Satria Mulya yang telah membantu dalam hal pengerjaan bagian implementasi program manajemen ruang *meeting*.
8. Dary Hilmy Iswara yang telah membantu meneliti kesalahan dan membantu menyelesaikan beberapa kekurangan pada bagian implementasi.
9. Akmilatul Maghfiroh dan Rofiqoh Ainun Zakiyah yang telah membantu dalam kegiatan seminar hasil dan sidang ujian skripsi.
10. 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, 10 Agustus 2016

Penulis

Imamhambali@hotmail.co.id

ABSTRAK

Rapat merupakan sebuah kegiatan yang sering dilakukan di dalam lingkup perkantoran maupun perusahaan, yang bertujuan untuk memberikan pemecahan masalah dan merumuskan suatu solusi dari permasalahan yang ada. Dalam kegiatan rapat, diperlukan suatu kontrol untuk melangsungkan kegiatan rapat. Hal tersebut bertujuan untuk menghindari hal yang dapat menghambat jalannya sebuah rapat. Proses kontrol kegiatan rapat yang ada yaitu, menggunakan catatan yang bersifat manual yang cenderung konvensional. Manajemen atau kontrol ruang rapat difokuskan pada kegiatan pencatatan penggunaan ruangan rapat dan pemesanan ruang rapat pada waktu yang telah ditentukan. Pengembangan aplikasi manajemen ruang *meeting* dilakukan berdasarkan permasalahan tersebut dan dengan memanfaatkan model pengembangan SDLC *waterfall*. Aplikasi manajemen ruang *meeting* menggunakan pendekatan MVC dimana proses dalam pengembangannya berdasarkan fase didalam MVC tersebut. Berdsarkan pada proses pengembangan aplikasi tersebut, peneliti juga melakukan pengujian dari sisi performa aplikasi dengan menggunakan *tool* yang bernama Yslow. Yslow merupakan *tool* berbasis online yang dapat melakukan pengujian dari sisi performa perangkat lunak.

Kata kunci: pengembangan, manajemen ruang rapat, MVC, performa, Yslow

ABSTRACT

Meetings is an activity that is often carried out within the scope of offices and companies, which aims to make solutions and formulate a solution of the existing problems. Meeting activities, needed establish a control meeting activities. It aims to avoid things that can implementation the course of a meeting. Activities control existing meeting that is, manual record tend to be conventional. Management or control conference recording activities focused on meeting room usage and booking meeting rooms at a predetermined time. Meeting room management application development is based on these issues and to take advantage of SDLC waterfall development model. Application management using the meeting room where the MVC approach in its development process based on the phase in the MVC. Base on the application development process, the researchers also conducted tests of the application performance by using a tool called YSlow. YSlow is an online-based tool that can test the performance of the software.

Keyword: *development, meeting room management, MCV, performance. Yslow*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika pembahasan.

1.1 Latar belakang

Di dalam dunia kerja terdapat banyak kegiatan yang dilakukan sebagai bentuk dari kewajiban suatu pegawai di suatu instansi maupun perusahaan dengan beragam kegiatan yang memiliki prioritas dari mulai tinggi sedang dan rendah. Kegiatan yang dijalankan sehari – hari oleh seluruh pegawai memiliki lingkup masing – masing dengan prioritas yang berbeda pula. Kegiatan yang peneliti amati ada beberpa yang menjadi sebuah rutinitas kewajiban ada pula yang menjadi sebuah pekerjaan tambahan bahkan menjadi tugas khusus yang memerlukan jam kerja lebih larut untuk mencapai sebuah target dengan tingkat profesionalitas yang tinggi. Sehingga kegiatan yang ada di dalam lingkup perusahaan menjadi terstruktur sesuai dengan alur kerja serta di dukung dengan sumber daya yang profesional. Kegiatan di sebuah perusahaan ataupun instansi memilki karakteristik yang berbeda dilihat dari sisi perusahaan atau intstansi tersebut bergerak dibindang apa. Namun dari seluruh kegiatan yang menjadi rutinitas dari perusahaan atau instansi terdapat kegiatan yang penting dalam berjalannya sebuah pencapaian tertentu diantaranya yaitu, kegiatan *meeting*.

Meeting adalah kegiatan yang kerap kali berjalan pada suatu organisasi, bertujuan untuk memberikan gambaran terhadap permasalahan serta merumuskan sebuah kesimpulan atau solusi, penjadwalan kerja, pemecahan permasalahan, dan penyampaian informasi (Jennifer et al., 2015). *Meeting* dilakukan atas dasar keperluan yang memerlukan suatu keputusan bersama di dalam suatu permaslahan yang ada di dalam suatu organisasi. Kegiatan tersebut tidak semata – mata dilakukan dengan mementingkan kepentingan pribadi, namun memprioritaskan sesuatu yang dianggap penting dalam sebuah organisasi dengan tujuan dapat menyelesaikan sebuah permasalahan yang kerap kali muncul di dalamnya. Kegiatan *meeting* memerlukan sebuah cara dalam hal mengkoordinasikan agar terselenggaranya kegiatan *meeting* tersebut. Koordinasi yang diakukan atas dasar utama memberikan kenyamanan berlangsungnya kegiatan *meeting* dari sisi peserta seperti adanya kebutuhan yang menjadi penunjang berjalannya *meeting* yaitu, LCD, Proyektor, Soundsystem, konsumsi, dan beberapa kebutuhan lainnya.

Permasalahan yang sering timbul terlihat pada kegiatan yang memanfaatkan ruang meeting cenderung menggunakan prosedur yang terbilang mudah namun tidak praktis dalam penerapannya dan konvensional yaitu, dengan melakukan konfirmasi kepada pihak reseptionist bahwa manakah ruangan meeting di suatu instansi kerja yang tidak terpakai dan tidak terpakai untuk meeting. jika salah ruangan yang tersebut tidak terpakai, maka kegiatan meeting dapat segera berlangsung oleh suatu departemen ataupun sebuah kelompok rapat, namun jika didapati tidak ada satupun ruangan yang tersedia untuk kegiatan meeting di sautu

instansi kerja tersebut, maka suatu departemen atau sebuah kelompok rapat harus melakukan pencarian ruangan meeting lain yang tidak terpakai dengan prosedur yang serupa di ruangan lain, yaitu, dengan melakuakn konfirmasi kembali kepada receptionist. Hal tersebut tentu menjadi faktor utama yang sangat mengganggu kinerja dari setiap departemen maupaun kelompok rapat di sebuah instansi kerja tertentu. Karena banyaknya upaya dikeluarkan dan waktu yang dihabiskan hanya untuk menggunakan ruang meeting.

Berdasarkan pada permasalahan yang timbul di suatu instansi perusahaan seperti dijelaskan diatas, peneliti membuat sebuah aplikasi manajemen ruang meeting yang dapat mencangkup kebutuhan dari setiap departemen maupun kelompok rapat pada sebauh instansi perusahaan untuk dapat menggunakan ruang meeting dengan mudah. Oleh karena itu dengan adanya aplikasi yang dikembangkan oleh peneliti, kegiatan kontrol maupun konfirmasi terhadap ruang meeting dapat berjalan secara efektif dan efisien, serta proses pemesanan ruang meeting dapat dilaksanakan dengan mudah, dan penggunaan ruang meeting akan menjadi lebih optimal.

Dalam usulan penliti, terhadap pengembangan aplikasi website, perlu dilakukan pengujian sebelum aplikasi tersebut dapat digunakan secara langsung oleh pihak stakeholder. Pengujian merupakan salah satu teknik yang digunakan untuk melihat menjamin kualitas daripada perangkat lunak (Fevzi Belli et al., 2016). Terdapat beberapa dimensi uji kualitas yang diterapkan pada sebuah *website* untuk mengetahui apa saja yang mejadi parameter yang dapat diuji serta dapat dilakukan evaluasi dari hasil pengembangan perangkat lunak seperti *security, usability, performance*, *realibility,* serta dimensi uji lainnya yang juga mendukung proses pengujian perangkat lunak. Dalam pembahasan terkait pengujian dari perangkat lunak berbasis *web* sesuai dengan aplikasi yang dikembangkan, peneliti mengambil dimensi uji performa untuk mendapatkan hasil yang diinginkan serta kemudahan dalam hal proses pembelajaran dan serta penggunaan dari aplikasi manajemen ruang *meeting*. Pengujian performa di dalam sebuah aplikasi berbasis *web* ini dilakukan untuk mendapatkan respon dengan parameter waktu yang baik untuk memuat sebuah halaman *web* didalam waktu tertentu. Hal tersebut merupakan salah satu yang menjadi dasar peneiti untuk mengambil pengujian berdasarkan performa dari aplikasi *web* tersebut.

Dalam pengujiannya, peneliti memanfaatkan alat atau *tool* yang digunakan untuk membantu proses pengujian dari sisi performa *website*. *Tool* pengujian performa sebuah *web* tidak hanya satu namun ada beberapa *tool* uji diantaranya yang banyak digunakan adalah GT Matrix, Pigdom, Pagespeed, Yslow, dan beberapa *tool* uji lainnya yang serupa. Peneliti memilih *tool* Yslow karena *tool* ini memberikan evaluasi terhadap hasil uji performa yang lengkap dalam hal pengembangannya serta *tool* ini dapat memberikan titik fokus terhadap proses pengembangan kembali untuk mendapatkan performa yang lebih baik berdasarkan hasil uji sebelumnya. Cara kerja serta penggunaan *tool* Yslow tersebut antara lain memberikan penilaian terhadap aplikasi *web* berupa pemberian grade berdasarkan pada matrik uji yang terdapat pada tool Yslow.

Berdasarkan pada paparan informasi tersebut, peneliti melakukan pengembangan aplikasi manajemen ruang *meeting* untuk mengontrol, mengatur serta menjadwalkan sebuah kegiatan *meeting* yang bertujuan untuk mengoptimalkan penggunaan ruang *meeting.* Pengembangan aplikasi manajemen ruang *meeting* berdasarkan model *object oriented*. Penulis mengambil judul "**Pengembangan Aplikasi Manajemen Ruang *Meeting* Dengan Menerapkan Tool YSlow Sebagai Metode Pengujian Perangkat Lunak** "dengan harapan aplikasi ini dapat membantu memduahkan proses pemesanan ruangan serta penjadwalan ruang *meeting* yang lebih optimal serta efektif dan efisien.

1.2 Rumusan masalah

Berdasarkan dengan pengembangan dari aplikasi manajemen ruang *meeting,* penulis dapat merumuskan beberapa permasalahan yang jelas, relevan, fokus, menarik, dan dapat menjawab permasalahan yang ada dengan aplikasi tersebut. Perumusan masalah dalam pengembangan aplikasi manajemen ruang *meeting* antara lain sebagai berikut:

1. Bagaimana melakukan proses *gathering* pada analisa kebutuhan kebutuhan untuk proses pengembangan aplikasi manajemen ruang *meeting* agar tidak terjadi *crash* jadwal *meeting*?
2. Bagaimana proses pengembangan aplikasi manajemen ruang *meeting* dengan menggunakan model SDLC *waterfall*?
3. Bagaimana evaluasi hasil pengujian fitur pada aplikasi manajemen ruang *meeting*?
4. Bagaimana hasil implementasi serta hasil evaluasi aplikasi manajemen ruang *meeting* dengan menggunakan *tool* Yslow?

1.3 Tujuan

Pada pengembangan aplikasi manajemen ruang *meeting* tersebut, terdapat tujuan yang menjadi dasar dalam pengembangannya. Tujuan dari pengembangan aplikasi manajemen ruang *meeting* adalah untuk mengembangkan aplikasi yang ada dengan pengembangan fitur yang relevan dan lebih memanfaatkan keberadaan *website* dan dukungan dari media internet untuk mengembangkan aplikasi penjadwalan antar pengguna dengan sistem yang ada.

1.4 Manfaat

Manfaat dari penelitian yang terkait dengan pengembangan aplikasi manajemen ruang meeting adalah untuk memberikan dampak positif dari kemudahan dalam proses pemesanan ruang meeting hingga pengolahan keseluruhan data dalam aplikasi manajemen ruang meeting pada suatu instansi kerja tertentu dari sisi aplikasi. kemudian dapat mengetahui apa saja yang menjadi keunggulan dari kebutuhan *non*-fungsional yang lebih spesifik di dalam aplikasi manajemen ruang *meeting* melalui pengujian dari *tool* Yslow dari sisi metode pengujian perangkat lunak. Terhadap pengguna, manfaat yang diberikan dapat berupa efektifitas pelaksanaan *meeting* yang akan dilaksanakan tanpa harus

melakukan pemesanan atau peminjaman secara manual melainkan dapat memesan ruangan *meeting* secara mudah dan dapat terjadwal secara baik.

1.5 Batasan masalah

Batasan masalah yang menjadi titik fokus dari pada aplikasi manajemen ruang *meeting* agar supaya tidak memperluas bahasan yang terkait dengan penelitian diantaranya:

1. Permaslahan diambil dari sebuah instansi perusahaan yang membutuhkan aplikasi manajemen dalam melakukan kontrol terhadap kegiatan *meeting*.
2. Pengembangan aplikasi manajemen ruang *meeting* adalah berbasis *web*.
3. Pengujian terhadap aplikasi manajemen ruang *meeting* dilakukan dengan menggunakan *tool* Yslow.

Penelitian difokuskan pada pengembangan aplikasi dan pengunjian dengan *tool* yang sudah didefinisikan.

1.6 Sistematika pembahasan

Pengembangan aplikasi manajemen ruang *meeting* memliki sturktur terkait dengan pembahasan yang dapat menjelaskan secara detail tentang pengembangan aplikasi tersebut sesuai dengan bab dan subab yang ada. Berikut sistematika pembahasan terkait dengan aplikasi manajemen ruang *meeting*:

BAB I : Pendahuluan

Bab ini menjelaskan mengenai latar belakang permasalahan, rumusan masalah, batasan masalah, tujuan, manfaat, serta sistematika penulisan.

BAB II : Landasan Kepustakaan

Bab ini menjelaskan kajian pustaka dan dasar teori yang menjadi acuan dalam penulisan skripsi yang diperoleh dari beberapa literatur, jurnal, dan halaman website.

BAB III : Metodologi

Bab ini berisi langkah-langkah yang digunakan dalam penelitian yang terdiri dari tahap penelitian, kebutuhan sistem, formulasi permasalahan, siklus penyeleseaian dan contoh perhitungan manual.

BAB IV : Analisis dan Perancangan

Bab ini berisi perancangan yang terdiri dari perancangan database, perancangan antarmuka, serta perancangan uji coba dan evaluasi.

BAB V : Implementasi dan Pengujian

Bab ini menjelaskan tentang tingkat akurasi dan analisa hasil terhadap metode yang digunakan

BAB VI : Analisa dan Pembahasan Hasil

Bab ini berisi analisis dari proses pengerjaan aplikasi manajemen ruang *meeting* serta pembahasan hasil akhir pengembangannya.

BAB VII : Penutup

Bab ini berisi kesimpulan dan saran atas kekurangan di dalamnya sehingga dapat dikembangkan menjadi penelitian yang lebih baik.

BAB 2 LANDASAN KEPUSTAKAAN

Secara umum pada bab ini berisi tentang kajian pustaka dan dasar teori yang digunakan dalam penelitian. Kajian pustakan difokuskan pada metode pengujian dengan menggunakan *tool* Yslow yang merupakan *tool* untuk melakukan pengujian dari fungsi *performance*. Dasar teori di fokuskan pada apa saja yang mendukung proses pengembangan aplikasi manajemen ruang *meeting*.

2.1 Kajian Pustaka

Apliksi ruang *meeting* dikembangkan dengan menambahhkan sebauh fitur untuk nantinya dilakukan pengujian dari aplikasi tersebut untuk mendapatkan hasil beruapa data matrik pengujian berupa kebutuhan *non*-fungsional yang terdapat di dalam aplikasi manajemen ruang *meeting.* Matrik pengujian secara umum merupakan urutan atau acuan yang nantinya akan menjadi sebuah tolak ukur dari kebutuhan *non*-fungsional yang akan diuji. Beberpa paper atau jurnal yang terkait dengan pengujian perangkat lunak membahas tentang perbandingan *tool* dengan matrik pengujinya terhadap sebauh halaman web yang akan di uji. Bebrapa *tool* yang menjadi alat ukur tersebut memiliki matrik yang serupa dan dapat menjadikan acuan terhadap performa dari sebuah halaman web. Dari beberapa paper atau jurnal yang ada dan terkait dengan pembahasan yang diangkat, peneliti mengambil jurnal dengan judul "*Analysis of Yslow Performance Test tool & Emergences on Web Page Data Extraction".* Peneliti mengambil jurnal tersebut dengan dasar pembahasan yang kuat di dalam penggunaan *tool* Yslow untuk menguji sebuah ekstraksi di dalam sebuah halaman web. Pada bagian abstrak dalam penelitian tersebut secara umum menggambarkan bagaimana data melakukan ekstraksi dari sebuah halaman web didalam beberapa aplikasi web serta performa yang lambat dari sebuah akses halaman web. Di dalam penelitian sebelumnya dilakukan teknik ekstraksi data dalam sebuah tingkatan level kebutuhan. Dengan dasar tersebut penelitian ini dibuat untuk memperkenalkan konsep dari cara kerja *tool* Yslow dalam menguji proses dari masing-masing kebutuhan dalam matrik pengujian seperti HTTP *request*, *response & Domain Network System, JavaScript* dan juga improve dari level dari performa ektraksi data di dalam tingkat yang baik (Indira, 2013).

Didalam pengujian perangkat lunak berbasis *web, tool* yang digunakan mayoritas memiliki matrik pengujian yang serupa dengan *tool* yang lainnya. Salah satu yang menjadi pembahasan dalam jurnal yang peneliti gunakan sebagai tinjauan pusataka, yaitu *tool* Yslow yang merupakan *tool* besutan Yahoo yang digunakan sebagai pengembangan dan analisis halaman web berdsarkan pada keseluruhan komponen dengan menggunakan JavaScript sebagai cara untuk membentuk sebuah komponen halaman *web* yang bersifat dinamis. Kompnen dan statistik serta peningkatan performa berdasarkan pada 22 buah matrik yang dikategorikan kedalam 6 konten CSS JavaScript dan server. Beberapa *tool* yang melakukan pengujian performa dengan merangkingkan kecepatan akses, tidak terlalu memberikan dampak besar untuk kelambatan *website* dan menurunkan

jumlah pengunjung pada *website* tersebut. Satu dari add-ons yang popular untuk menguji sebuah performa *website* yaitu, Yslow dan Google page speed yang memberikan sebuah pandangan bagi pengembang untuk dapat melakukan improvisasi terhadap *website* sesuai dengan rekomendasi dari Firefox dan juga dapat diterapkan untuk *browser* lainnya. Di dalam proses *load* untuk mengambil data berupa gambar, CSS, JS secara berkesinambungan seperti pada saat pengunjung suatu *website* melakukan akses secara langsung dan saat itu juga dapat diamati bagaimana kecepatan akses *website* tersebut, hal tersebut memerlukan pelaporan performa untuk mendapatkan elemen yang perlu untuk dilakukan improvisasi terhadap waktu *load* sebuah *website.* Fitur dari Yslow, user dapat memilih *rule* yang telah di tetapkan oleh Yslow di dalam sebuah halaman *web* yang kecil atau blog untuk melakukan *rating* sebuah *website* dari parameter yang tersedia, selain itu Yslow juga menyediakan kepada user untuk memfilter parameter yang terkait dengan konten cookie, CSS dan JavaScript. Didalam sekumpulan komponen yang didalamnya terdiri dari sub komponen seperti doc, js, css, iframe, cssimage, dan gambar yang menampilkan total dari HTTP *request* dan bobot term dari *cache* kosong dan *chace* utama sebagai bobot dari tamplian *website*.

Hal tersebut dapat mengambil seluruh komponen dari sebuah halaman *web* berdasarkan pada penerapan *Domain Object Model* (DOM). Jika terdapat komponen yang tidak tersedia di dalam Net panel seperti komponen yang terbaca dari *chace* atau terdapat pemberitahuan *error* 304, maka Yslow akan memberikan respon berupa membuat sebuah XHLHttp *request* untuk mengambil komponen yang ditujukan dengan meletakkannya pada *header* dan informasi yang diperlukan. Yslow juga berfungsi untuk melakukan proses *generate rule* yang akan digunakan untuk pengambilan sebuah data serta memberikan nilai untuk sebuah halaman *web*. Beberpa keuntungan yang diberikan pada user terkait dengan penggunaan Yslow sebagai *tool* uji perangkat lunak berbasis *web* antara lain:

1. User dapat melakukan improvisasi lebih siginifikan terhadap performa dari sebuah *website*.
2. User dapat melakukan *design* dengan lebih cepat.
3. User dapat mengurangi waktu tunggu dari akses halaman *web*.

User mendapatkan improvisasi yang lebih potensial dan untuk lebih fokus terhadap tatap muka pengguna.

2.2 SIstem Informasi

Sistem informasi secara umum diagambarkan sebagai suatu informasi yang dirangkai atau bahkan disampaikan dengan suatu cara yang di dasarkan oleh sistem yang sudah di buat.

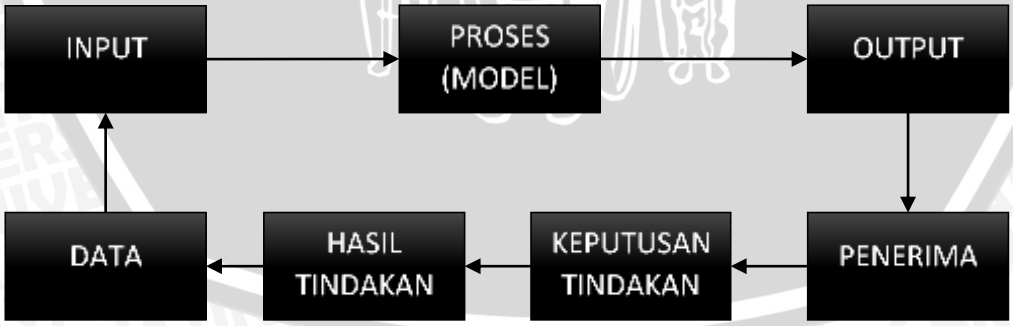
a) Definisi Sistem

Sistem adalah sekelompok komponen yang saling berhubungan, bekerja bersama untuk mencapai tujuan bersama dengan menerima input serta

menghasilkan output dalam proses transformasi yang teratur (O'brien, 2005). Konsep dasar sistem terdapat dua pendekatan yaitu penekanan pada prosedur dan penekanan pada komponen. Pada prosedur meliputi suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Prosedur dapat diartikan sebagai suatu urutan operasi klerikal (tulis-menulis), biasanya melibatkan beberapa orang di dalam satu atau lebih departemen, yang diterapkan untuk menjamin penaganan yang seragam dari transaksi bisnis yang terjadi. Selain itu prosedur juga berupa urutan yang tepat dari tahapan-tahapan instruksi yang menerangkan apa yang harus dikerjakan, siapa yang mengerjakan, kapan dikerjakan dan bagaimana mengerjakannya. Selanjutnya pada komponen berupa kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Sistem memliki karakteristik seperti memiliki komponen, batas sistem, lingkungan luar sistem, penghubung, *input*, *process*, *output*, dan sasaran atau tujuan.

b) Definisi Informasi

Informasi adalah data yang telah diatur dan diporses untuk menyediakan arti kepada pengguna (Ruri, 2010). Dapat diartikan juga bahwa informasi adalah suatu data yang telah diolah dan diproses menjadi sesuatu yang bernilai, dan bermanfaat bagi para pengguna. Informasi juga dapat didefinisikan sebagai data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi pihak penerima. Data dapat diartikan sebagai kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan yang nyata. Atau data adalah representasi dari dunia nyata yang mewakili suatu objek seperti manusia (pegawai, mahasiswa, pelanggan), hewan, peristiwa, konsep, keadaan dan lain sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau berupa kombinasinya. Model yang digunakan untuk mengolah data tersebut disebut model pengolahan data atau dikenal dengan siklus pengolahan data (siklus informasi).



Gambar 2.1 Alur Sistem Informasi

Sumber: Pengantar Sistem Informasi (BINUS)

c) Definisi Sistem Informasi

Suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan (Universitas Sriwijaya, 2010). Adapun arti singkat dari sistem informasi adalah kerangka kerja yang mengkoordinasikan sumber daya (manusia dan komputer) untuk mengubah masukan (*input)* menjadi keluaran (*output*) informasi guna mencapai sasaran-sasaran perusahaan.

d) Aplikasi Manajemen Ruang *Meeting*

Aplikasi ini dirancang dan dibangun untuk memenuhi kebutuhan dari sebuah instansi perusahaan penyedia layanan *provider* PT. Telkomsel Indonesia Tbk regional jawa timur. Rancang bangun aplikasi ini didasari dari kebutuhan akan kegiatan *meeting* yang memiliki *traffic* tertinggi di dalam perusahaan tersebut. Atas dasar tersebut dilakukan rancang bangun aplikasi manajemen ruang *meeting* dengan tujuan utama agar supaya kegiatan *meeting* menjadi lebih terorganisir dari kegiatan *meeting* satu dengan kegiatan *meeting* lainnya. Permasalahan utama ada pada tidak terkoordinasikan antara satu pengguna ruang *meeting* dengan pengguna lain yang lebih menginginkan ruang *meeting* tersebut lebih dahulu serta tidak adanya tingkat efektivitas penggunaan ruang *meeting* tersebut di dalam jam kerja sehingga terkadang ruangan *meeting* tersebut kosong dalam waktu 1 hari bahkan 2 hari dan pada saat penggunaan ruang *meeting* penuh, pengguna lain yang ingin mengadakan *meeting* pada hari yang sama tidak dapat menggunakan ruangan tersebut alhasil *meeting* dibatalkan atau ditunda hingga mendapatkan ruangan *meeting* yang kosong.

Aplikasi manajemen ruang *meeting* dirancang dan dibangun berjalan diseluruh *operating system* dengan berbasis *web* dan dibangun dibangun menggunakan Bahasa pemrograman PHP, HTML, JavaScript, dan jQuery dengan menggunakan Bootstrap sebagai *library*. Aplikasi ini menyimpan segala inputan dan *design* dari admin dalam sebuah file aplikasi dengan dengan ekstensi PHP, HTML, CSS, dan JavaScript yang kemudian di *upload* dalam *domain* yang sudah ditentukan untuk berjalan dalam beberapa server dengan masing-masing fungsinya.

2.3 Testing

Testing merupakan salah satu teknik yang digunakan untuk melihat menjamin kualitas daripada perangkat lunak industri. Hal tersebut menyebabkan eksekusi dari sebuah perangkat lunak berada didalam suatu ruang lingkup yang nyata. langkah awal dalam proses pengembangan perangkat lunak sering kali berasal dari kebutuhan import, yang menghasilkan spesifikasi dari jalannya sebuah sistem (Fevzi Belli et al, 2016). hal tersebut penting untuk menghasilkan tes case yang baik serta membatasi proses testing di awal, sebelum proses implementasi berjalan, dan proses kompilasi dengan berdasarkan argumen user tentang bagaimana sistem seharusnya berjalan. Keuntungan akan didapat dari pencegahan kesalahan yang terdeteksi di langkah akhir, jika user melakukan

deteksi keseluruhan diawal step dan mengurangi kerugian. Pentingnya testing untuk perangkat lunak adalah pada keterlibatan sederetan aktivitas produksi dimana terdapat peluang yang mengakibatkan terjadinya kesalahan manusia yang cukup fatal serta area ketidakmampuan manusia untuk menjalankan dan berkomunikasi dengan baik sehingga pengembangan perangkat lunak diimbangi dengan kualitas sebuah jaringan.

2.3.1 Performance Testing

Performa testing diperlukan untuk pengembangan dan "pembuktian praktis" dari suatu ruang lingkup nyata dengan memberikan suatu tes metode dan menempatkannya sesuai dengan keterbatasan kemampuan perangkat lunak. Tedapat kasus khusus dimana sebuah service life models tidak baik untuk diterapkan pada proses pengembangan, kekurangan tersebut datang dari tanggapan serta pengalaman dengan maksud untuk memodifikasi atau melakukan improvisasi terhadap kekurangan tersebut berdasarkan model yang lebih baik untuk proses pengembangan perangkat lunak. Selanjutnya, performa testing dapat digunakan pada cata yang berbeda. Tipe dari penggunaan performa testing adalah untuk melakukan kualifikasi terlebih dahulu untuk kemudian dilakukan pembangunan perangkat lunak. selain itu penerapan dari performa testing adalah pada tahapan quality control selama pembuatan perangkat lunak berjalan, disaat terdapat dua pilihan antara lain: peratama, testing pada pengujian yang berlangsung rutin terhadap sebuah perangkat lunak atau kedua, struktur yang telah di uji menggunkana in-situ testing atau menggunakan sample yang diambil dari struktur itu sendiri dan dilakukan testing di sebuah laboratorium, kedua kasus tersebut objek yang ada akan di akses dengan pengujian kualitas as-built dari sebuah struktur perangkat lunak (Mark Alexander, Michael Thomas, 2015).

2.3.1.1 Performance Testing Website

Pengujian performa dalam perangkat lunak sangatlah beragam dari sisi objek uji yang ada seperti pengujian performa pada aplikasi *desktop,* pengujian pada aplikasi berbasis *mobile,* hingga pengujian aplikasi berbasis *website* (Dickinger Asrtid, Stangl Brigitte., 2011). Dalam pengujiannya memiliki masing-masing karakteristik atau parameter yang dijadikan sebagai bahan uji untuk melihat kualitas dari aplikasi perangkat lunak tersebut. Dari beberapa objek model yang diuji dari sisi performa tersebut, peneliti mengambil pengujian performa pada *website* dengan ancuan berdasarkan pada aplikasi manajemen ruang *meeting* yang merupakan aplikasi berbasis *web.* Pegujian performa dalam sebuah website dipengaruhi oleh beberapa faktor yang mendukung untuk digunakan dalam memberikan penilaian seberapa baik aplikasi website tersebut dari sisi performa. Pengujian performa *website* lebih ditekankan pada pengujian dari sisi *load page* dari sebuah halaman *website*. Melalui hal tersebut dapat diamati bagaimana *website* dapat berinteraksi dengan pengguna secara baik tanpa adanya keluhan maupun suatu ketidak nyamanan dalam penggunaan aplikasi *web* tersebut.

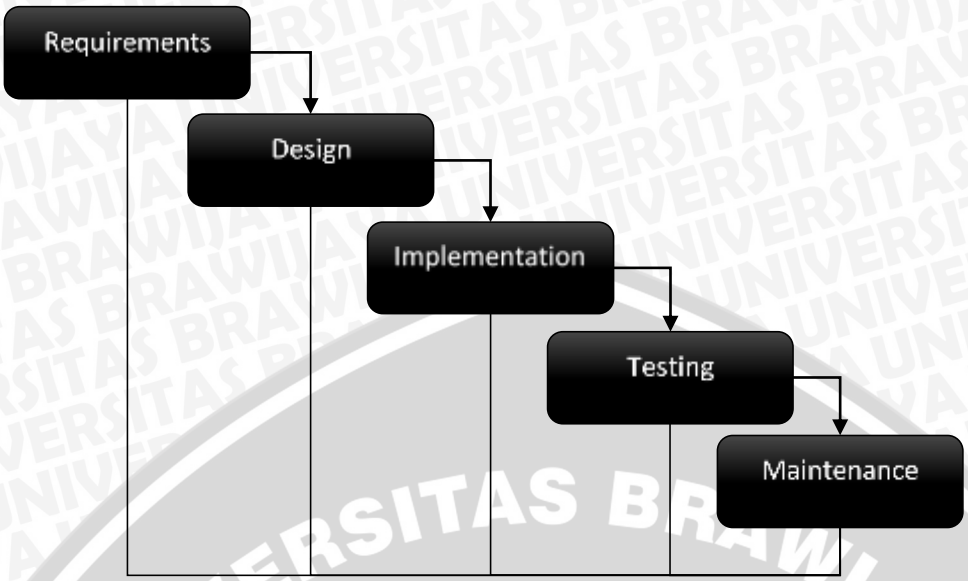
2.4 Matrik Pengujian Perangkat Lunak

Matrik didalam suatu pengujian perangkat lunak memiliki masing-masing standar yang dibentuk untuk mendapatkan hasil pengujian yang baik dan sesuai dengan kontek yang akan diujiakan dari suatu aplikasi atau sistem perangkat lunak. Dari beberapa pengamatan peneliti terhadap standar faktor *usability* dan model terdapat 127 matrik spesifik *usability* (Seffa, Donyaee.M, 2006). Bebrapa matrik didefinisikan sebagai fungsi dan formula, sedangkan lainnya hanya berupa sebuah data yang dapat dihitung. Perhitungan matrik diperolah dari data mentah yang dikumpulkan melalui data nyata serpeti *log* file, video observasi, wawancara, atau survey.

Berdasarkan fase dari *software life cycle*, matrik *usability* terbagi atas 2 kategori klasifikasi antara lain *testing* dan *predictive* (Seffa, Donyaee.M 2006). Data yang berasal dari matrik *testing* dikumpulkan untuk mengatahui cara kerja actual dari aplikasi maupun sistem terhadap identifikasi masalah yang ada. Kebutuhan data yang dikumpulkan seluruhanya merupakan sistem fungsional atau dapat berupa gambaran umum yang ada. Matrik *predictive* adalah sebuah nilai yang mendasari beberapa aspek dari mulai yang membangun maupun yang memperdiksi sistem *usability*. Didalam terbentuknya sebuah matrik pegnukuran di perlukan 2 buah kontek yang mendukung proses pembentukan matrik antara lain, faktor dan kriteria. Kriteria sebagai faktor utama dalam matrik pengukuran yang digunakan untuk mengetahui kemampuan *usability* dari sebuah aplikasi maupun sistem perangkat lunak.

2.5 Software Development Life Cycle

Pengembangan aplikasi perangkat lunak baik berbasis website maupun berbasis desktop memiliki siklus berupa proses yang berjalan untuk media pendukung pengembangannya. Proses tersebut terbagi atas beberapa fase yang meliputi analisis kebutuhan, *design*, *coding*, pengujian, instalasi, dan pemeliharaan. Fase-fase tersebut akan berubah-ubah sesuai dengan apa yang dibutuhkan oleh user terhadap sistem. Pengembangan perangkat lunak harus memiliki siklus tersebut baik dalam skala kecil hingga dalam siklus besar. Hal tersebut menjadi penting karena pada sebuah penelitian dalam pengembangan perangkat lunak memiliki urutan fase yang harus dilewati untuk dapat mencapai hasil yang sesuai dengan kebutuhan user terhadap aplikasi perangkat lunak. Adapun beberapa model yang menjadi faktor perbedaan urutan fase yang berjalan pada proses pengembangan perangkat lunak, diantaranya adalah model waterfall, model prototyping, model spiral, model, incremental serta beberapa model pengembangan lainnya. Peneliti mengambil model waterfall untuk proses pengembangan perangkat lunak. Waterfall merupakan model siklus pengembangan yang keseluruhan fasenya berjalan secara linier dengan fase lainnya, sehingga yang menjadi fase berikutnya akan berjalan setelah proses pada fase sebelumnya telah berkahir dan layak untuk dilanjutkan pada fase berikutnya (Kumar Naresh, 2013). Berikut merupakan gambaran dari model waterfall pada gambar 2.2 Model Waterfall.



Gambar 2.2 Model Waterfall

Sumber: (Sadaf Ateeq, 2014)

2.6 *Meeting*

Meeting adalah sebuah kegiatan yang kerap kali berjalan pada suatu organisasi, bertujuan untuk memberikan gambaran terhadap permasalahan serta merumuskan sebuah kesimpulan atau solusi, penjadwalan kerja, pemecahan permasalahan, dan penyampaian informasi (Jennifer et al, 2015). Kegiatan *meeting* yang berlangsung akan membentuk sebuah gambaran umum terkait dengan isu yang akan diangkat pada saat meeting berlangsung. Hal tersebut menjadikan para peserta meeting untuk melakukan sanggahan ataupun masukan terhadap isu yang telah diangkat. Kegiatan *meeting* yang dilakukan tidak semata-mata melakukan meeting dengan tanpa melihat informasi yang menjadi suatu kebutuhan dari kegiatan *meeting*.

Secara detail menurut (cohen et al, 2011) terdapat 18 karakteristik desain *meeting* yang di representasikan seperti *temporal* (*promptness*), *attendee* (*fasilitator*), *physical* (*meeting setting*), dan *procedural* (*formal agenda, meeting minutes*). Karakteristik desain secara umum bertujuan untuk mengatur perencanaan meeting akan diadakan baik sebelum kegiatan meeting berlangsung atau pada saat meeting sedang berjalan (Jennifer et al, 2015). Hal tersebut dapat memberikan kemudahan dalam hal pelaksanaan kegiatan meeting didalam sebuah organisasi. Pelaksanaan *meeting* dilakukan dengan mengkoordinasikan seluruh perangkat dari kegiatan meeting agar supaya berjalan dengan baik dan mendapatkan solusi dari permasalahan yang diangkat dalam kegiatan tersebut.

2.6.1 Manajemen Meeting

Ilmu manajemen memiliki usia selayaknya kehidupan manusia, karena pada dasarnya manusia dalam kehidupan sehari-hari tidak dapat terlepas dari prinsip-prinsip manajemen. Baik secara langsung maupun tidak langsung, baik disadari maupun tidak disadari manusia menggunakan prinsip-prinsip manajemen dalam menjalani kehidupan sehari-hari. Ilmu manajemen ilmiah timbul sekitar abad ke 20 di benua Eropa barat dan Amerika. Dimana negara-negara tersebut sedang dilanda revolusi yang dikenal dengan sebutan "revolusi industri", yaitu percobaan-percobaan dalam pengelolaan produksi yang efektif dan efisien (Manulang, 1983).

Manajemen meeting merupakan cara untuk melakukan prosedur pelaksanaan meeting dimulai dari proses koordinasi dalam hal pelaksaan maupun kelengkapan hingga jalannya meeting hingga akhir. secara umum manajemen meeting memberikan kemudahan dalam proses jalannya meeting baik dari sisi peserta maupun pihak pengelenggara. secara umum koordinasi meeting dilakukan dengan tujuan mempersiapkan kelengkapan meeting yang bersifat objektif seperti pemimpin meeting hingga notulesi meeting. Selain itu koordinasi dilakukan untuk membentuk forum meeting agar supaya peserta mauapun penyelenggara meeting dapat dengan mudah untuk menyampaikan permasalahan maupun argumen terkait dengan hal tersebut sehingga dapat di dengarkan oleh peserta rapat lain dan diharapkan menemui solusi terhadap permasalahan tersebut.

2.6.1.1 Manajemen Ruang Meeting

Manajemen ruang *meeting* merupakan kegiatan yang bertujuan untuk mengendalikan *meeting* pada suatu ruangan tertentu yang tersedia dan telah dilakukan identifikasi sesuai dengan kebutuhan *meeting* sebelumnya. Aktifitas tersebut memberikan kemudahan dalam terselenggaranya *meeting* dari sisi penggunaan ruang yang tersedia. Hal tersebut menjadi penting disaat ruang *meeting* yang tersedia hanya beberapa ruang saja dan dengan kebutuhan pelengkap ruangan yang minimum. Oleh sebab itu manajemen dilakukan dalam hal mengendalikan proses penggunaan ruang dengan beberapa kebutuhannya disesuaikan berdasarkan kegiatan *meeting* yang akan dilaksanakan. Selain itu yang menjadi tugas utama dari manajemen ruang *meeting* adalah sebagai media dalam memantau bagaimana penggunaan ruang yang ada dan ketersediaan ruang yang memenuhi standar *meeting* yang ada.

2.7 Rekayasa Perangkat Lunak

Menurut IEEE 610.12 rekayasa perangkat lunak adalah suatu studi dan aplikasi dari suatu pendekatan kuantitatifable, dispilin, dan sistematis kepada pengembangan, operasi, dan pemeliharaan perangkat lunak yang kesemuanya itu merupakan aplikasi rekayasa yang berkaitan dengan perangkat lunak (Pressman, 2002). Peneliti dapat menarik kesimpulan bahwa rekayasa perangkat lunak adalah disiplin ilmu yang menangani perancangan, pembuatan, dan pemeliharaan suatu perangkat lunak dengan menggunakan sistem ataupun prinsip aturan tertentu yang sistematis untuk menghasilkan suatu perangkat lunak yang sesuai dengan

kebutuhan nyata pengguna dengan tingkat fungsi dan efisiensi serta efektifitas yang maksimal.

Evolusi perangkat lunak dirasa penting karena sebuah perangkat lunak harus berkembang untuk memnuhi kebutuhan pelanggan yang setiap sasatnya berubah-ubah seiring dengan kebutuhan akan teknologi pada era informasi. Metode perangkat lunak merupakan pendekatan terstruktur terhadap pengembangan suatu perangkat lunak yang bertujuan untuk memberikan fasilitas produksi perangkat lunak berkualitas tinggi dengan cara meningkatkan efektifitas penggunaan biaya. Metode didalam perangkat lunak yang sering digunakan adalah metode berorientasi fungsi, metode berorientasi objek dan metode gabungan yang sering dinamakan *Unified Modeling Language* (UML) (Pressman, 2002).

2.8 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah sebuah "Bahasa" yang telah menjadi standart dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah system. "Dalam kerangka spesifikasi, *Unified Modelling Languange* (UML) menyediakan model-model yang tepat, tidak mendua arti *(ambigu)* serta lengkap (Nugroho, 2005).

Dengan menggunakan UML peneliti dapat membuat model untuk semua jenis aplikasi perangkat lunak, dimana aplikasi tersebut dapat berjalan pada perangkat keras, sistem operasi, dan jaringan apapun, serta ditulis dalam Bahasa pemrograman apapun. Penggunaan UML dengan dasar penggunaan *class* pada metodenya sehingga dapat digunakan pada Bahasa pemrograman berorientasi objek seperti C++, Java, C# atau VB.NET.

Seperti Bahasa pemrograman pada umumnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML utama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Obejct Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*).

Abstaksi konsep dasar UML yang terdiri dari *structural classification, dynamic behavior,* dan *model management,* yang dapat dipahami dengan mudah oleh pihak pengembang aplikasi perangkat lunak (Dharwiyanti, 2003).

2.8.1 Use case Diagram

Usecase diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Dalam usecase yang menjadi kebutuhan utama adalah "apa" yang diperbuat oleh sistem, dan bukan "bagaimana". Suatu usecase mempresentasikan sebuah interaksi antar aktor dengan sistem. Usecase merupakan sebuah

pekerjaan terntu. Sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan terntu. *Use Case Diagram* menampilkan aktor yang menggunakan *use case-use case* mana yang dimasukkan kedalam *use case* lain dan berhubungan antara aktor dan *use case* (Fowler,2005).

Sebuah *use case* dapat melakuakan *include* fungsionaltisas *usecase* lain sebagai bagian dari proses dalam *usecase* itu sendiri. Secara umum diasumsikan bahwa *usecase* yang di-*include* akan dipanggil setiap kali *usecase* yang meng-*include* dieksekusi secara normal. Sebuah *usecase* dapat di-*include* oleh lebih dari satu *usecase* lain. Sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common* (Dharwiyanti, 2003).

Sebuah *usecase* juga dapat melakukan *extends* terhadap *usecase* lain dengan *behavior* pada *usecase* itu sendiri. Sementara itu hubungan generalisasi antar *usecase* menujukkan bahwa *usecase* yang satu merupakan spesialisasi dari yang lain.

Tabel 2.1 Simbol *Use Case* Diagram

Nama	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*		Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi 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 interksi dengan aktor.
Menggunakan/*Include*	<<include>> 	Include memilki makna *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan
Ekstensi/*Extend*	<<extend>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan, serupa dengan prinsip dari notasi *inheritance* pada pemrograman berorientasi objek
Generalisasi/*Generalization*		Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

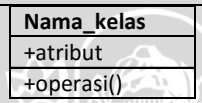
Sumber: (Fowler, 2005)

2.8.2 Class Diagram

Class merupakan sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek (Dharwiyanti, 2003). *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi). *Class* dapat merupakan implementasi dari sebuah *interface,* yaitu *class* abstrak yang hanya memiliki metode. *Interface* tidak dapat langsung diintansiasi, tetapi harus diimplementasi dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metode pada saat *run-time*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package* (Dharwiyanti, 2003).

Hubungan antar *class* meliputi asosiasi, agregasi, pewarisan, dan hubungan dinamis. Relasi asosiasi *class* diagram merupakan hubungan statis antar *class.* Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class.* Hubunga agregasi merupakan hubungan yang menyatakan bagian ("terdiri atas…"). Hubungan pewarisan yaitu hubungan hirarki antar *class. Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metode *class* asalnya dan menambahkan fungsionalitas baru, sehingga dapat disebut anak dari *class* yang mewarisinya. Kebalikan dari pewarisan adalah generalisasi. Hubungan dinamis yang jarang digunakan dalam menggambarkan *class* diagram yaitu hubungan rangkaian pesan yang di-*passing* dari suatu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence* diagram yang akan dijelaskan kemudian.

Tabel 2.2 Simbol *Class* Diagram

Nama	Simbol	Deskripsi
Kelas		Blok pembangunan pada pemrograman berorientasi objek. Bagian atas adalah bagian dari *class.* Bagian bawah tengah mengidentifikasi property/atribu *class.* Bagian akhir mendefinisikan metod dari sebuah *class.*
Asosiasi/*Association*		Garis ini dapat melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hokum-hukum multiplisitas pada sebuah *relationship.*
Asosiasi berarah/*directed association*		Relasi antar kelas dengan makna kelas yang satu digunakan kelas yang lain, asosiasi biasanya disertai dengan multiplicity.
Generalisasi/ *Generalization*		Hubungan dimana objek anak berbagi perilaku dan struktur data dari objek yang ada diatas objek induk.

Kebergantungan/ *Dependency*		Hubungan dimana perubahan terjadi pada suatu emelen mandiri akan mempengaruhi elemen bergantung pada elemen yang tidak mandiri.
Agregasi/*Aggregation*		Mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut relasi.

Sumber: (Fowler, 2005)

2.8.3 Activity Diagram

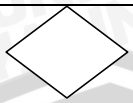
Activity diagram menggambarkan berbagai alir aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berkahir. *Activity* diagram juga dapat menggambarkan proses parallel yang mungkin terjadi pada beberapa eksekusi (Dharwiyanti, 2003).

Activity diagram merupakan *state* diagram khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*) (Fowler, 2005). Oleh karena itu *activity* diagram tidak menggambarkan *behavior* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktifitas dari level atas secara umum. *Node* pada sebuah *activity diagram* disebut sebagai *action,* sehingga diagram tersebut menampilkan sebuah *activity* yang tersusun dari *action* (Fowler, 2005).

Sebuah aktifitas dapat direalisasikan oleh satu *use case* atau lebih. Aktifitas menggambarkan proses yang sedang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktifitas. Serupa dengan *state,* standar UML menggunakan segiempat dengan sudut membualat untuk menggambarkan aktifitas. *Decision* digunakan untuk menggambarkan *behavior* pada kondisi tertentu. Untuk mengilustrasikan proses-proses parallel (*fork* dan *join*) digunakan titik sinkronasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktifitas tertentu.

Tabel 2.3 Simbol *Activity* Diagram

Nama	Simbol	Deskripsi
Status Awal/*Initial*		Titik awal, untuk memulai suatu aktivitas.

Status Akhir/Dependency		Titik akhir, untuk mengakhiri aktivitas.
Aktivitas*/Activity*		Menandakan sebuah aktivitas.
Percabangan/Decision		Pilihan untuk mengambil keputusan.
Fork/Join		Menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan parallel menjadi satu.
Rake		Menunjukkan adanya dekomposisi.

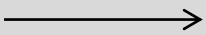
Sumber: (Fowler, 2005)

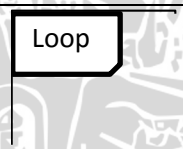
2.8.4 Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek didalam dan disekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadapt waktu (Fowler, 2005). *Sequence* diagram terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence* diagram dapat digunakan untuk menggambarkan scenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktifitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari suatu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi atau metode dari *class. Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah pesan. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller,* dan *persistent entity* (Dharwiyanti, 2003).

Tabel 2.4 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
Boundary		Terletak diantara sistem dengan duni disekitarnya. Semua form, laporan, antarmuka ke perangkat keras seperti printer atau

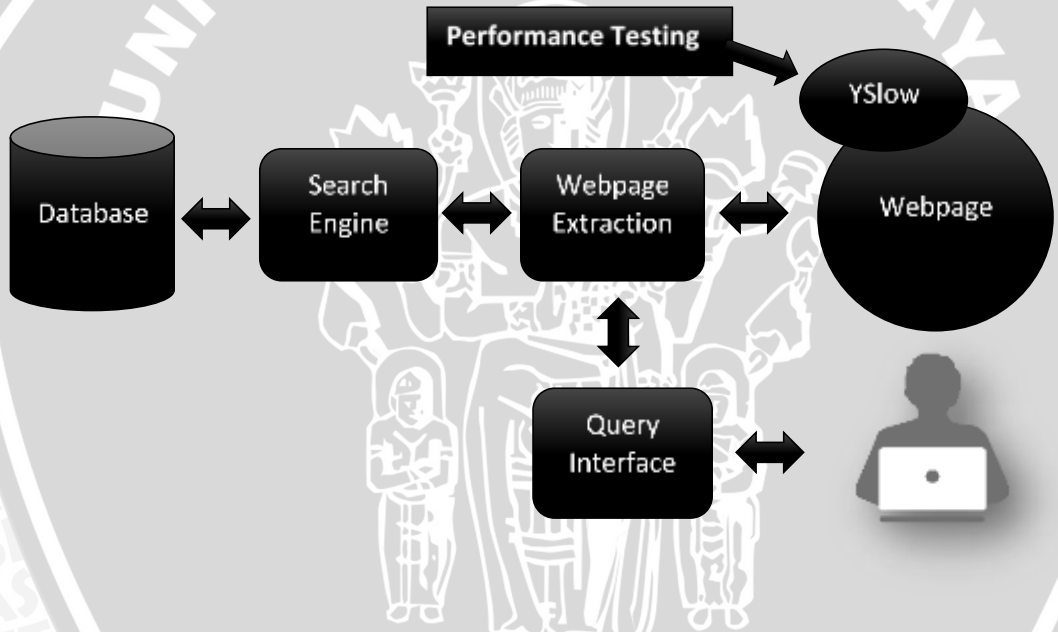
		scanner lainnya termasuk dalam kategori.
Control		Berhubungan dengan fungsionalitas seperti pemanfaatan sumber daya, pemrosesan disrtibusi atau peneganan kesalahan.
Entity		Digunakan untuk menagani informasi yang mungkin akan disampaikan secara permanen.
Pesan/*Message*	Message	Mengindikasikan komunikasi antar objek.
Self-Message		Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.
Loop		Mengeksekusi berulang kali dan melihat iterasi dari sistem.
Pesan tipe *call*	1 : nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri

Sumber: (Fowler, 2005)

2.9 Yslow

Yslow merupakan sebuah *tool* dengan kemampuan untuk menguji performa suatu aplikasi berbasis *website*. *Tool* Yslow merupakan *tool* besutan Yahoo yang memiliki tingkat akurasi dalam pengujian yang baik untuk mendapatkan sebuah optimasi aplikasi *website*. Yslow adalah *tool* pengembangan dan analisis besutan yahoo untuk menguji seluruh komponen dari sebuah halam web dinamis yang dibangun dari penggunaan javascript, component, statistik dan tools upgrade pada 23 matiks kemampuan pengukuran yang dibagi pada enam kategori konten cookies CSS image javascript dan server (Indira, 2013). Enam buah kategori tersebut dapat dijabarkan pada **LAMPIRAN B.**

Tujuan dari pengujian menggunakan *tool* Yslow utamanya untuk mengetahui seberapa baik aplikasi *website* tersebut layak digunakan, kemudian apakah aplikasi *website* tersebut dapat memenuhi *rules* yang terdapat pada *tool* Yslow. Ada beberpa *rules* yang telah didefinisikan oleh yahoo yang dinamakan *Yahoo Exceptional Performance.* Parameter uji yang ada didalam Yslow didasarkan pada *rules* tersebut. Masing – masing *rules* menghasilkan nilai yang berbeda. Nilai yang muncul ada pada *range* A hingga F. *Rules* serta grade pada Yslow dapat dilihat pada **LAMPIRAN C** terkait dengan bagaimana *tool* Yslow menguji perangkat lunak. Untuk meningkatkan kemampuan dari sebuah halaman web yang bekerja dengan menganalisis kemampuan pengujian menggunakan Yslow dan HTTP watch. Lambatnya suatu kemampuan *load* CSS dikarenakan lambatnya menjalankan file JavaScript pada internet explorer, pihak pengembang sering kali menggunakan *method* untuk menjalankan file CSS dengan memanfaatkan halaman kerja JavaScript seperti jQuery atau *prototype.* Hal tersebut membutuhkan iterasi dari pohon DOM untuk menampilkan sebuah kinerja dari JavaScript. Berikut gambar 2.3 merupakan dari proses *load* sebuah halaman website.

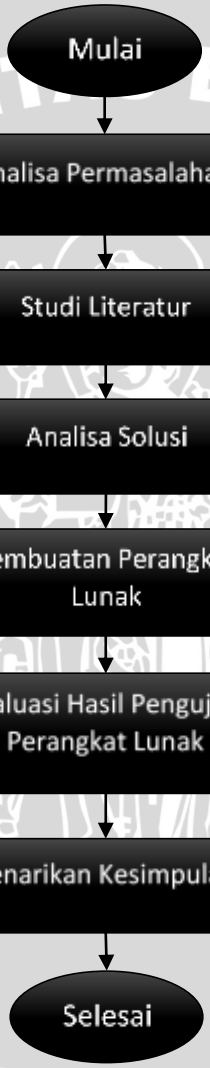


Gambar 2.3 Alur *Load* Halaman *Web*

Sumber: (Indira, 2013)

BAB 3 METODOLOGI

Demi terselesaikannya penelitian ini, maka dibutuhkan susunan tahapan kegiatan penelitian yang terstruktur dan tepat serta perancangan yang baik. Pada metodologi penelitian ini akan dibahas langkah-langkah dan rancangan yang digunakan dalam pembangunan aplikasi manajemen ruang *meeting* dengan pengujian menggunakan tools Yslow Tahapan penelitian ini meliputi analisa permasalahan, studi literatur, analisa solusi, pembuatan perangkat lunak, analisa hasil penelitian, dan penarikan kesimpulan yang ditunjukkan pada Gambar 3.1 berikut ini:



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Analisa Permasalahan

Analisa permasalahan terkait dengan pengembangan sebuah perangkat lunak dengan dasar dan teknologi yang telah di gunakan sebelumnya. Analisa permasalahan dilakukan dalam sebuah studi lapangan terkait dengan aplikasi yang akan dikembangkan. Teknik yang dilakukan dalam proses analisa permasalahan yang akan dikembangankan adalah dengan melakukan wawancara secara langsung dengan pihak *stakeholder* sebagai *user* yang berada dalam sebuah department yang mengelola kegiatan *meeting* (studi kasus pada PT. Telkomsel Indonesia Tbk). Proses wawancara tersebut terdapat pada **LAMPIRAN A**. Disela wawancara tersebut informasi juga didapatkan disaat kondisi peminjaman ruangan untuk proses pelaksanaan wawancara. Selain informasi tersebut, dalam wawancara timbul informasi terkait dengan bagaimana cara memaksimalkan penggunaan ruang *meeting* sesuai dengan keigatan *meeting* yang berlangsung serta dapat sesuai dengan jam operasional kantor. Kemudian pihak *stakeholder* pun memberikan masukan agar setiap permintaan *meeting* dapat dilihat dari sisi kebutuhan dari permintaan tersebut secara detail sehingga user *admin* dapat melihat apa saja yang akan dibutuhkan pada saat pelaksaan kegiatan *meeting*. Diakhir proses wawancara pihak *stakeholder* memberikan data abstrak berupa daftar gedung dan daftar ruangan sebagai acuan dalam proses pengembangan sistem manajemen ruang *meeting*.

3.2 Studi Literatur

Tahapan studi literature dilakukan berdasarkan pada analisa permasalahan yang telah dijelasan sebelumnya sebagai landasan penelitian yang dikerjakan. Studi literatur dilakukan untuk mengumpulkan informasi yang bersumber dari buku, naskah penelitian, dan informasi melalui internet. Referensi dari studi literatur didapat dari beberapa buku, jurnal, *website* dan artikel. Studi literatur berguna untuk untuk menunjang pengerjaan skripsi. Bahasan teori dari referensi yang diambil mengenai sistem informasi, manajemen ruang *meeting*, *website, database, bootstrap,* UML, SDLC *waterfall* model, dan pemahaman tentang *tool* Yslow yang akan diterapkan sebagai media pengujian perangkat lunak. Studi literatur yang berhubungan secara langsung dengan pengembangan aplikasi manajemen ruang *meeting* yaitu dengan model *software life cycle waterfall* yang menjadi model pengembangan aplikasi tersebut. Selain itu, dalam pengembangannya, dilakukan pengujian untuk dapat melihat hasil pengembangan aplikasi tersebut. Adapun pengujian dilakuan dengan memanfaatkan *tool testing* Yslow yang akan diterpakan sebagai media dalam melakukan pengujian dari sisi performa perangkat lunak.

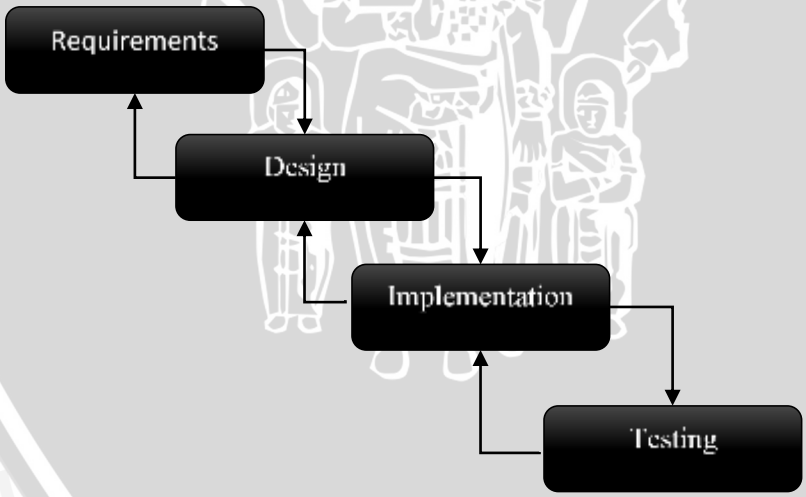
3.3 Analisa Solusi

Pada tahapan analisa solusi didapatkan sebuah perumusan solusi terkait dengan permasalahan yang timbul. Berdasarkan pada penjelasan fase analisa permasalahan dan didukung dengan studi litertur sebagai penunjang, didapatkan inti dari permasalahan tersebut yang menjadi landasan dalam pengembangan

aplikasi perangkat lunak tersebut. Peneliti mengambil solusi dari permasalahan yang ada pada fase analisa permasalahan yaitu pihak *stakeholder* menginginkan suatu aplikasi yang dapat merubah perlikau kerja yang cenderung bersifat konvensional kedalam kerja yang bersifat praktis dan efektif serta memiliki efisiensi yang tinggi. *Stakeholder* berharap aplikasi tersebut dapat memaksimalkan kinerja yang ada sebelumnya seperti pada penggunaan ruang *meeting* serta penjadwalan antar *meeting* agar supaya dapat di lakukan kontrol baik secara langsung maupun secara berkala. Analisa solusi menghasilkan sebuah deskripsi umum sistem yang akan dikembangankan oleh peneliti. Sehingga atas dasaran analisa solusi, peneliti mengambil langkah untuk melakukan pengembangan aplikasi manajemen ruang *meeting* dengan menerapkan metode pengembangan berupa model *waterfall* serta menganalisa permasalahan yang telah disebutkan oleh pihak *stakeholder.*

3.4 Pembuatan Perangkat Lunak

Tahapan dalam pembuatan perangkat lunak ini dilakukan untuk menentukan urutan pembuatan sebuah perangkat lunak sesuai dengan alur proses atau fase pembuatan berdasarkan *System Development Life Cycle* pada model *waterfall*. Fase yang terdapat didalam model *waterfall* meliputi *requirements*, *design*, *implementation*, dan *testing.* Fase tersebut berjalan secara linier dengan batasan setiap fase dilakukan hingga mencapai target yang sesuai kemudian fase lainnya juga akan berjalan hingga fase akhir pada model *waterfall.* Adapun penjelasan setiap fase dalam model *waterfall* terdapat pada gambar 3.2.



Gambar 3.2 Model *Waterfall*

3.4.1 Fase *Requirements*

Pada fase ini proses *requirements* yang dilakukan peneliti adalah identifikasi kebutuhan berupa apa saja yang menjadi pendukung dalam melakukan pengembangan aplikasi manajemen ruang *meeting*. Kebutuhan yang diambil sercara umum berupa kebutuhan fungsional dan non fungsional. Kebutuhan fungsional didapatkan dari proses pengambilan kebutuhan melalui media internet

sebagai referensi dan melihat aplikasi manajemen ruang *meeting* sejenis yang sudah ada sebelumnya serta literatur yang mendukung serta mendasari pengambilan kebutuhan untuk pengembangan aplikasi manajemen ruang *meeting*. Hasil dari tahap ini berupa spesifikasi kebutuhan perangakat lunak yang disertai dengan identifikasi aktor pada sistem dan diagram *use case* beserta skenarionya.

3.4.2 Fase *Design*

Pada fase *design* ini akan dilakukan jika seluruh kebutuhan dari sistem telah dipenuhi melalui fase *requirements*. Berdasarkan dengan prinsip model *waterfall,* fase *design* disini menggambarkan sebuah rancangan umum terkait dengan aplikasi manajemen ruang *meeting. Design* dapat berupa *blueprint* yang menjadi dasar untuk proses implementasi. Hasil dari tahap ini berupa *design usecase design arcitehcture*, *design class diagram, squance diagram,* dan *design* antarmuka.

3.4.3 Fase *Implementation*

Dalam tahapan ini, aplikasi akan di implementasikan berdasrkan perancangan sistem yang telah dibuat pada tahap *design* akan diubah menjadi sebuah kode program. Kode program tersebut diimplementasikan dengan Bahasa PHP pada sisi *back end* dan *syntax* HTML, CSS, dan JavaScript dari sisi *front end*. Hasil dari tahap ini berupa sistem yang sesuai dengan rancangan sisem pada fase *design* kemudian dialakukan proses *testing* untuk dapat mengetahui apakah proses implementasi telah dilakukan dengan baik dan sesuai.

3.4.4 Fase *Testing*

Pada fase ini akan dilakukan *testing* dasar berupa *black box* dan *white box*. Sistem akan diuji berdasarkan kesesuaian antara hasil dari fase *implementation* dengan hasil pada fase *requirements*. Sehingga didapatkan aplikasi yang baik dan sesuai serta layak digunakan oleh pihak *stekholder*. Hasil dari fase ini berupa sistem yang telah memenuhi standar pengujian dasar dan sesuai pada kaidahnya.

3.5 Evaluasi Hasil Pengujian Perangkat Lunak

Tahapan evaluasi pengujian aplikasi manajemen ruang *meeting* dilakukan agar didapatkan hasil pengujian dari fase *testing* pada model *waterfall* yang merupakan pengujian dasar dan menghasilkan kesesuaian antara kebutuhan yang telah didefinisikan berupa kebutuhan fungsional serta implementasi yang telah dilaksanakan. Pengujian *black box* sebagai fase kesesuaian dengan kebutuhan yang telah di analisa pada fase *requirements* dan fase *implementation* perangkat lunak. Sedangkan pengujian *white box* berupa pengujian aplikasi yang dilakukan berdasrkan pada *test case* yang merupakan pengujian untuk mendapatkan alur atau cara kerja program secara internal. Pada tahapan ini, dalam fase *testing* pada model *waterfall* dilakukan untuk mendapatkan hasil yang sesuai dengan kaidah pengembangan dan layak untuk digunakan secara umum kepada pihak *stakeholder.* Penelitian ini dilakukan agar supaya mendapatkan hasil yang

maksimal dalam proses pengembangannya untuk dapat memberikan nilai lebih yang akan didapatkan melalui pihak *stakeholder.* Peneliti melakukan pengujian lanjutan untuk mendapatkan hasil yang maskimal terhadap pengembangan aplikasi perangkat lunak. Peneliti melihat sisi performa sebuah perangkat lunak yang dapat diuji untuk menunjang proses pengujian dasar dalam fase *testing* pada model *waterfall*. Selain itu, pengujian *performa* dllakukan untuk mendapatkan hasil yang lebih baik dan layak untuk digunakan secara umum oleh pihak *stakeholder*.

Peneliti menerpakan sebuah *tool* yang dapat menguji apakah aplikasi tersebut memiliki tingkat performa yang baik, sedang atau bahkan kurang. *Tool* Yslow yang diterapkan oleh peneliti untuk proses pengujian performa aplikasi. *Tool* Yslow merupakan *tool* yang dibangun oleh Yahoo untuk menguji tingkat performa aplikasi agar pihak *end-user* menjadi lebih nyaman dalam melakukan akses terdapat aplikasi tersebut. Aplikasi perangkat lunak yang diuji merupakan aplikasi perangkat lunak berbasis *web.* Pada *tool* ini pengujian dilakukan berdasarkan pada performa yang meliputi kecepatan akses aplikasi berdasarkan faktor komponen yang terdapat didalam sistem serta faktor environment suatu alamat *web* yang tersedia. Aplikasi dapat dikatakan memiliki performa yang baik jika hanya kedua faktor tersebut dapat terpenuhi dengan baik. Terkait dengan pengujian performa aplikasi, pada penelitian ini termasuk kedalam lingkup dari kebutuhan non-fungsional sebagai penujang tambahan dalam proses pengembangnya. Terdapat beberapa alat ukur atau dapat disebut matriks uji pada *tool* Yslow yang merepresentasikan daftar pengujian perangkat lunak untuk memudahkan proses pengukuran. Pengujian performa yang dilakukan oleh *tool* Yslow berkaitan langsung dengan proses kecepatan akses sistem secara keseluruhan. Peneliti melakukan pengujian performa dari sisi efisiensi yang merupakan salah satu pengujian yang terdapat dalam *tool* Yslow. Efisiensi yang dimaksud didalamnya yaitu pengujian sistem yang dilalukan secara langsung oleh *end-user* yang diantaranya membawahi kecepatan akses, pemakaian *resource*, dan kecepatan proses sistem secara keseluruhan.

Hasil dari tahapan evaluasi pengujian perangkat lunak ini adalah aplikasi dapat memberikan nilai performa yang baik terhadap *end-user* dengan beberapa aspek pengujian dari sisi efisiensi dalam lingkup performa aplikasi. Kemudian dari evaluasi tersebut dilakukan analisa dari proses *load test* untuk mengetahui bagaimana aplikasi yang baik dapat di pertahankan serta dilakukan *maintanace* secara berkala dan aplikasi yang mendapatkan nilai yang kurang baik dapat ditingkatkan dengan melakukan evaluasi efisiensi dengan berdasarkan pada penerapan pengujian *tool* Yslow.

3.6 Penarikan Kesimpulan dan Saran

Pada tahap akhir dalam penelitian adalah penarikan kesimpulan. Tahap ini dilakukan setelah seluruh tahapan pengembangan perangkat lunak yaitu analisis permasalahan hingga evaluasi hasil pengujian perangkat lunak selesai. Tahap ini terdiri dari dua bagian yaitu kesimpulan dan saran. Kesimpulan diambil dari hasil

pengujian dan analisis sistem dan analisa hasil penelitian yang ditujukan untuk menjawab rumusan masalah yang telah diuraikan sebelumnya. Sedangkan saran digunakan sebagai perbaikan kesalahan-kesalahan yang terjadi dan penyempurnaan penelitian yang akan dilakukan selanjutnya. Penarikan kesmpulan dan saran tersebut mengacu kepada hasil dari pengujian dan analisis.

BAB 4 ANALISIS DAN PERANCANGAN

Bab ini memaparkan perancangan dan implementasi dari aplikasi manajemen ruang *meeting.* Perancangan dijabarkan menjadi dua tahap. Tahap pertama yang dilakukan adalah analisis kebutuhan. Sedangkan tahap kedua yang dilakukan adalah perancangan sistem. Kemudian tahap implementasi terdiri dari pembahasan terkait dengan spesifikasi lingkup implementasi, batasan dalam implementasi, implementasi basis data, kelas yang terdapat pada program, dan impelmentasi antarmuka.

4.1 Deskripsi Umum Sistem

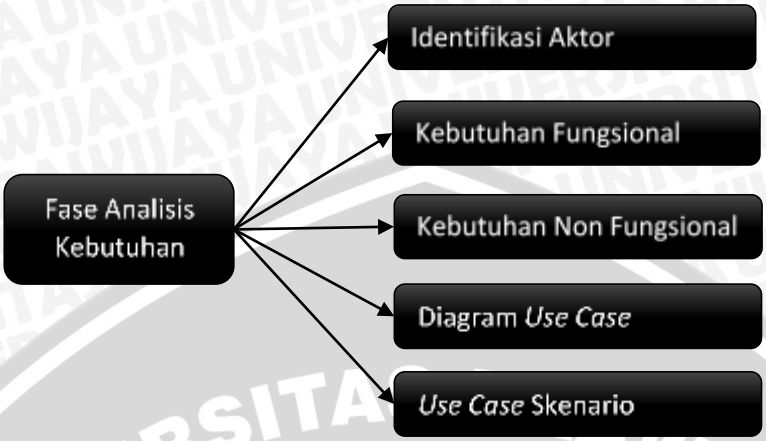
Aplikasi manajemen ruang *meeting* adalah aplikasi yang dapat membantu proses penjadwalan kegiatan *meeting.* Aplikasi ini mencangkup kebutuhan dari pihak *end user.* Aplikasi manajemen ruang *meeting* bersifat general atau umum namun lebih spesifik pada sebuah instansi yang memliki department yang membawahi pengelolahan ruang *meeting*. Sehingga aplikasi tersebut dapat berjalan dengan struktur yang ada pada department tersebut. Aplikasi ini berbasis *web* sehingga proses pemesanan ruang *meeting* dapat dilakukan dengan mudah dan cepat serta efektif dan efisien. Hal ini melibatkan penggunaan koneksi yang memanfaatkan area dalam suatu instansi peruashaan. Oleh sebab itu aplikasi lebih spesifik dalam penggunaannya didalam lingkup suatu instansi perusahaan.

Aplikasi manajemen ruang *meeting* bertujuan untuk meng-*handle* kegiatan *meeting* agar lebih terjadwal dan tidak mengalami bentrok dari segi jam hingga penggunaan ruang *meeting.* Untuk mendukung hal tersebut peneliti memanfaatkan beberapa user lain untuk terlibat dalam pengoperasian aplikasi manajemen ruang *meeting*. Diantarnya admin yakni sebagai user yang mengatur keseluruhan sistem yang berjalan dan sebagai pihak yang memiliki hak akses tertinggi dalam proses pemeliharaan aplikasi manajemen ruang *meeting*. Kemudian requestor yakni user yang melakukan pemesanan ruang *meeting* didalam sistem. Requestor memiliki atribut username dan password sebagai syarat utama untuk masuk kedalam sistem sebelum dapat melakukan pemesanan ruangan. Hak akses requestor diberikan oleh admin setelah user mendaftarkan diri sebagai requestor. Selain itu terdapat receptionist yakni user yang melakukan kontrol terhadap jalannya kegiatan *meeting.* Receptionist juga merupakan user yang memiliki hak akses serupa dengan requestor dari sisi pendaftaran user namun disini hak yang diberikan berbeda.

4.2 Analisis Kebutuhan

Pada tahap analisis kebutuhan pada penelitian ini diawali dengan elisitasi kebutuhan yang menjelaskan kebutuhan berdasarkan pada masing-masing aktor yang terlibat, identifikasi aktor yang merupakan penjabaran dari masing-masing aktor, kebutuhan fungsional yang meliputi kebutuhan yang spesifik pada aplikasi manajemen ruang *meeting*, kebutuhan non fungsional yang merupakan kebutuhan luar sistem terkait dengan kemampuan yang dimiliki oleh aplikasi

manajemen ruang *meeting*, diagram *use case,* dan *use case* skenario. Gambar 4.1 mempresentasikan tahapan dari analisis kebutuhan.



Gambar 4.1 Diagram alir analisa kebutuhan

Sumber: (Sutcliffe Alistair, 2010)

4.2.1 Identifikasi Aktor

Didalam tahapan ini dilakukan identifikasi aktor yang berperasn dalam menjalankan aplikasi manajemen ruang *meeting.* Identifikasi didapat melalui fase gambaran umum sistem. Berikut pada table 4.1 representasi aktor beserta dekripsi dan perannya.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
Admin	Admin merupakan user yang memliki hak akses terting dalam sistem. Admin dapat menjalankan aplikasi secara keseluruhan serta dapat melakukan perawatann sistem.
Requestor	Requetor merupakan user yang memiliki hak akses hanya untuk memesan ruang *meeting* serta melihat daftar *meeting* yang telah dipesan.
Receptionist	Receptionist merupakan user yang memiliki hak akses berupa melakukan alih jalannya kegiatan *meeting*.

4.2.2 Kebutuhan Fungsional

Tahapan analisis kebutuhan fungsional sistem adalah analisa utama untuk menentukan kebutuhan apa yang mendasari pengembangan sistem. Dalam analisa yang dilakukan peneliti ini, diperlukan pengetahuan khusus di bidang pengembangan perangkat lunak agar supaya hasil dari analisa dapat diperoleh hasil yang optimal. Berikut kebutuhan fungsional yang terdapat pada aplikasi manajemen ruang *meeting* terdapat pada table 4.2.

Tabel 4.2 Kebutuhan Fungsional

No.	Kode Kebutuhan	Kebutuhan	Nama Kebutuhan	Prioritas	Peran
1.	MR_01	Sistem dapat melakukan pemesanan ruang rapat.	Memesan ruangan rapat	Tinggi	Requestor
2.	MR_02_01	Sistem dapat melakukan perubahan informasi permintaan rapat yang telah dibuat.	Mengubah informasi permintaan rapat	Tinggi	Requestor
3.	MR_02_02	Sistem dapat melakukan pembatalan permintaan rapat.	Membatalkan permintaan rapat	Tinggi	Requestor
4.	MR_03	Sistem dapat memberikan informasi terkait kegiatan permintaan rapat melalui teknologi SMS *Gateaway*.	Memberikan informasi rapat	Tinggi	Admin
5.	MR_04	Sistem melakukan pembuatan file PDF sebagai sarana daftar hadir bagi peserta rapat.	Menghasilkan daftar absensi	Tinggi	Receptionist
6.	MR_05	Sistem dapat menampilkan rincian permintaan rapat yang telah di terima oleh admin.	Melihat rincian permintaan rapat.	Sedang	Admin
7.	MR_02_03	Sistem dapat menerima permintaan rapat yang masuk	Menerima permintaan rapat	Sedang	Admin

		kedalam daftar permintaan rapat.			
8.	MR_02_04	Sistem dapat menerima pemabatalan permintaan rapat.	Menerima pembatalan permintaan rapat	Sedang	Admin
9.	MR_06_01	Sistem dapat melakukan manajemen pengguna seperti penambahan pengguna baru dan memperbarui informasi dari pengguna tersebut yang masing-masing berdasarkan pada *privillage* yang terdaftar.	Manajemen pengguna	Sedang	Admin
10.	MR_06_02	Sistem dapat melakukan manajemen ruangan seperti penambahan ruangan baru beserta dengan fasilitas yang ada serta pembaruan informasi ruangan.	Manajemen ruangan	Sedang	Admin
11.	MR_06_03	Sistem dapat melakukan manajemen gedung seperti penambangan gedung baru dan pembaruan informasi gedung teresbut.	Manajemen gedung	Sedang	Admin
12.	MR_07	Sistem dapat melihat frekuensi penggunaan	Melihat frekuensi penggunaan ruangan	Sedang	Admin

		raungan rapat yang telah terpakai.			
13.	MR_08	Sistem dapat melihat daftar ruangan rapat kosong yang dapat di pesan.	Melihat daftar ruangan kosong	Sedang	Requestor
14.	MR_09	Sistem dapat melakukan penyaringan terhadap daftar permintaan rapat.	Menyaring daftar permintaan ruangan	Sedang	Receptionist
15.	MR_10_01	Sistem dapat mengunduh file daftar hadir untuk melakukan absensi kehadiran rapat.	Mengunduh daftar absensi	Sedang	Receptionist
16.	MR_10_02	Sistem dapat menggungah file daftar hadir agar ruangan rapat dapat di tutup.	Mengunggah daftar absensi	Sedang	Receptionist
17.	MR_11	Sistem dapat menutup ruangan rapat setelah kegiatan rapat selesai dilakukan dan daftar hadir telah di unggah kedalam sistem.	Menutup ruangan rapat	Sedang	Receptionist
18.	MR_12_01	Sistem dapat melihat seluruh daftar permintaan rapat yang telah di terima oleh admin.	Melihat daftar permintaan rapat yang diterima	Rendah	Admin, receptionist
19.	MR_12_02	Sistem dapat melihat daftar permintaan rapat yang dibuat oleh requestor.	Melihat daftar permintaan rapat.	Rendah	Admin
20.	MR_12_03	Sistem dapat melihat daftar	Melihat daftar permintaan	Rendah	Requestor

		permintaan rapat yang telah dibuat.	rapat yang dibuat.		
21.	MR_12_04	Sistem dapat menampilkan daftar pembatalan permintaan rapat yang telah dibuat.	Melihat daftar pembatalan permintaan rapat.	Rendah	Admin
22.	MR_13	Seluruh aktor dapat masuk kedalam sistem dengan memasukkan username dan password yang telah terdaftar.	*Login*	Rendah	Admin, requestor, Receptionist

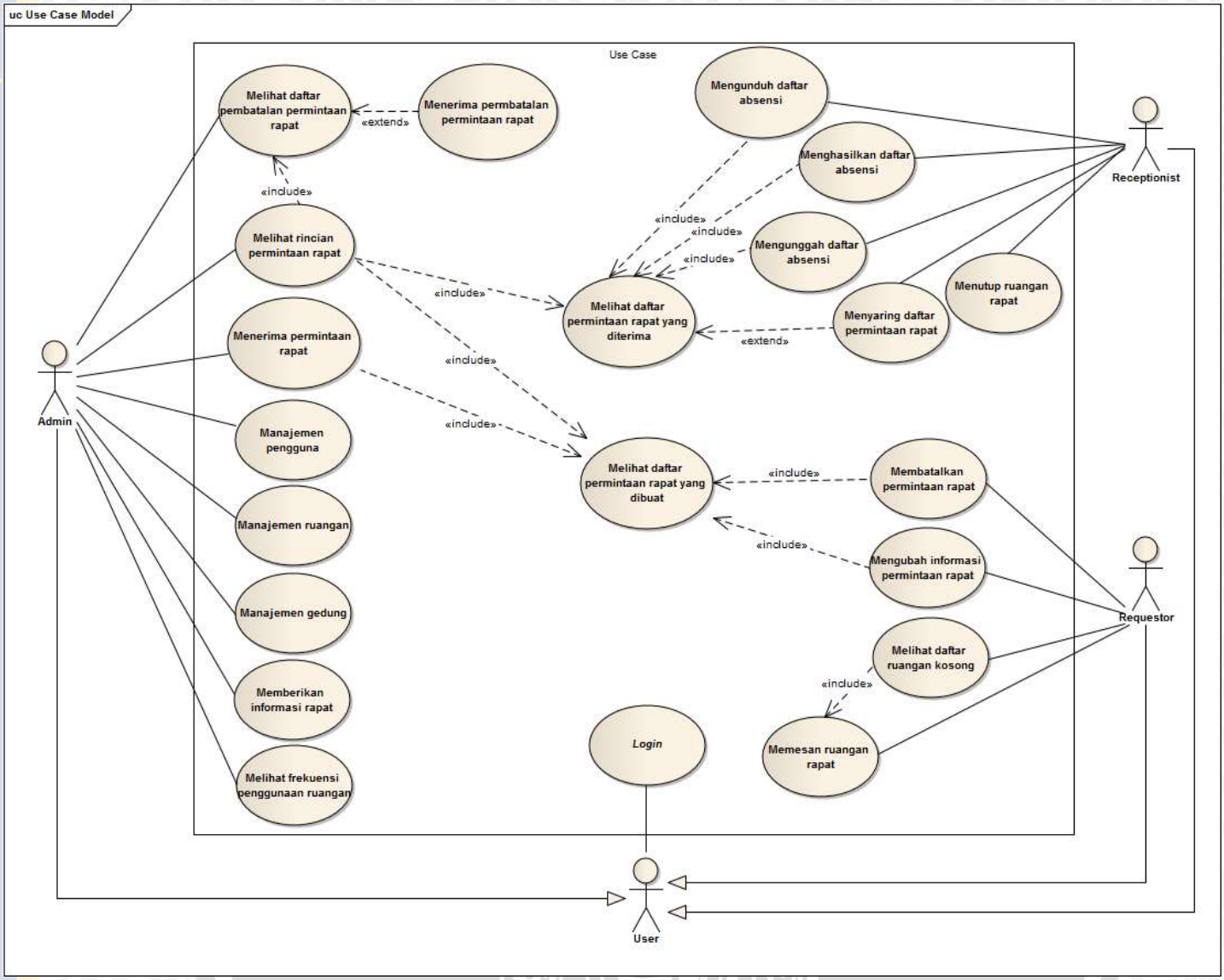
4.2.3 Kebutuhan Non Fungsional

Kebutuhan non-fungsional yang terdapat didalam sistem meliputi *performance system, security system,* dan lainnya. Untuk melakukan sebauh analisa terkait dengan kebutuhan non-fungsional, agar dapat diintegrasikan dengan hasil dari analisa kebutuhan fungsional sebelumnya agar nantinya tidak terjadi kesalahan yang mengakibatkan harus digantinya daftar kebutuhan fungsional. Berikut list dari kebutuhan non fungsional yang mendukung pengembangan aplikasi manajemen ruang *meeting* terdapat pada table 4.3.

Tabel 4.3 Kebutuhan Non Fungsional

No.	Kode Kebutuhan	Parameter	Deskripsi
1.	MR_14	Performance	Sistem memberikan *response time* terhadap aksi user kurang dari 1 detik serta tingkat efisiensi aplikasi yang juga berkaitan secara langsung dengan *load test* aplikasi berbasis *web*.

4.2.4 Diagram Use Case



Gambar 4.2 Diagram Use Case

4.2.5 Use Case Skenario

Use case scenario memberikan keterangan yang lebih detail dengan berdasarkan pada diagram use case yang telah di bentuk. Berikut masing-masing penjelasan case berdasarkan masing-masing aktor yang bersangkutan.

1. *Use case* pemesanan ruangan rapat

Tabel 4.4 Pemesanan Ruangan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_01
Nama kebutuhan	Memesan ruangan rapat

Tujuan	Untuk melakukan pemesanan ruangan rapat yang dilakukan berdasarkan jadwal yang dipilih oleh aktor.
Deskripsi	*Use case* ini menjelaskan bagaimana proses pemesanan ruangan yang dilakukan dengan memasukan beberapa informasi kemudian aktor dapat memilih ruangan yang tersedia.
Aktor	Requestor
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan pemesanan ruangan rapat.
Aksi aktor	**Reaksi sistem**
1. Aktor masuk kedalam menu *Room*	2. Sistem mengecek status autentikasi.
3. Aktor masuk kedalam sub menu *request room*	4. Sistem merespon perintah dari sub menu *request* untuk selanjutnya aktor dapat memesan ruangan *meeting*.
5. Aktor mulai meng-*input*-kan data kelangkapan untuk mendapatkan jadwal yang tepat serta kebutuhan yang tepat	6. Sistem memproses inputan yang ada untuk menampilkan hasil berupa jadwal ruangan yang kosong sesaui dengan kebutuhan aktor.
7. Aktor melakukan pemilihan ruangan berdasarkan dengan kebutuahan yang ada.	8. Sistem memproses perintah tombol *booking* untuk selanjutnya sistem menampilkan kebutuhan yang telah di *input* kan oleh aktor.
9. Aktor menekan tombol *booking* untuk melakukan pemesanan ruangan.	10. Sistem merespon tombol save untuk kemudian menuju halaman utama aktor untuk menunggu proses penerimaan oleh admin.
11. Aktor menekan tombol save untuk memastikan data yang dimasukkan tepat.	12. Sistem menyimpan data yang telah dimasukkan oleh aktor.
Skenario Alternatif 1: Jika aksi aktor tidak valid	
	Sistem akan kembali pada halaman *home* untuk dilakukan *input* kembail pada form yang tersedia.

Kondisi akhir	Pemesanan ruangan berhasil disimpan kedalam sistem dan masuk kedalam daftar *request*.

2. Use case mengubah informasi permintaan rapat

Tabel 4.5 Mengubah Informasi Permintaan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_02_01
Nama kebutuhan	Mengubah informasi permintaan rapat
Tujuan	Untuk merubah informasi prmintaan rapat yang telah dibuat.
Deskripsi	*Use case* ini menjelaskan bagaimana informasi permintaan rapat dapat dirubah oleh admin.
Aktor	Requestor
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat merubah permintaan ruangan dari daftar yang ada pada halaman *home*.
Aksi aktor	**Reaksi sistem**
1. Aktor menekan tombol *edit* pada kolom aksi	2. Sistem mengecek status autentikasi.
	3. Sistem menampilkan halaman *edit* berupa form yang dapat ubah berdasarkan informasi yang tersedia.
4. Aktor mengisi kolom judul *meeting*, total peserta, fasilitas, dan konsumsi	5. Sistem memproses perintah *edit* dan informasi yang telah diperbarui disimpan kedalam sistem.
Skenario Alternatif 1: Jika aktor tidak ingin merubah permintaan	
	Aktor menekan tombol *cancel* untuk membatalkan perubahan informasi.
Kondisi akhir	Informasi baru berhasil disimpan kedalam sistem.

3. Use case membatalkan permintaan rapat

Tabel 4.6 Membatalkan Permintaan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_02_02
Nama kebutuhan	Membatalkan permintaan rapat
Tujuan	Untuk melakukan pemabatalan permintaan rapat yang telah diterima oleh admin.
Deskripsi	*Use case* ini menjelaskan bagaimana proses pembatalan permintaan rapat yang telah diterima oleh admin.
Aktor	Requestor
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan pembatalan permintaan rapat penggunaan ruangan.
Aksi aktor	**Reaksi sistem**
1. Aktor menekan tombol batal pada kolom aksi	2. Sistem mengecek status autentikasi.
	3. Sistem merespon perintah dari tombol batal kemudian masuk kedalam halaman batal *request.*
4. Aktor mengisi form yang tersedia untuk menjelaskan pembatalan permintaan rapat yang dibuat.	5. Sistem memproses perintah batal dan data yang telah disimpan kedalam sistem.
Skenario Alternatif 1: Jika aktor tidak ingin membatalkan permintaan	
	Aktor menekan tombol *cancel* untuk membatalkan perubahan informasi.
Kondisi akhir	Pemesanan pembatalan ruangan berhasil dibatalkan dan dihapus dari daftar *request*.

4. Use case memberikan informasi rapat

Tabel 4.7 Memberikan Informasi Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_03
Nama kebutuhan	Memberikan informasi rapat
Tujuan	Untuk memberikan informasi kepada requestor terkait dengan rapat yang akan diadakan pada jadwal yang ditentukan.
Deskripsi	*Use case* ini menjelaskan bagaimana alur proses penyampaian informasi permintaan rapat dengan memanfaatkan teknologi *SMS Gateaway*.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah login dan mendapatkan informasi terkait dengan kegiatan rapat yang akan berlangsung.
Aksi aktor	**Reaksi sistem**
1. Aktor menekan tombol sms pada kolom aksi	2. Sistem mengecek status autentikasi.
	3. Sistem merespon perintah dari tombol sms kemudian sistem menampilkan pop up pemberitahuan bahwa sms telah dikirim.
Kondisi akhir	SMS berhasil tersampaikan oleh sistem kepada requestor dengan baik.

5. Use case menghasikan daftar absensi

Tabel 4.8 Menghasilkan Daftar Absensi

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_04
Nama kebutuhan	Menghasilkan daftar absensi
Tujuan	Untuk menghasilkan file daftar absensi yang berisi kolom-kolom sejumlah peserta rapat.
Deskripsi	*Use case* ini menjelaskan bagaimana proses pembentukan sistem mannghasilkan file daftar hadir

	berdasarkan total peserta rapat yang di definisikan sebelumnya.
Aktor	Receptionist
Skenario Utama	
Kondisi awal	Aktor telah *login* dan sistem dapat menampilkan daftar absensi yang dapat di unduh oleh aktor sebagai absensi peserta rapat.
Aksi aktor	**Reaksi sistem**
1. Aktor menekan tombol PDF pada kolom daftar peserta rapat.	2. Sistem mengecek status autentikasi.
	3. Sistem merespon perintah dari tombol PDF untuk kemudian sistem menampilkan file PDF berisi kolom sebanyak perserta rapat.
Kondisi akhir	File PDF dapat di edarkan di dalam ruangan rapat untuk dapat disi oleh para peserta rapat.

6. Use case melihat rincian permintaan rapat

Tabel 4.9 Melihat Rincian Permintaan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_05
Nama kebutuhan	Melihat rincian permintaan rapat
Tujuan	Agar dapat mengetahui rincian dari permintaan rapat yang terdapat pada daftar permintaan rapat aktor.
Deskripsi	*Use case* ini menjelaskan bagaimana sistem dapat menampilkan keseluruhan kebutuhan yang di definisikan oleh requestor untuk mendukung kegiatan rapat yang akan berlangsung.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melihat rincian permintaan rapat.
Aksi aktor	**Reaksi sistem**

1. Aktor masuk kedalam halaman *home*.	2. Sistem mengecek status autentikasi
	3. Sistem menampilkan seluruh daftar dari permintaan rapat yang dibuat.
4. Aktor melihat seluruh daftar pemrintaan aktor dan memilih mana yang akan dilihat rinciannya.	
5. Aktor menekan tombol *detail* pada kolom aksi.	6. Sistem memproses perintah dan menampilkan informasi yang berdasarkan pada kebutuhan.
Kondisi akhir	Sistem dapat menampilkan rincian permintaan rapat sesuai dengan permintaan rapat yang diinginkan.

7. Use case menerima permintaan rapat

Tabel 4.10 Menerima Permintaan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_02_03
Nama kebutuhan	Menerima permintaan rapat
Tujuan	Aktor dapat menerima permintaan rapat yang dibuat oleh requetor.
Deskripsi	*Use case* ini menjelaskan bagaimana aktor dapat melakukan penerimaan permintaan rapat dari requestor untuk menunjang dalam pelaksanaan kegiatan rapat.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan aksi *approve* permintaan rapat pada daftar permintaan rapat yang ada.
Aksi aktor	**Reaksi sistem**
1. Aktor masuk kedalam halaman *home*.	2. Sistem mengecek status autentikasi
	3. Sistem menampilkan seluruh daftar dari *request* yang dibuat.

4. Aktor melihat seluruh daftar dari permintaan rapat yang ada dan kemudian aktor memilih permintaan rapat yang akan di terima.	
5. Aktor menekan tombol detail pada kolom aksi.	
6. Aktor memilih tombol accept untuk menerima *request*	7. Sistem memproses perintah dan menampilkan informasi yang berdasarkan pada kebutuhan serta secara langsung memberikan update terhadap request yang telah di terima pada halaman home requestor.
Kondisi akhir	Sistem menerima permintaan rapat yang dibuat oleh requestor dan dapat menjalankan kegiatan rapat sesuai dengan jadwal yang ada.

8. Use case menerima pembatalan permintaan rapat

Tabel 4.11 Menerima Pembatalan Permintaan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_02_04
Nama kebutuhan	Menerima pembatalan permintaan rapat
Tujuan	Agar aktor dapat melakukan pembatalan *request* dengan alasasn yang kuat.
Deskripsi	*Use case* ini menjelaskan bagaimana ketika aktor mendapat kegiatan yang mendesak sehingga aktor melakukan penerimaan pembatalan permintaan rapat dari requestor kepada admin.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat menerima pembatalan permintaan rapat.
Aksi aktor	**Reaksi sistem**

1. Aktor masuk kedalam halaman *home*.	2. Sistem mengecek status autentikasi
	3. Sistem menampilkan seluruh daftar permintaan rapat yang dibuat.
4. Aktor melihat seluruh permintaan pembatalan rapat dan memilih salah satu permintaan pembatalan rapat.	
5. Aktor memilih tombol accept untuk membatalkan permintaan rapat	6. Sistem memproses perintah dan menampilkan informasi serta secara langsung memberikan update terhadap permintaan rapat yang telah di batalkan pada halaman *home* requestor.
Kondisi akhir	Sistem menerima pembtalan permintaan rapat yang dibuat oleh requestor.

9. Use case manajemen pengguna

Tabel 4.12 Manajemen Pengguna

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_06_01
Nama kebutuhan	Manajemen pengguna
Tujuan	Untuk melakukan manajemen pengguna dari sistem berdasarkan *privillage* dan manajemen informasi sesuai dengan kebutuhan pengguna.
Deskripsi	*Use case* ini menjelaskan bagaimana sistem melakukan manajemen pengguna dalam hal penambahan pengguna baru kedalam sistem hingga pembaruan informasi pengguna sesuai dengan *privillage* dan kebutuhan dari masing-masing pengguna.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan manajemen pengguna.
Aksi aktor	**Reaksi sistem**

Penambahan Pengguna Baru	
1. Aktor menekan tombol menu *manage*.	2. Sistem mengecek status autentikasi
3. Aktor memilih tombol *new* pada kolom *user* didalam sub menu *manage*.	4. Sistem memproses perintah dan masuk kedalam form pengisian data diri *user*.
5. Aktor mengisi data diri sesuai dengan kebutuhan beserta dengan *privillage* dari *user* baru	
6. Aktor menekan tombol *save* untuk menyimpan seluruh data diri yang telah diisikan kedalam form	7. Sistem menyimpan data diri user berserta dengan *privillage* yang ditentukan oleh user.
	8. Sistem menampilkan pop up berupa user baru telah ditambahkan.
Pembaruan Informasi Pengguna	
1. Aktor menekan tombol menu *manage*.	2. Sistem mengecek status autentikasi.
3. Aktor memilih tombol update pada kolom *user* didalam sub menu *manage*.	4. Sistem memproses perintah dan masuk kedalam form pembaruan informasi *user*.
5. Aktor melihat daftar pengguna dan memilih penggua yang akan diperbarui informasinya.	
6. Aktor mengisi pembaruan data diri sesuai dengan kebutuhan beserta	

dengan *privillage* yang ada.	
7. Aktor menekan tombol save untuk menyimpan seluruh data diri yang telah diisikan kedalam form	8. Sistem menyimpan data diri user berserta dengan *privillage* yang ditentukan oleh user.
	9. Sistem menampilkan pop up berupa pembaruan informasi user telah tersimpan.
Skenario Alternatif 1: Jika aksi aktor tidak valid	
	Pengisian data diri *user* baru tidak terpenuhi secara keseluruhan maka sistem akan melakukan *reload* halaman form penambahan *user* baru
Scenario Alternatif 2: jika *input* data tidak valid	
	Sistem menampilkan <!> *warning* bahwa form harus diisi.
Kondisi akhir	Penambahan *user* baru telah berhasil dan tersimpan didalam database aplikasi.

10. Use case manajemen ruangan

Tabel 4.13 Manajemen Ruangan

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_06_02
Nama kebutuhan	Manajemen ruangan
Tujuan	Aktor dapat melakukan manajemen ruangan yang terkait dengan penambahan ruangan dan pembaruan informasi ruangan.
Deskripsi	*Use case* ini menjelaskan bagaimana aktor melakukan manajemen ruangan dengan berdasarkan ketersediaan ruangan yang baru untuk dapat didaftarkan kedalam sistem serta dapat dilakukan pembaruan informasi terkait dengan ruangan sesuai dengan kebutuhan.
Aktor	Admin
Skenario Utama	

Kondisi awal	Aktor telah *login* dan dapat melakukan manajemen ruangan sesuai dengan kebutuhan.
Aksi aktor	**Reaksi sistem**
Penambahan Ruangan Baru	
1. Aktor menekan tombol menu *manage*.	2. Sistem mengecek status autentikasi
3. Aktor memilih tombol *new* pada kolom *room* didalam sub menu *manage*.	4. Sistem memproses perintah dan masuk kedalam form pengisian data ruangan yang ingin ditambahkan.
5. Aktor mengisi data ruangan sesuai dengan kebutuhan beserta dengan fasilitasnya.	
6. Aktor menekan tombol *save* untuk menyimpan seluruh data ruangan yang telah diisikan kedalam form	7. Sistem menyimpan data ruangan berserta dengan fasilitas yang telah ditentukan.
	8. Sistem menampilkan pop up berupa ruangan baru telah ditambahkan.
Pembaruan Informasi Ruangan	
1. Aktor menekan tombol sub menu manage pada header halaman.	2. Sistem mengecek status autentikasi.
3. Aktor memilih tombol *update* pada kolom *room* didalam sub menu *manage*.	4. Sistem memproses perintah dan masuk kedalam form pembaruan informasi ruangan.
5. Aktor memilih ruangan yang ingin di lakukan pembaruan informasi.	

6. Aktor mengisi pembaruan informasi ruangan sesuai dengan kebutuhan beserta dengan fasilitas yang ada.	
7. Aktor menekan tombol *save* untuk menyimpan seluruh informasi ruangan yang telah diisikan kedalam form	8. Sistem menyimpan informasi ruangan berserta dengan fasilitas yang telah ditentukan.
	9. Sistem menampilkan pop up berupa pembaruan informasi ruangan telah tersimpan.
Skenario Alternatif 1: Jika aksi aktor tidak valid	
	Pengisian data diri user baru tidak terpenuhi secara keseluruhan maka sistem akan melakukan *reload* halaman form penambahan ruangan baru
Skenario Alternatif 2: Jika input data tidak valid	
	Sistem menampilkan <!> *warning* bahwa form harus diisi.
Kondisi akhir	Penambahan ruangan baru telah berhasil dan tersimpan didalam sistem.

11. Use case manajemen gedung

Tabel 4.14 Manajemen Gedung

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_06_03
Nama kebutuhan	Manajemen gedung
Tujuan	Aktor dapat melakukan manajemen ruangan yang terkait dengan penambahan gedung dan pembaruan informasi gedung.
Deskripsi	*Use case* ini menjelaskan bagaimana aktor melakukan manajemen gedung dengan berdasarkan ketersediaan gedung yang baru untuk dapat didaftarkan kedalam sistem serta dapat dilakukan

	pembaruan informasi terkait dengan gedung sesuai dengan kebutuhan.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan manajemen gedung sesuai dengan kebutuhan.
Aksi aktor	**Reaksi sistem**
Penambahan Gedung Baru	
1. Aktor menekan tombol menu *manage*.	2. Sistem mengecek status autentikasi
3. Aktor memilih tombol *new* pada kolom *building* didalam sub menu *manage*.	4. Sistem memproses perintah dan masuk kedalam form pengisian data gedung yang ingin ditambahkan.
5. Aktor mengisi data g sesuai gedung dengan kebutuhan beserta dengan data yang tersedia.	
6. Aktor menekan tombol *save* untuk menyimpan seluruh data gedung yang telah diisikan kedalam form	7. Sistem menyimpan data gedung berserta dengan data yang telah ditentukan.
	8. Sistem menampilkan pop up berupa gedung baru telah ditambahkan.
Pembaruan Informasi Ruangan	
1. Aktor menekan tombol sub menu manage pada header halaman.	2. Sistem mengecek status autentikasi.
3. Aktor memilih tombol *update* pada kolom *building* didalam sub menu *manage*.	4. Sistem memproses perintah dan masuk kedalam form pembaruan informasi gedung.

5. Aktor memilih gedung yang ingin di lakukan pembaruan informasi.	
6. Aktor mengisi pembaruan informasi gedung sesuai dengan kebutuhan beserta dengan fasilitas yang ada.	
7. Aktor menekan tombol *save* untuk menyimpan seluruh informasi gedung yang telah diisikan kedalam form	8. Sistem menyimpan informasi gedung berserta dengan fasilitas yang telah ditentukan.
	9. Sistem menampilkan pop up berupa pembaruan informasi gedung telah tersimpan.
Skenario Alternatif 1: Jika aksi aktor tidak valid	
	Pengisian data gedung baru tidak terpenuhi secara keseluruhan maka sistem akan melakukan *reload* halaman form penambahan gedung baru
Skenario Alternatif 2: Jika input data tidak valid	
	Sistem menampilkan <!> *warning* bahwa form harus diisi.
Kondisi akhir	Penambahan gedung baru telah berhasil dan tersimpan didalam sistem.

12. Use case melihat frekuensi penggunaan ruangan

Tabel 4.15 Melihat Frekuensi Penggunaan Ruangan

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_07
Nama kebutuhan	Melihat frekuensi penggunaan ruangan
Tujuan	Untuk melakukan melihat frekuensi penggunaan ruang rapat.

Deskripsi	*Use case* ini menjelaskan bagaimana ruang rapat di rekap penggunaannya untuk dapat dievaluasi terhadap penggunaan ruang lainnya.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melihat frekuensi penggunaan ruang.
Aksi aktor	**Reaksi sistem**
1. Aktor menekan tombol menu *manage*.	2. Sistem mengecek status autentikasi
3. Aktor memilih tombol *user freq room* pada sub menu *manage*.	4. Sistem memproses perintah dan masuk kedalam halaman *sort* untuk menentukan penggunaan ruang pada bulan dan tahun berapa.
5. Aktor mengisi bulan dan tahun yang ingin dilakukan rekapan penggunaan ruang.	
	6. Sistem menampilkan frekuensi penggunaan ruang dalam bentuk diagram batang.
Skenario Alternatif 1: Jika aksi aktor tidak valid	
	Pengisian bulan dan tahun tidak terpenuhi, maka sistem akan melakukan *reload* halaman tersebut.
Kondisi akhir	Frekuensi penggunaan ruang dapat ditampilkan dalam bentuk diagram batang sesuai dengan bulan dan tahun yang telah di inputkan.

13. Use case melihat daftar ruangan kosong

Tabel 4.16 Melihat Daftar Ruangan Kosong

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_08
Nama kebutuhan	Melihat daftar ruangan kosong
Tujuan	Untuk melihat daftar ruangan kosong yang dapat segera di pesan.

Deskripsi	*Use case* ini menjelaskan bagaimana aktor melihat ruang kosong berdasarkan beberapa inputan yang tersedia.
Aktor	Requestor
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melihat daftar ruangan kosong yang siap untuk dipesan.
Aksi aktor	**Reaksi sistem**
1. Aktor menekan tombol menu *room*.	2. Sistem mengecek status autentikasi
3. Aktor memilih tombol *request for room* pada sub menu *room*.	4. Sistem memproses perintah dan masuk kedalam halaman *check availability*
5. Aktor mengisi inputan pada *field* yang tersedia.	
6. Aktor menekan tombol *check*.	7. Sistem memproses perintah dan menampilkan ruangan-ruangan yang dapat dipesan.
Skenario Alternatif 1: Jika aksi aktor tidak valid	
	Pengisian inputan tidak terpenuhi, maka sistem akan melakukan *reload* halaman tersebut.
Kondisi akhir	Sistem menampilkan ruangan-ruangan berdasarkan inputan aktor.

14. Use case menyaring daftar permintaan rapat

Tabel 4.17 Menyaring Daftar Permintaan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_09
Nama kebutuhan	Menyaring daftar permintaan rapat
Tujuan	Untuk melakukan filter dari daftar permintaan rapat sesuai dengan inputan.
Deskripsi	*Use case* ini menjelaskan bagaimana aktor melakukan penyaringan dalam melihat permintaan rapat.
Aktor	Receptionist

Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan penyaringan daftar permintaan rapat.
Aksi aktor	**Reaksi sistem**
1. Aktor kedalam halaman *home*.	2. Sistem mengecek status autentikasi
	3. Sistem menampilkan seluruh daftar permintaan rapat yang diterima oleh admin.
4. Aktor melakukan penyaringan dengan memasukkan tanggal	
	5. Sistem menampilkan daftar permintaan rapat berdasrkan inputan *user*.
Kondisi akhir	Sistem melakukan filter berdasarkan tanggal agar aktor dapat dengan mudah memilih apa yang kegiatan rapat yang harus didahulukan persiapannya.

15. Use case mengunduh daftar absensi

Tabel 4.18 Mengunduh Daftar Absensi

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_10_01
Nama kebutuhan	Mengunduh daftar absensi
Tujuan	Untuk memudahkan dalam hal rekapan data perserta rapat dengan menggunakan absensi file tersebut.
Deskripsi	*Use case* ini menjelaskan bagaimana aktor dapat mengunduh daftar absensi bagi peserta rapat.
Aktor	Receptionist
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan unduh file daftar hadir yang tersedia.
Aksi aktor	**Reaksi sistem**
1. Aktor kedalam halaman *home*.	2. Sistem mengecek status autentikasi

	3. Sistem menampilkan seluruh daftar permintaan rapat yang diterima oleh admin.
4. Aktor memilih permintaan rapat dan kemudian menekan tombol unduh.	
	5. Sistem menampilkan daftar absensi yang siap untuk di unduh.
Kondisi akhir	Sistem menampilkan daftar absensi yang berisi sejumlah perserta rapat yang telah di deskripsikan sebelumnya dan dapat di unduh.

16. Use case mengunggah daftar absensi

Tabel 4.19 Mengunggah Daftar Absensi

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_10_02
Nama kebutuhan	Mengunggah file daftar hadir peserta
Tujuan	Untuk syarat dalam melakukan tutup ruangan agar dapat digunakan untuk kegiatan rapat lainnya.
Deskripsi	*Use case* ini menjelaskan bagaimana aktor dapat mengunggah daftar absensi dan sistem dapat menyimpan daftar absensi tersebut dan secara langsung melalui aktor rapat dapat ditutup.
Aktor	Receptionist
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melakukan unggah file daftar hadir yang tersedia.
Aksi aktor	**Reaksi sistem**
1. Aktor kedalam halaman *home*.	2. Sistem mengecek status autentikasi
	3. Sistem menampilkan seluruh daftar permintaan rapat yang diterima oleh admin.
4. Aktor menekan tombol unggah PDF	

	5. Sistem menampilkan halaman untuk memilih file hasil scan beruapa absensi kehadiran perserta yang telah di unduh sebelumnya.
6. Aktor menekan tombol *upload* untuk mengunggah file hasil scan tersebut.	7. Sistem akan menyimpan file tersebut kedalam sistem.
Skenario Alternatif 1: Jika aktor tidak ingin mengunggah file	
	Aktor dapat menekan tombol *back* untuk membatalkan proses unggah daftar absensi.
Kondisi akhir	Sistem menyimpan daftar absensi yang telah di pilih oleh aktor dan telah di unggah untuk disimpan dalam sistem.

17. Use case menutup ruangan rapat

Tabel 4.20 Menutup Ruangan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_11
Nama kebutuhan	Menutup ruangan rapat
Tujuan	Agar ruangan rapat dapat digunakan kembali oleh requestor lain.
Deskripsi	*Use case* ini menjelaskan bagaimana proses aktor menutup ruang rapat dengan syarat mengunggah file hasil absensi peserta rapat.
Aktor	Receptionist
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat mentutup ruangan rapat.
Aksi aktor	**Reaksi sistem**
1. Aktor kedalam halaman *home*.	2. Sistem mengecek status autentikasi
	3. Sistem menampilkan seluruh daftar permintaan rapat yang dapat ditutup oleh aktor.
4. Aktor menekan tombol close room	5. Sistem akan menghapus list rapat yang telah di close.
Kondisi akhir	Sistem akan menghapus kegiatan rapat dimana aktor telah melakukan unggah daftar absensi yang

	kemudian akan dilakukan penutupan ruangan sekaligus ruangan tersebut dapat digunakan kembali oleh requestor lain.

18. Use case melihat seluruh daftar permintaan rapat yang diterima

Tabel 4.21 Melihat Seluruh Daftar Permintaan Rapat yang Diterima

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_12_01
Nama kebutuhan	Melihat seluruh daftar permintaan rapat yang diterima.
Tujuan	Untuk melihat seluruh daftar dari permintaan rapat yang telah diterima oleh admin.
Deskripsi	*Use case* ini menjelaskan bagaimana sistem dapat menampilkan daftar permintaan rapat yang tela diterima oleh admin.
Aktor	Admin, Receptionist
Skenario Utama	
Kondisi awal	Aktor telah *login* dan sistem menampilkan daftar dari permintaan rapat yang diterima oleh admin.
Aksi aktor	**Reaksi sistem**
1. Aktor masuk kedalam halaman *home*.	2. Sistem mengecek status autentikasi.
	3. Sistem menampilkan seluruh daftar permintaan rapat.
Kondisi akhir	Sistem dapat menampilkan status dari keseluruhan permintaan rapat yang diterima oleh admin.

19. Use case melihat daftar permintaan rapat requestor

Tabel 4.22 Mellihat Daftar Permintaan Rapat Requestor

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_12_02
Nama kebutuhan	Melihat daftar permintaan rapat requestor
Tujuan	Untuk melihat daftar permintaan rapat yang dibuat oleh requestor.

Deskripsi	*Use case* ini menjelaskan bagaimana sistem dapat menampilkan daftar permintaan rapat dari requestor.
Aktor	Admin
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melihat daftar permintaan rapat dari requestor.
Aksi aktor	**Reaksi sistem**
1. Aktor masuk kedalam menu *room*	2. Sistem mengecek status autentikasi.
3. Aktor menekan tombol *request room list* pada sub menu *room*	
	4. Sistem menampilkan seluruh daftar permintaan rapat yang dibuat oleh requestor.
Kondisi akhir	Sistem dapat menampilkan daftar permintaan rapat keseluruhan yang dibuat oleh requestor.

20. Use case melihat daftar permintaan rapat yang dibuat

Tabel 4.23 Melihat Daftar Permintaan Rapat yang Dibuat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_12_03
Nama kebutuhan	Melihat daftar permintaan rapat yang dibuat
Tujuan	Untuk melihat daftar dari permintaan rapat yang telah dibuat.
Deskripsi	*Use case* ini menjelaskan bagaimana sistem dapat menampilkan daftar dari permintaan rapat yang dibuat oleh aktor yang dapat di sajikan pada halaman utama aktor.
Aktor	Requestor
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melihat daftar permintaan rapat yang telah dibuat.
Aksi aktor	**Reaksi sistem**

1. Aktor masuk kedalam halaman *home*.	2. Sistem mengecek status autentikasi.
	3. Sistem menampilkan seluruh permintaan rapat yang telah dibuat oleh aktor.
Kondisi akhir	Sistem dapat menampilkan keseluruhan dari daftar permintaan rapat yang telah dibuat.

21. *Use case* Melihat daftar pembatalan permintaan rapat

Tabel 4.24 Melihat Daftar Pembatalan Permintaan Rapat

Skenario Kasus pada Sistem	
Kode kebutuhan	MR_12_04
Nama kebutuhan	Melihat daftar pembatalan permintaan rapat
Tujuan	Untuk melihat daftar dari pembatalan permintaan rapat.
Deskripsi	*Use case* ini menjelaskan bagaimana sistem dapat menampilkan daftar dari pembatalan permintaan rapat yang dibuat oleh aktor yang dapat di sajikan pada halaman *cancel room* aktor.
Aktor	Requestor
Skenario Utama	
Kondisi awal	Aktor telah *login* dan dapat melihat daftar pembatalan permintaan rapat.
Aksi aktor	**Reaksi sistem**
1. Aktor masuk kedalam halaman *cancel room*	2. Sistem mengecek status autentikasi.
	3. Sistem menampilkan seluruh pembatalan permintaan rapat yang telah dibuat oleh aktor.
Kondisi akhir	Sistem dapat menampilkan keseluruhan dari daftar pembatalan permintaan rapat yang telah dibuat.

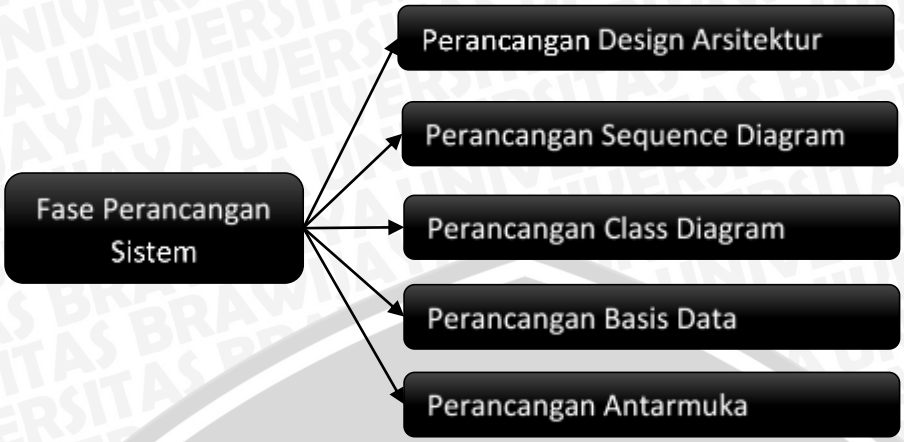
22. *Use case login*

Tabel 4.25 Login

Skenario Kasus pada Sistem	
Kode Kebutuhan	MR_13
Nama Kebutuhan	*Login*
Tujuan	Untuk masuk kedalam sistem dan dapat mengakses aplikasi sesuai dengan *privillage* dari masing-masing aktor.
Deskripsi	Use case ini menjelaskan bagaiamana user dapat masuk kedalam sistem dengan menggunakan *username* dan *password* yang sesuai
Aktor	Admin, Requestor, Receptionist
Skenario Utama	
Kondisi awal	Aktor telah *login* dan masuk kedalam halaman *home* masing-masing.
Aksi Aktor	**Reaksi Sistem**
1. Aktor meng-*input*-kan *username* dan *password*	2. Sistem mengecek username dan password aktor sesuai dengan *privillage*
	3. Sistem menampilkan halaman home masing-masing aktor.
Skenario Alternatif 1: Jika sesi aktor tidak valid	
	Sistem akan kembali melakukan *reload* pada halaman login
Kondisi akhir	Sistem menampilkan halaman *home* dari masing-masing aktor.

4.3 Perancangan Sistem

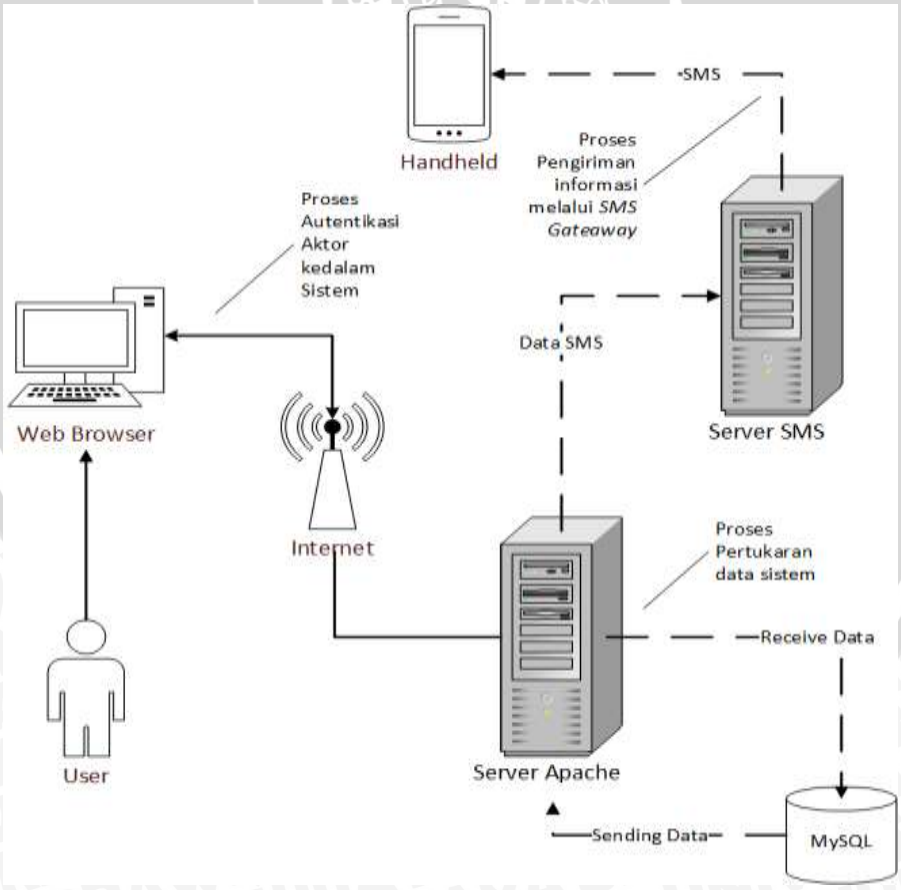
Tahapan perancangan sistem pada penelitian ini dilakukan dengan lima tahapan pengerjaan yaitu peracangan arsitektur dari sistem yang dibangun, memodelkan diagram class, kemudian melakukan pemodelan diagram ER (Entity Relation), lalu memodelkan diagram sequence berdasarkan aktifitasnya, dan pada tahap akhir perancangan sistem dilakukan pemodelan antarmuka. Perancangan sistem dilakukan dengan pendekatan berorientasi objek dengan memanfaatkan diagram UML (Unifiend Modeling Language). Gambar 4.3 merepresentasikan urutan fase perancangan sistem.



Gambar 4.3 Diagram Alir Fase Perancangan Sistem

4.3.1 Perancangan Design Arsitektur Diagram

Diagram arstiketur merupakan diagram yang menggambarkan rancangan atau *design* alur berjalannya program di lapangan secara langsung. Pada diagram ini digambarkan komponen pendukung untuk melakukan rancangan atau *design* sistem. Pada gambar 4.4 merepresentasikan diagram arsitektur.



Gambar 4.4 Arsitektur Diagram

Perancangan diagram arsitektur berdasarkan pada kegiatan yang berlangsung pada sistem dan bertujuan untuk memudahkan *stakeholder* membaca alur sistem berjalan secara general atau umum. Penjelasan terkait dengan perancangan diagram arsitektur didasarkan pada kebutuhan fungisonal yang telah di definiskan sebelunya pada sub bab analisa kebutuhan sistem. Berikut pada table 4.26 menjelaskan keterkaitan diantar diagram arsitektur dengan kebutuhan fungsional.

Tabel 4.26 Runutan Aktifitas Arsitektur Diagram

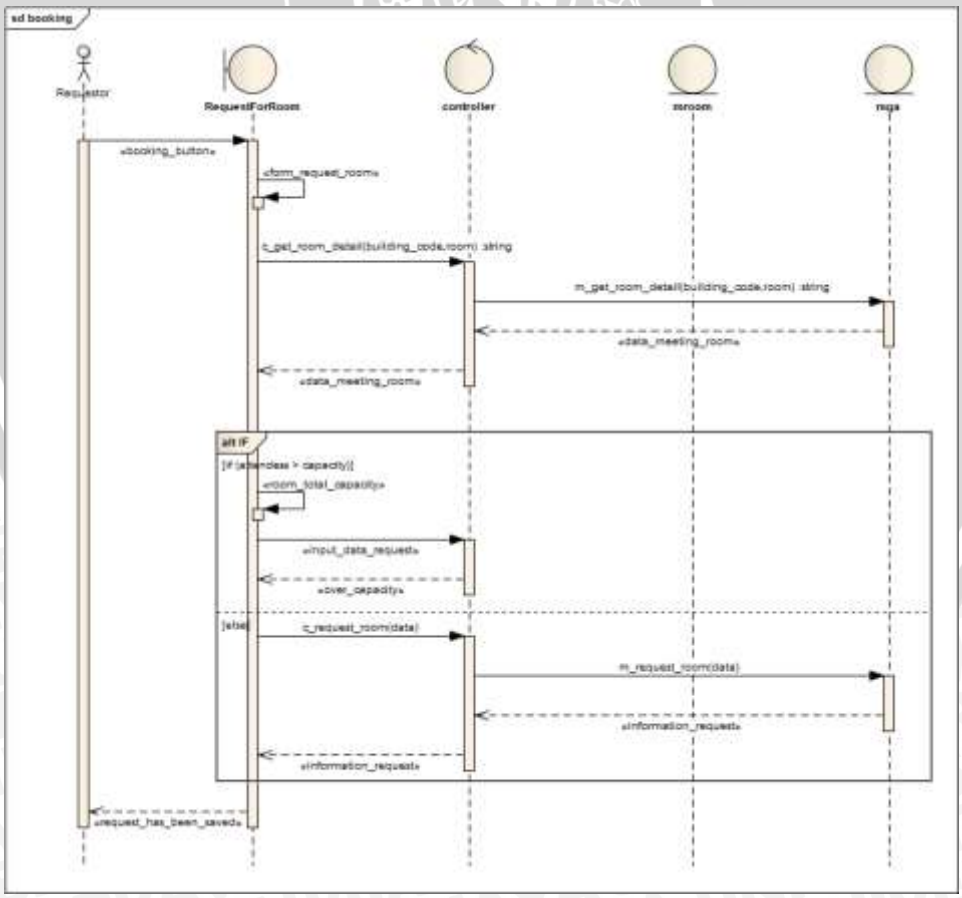
Aktifitas	Deskripsi	Kode kebutuhan
Proses autentikasi aktor kedalam sistem *User → Web Browser ↔ Internet ↔ Server Apache ↔ Database*	Dijelaskan pada gambar arsitektur diagram diatas adalah aktor dapat masuk kedalam sistem dengan memanfaatkan perangkat komputer dengan dukungan berupa sebuah koneksi internet untuk dapat melakukan autentikasi kedalam sistem berdasarkan hak akses yang telah diberikan.	MR_14
Proses pertukaran data sistem *User → Web Browser ↔ Internet ↔ Server Apache ↔ Database*	Berdasarkan diagram proses kedua ini aktor telah masuk kedalam sistem dan dapat melakukan pengolahan data sesuai dengan hak dari masing-masing aktor. Pertukaran data pada sistem dibutuhkan satu buah server sebagai penghubung dan satu buah database sebagai ruang penyimpanan seluruh data dari sistem tersebut.	MR_01 MR_02_01 MR_02_02 MR_04 MR_05 MR_06 MR_02_03 MR_02_04 MR_07 MR_08 MR_09 MR_10 MR_11_01 MR_11_02 MR_12 MR_13_01 MR_13_02

		MR_13_03 MR_13_04
Proses pengiriman informasi dengan menggunakan *SMS Gateaway* *User → Web Browser ↔ Internet ↔ Server Apache ↔ Database ↔ Server SMS → Handheld*	Pada proses ketiga yaitu pengiriman informasi dilakukan dengan memanfaatkan teknologi *SMS Gateaway* yang dapat terintegrasi dengan menggunakan satu buah server yang terhubung.	MR_03

4.3.2 Perancangan Sequence Diagram

Tahap perancangan *sequence diagram* dilakuakan bedasrkan aktifitas dari aplikasi manajemen ruang *meeting* yang berlangsung. Didalam diagram tersebut menjelaskan setiap alur berdasarkan dengan impelmentasi yang terdapat dalam program. Berikut *sequence diagram* yang terdapat dalam aplikasi manajemen ruang *meeting.*

1. Sequence diagram pemesanan ruang rapat

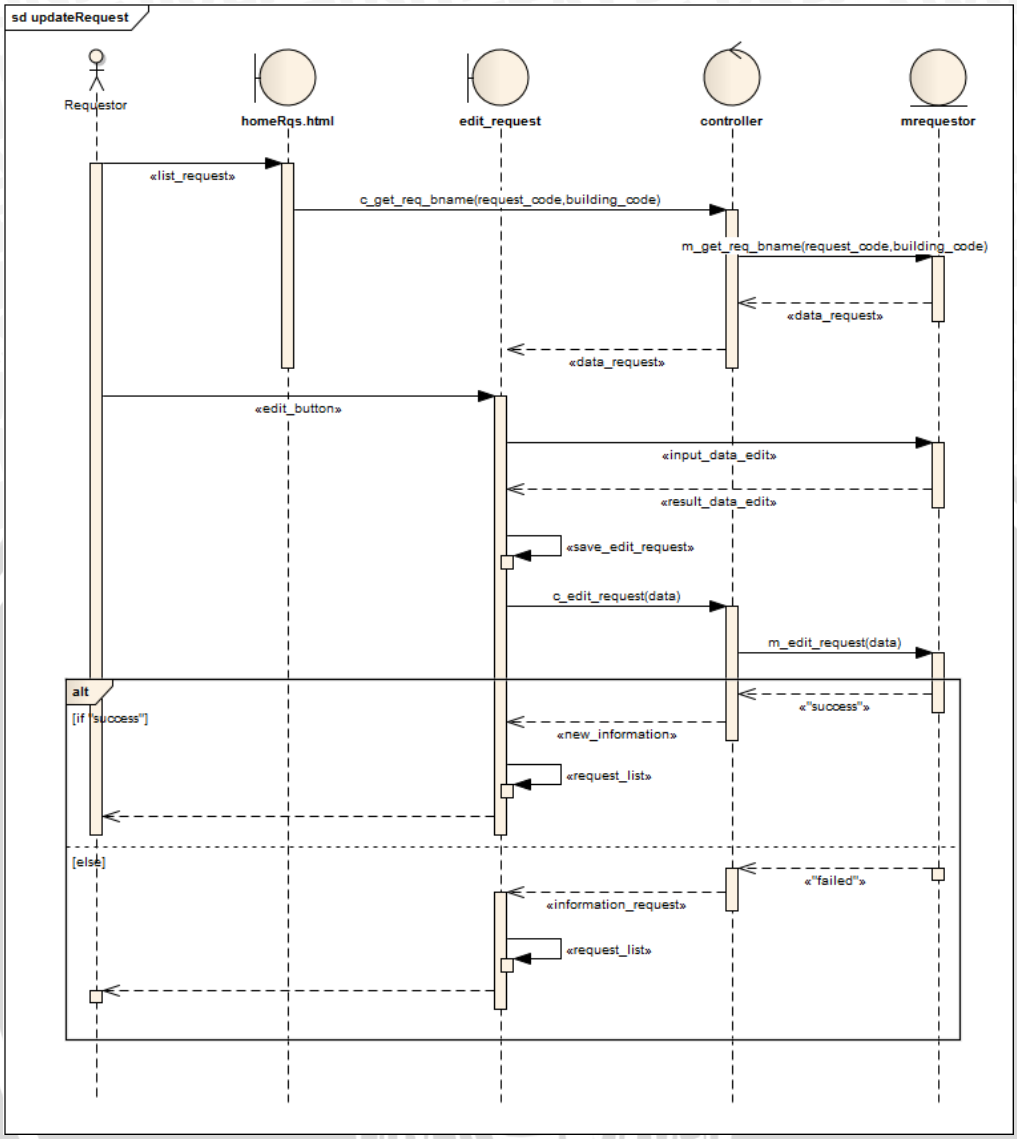


Gambar 4.5 Sequence Pemesanan Ruang *Meeting*

Pada diagram sequence pemesanan ruangan menggambarkan salah satu aktor yaitu requestor dapat melakukan pemesan ruang rapat yang menjadi aktifitas utama dari aplikasi manajemen ruang *meeting*. Pada diagram tersebut terdapat satu buah aktor, view, controller, dan dua buah model. Adapun alur aktifitas pada diagram tersebut meliputi requestor berhubungan langsung dengan view requerForRoom untuk melakukan pemesana ruang *meeting*. Kemudian pada view tersebut mengirimkan method pada controller untuk proses menampilkan seluruh gedung yang terdaftar didalam sistem dengan menggunakan perulangan. Kemudian pada pemilihan kapasistas ruangan, tidak berbeda dengan alur proses menampilkan seluruh gedung yang terdaftar dengan memanfaatkan proses perulangan yang terhubung dengan database sistem. Selanjutnya alur ini mengolah informasi yang didapatkan dari proses input gedung, kapasitas serta tanggal.

Sehingga hasil yang muncul adalah daftar ruangan berdasarkan data inputan yang tersedia serta status dari masing – masing ruangan di masing – masing jam yang tersedia. Selanjutnya requestor hanya tinggal memilih ruangan mana yang ingin dipesan di jam yang telah tersedia. Setelah requestor menekan tombol booking, akan muncul form yang digunakan untuk mengisi ketentuan data pemesana ruangan *meeting* sesuai dengan field yang tersedia. Hasil akhir pada diagram tersebut adalah requestor dapat memesan ruangan berdasarkan jam yang tersedia dan sistem dapat memberikan keterangan bahwa pemesanan ruang *meeting* berhasil diproses.

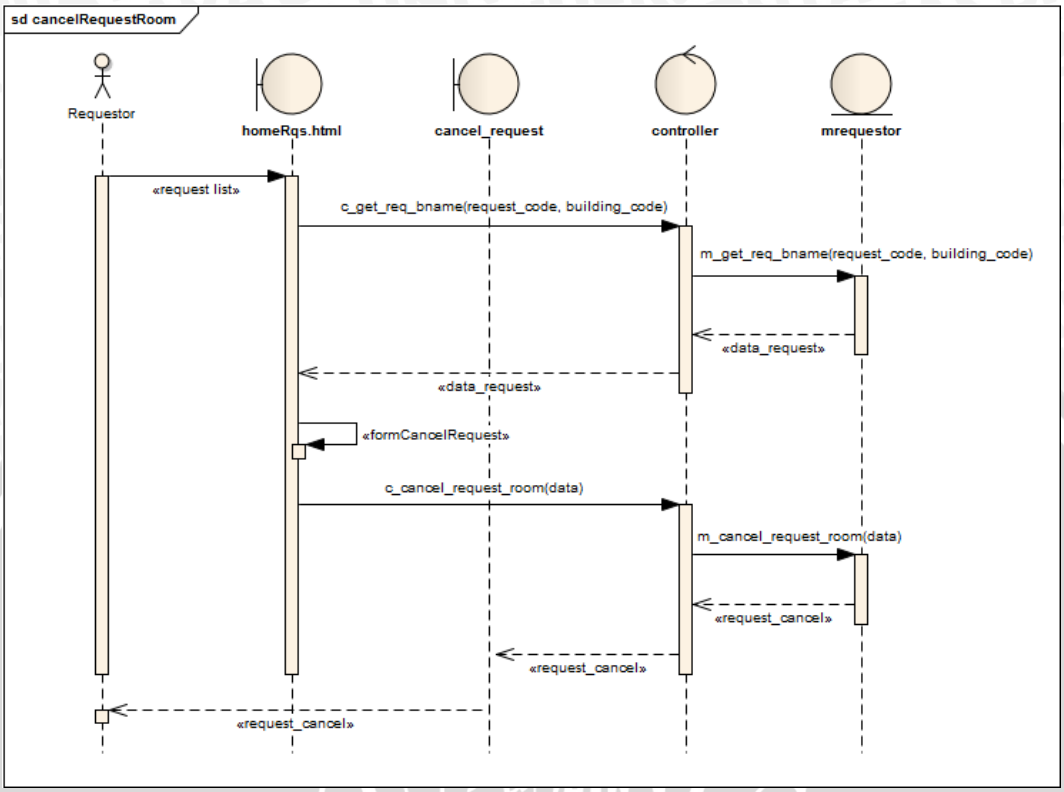
2. Sequence diagram mengubah informasi permintaan rapat



Gambar 4.6 Sequence Diagram Merubah Informasi Rapat

Pada diagram sequence mengubah informasi permintaan rapat menggambarkan salah satu aktor yaitu requestor dapat melakukan perubahan informasi permintaan rapat yang dibuat, namun dengan kondisi dimana permintaan rapat belum diterima oleh admin. Pada diagram tersebut terdapat satu buah aktor, controller, model, dan dua buah view. Adapun runtutuan aktifitas dari diagram tersebut meliputi requestor melalui homeRqs.html mengirimkan method pada controller untuk menampilkan seluruh permintaan rapat berdasarkan dengan data yang tersedia termasuk dengan nama requestor atau nama pemesan. Setelah data permintaan rapat didapatkan proses berlanjut dengan menggunakan view edit_request yang menampilkan form perubahan permintaan rapat yan telah dibuat. Requestor dapat mengisikan perubahan informasi sesuai dengan field yang tersedia. Hasil akhir informasi perubahan permintaan rapat dapat disimpan kedalam database sistem.

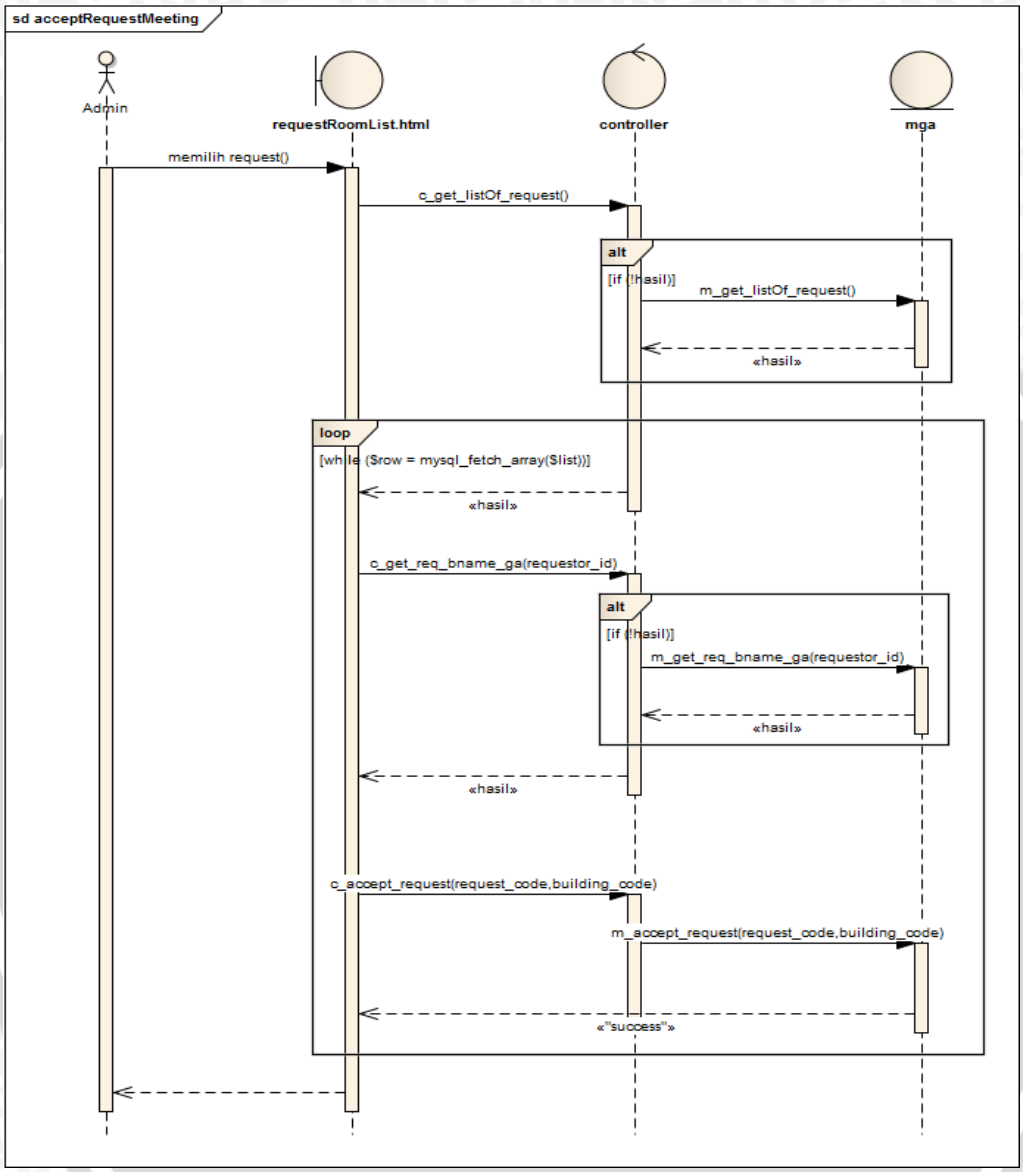
3. Sequence diagram membatalkan permintaan rapat



Gambar 4.7 Sequence Diagram Membatalkan Permintaan Rapat

Pada diagram sequence membatalkan permintaan rapat menggambarkan salah satu aktor yaitu requestor dapat melakukan pembatalan permnitaan rapat berdasarkan permintaan yang telah diterima oleh admin maupun belum diterima oleh admin. Pada diagram tersebut terdapat satu buah aktor, controller, model, dan dua buah view. Adapun runtutuan aktifitas dari diagram tersebut meliputi requestor menuju halaman homeRqs.html untuk melihat *request* yang ingin dilakukan pembatalan. Masuk kepada controller dengan mengirimkan method c_get_req_bname hingga kepada model mrequestor untuk diproses dan dikembalikan dengan data berupa *request* keseluruhan dari database sistem. Kemudian pada halaman view homeRqs.html muncul form berisi informasi utama terkait dengan *request* yang telah dibuat. Selanjutnya halaman view mengirimkan method c_cancel_request_room untuk proses pembatalannya hingga menuju model mrequestor. Hasil akhir dari proses tersebut *request* yang dibatalkan akan muncul pada halaman cancel_room dan menunggu konfirmasi pihak admin.

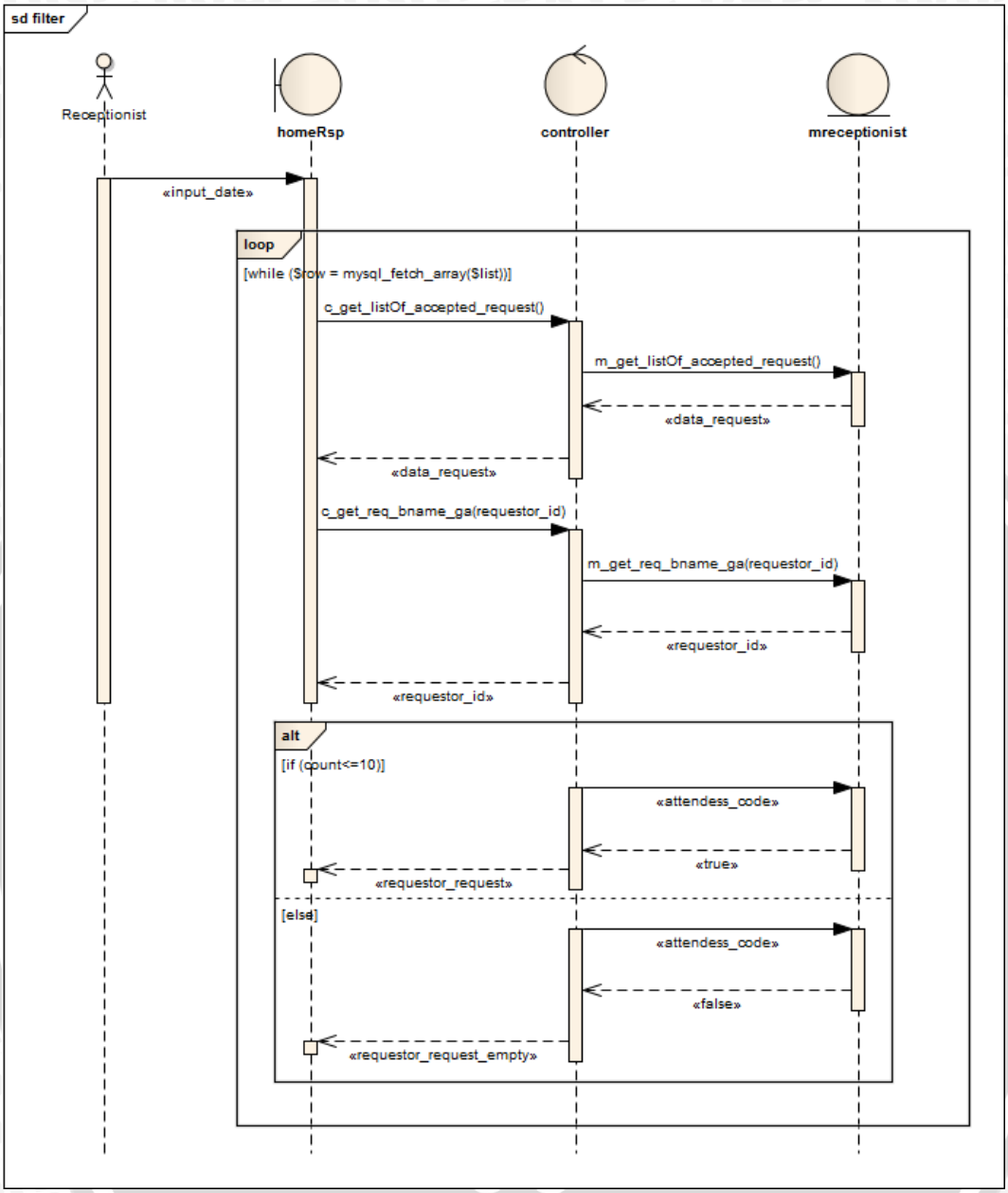
4. Sequence diagram menerima permintaan rapat



Gambar 4.8 Sequence Diagram Menerima Permintaan Rapat

Pada diagram sequence menerima permintaan rapat menggambarkan salah satu aktor yaitu admin dapat melakukan menerima permnitaan rapat berdasarkan permintaan yang telah masuk pada daftar permintaan rapat. Pada diagram tersebut terdapat satu buah aktor, controller, model, dan view. Adapun runtutuan aktifitas dari diagram tersebut meliputi admin menuju halaman requestRoomList.html untuk melihat seluruh daftar dari *request* yang masuk. Kemudian pada controller dikirmkan method hingga model berupa c_accept_request untuk dilakukan *approve* terhadap *request* yang ada. Hasil akhir dari aktifitas tersebut tampil *request* baru pada halaman home admin dan pada home requestor akan muncul status bahwa *request* telah diterima oleh admin.

5. Sequence diagram filter

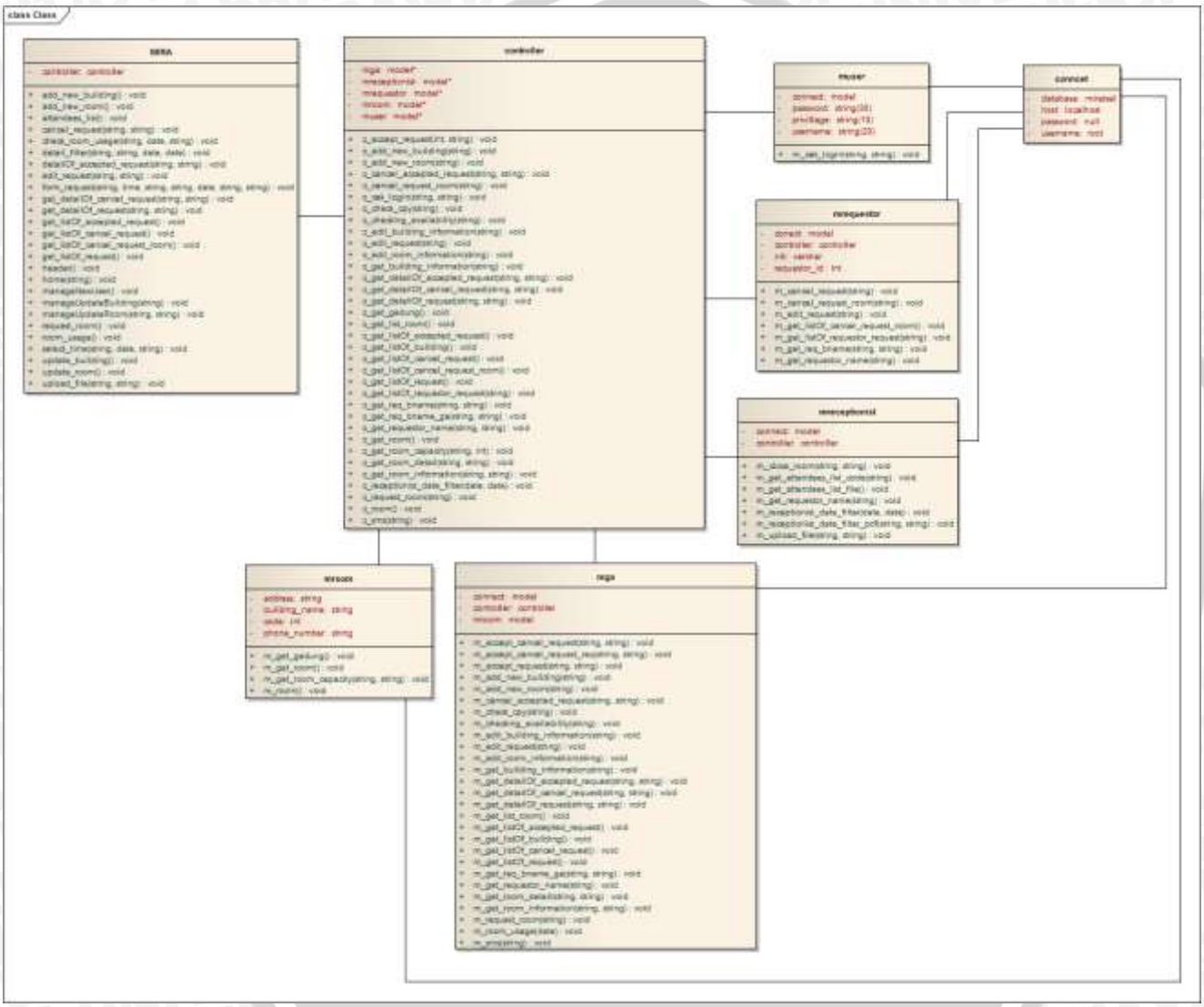


Gambar 4.9 Sequnce Diagram Filter

Pada diagram sequence filter menggambarkan salah satu aktor yaitu receptionist dapat melakukan filtering untuk mendapatkan daftar permintaan rapat berdasarkan inputan tanggal yang tersedia. Pada diagram tersebut terdapat satu buah aktor, view, controller, dan model. Adapun alur dari diagram tersebut meliputi receptionist terhubung dengan view homers.html untuk melihat daftar permintaan rapat yang telah diterima oleh admin. Selanjutnya pada view mengirimkan method kepada controller dan diteruskan menuju model untuk mendapatkan seluruh data pemesanan ruang *meeting* melalui database sistem. Kemudian dilakukan proses inputan tanggal untuk mendapatkan hasil beruapa daftar permintaan rapat yang sesuai dengan inputan tersebut.

4.3.3 Perancangan Class Diagram

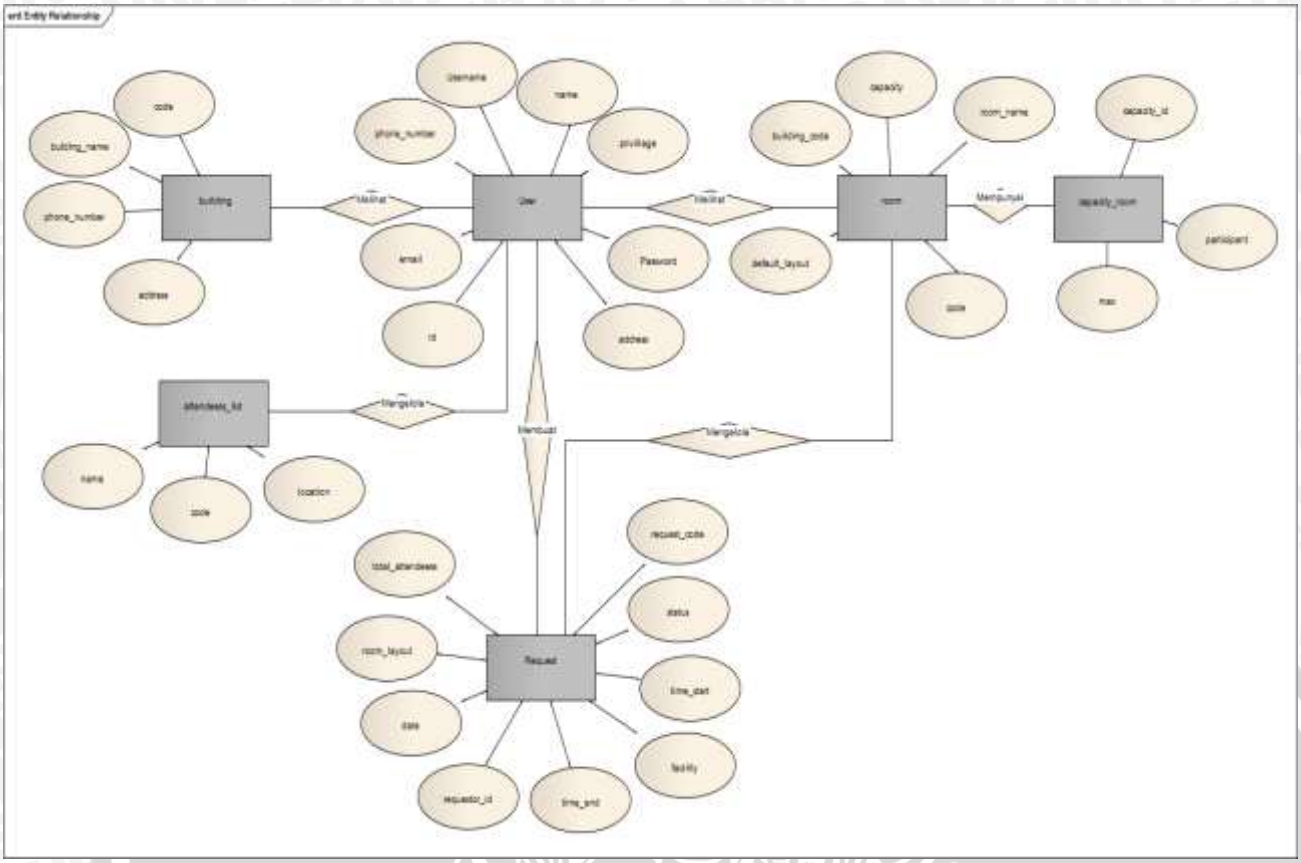
Pada perancangan *class* diagram dilakukan berdasarkan kebutuhan yang telah didefinisikan oleh pihak *stakeholder.* Kebutuhan tersebut nantinya akan dibagi kedalam tiga tempat yang berfungsi untuk membagi kondisi antara sistem dengan *stakeholder*. Tiga pembagian tersebut adalah *view, controller,* dan *model.* Pada gambar 4.10 merepresentasikan *class diagram* berdasrkan dengan kebutuhan yang teah di definisikan.



Gambar 4.10 Class Diagram

4.3.4 Perancangan Basis Data

Tahap perancangan basis data ini dilakukan dengan membangun sebuah diagram ERD (*Entity Relational Diagram*). Didalam diagram tersebut dapat menjelaskan secara detail tentang kebutuhan dari pada sistem yang akan dikembangkan.

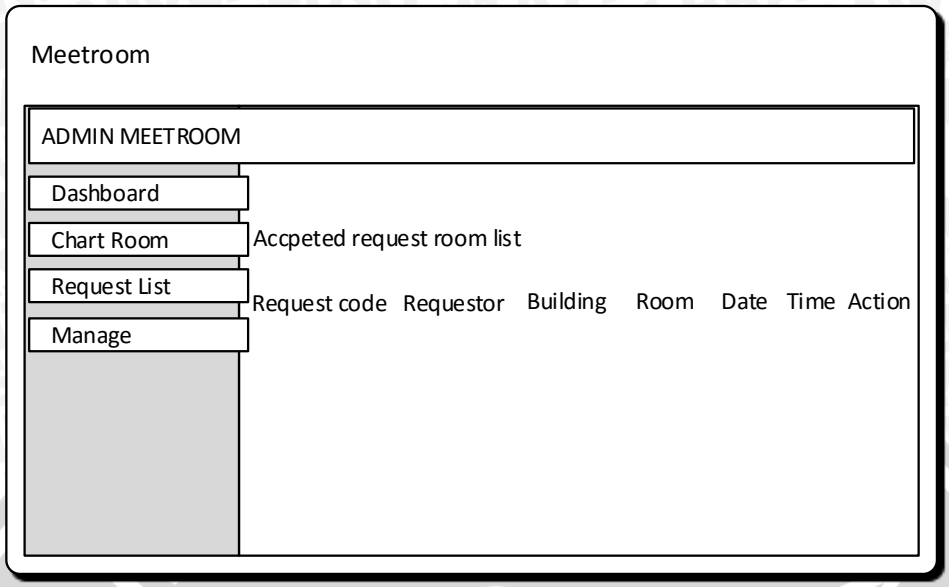


Gambar 4.11 Diagram Basis Data

4.3.5 Perancangan Antarmuka

Tahapan terkahir yaitu merancang antar muka pengguna yang akan di terapkan pada aplikasi manajemen ruang *meeting.* Penggunaan tools online seperti css bootstarp diperlukan untuk mempercantik hasil dari tampilan antar muka. Berikut gambaran umum perancangan antarmuka dari aplikasi manajemen ruang *meeting*.

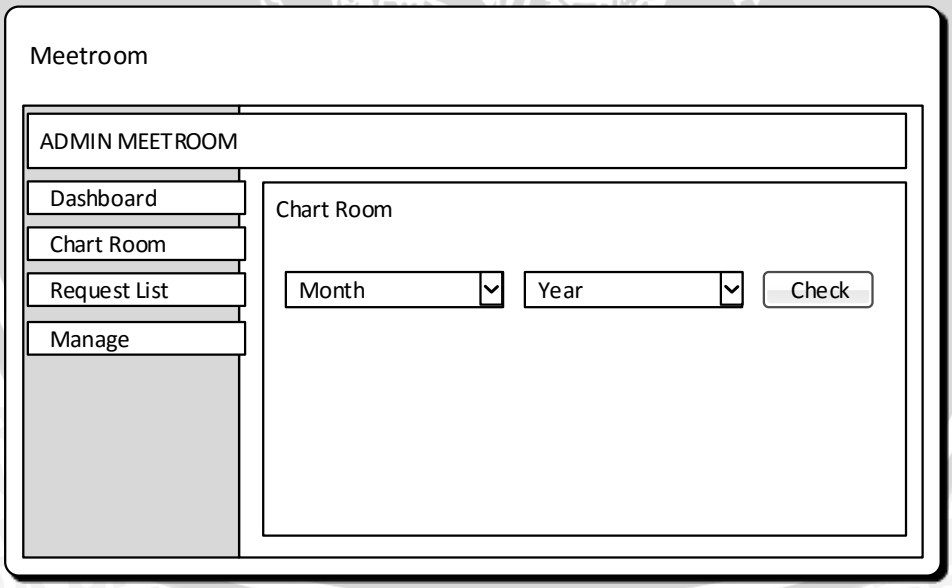
4.3.5.1 Halaman Utama admin



Gambar 4.12 Rancangan Halaman Utama Admin

Pada gambar 4.12 Halaman utama admin menampilkan daftar *request* yang telah diterima oleh admin dan telah dapat dilaksanakan. Halaman tersebut beruapa table yang berisi urutan dari *request* yang telah diterima oleh admin.

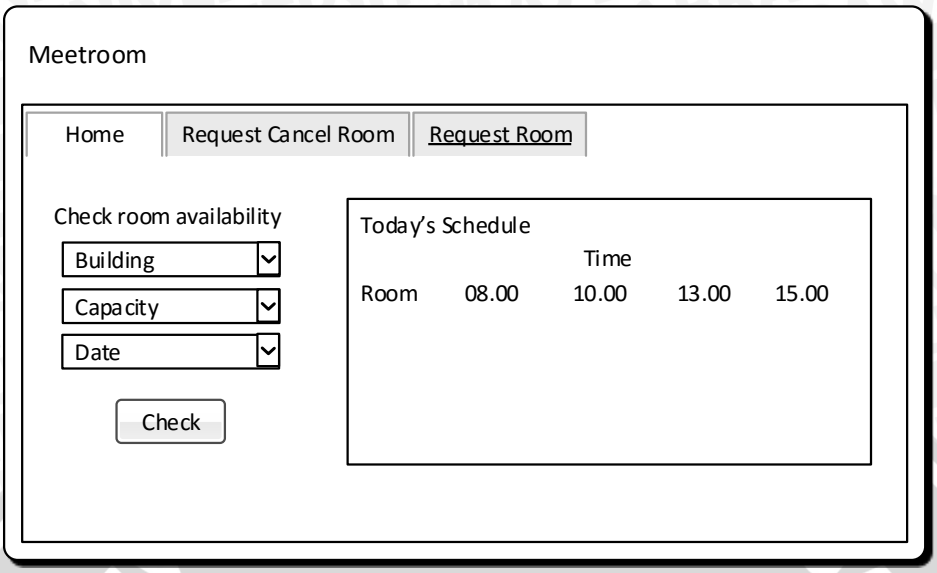
4.3.5.2 Halaman *Chart Room* Admin



Gambar 4.13 Rancangan Halaman *Chart Room* Admin

Pada gambar 4.13 Halaman ini bertujuan untuk menampilkan penggunaan ruangan per bulan pada tahun tertentu. Proses dalam menampilkan penggunaan ruangan tersebut dengan memanfaatkan library yang ada dengan beberapa perubahan untuk dapat memenuhi kriteria dari ruangan yang tersedia didalam sistem. Hasil yang muncul pada halaman tersebut adalah diagram batang dengan beberapa keterangan yang terkait pada ruangan tersebut.

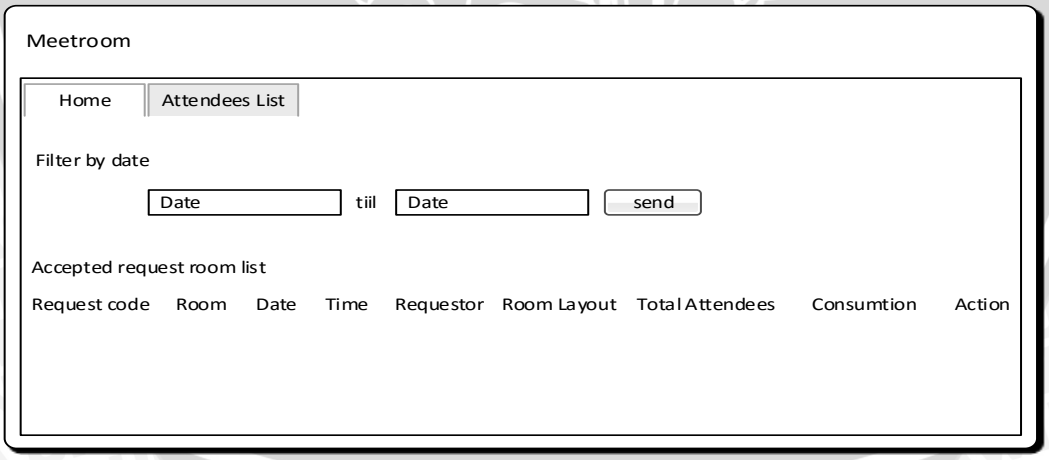
4.3.5.3 Halaman *Request Room*



Gambar 4.14 Rancangan Halaman *Request Room*

Halaman *request room* ini merupakan tampilan untuk melakukan pemesanan ruang *meeting*. Pada kolom kiri terdapat kolom *check room availability* yang merupakan kolom dengan field yang merupakan inputan untuk melakukan *checking* ruangan kosong. Kemudian pada kolom kanan bertujuan untuk menampilkan hasil dari proses *checking* ruangan baik yang kosong maupun yang telah dipesan.

4.3.5.4 Halaman Filter *Attendees List*



Gambar 4.15 Rancangan Halaman Filter *Attendees List*

Halaman filter diatas bertujuan untuk memudahkan receptionist untuk melakukan pengelompokan data keperluan rapat yang akan dijalankan. Proses didalamnya beruapa pengambilan data berupa tanggal awal dan tanggal akhir yang akan dicocokkan untuk dilihat kesamaan dalam hal tanggal pada masing – masing *request* yang ada.

BAB 5 IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan tentang implementasi dan pengujian dari hasil analisa dan perancangan yang telah diperoleh dari bab sebelumnya. Implementasi terdiri dari penjelasan terkait dengan spesifikasi sistem serta batasan implementasi, implementasi program dan implementasi antarmuka. Sedangkan pada pengujian dibagi menjadi dua proses uji meliputi pengujian fungsional dan pengujian performa. Pada pengujian fungsional, diawali dari pengujian *code* program yang akan dilakukan pada pengujian unit, selanjutnya dilakukan pengujian terdapat arsitektur atau desain yang akan dijabarkan pada pengujian integrasi, kemudian dilanjutkan dengan pengujian sistem dari sisi fungsionalitas yang akan dilakaukan pada pengujian validasi. Sedangkan pada pengujian performa dilakukan dengan memanfaatkan *tools* Yslow yang merupakan pegujian dari sisi performa sebuah aplikasi *web*.

5.1 Implementasi

Pada sub bab ini membahas tentang implementasi berdasarkan hasil analisis kebutuhan dan proses perancangan sistem dari aplikasi manajemen ruang *meeting*. Pembahasan pada sub bab ini terdiri dari penjelasan tentang spesifikasi sistem, batasan implementasi, implementasi program, dan implementasi antarmuka.

5.1.1 Spesifikasi Sistem

Berdasarkan hasil dari analisis kebutuhan dan perancangan sistem yang telah diajabarkan menjadi pedoman untuk melakukan implementasi sistem yang sesuai dengan yang diharapkan serta akan berfungsi sesuai dengan kebutuhan pengguna. Spesifikasi sistem terdiri dari spesifikasi perangkat keras dan perangkat lunak sebagai berikut

5.1.1.1 Spesifikasi Perangkat Keras

Pembangunan aplikasi manajemen ruang *meeting* menggunakan spesifikasi perangkat keras dapat dilihat pada tabel 5.1 berikut

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	Intel(R) Core(TM) i5-3317U CPU @ 1.70GHz (4 CPUs), ~1.7GHz
Memori (RAM)	2048 MB
Harddisk	1 Tb
Kartu Grafis	Intel(R) HD Graphics 4000
Monitor	14.0”, Resolusi 1366 x 768

5.1.1.2 Spesifikasi Perangkat Lunak

Pembangunan aplikasi manajemen ruang *meeting* menggunakan spesifikasi perangkat lunak dapat dlilihat pada table 5.2 berikut

Tabel 5.2 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
Operating System	Windows 10 Pro 64-bit (10.0, Build 10240) (10240.th1.160104-1507)
Browser	Google Chrome versi 51.0.2704.84
Code Editor	Sublime Text build 3103
Image Editor	Corel Draw X7
Database	MySQL 5.6
Pemodelan, UML	Power Designer 15, Enterprise Architect 8.0
Framework	Bootstrap 3
Documentation	MS Word 2016

5.1.2 Batasan Implementasi

Beberapa batasan implementasi aplikasi manajemen ruang *meeting* dengan memanfaatkan pola MVC (Model View Controller) diantaranya:

1. Aplikasi manajemen ruang *meeting* dirancang dan dibangun menggunakan bahasa pemrograman PHP, *framework* Bootsrap untuk pembuatan website, serta menggunakan MySQL sebagai pendukung basis data.
2. Aplikasi manajemen ruang *meeting* memiliki fitur utama berupa kelola ruang *meeting* (*Booking*), pembatalan *request* ruang *meeting*, penyebaran informasi *request* dengan memanfaatkan SMS *Getaway*, rekapan penggunaan ruang *meeting* berdasarkan bulan pada tahun tertentu, dan *generate* PDF untuk proses absensi peserta *meeting*.
3. Data yang digunakan dalam aplikasi manajemen ruang *meeting* merujuk pada perusahaan telekomunikasi PT. Telkomsesl Indonesia Tbk. Selain itu data juga diambil melalui beberapa orang yang memiliki kegiatan dengan mobilitas tinggi dalam hal *meeting.*

5.1.3 Implementasi Program

Proses implementasi aplikasi manajemen ruang *meeting* dilakukan dengan menggunakan bahasa pemrograman PHP dengan menggunakan konsep MVC dan *framework* bootstrap sebagai template CSS serta javascript. Pada perancangan sebelumnya telah didefinisikan kelas-kelas yang masuk dalam pola perancangan MVC. Komponen yang terlibat dalam MVC adalah controller sebagai pusat dari beberapa model yang dapat mengakses method didalamnya. Model yang berinteraksi dengan controller mengalirkan data secara berkesinambungan untuk menghasilkan fungsi yang dapat berjalan sesuai dengan kebutuhan aplikasi. Berikut implementasi program dari masing-masing komponen MVC berdasarkan dengan beberapa fitur yang terkait didalamnya.

5.1.3.1 Implementasi Program *Checking Free Room*

Implementasi pada proses *cheking free room* menggambarkan alur bagaimana requestor dapat melakukan cek terhadap ruangan mana yang dapat dipesan dengan memberikan status dari masing – masing ruangan berdasarkan dengan jam yang telah ditentukan. Implementasi program ini merupakan bagian kecil dari implementasi proses pemesanan ruang *meeting*. Komponen berupa kelas yang terlihat pada kutipan program berikut yaitu kelas controller, kelas main, dan kelas mga yang merupakan kelas model untuk admin. Implementasi dari porses *checking free room* dapat dilihat pada table 5.3 berikut.

Kelas controller

Tabel 5.3 Implementasi *Check Free Room* Controller

```
1.  public function c_request_room($data){
2.        $mga = new mga();
3.        $hasil = $mga->m_request_room($data);
4.        return $hasil;
5.  }
```

Membuat method c_request_room dengan variable $data yang memanggil kelas model dari admin pada model mga (baris 2) dengan method m_request_room dengan variable $data. Fungsi ini akan mengambalikan nilai pada variable $hasil (baris 3) untuk kemudian dapat diakses oleh sebuah fungsi pada kelas main.

Kelas Main

Tabel 5.4 Implementasi *Check Free Room* Main

```
1.  public function request_room(){
2.  ?>
3.  <div id="konten">
4.  <?php
5.     $controller = new controller();
6.     $gedung = $controller->c_get_gedung();
7.     $capacity = $controller->c_get_room();
8.        echo "
9.        <div class=\"page-wrapper\">
10.         <div class=\"row\">
11.           <div class=\"col-lg-12 text-center\">
12.           <h2>Request for Room</h2><br />
13.           </div>
14.         </div>
15.         <div class=\"row\">
16.           <div class=\"col-lg-4\">
17.             <div class=\"panel panel-default\">
18.               <div class=\"panel-heading\">
19.               <h8>Checking Room Availability</h8>
20.               </div>
21.              <div class=\"panel-body\">
22.              <center>
23.       <form id=\"check_form\" class=\"form-inline\"
24.  method=\"post\" action=\"\">
25.       <select required class=\"form-control\"
```

```
name=\"building_code\"
id=\"building_code\" style=\"width:60%\">
            <option value=\"\">Building</option>;”
            while($row = mysql_fetch_assoc($gedung)){
            echo
"<option
value=\"".$row['code']."\">".$row['building_name']."
</option>\n";                }
            echo
"</select><br /><br />\n
      <select class=\"form-control\" name=\"capacity\"
 id=\"capacity\"
style=\"width:60%\" placeholder=\"Capacity\">
      <option value=\"\">Participant....</option>";
            while ($row = mysql_fetch_assoc($capacity)) {
            echo
"<option
value=\"".$row['capacity_id']."\">".$row['participant'].
"</option>\n";              }
            echo        "</select><br /><br />
                        <input       class=\"form-control\"
type=\"text\" name=\"tanggal\"
id=\"datepicker\" style=\"width:60%\" required/><br /><br />
            <input type=\"submit\"
id=\"checking_availability\"
name=\"checking_availability\" class=\"btn
btn-success\" value=\"Check\"/>
      </form>
      \n<br /><br />
      </center>
      </div>
      </div></div>
      <div class=\"col-lg-8\">
         <h4>Today's Schedule</h4><br />
      <center>
      <table  id=\"today_schedule\"  class=\"table  table-
striped   table-bordered   table-hover\"   style=\"table-
layout:fixed\">
      <tr class=\"text-center\">
                  <td rowspan=\"2\">Time /<br />Room</td>
      <td colspan=\"4\">Time</td>
      </tr>
      <tr class=\"text-center\">
      <td>08.00-10.00</td>
      <td>10.00-12.00</td>
      <td>13.00-15.00</td>
      <td>15.00-17.00</td>
      </tr>
      <tr class=\"text-center\">
      <td class=\"room_code\">Room</td>
      <td><input type=\"button\" id=\"bk\" name=\"booking\"
value=\"no   action\"   class=\"btn   btn-danger   btn-xs
disabled\"/></td>
      <td><input type=\"button\" id=\"bk\" name=\"booking\"
value=\"no   action\"   class=\"btn   btn-danger   btn-xs
disabled\"/></td>
<td><input   type=\"button\"   id=\"bk\"   name=\"booking\"
value=\"no   action\"   class=\"btn   btn-danger   btn-xs
disabled\"/></td>
```

```
85.         <td><input type=\"button\" id=\"bk\" name=\"booking\"
86. value=\"no    action\"    class=\"btn    btn-danger    btn-xs
87. disabled\"/></td>
88.         </tr>
89.         </table>
90.         </center>
91.         </div></div>";
92. }
93.
94.
95.
```

Dalam implementasi algoritma *cheking free room* diimplementasikan dengan fungsi public yang memanggil method untuk memanggil data berupa gedung dengan proses perulangan (baris 29) dan berupa kapasitas (baris 40) sesuai dengan yang ditentukan serta tanggal yang menentukan seluruh inputan untuk mencari ruangan kosong yang dapat dipesan.

Kelas mga

Tabel 5.5 Implementasi *Check Free Room* mga

```
1.  public function m_request_room($data){
2.    date_default_timezone_set('Asia/Jakarta');
3.    $request_code = $data['request_code'];
4.    $time_start = $data['time_start'];
5.    $building_code = $data['building_code'];
6.    $requestor_id = $data['id'];
7.    $room = $data['room'];
8.    $judul = ucwords($data['judul']);
9.    $room_layout = $data['room_layout'];
20.   $attandees = $data['attandees'];
11.   $facility = $data['facility'];
12.   $snack = $data['snack'];
13.   $date = $data['date'];
14.   $status = 0;
15.   if($time_start=="08.00"){
16.       $time_end = "10.00";
17.   }
18.   else if($time_start=="10.00"){
19.       $time_end = "12.00";
20.   }
21.   else if($time_start=="13.00"){
22.       $time_end = "15.00";
23.   }
24.   else{
25.       $time_end = "17.00";
26.   }
27.   echo            "<h4>".$snack.",           ".$request_code.",
28.   ".$building_code.", ".$date."
29.   , ".$room."</h4>";
30.       $query =
31. "UPDATE request
32.  SET requestor_id='$requestor_id',judul_meeting='$judul',
33. time_end='$time_end',room_layout='$room_layout',
34. total_attendees='$attandees',facility='$facility',
35. snack_description='$snack',status='$status'
36. WHERE
37. (((request_code='$request_code' AND
```

```
38.   building_code='$building_code') AND
39.  tanggal='$date') AND Substring(request_code,1,3)='$room')";
40.     $hasil = mysql_query($query);
41.         if(!$hasil){
42.         echo mysql_error();
43.         }
44.         else{
45.         return "Berhasil";
46.         }
47.  }
```

Pada proses pemesanan ruang *meeting* model yang dibutuhkan adalah model milik admin dengan nama kelas mga. Fungsi yang dipanggil untuk mendukung proses pemesanan ruang *meeting* adalah m_request_room dengan variable $data.

5.1.3.2 Implementasi Program *Select Time*

Pada konsep MVC yang diterpakan pada aplikasi manajemen ruang *meeting* menggabungkan seluruh fungsi controller pada satu kelas yang diberi nama controller. Kutipan *code* program yang diambil yaitu pemesanan ruang *meeting* (*Booking*). *Booking* merupakan kegiatan dimana user yang diberi hak akses sebagai requestor (pemesan) melakukan pemesanan ruang *meeting* sesuai dengan kebutuhan dan kegiatan yang *meeting* yang diinginkan. Sehingga sistem requestor dapat melakukan pemesanan ruangan sesuai dengan inputan yang telah disedikan oleh aplikasi manajemenr ruang *meeting.* Komponen berupa kelas yang terlihat pada kutipan program berikut yaitu kelas controller, kelas main, dan kelas mga yang merupakan kelas model untuk admin. Implementasi dari porses *booking* dapat dilihat pada table 5.6 berikut.

Kelas controller

Tabel 5.6 Implementasi *Select Time* Controller

```
1.  public function c_request_room($data){
2.        $mga = new mga();
3.        $hasil = $mga->m_request_room($data);
4.        return $hasil;
5.  }
```

Pada kelas controller terbentuk method c_request_room dengan variable $data. Fungsi ini melibatkan model dari requestor dengan nama kelas mrequestor. Method m_ requet_room dengan variable $data merepresentasikan fungsi utama dari kelas controller pada proses pembatalan *request.*

Kelas main

Tabel 5.7 Implementasi *Select Time* Main

```
1.  public function select_time($building_code,$date,$kapasitas){
2.     ?> <div id="konten"><?php
3.     $controller = new controller();
4.     $capacity = $controller->c_get_room();
5.     $gedung = $controller->c_get_gedung();
6.     $room_capacity =
7.     $controller->c_get_room_capacity($building_code
8.     ,$kapasitas);
```

```
9.    $room = $controller->c_room();
10.   $time = " ";$count = 4;$j=0;
11.      echo "
12.   <div class=\"page-wrapper\">
13.   <div class=\"row\">
14.   <div class=\"col-lg-12 text-center\">
15.   <h2>Request for Room</h2><br />
16.   </div>
17.   </div>
18.   <div class=\"row\">
19.   <div class=\"col-lg-4\">
20.   <div class=\"panel panel-default\">
21.   <div class=\"panel-heading\">
22.   <h8>Checking Room Availability</h8>
23.   </div>
24.   <div class=\"panel-body\">
25.   <center>
26.   <form id=\"check_form\"
27.   class=\"form-inline\" method=\"post\"
28.   action=\"\">
29.   <select required
30.   class=\"form-control\" name=\"building_code\"
31.   id=\"building_code\" style=\"width:60%\">
32.   <option value=\"\">Building</option>
33.      ";while($row = mysql_fetch_assoc($gedung)){
34.      echo"<option
35. value=\"".$row['code']."\">".$row['building_name'].
36.   "</option>\n";
37.      }
38.      echo"</select><br /><br />\n
39.   <select class=\"form-control\"
40.   name=\"capacity\" id=\"capacity\"
41.   style=\"width:60%\" placeholder=\"Capacity\">
42.   <option value=\"\">Participant....</option>";
43.      while ($row = mysql_fetch_assoc($capacity)) {
44.      echo "<option value=\"".$row['capacity_id']."\">".
45.   $row['participant']."</option>\n";
46.      }
47.      echo "</select><br /><br />
48.   <input class=\"form-control\"
49.   type=\"text\" name=\"tanggal\"
50.   id=\"datepicker\" style=\"width:60%\"required/>
51.   <br /><br />\n
52.   <input type=\"submit\" id=\"checking_availability\"
53.   name=\"checking_availability\" class=\"btn btn-success\"
54.   value=\"Check\"/>
55.   </form>
56.   </center>
57.   </div>
58.   </div>
59.   </div>
60.   <div class=\"col-lg-8\">
61.   <h4>Today's Schedule</h4><br />
62.   <center>
63.   <div id=\"lock\"></div>
64.   <form method=\"post\" action=\"\">
65.   <input type=\"hidden\"
66.   id=\"buil\" value=\"".$building_code."\"/>
67.   <input type=\"hidden\"
```

```
68.    id=\"cpy\" value=\"".$capacity."\"/>
69.    <input type=\"hidden\" id=\"dt\" value=\"".$date."\"/>
70.    </form>
71.    <table id=\"today_schedule\" class=\"table table-striped
72.    table-bordered table-hover\">
73.    <tr class=\"text-center\">
74.    <td rowspan=\"2\">Time /<br />Room</td>
75.    <td colspan=\"4\">Time</td>
76.    </tr>
77.    <tr class=\"text-center\">
78.    <td>08.00-10.00</td>
79.    <td>10.00-12.00</td>
80.    <td>13.00-15.00</td>
81.    <td>15.00-17.00</td>
82.    </tr>";
83.       while                    ($row_capacity                    =
84. mysql_fetch_assoc($room_capacity)) {
85.       echo"<tr class=\"text-center\">";
86.       echo"<td
87. class=\"room_code\">".$row_capacity['room_name']."</td>";
88.       for($i=0; $i<4; ){
89.             if($i==0){
90.             $book = "08.00"
91.             elseif ($i==1) {
92.             $book = "10.00";
93.             } elseif ($i==2) {
94.             $book = "13.00
95.             } elseif ($i==3) {
96.             $book = "16.00";
97.
98.             }
99.       echo "<td>";
100 $array = array('room_code'=>$row_capacity['code'],
101 'building_code'=>$building_code,'time_start'=>$book,'date'=>$
102 date);
103 $hasil = $controller->c_checking_availability($array);
104 $jumlah = mysql_num_rows($hasil);
105    /*
106    $check = "";
107    if($jumlah > 0) $check = "checked";
108    echo
109    "<input type='checkbox' name='".$row_capacity['code'].
110    "' $check>";
111     */
112    if($jumlah == 0){
123    $button = "success";
124    $status = "booking";
125    $link =
126    "?booking=".$book."&room=".$row_capacity['code']."
127    &buil=".$building_code."&capacity=".$kapasitas."
128    &date=".$date."
129    &cpy=".$kapasitas."&dl=".$room['default_layout'];
130
131    }
132    else{
133    $button = "warning";
134    $status = "booked";
135    $link = "#";
136    }
```

```
137     echo "<a href=".$link."><button value='".$book."'
138     class=\"btn btn-".$button." btn-xs\"
139     style=\"width:90px\">".$status."</button></a></td>";
140     }
141     echo "</tr>";
142     }
143     echo      "</table>
144     </center>
145     </div>
146     </div>";
147 }
```

Dalam implementasi algoritma pemesanan ruang *meeting* diperlukan dua buah fungsi *public* yang masaing-masing melakukan tahapan pemesanan secara runut namun tetap berdasrkan pada pemanggilan kelas controller dan model sesuai dengan kebutuhan dari suatu fungsi. Fungsi yang terlibat dalam proses pemesanan ruang *meeting* antara lain, request_room dan select_time dengan variable $building_code, $date, dan $kapasitas.

Kelas mga

Tabel 5.8 Implementasi *Select Time* mga

```
1.  public function m_request_room($data){
2.    date_default_timezone_set('Asia/Jakarta');
3.    $request_code = $data['request_code'];
4.    $time_start = $data['time_start'];
5.    $building_code = $data['building_code'];
6.    $requestor_id = $data['id'];
7.    $room = $data['room'];
8.    $judul = ucwords($data['judul']);
9.    $room_layout = $data['room_layout'];
20.   $attandees = $data['attandees'];
11.   $facility = $data['facility'];
12.   $snack = $data['snack'];
13.   $date = $data['date'];
14.   $status = 0;
15.   if($time_start=="08.00"){
16.       $time_end = "10.00";
17.   }
18.   else if($time_start=="10.00"){
19.       $time_end = "12.00";
20.   }
21.   else if($time_start=="13.00"){
22.       $time_end = "15.00";
23.   }
24.   else{
25.       $time_end = "17.00";
26.   }
27.   echo          "<h4>".$snack.",          ".$request_code.",
28.   ".$building_code.", ".$date."
29.   , ".$room."</h4>";
30.       $query =
31. "UPDATE request
32.  SET requestor_id='$requestor_id',judul_meeting='$judul',
33. time_end='$time_end',room_layout='$room_layout',
34. total_attendees='$attandees',facility='$facility',
35. snack_description='$snack',status='$status'
36. WHERE
```

```
37.  (((request_code='$request_code' AND
38.   building_code='$building_code') AND
39.  tanggal='$date') AND Substring(request_code,1,3)='$room')";
40.     $hasil = mysql_query($query);
41.         if(!$hasil){
42.         echo mysql_error();
43.         }
44.         else{
45.         return "Berhasil";
46.         }
47.  }
```

Pada proses pemesanan ruang *meeting* model yang dibutuhkan adalah model milik admin dengan nama kelas mga. Fungsi yang dipanggil untuk mendukung proses pemesanan ruang *meeting* adalah m_request_room dengan variable $data.

5.1.3.3 Implementasi Program *Edit Request*

Implemetnasi berikutnya adalah terkait dengan aktor requestor yang melakukan pembaruan informasi permintaan rapat yang telah dibuat sebelumnya. *Edit request* dibentuk dengan melibatkan tiga buah kelas meliputi, kelas controller, kelas main, dan kelas mga yang merupakan kelas dari aktor admin. Implementasi program pada kegiatan *edit request* dijelaskan pada gambar … berikut.

Kelas controller

Tabel 5.9 Implementasi *Edit Request* Controller

```
1.  public function c_edit_request($data){
2.        if($_SESSION['privillage']=="administrator"){
3.        $mga = new mga();
4.        $hasil = $mga->m_edit_request($data);
5.        return $hasil;
6.        }else{
7.        $mrequestor = new mrequestor();
8.        $hasil = $mrequestor->m_edit_request($data);
9.        return $hasil;
10.       }
11. }
```

Kelas controller untuk melakukan kegiatan *edit request* diperlukan fungsi c_edit_request dengan variable $data. Fungsi tersebut memanggil method m_edit_request dengan variable $data yang merupakan representasi dari fungsi tersebut.

Kelas main

Tabel 5.10 Implementasi *Edit Request* Main

```
1.  public function edit_request($request_code,$building_code){
2.  ?>
3.  <div id="konten">
4.  <?php
5.        $controller = new controller();
6.        if($_SESSION['privillage']=="requestor"){
7.        $data =
8.  $controller->c_get_req_bname($request_code,$building_code);
9.        echo"<center>
```

```
10.       <h4>Specify your request</h4><br /><br />
11.       <div id=\"lock\"></div>
12.       <form method=\"post\" action=\"?save_edit_request\">
13.       <input type=\"hidden\" id=\"req_code\"
14. name=\"request_code\" value=\"".$request_code."\"/>
15.       <input type=\"hidden\" id=\"bl\"
16. name=\"building_code\" value=\"".$building_code."\"/>
17.       <input type=\"text\" class=\"form-control\"
18. name=\"requestor\" value=\"".$_SESSION['username']."\"
19. style=\"width:20%\" disabled/><br />
20.       <input type=\"text\" class=\"form-control\"
21. name=\"judul\" placeholder=\"judul meeting\"
22. style=\"width:20%\"
23. value=\"".$data['judul_meeting']."\"/><br />
24.       <select class=\"form-control\" name=\"room_layout\"
25. style=\"width:20%\" required>";
26.       if($data['room_layout'] == "O Shape"){
27.       echo "
28.       <option value=\"O Shape\" selected>O Shape</option>
29.       <option value=\"U Shape\">U Shape</option>
30.       <option value=\"Class Room Shape\">Class Room
31. Shape</option>";
32.       }
33.       else if($data['room_layout']  == "U Shape"){
34.       echo "
35.       <option value=\"O Shape\">O Shape</option>
36.       <option value=\"U Shape\" selected>U Shape</option>
37.       <option value=\"Class Room Shape\">Class Room
38. Shape</option>";
39.       }
40.       else{
41.       echo "
42.       <option value=\"O Shape\">O Shape</option>
43.       <option value=\"U Shape\">U Shape</option>
44.       <option value=\"Class Room Shape\" selected>Class
45. Room Shape</option>";
46.       }
47.       echo"</select><br />
48.       <input type=\"text\" id=\"attendees\" class=\"form-
49. control\" name=\"attendees\" style=\"width:20%\"
50. value=\"".$data['attendees']."\"/><br />
51.       <select class=\"form-control\" name=\"facility\"
52. style=\"width:20%\" required>";
53.       if($data['facility']=="Internet"){
54.       echo "
55.       <option value=\"Internet\" selected>Internet</option>
56.       <option value=\"Intranet\">Intranet</option>
57.       <option value=\"Internet dan Intranet\">Internet &
58. Intranet</option>";
59.       }
60.       else if($data['facility']=="Intranet"){
61.       echo "
62.       <option value=\"Internet\">Internet</option>
63.       <option value=\"Intranet\" selected>Intranet</option>
64.       <option value=\"Internet dan Intranet\">Internet &
65. Intranet</option>";
66.       }
67.       else{
68.       echo "
```

```
69.         <option value=\"Internet\">Internet</option>
70.         <option value=\"Intranet\">Intranet</option>
71.         <option value=\"Internet dan Intranet\"
72. selected>Internet & Intranet</option>";
73.         }
74.         echo "</select><br /><br />
75.         <table>
76.         <tr>
77.         <td colspan=\"3\"><label for=\"snack\">Snack
78. :</label></td>
79.         </tr>
80.         <tr>
81.         <td><input type=\"radio\" name=\"snack\"
82. value=\"Coffe Break 1x\">Coffe Break 1x</td>
83.         <td>     </td>
84.         <td><input type=\"radio\" name=\"snack\"
85. value=\"Snack + Coffe Break 1x\">Snack + Coffe Break
86. 1x</td>
87.         </tr>
88.         <tr>
89.         <td><input type=\"radio\" name=\"snack\"
90. value=\"Coffe Break 2x\">Coffe Break 2x</td>
91.         <td>     </td>
92.         <td><input type=\"radio\" name=\"snack\"
93. value=\"Snack + Coffe Break 2x\">Snack + Coffe Break
94. 2x</td>
95.         </tr>
96.         <tr>
97.         <td colspan=\"3\"><input type=\"radio\"
98. name=\"snack\" value=\"No Snack\" checked=\"checked\">No
99. Snack</td>
100         </tr>
101         </table><br /><br />
102         <input type=\"submit\" class=\"btn btn-success\"
103 name=\"save_edit_request\" value=\"Save\"/>
104         </form>
105         </center><br /><br />";
106 }
```

Pada kelas main dijelaskan terkait dengan variable yang dipanggil serta method yang telah dijelaskan sebelumnya pada kelas controller. Aksi yang dilakukan pada antarmuka aplikasi berupa tanda edit pada masing-masing request yang di buat oleh requestor. Kemudian pada tombol tersebut dipanggil sebuah method edit_request untuk dilakukan *edit request*.

Kelas mga

Tabel 5.11 Implementasi *Edit Request* mga

```
1.  public function m_edit_request($data){
2.        $request_code = $data['request_code'];
3.        $building_code = $data['building_code'];
4.        $judul_meeting = $data['judul'];
5.        $room_layout = $data['room_layout'];
6.        $attendees = $data['attendees'];
7.        $facility = $data['facility'];
8.        $snack = $data['snack'];
9.        $query =
10.   "UPDATE request
```

```
11.   SET
12.  judul_meeting='$judul_meeting',room_layout='$room_layout',
13.  total_attendees='$attendees',facility='$facility',
14.  snack_description='$snack'
15.  WHERE request_code='$request_code' AND
16.  building_code='$building_code' AND
17.  status=1";
18.        $hasil = mysql_query($query);
19.        if(!$hasil){
20.        echo mysql_error();
21.        }
22.        else{
23.        return "Berhasil";
24.        }
25.  }
```

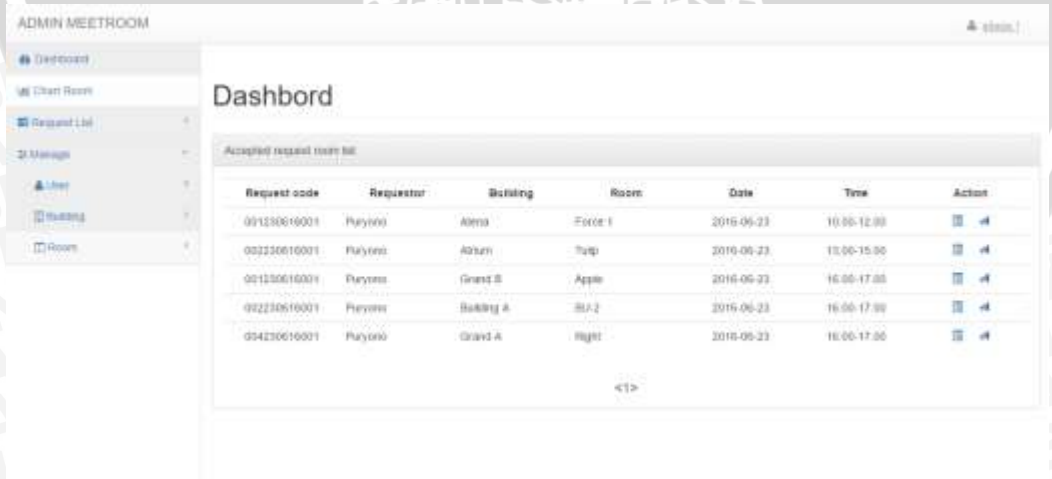
Kelas mga ini menjelaskan tentang representasi dari fungsi pada controller yang telah dijelaskan diawal. Fungsi yang dibuat bernama m_edit_request dengan variable $data.

5.1.4 Implementasi Antarmuka

Antarmuka sistem merupakan sarana yang digunakan pengguna untuk berinteraksi dengan sistem. Berikut penjelasan mengenai implementasi antramuka yang telah dibuat pada aplikasi manajemen ruang *meeting.*

5.1.4.1 Implementasi Antarmuka Lihat Daftar *Request* yang Diterima

Antarmuka lihat daftar *request* yang diterima merupakan tampilan dari halaman aktor admin. Halaman ini berisi keseluruhan *request* ruangan yang telah di *approve* oleh admin. Disajikan dalam bentuk table dengan beberapa informasi dan terdapat tombol aksi untuk melihat rincian dari masing-masing *request* dan tombol sms untuk menginirmkan informasi *request* ruangan. Implementasi antar muka lihat daftar *request* yang diterima dapat dilihat pada gambar 5.1 berikut.

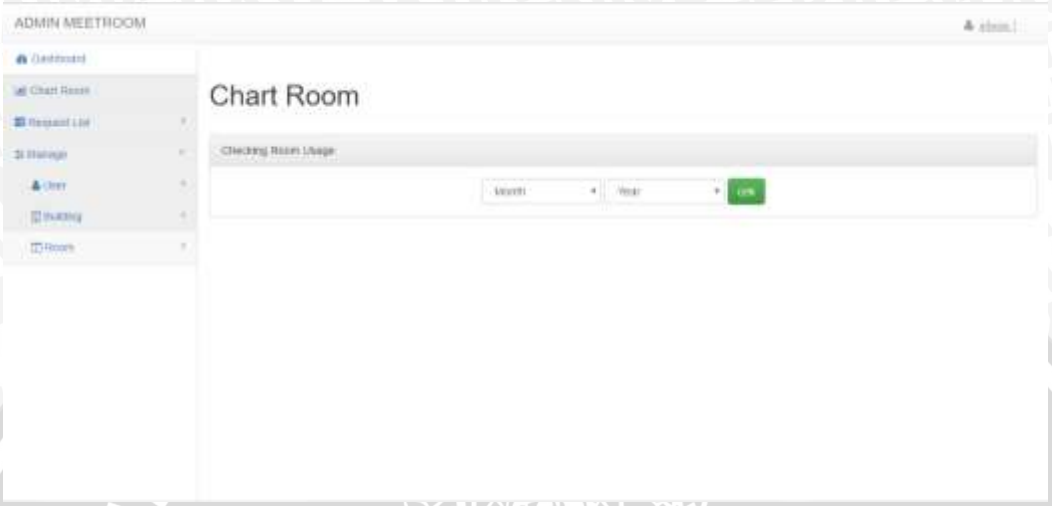


Gambar 5.1 Antarmuka *Home* Admin

5.1.4.2 Implementasi Antarmuka Lihat Bagan Penggunaan Ruangan

Antarmuka lihat bagan penggunaan ruangan bertujuan untuk menampilkan frekuensi penggunaan ruangan pada bulan dan tahun tertentu. Bagan yang

ditampilkan berupa nama ruangan yang digunakan pada pada bulan yang diinputkan oleh admin berdasarkan dengan tahun. Sistem akan melakukan rekapan hasil penggunaan ruangan yang sesuai dengan inputan admin tersebut. Implementasi antarmuka lihat bagan penggunaan ruangan dapat dilihat pada gambar 5.2 dan gambar 5.3 berikut

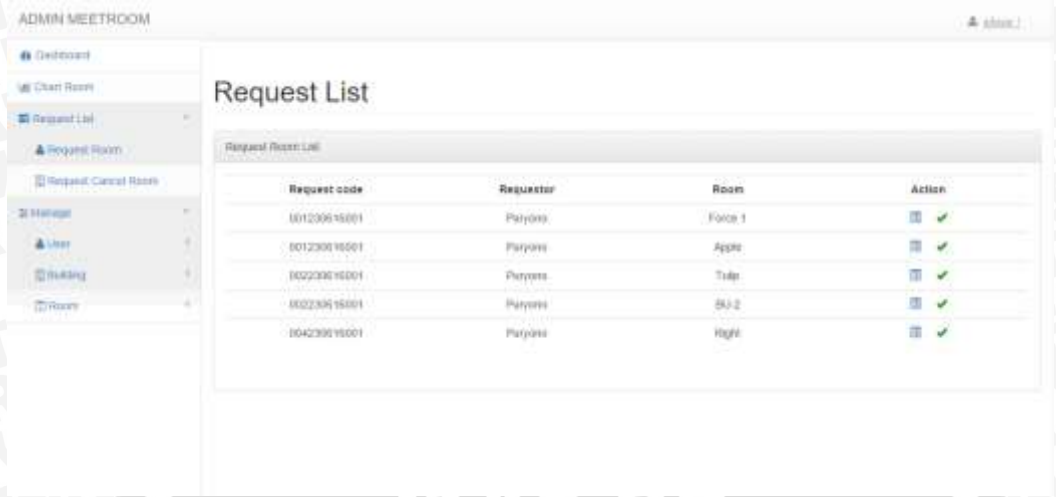


Gambar 5.2 Antarmuka *Chart Room*

Gambar 5.3 Antarmuka Hasil Input *Chart Room*

5.1.4.3 Implementasi Antarmuka Lihat Daftar *Request* Ruangan

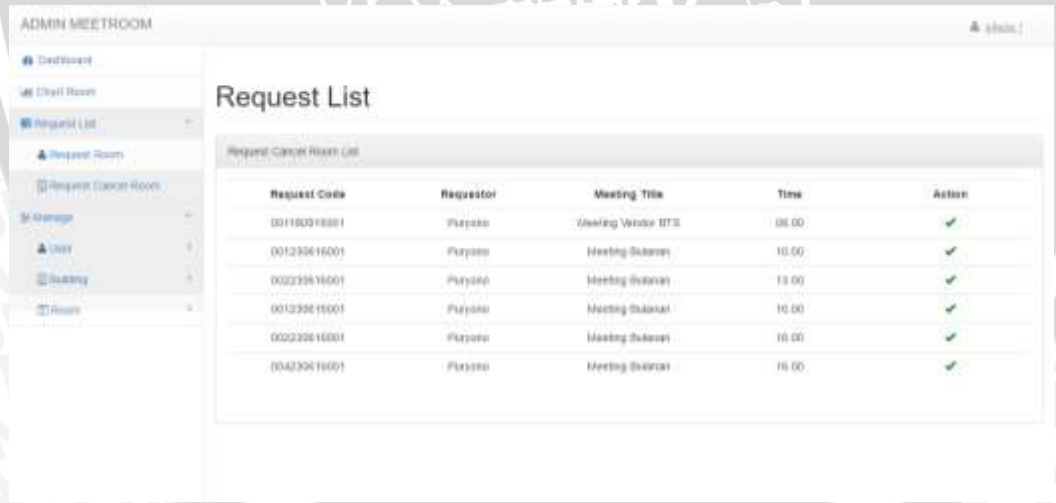
Antarmuka lihat daftar *request* ruangan merupakan tampilan untuk melihat daftar *request* yang telah dibuat oleh requestor namun belum dilakukan *approve* oleh admin. Implementasi lihat daftar *request* ruangan disajikan dalam bentuk table yang terdiri dari rician *request* yang masuk serta terdapat kolom aksi yang berisi dua buah dua buah tombol yaitu detail untuk melihat informasi secara detail pada request dan tombol *approve* untuk menyetujui *request* tersebut. Implementasi antarmuka lihat daftar *request* ruangan dapat dilihat pada gambar 5.4 berikut.



Gambar 5.4 Antarmuka *Request List Room*

5.1.4.4 Implementasi Antarmuka Lihat Daftar *Request Cancel*

Antarmuka lihat daftar *request cancel* tidak berbeda dengan tampilan lihat daftar *request* ruangan. Pada tampilan lihat daftar *request cancel* yang ditampilkan adalah daftar seluruh *request cancel* dari pihak requestor. Antarmuka lihat daftar *request cancel* disajikan dengan bentuk table yang berisi informasi request yang akan dicancel dan terdapat kolom aksi yang berisi tombol untuk menyetujui *request cancel* tersebut. Impelentasi antarmuka lihat daftar *request cancel* dapat dilihat pada gambar 5.5 berikut.

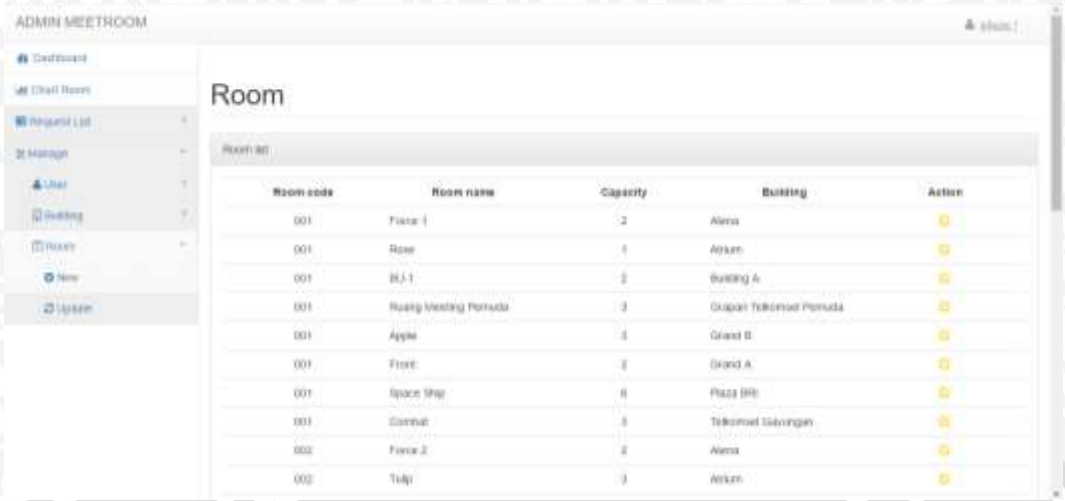


Gambar 5.5 Antarmuka *Request List Cancel*

5.1.4.5 Implementasi Antarmuka Lihat Daftar Ruangan

Antarmuka lihat daftar ruangan berisi seluruh ruangan yang terdaftar dalam sistem. Ruangan yang tersedia didaftarkan berdasarkan pada gedung yang terdaftar pada sistem. Sehingga daftar ruangan tidak berdiri sendiri, namun berdasarkan pada gedung yang ada. Antarmuka ini berisi daftar ruangan dengan

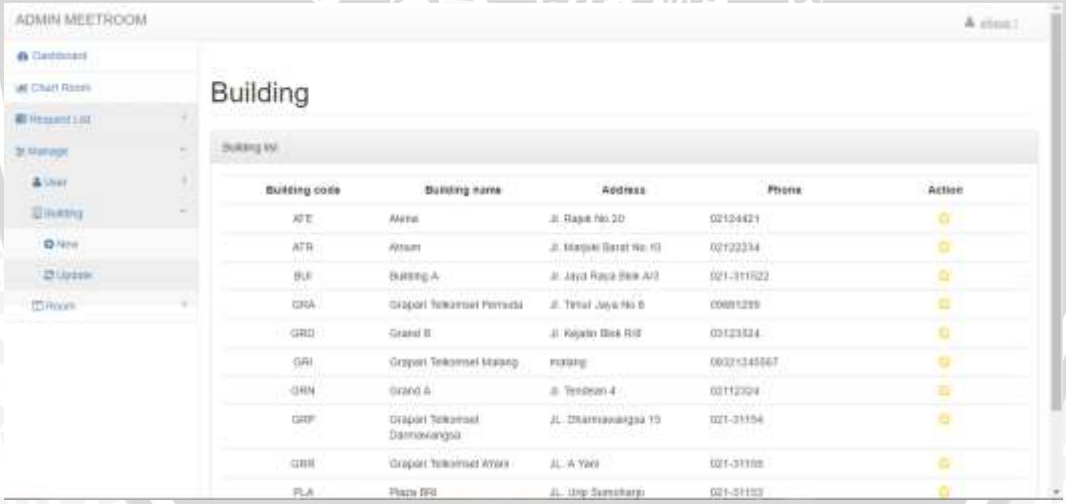
rincian informasi yang terkait dengan ruangan tersebut. Implemntasi antarmuka lihat daftar ruangan dapat dilihat pada gambar 5.6 berikut.



Gambar 5.6 Antarmuka Daftar Ruangan

5.1.4.6 Implementasi Antarmuka Lihat Daftar Gedung

Antarmuka lihat daftar gedung tidak berbeda dengan antarmuka lihat daftar raungan, pada antarmuka lihat daftar gedeung menampilkan seluruh gedung yang terdaftar kedalam sistem berserta dengan rincian informasinya. Implementasi antarmuka lihat daftar gedung dapat dilhat pada gambar 5.7 berikut.

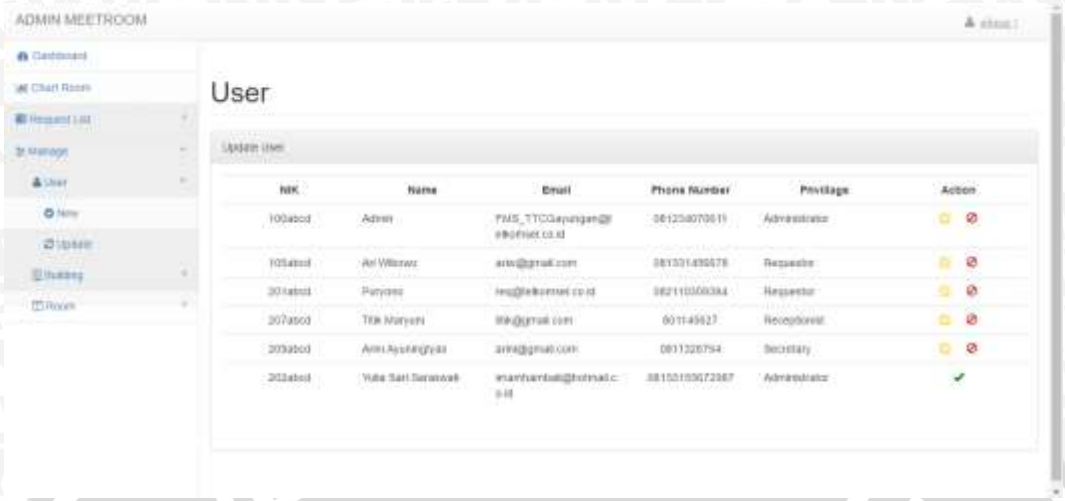


Gambar 5.7 Antarmuka Daftar Gedung

5.1.4.7 Implementasi Antarmuka Lihat Daftar *User*

Antarmuka lihat daftar user merupakan tampilan yang tidak berbeda dengan antarmuka lihat daftar ruangan dan daftar gedung, pada antarmuka lihat daftar user disajikan dalam table dengan informasi masing-masing user yang juga telah terdaftar kedalam sistem. Namun sedikit perbedaan terdapat pada kolom aksi yang berisi tombol *deactived* yang merupakan tombol untuk melakukan kegiatan non-active user apabila user tidak lagi menggunakan atau tidak lagi mendapat

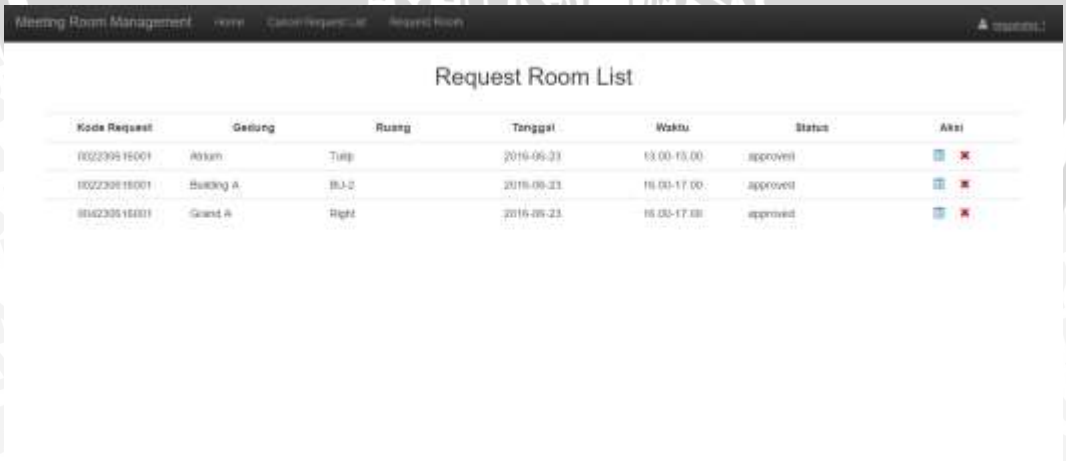
akses kedalam sistem aplikasi manajemen ruang *meeting*. Implementasi antarmuka lihat daftar user dapat dilihat pada gambar 5.8 Berikut.



Gambar 5.8 Antarmuka Daftar User

5.1.4.8 Implementasi Antarmuka Lihat Daftar *Request* Ruangan Requestor

Antarmuka daftar *request* ruangan hanya terdapat pada aktor requestor. Antarmuka ini menampilkan seluruh daftar *request* yang telah dibuat oleh requestor dan yang telah diteima oleh pihak admin. Antarmuka ini disajikan dalam bentuk table dengang beberapa informasi yang terkait dengan *request* tersebut. Terdapat beberapa kolom yang mengidentifikasikan informasi *request* tersebut dan terdapat satu buah kolom yang berisi dua buah tombol untuk melakukan aksi berupa melihat detail dari *request* yang dibuat dan tombol *cancel* untuk melakukan pembatalan *request*. Implementasi antarmuka lihat daftar *request* ruangan requestor dapa dilihat pada gambar 5.9 berikut.

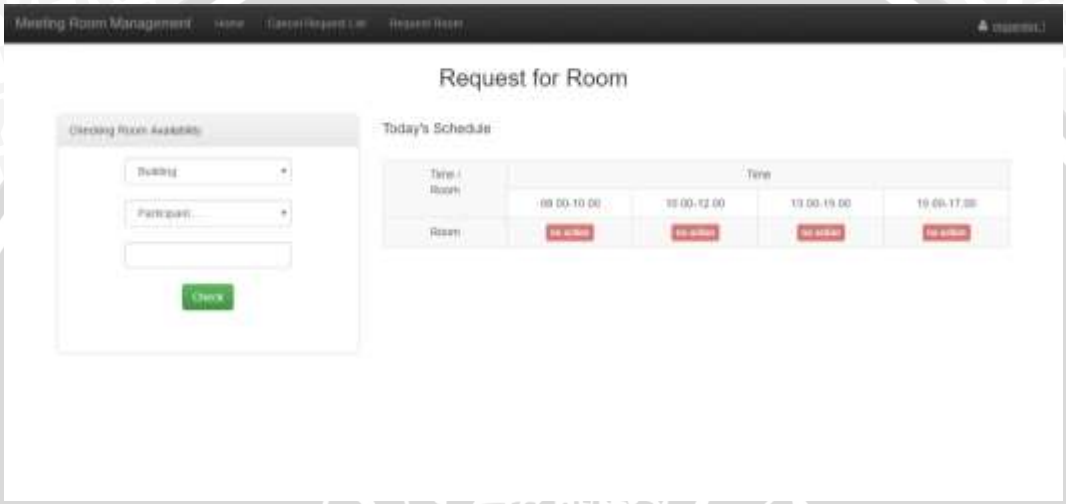


Gambar 5.9 Antarmuka *Home* Requestor

5.1.4.9 Implementasi Antarmuka *Request* Ruangan

Antarmuka *request* ruangan hanya terdapat pada aktor requestor, dimana proses *request* ruangan ini merupakan alur utama dalam pengembangan aplikasi

manajemen ruang *meeting.* Pada antarmuka *request* ruangan terdapat dua buah kolom yang pertama kolom untuk memasukkan informasi requestor terkait dengan *query* yang mempengaruhi hasil akhir berupa daftar ruangan berdasarkan jam serta status *booking.* Inputan yang berupa *query* dari antarmuka tersebut memiliki tiga buah field yang pertama field daftar nama gedung, kedua daftar peserta rapat berupa *range* dan ketiga field tanggal untuk memilih tanggal akan berlangsungnya kegiatan *meeting* serta terdapat tombol check untuk proses inputan yang telah dimasukkan ke dalam masing-masing field. kemudian pada kolom kedua berupa table yang berisi jam yang telah di tetapkan dan daftar list ruangan yang disesuaikan dengan inputan dari requestor. Implemntasi antarmuka *request* ruangan pada dilihat pada gambar 5.10 berikut.



Gambar 5.10 Antarmuka *Request Room*

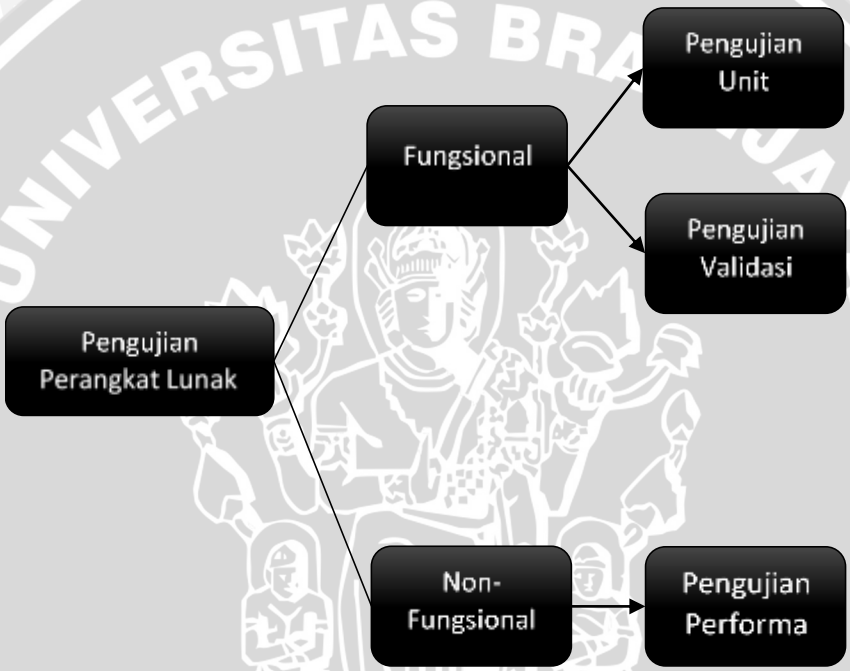
5.1.4.10 Implementasi Antarmuka Lihat Daftar Attendess List

Antarmuka lihat daftar *attendess list* merupakan tampilan receptionist yang dapat melihat seluruh daftar dari absensi dari keseluruhan kegiatan *meeting* yang dapat dijadikan sebagai rekapan untuk diproses lebih lanjut. Impelemntasi antarmuka lihat daftar *attendess list* dapat dilihat pada gambar 5.11 berikut.

Gambar 5.11 Antarmuka *Attendees List*

5.2 Pengujian

Pengujian yang dilakukan terdiri dari dua buah pengujian yaitu pengujian fungisonal dan pengujian non-fungsional. Pada pengujian fungsional terdiri dari pengujian unit, dan pengujian validasi. Sedangkan pengujian non-fungsional meliputi pengujian performa. Pengujian fungsional merupakan pengujian yang berdasarkan pada kebutuhan untuk dilakukan validasi pada sistem yang telah dibuat. Pada pengujian non-fungsional dilakukan bedasarkan pengujian diluar kebutuhan sistem yang telah terdefinisi dan bertujuan untuk melihat apakah sistem yang terbentuk siap untuk digunakan oleh stakeholder. Berikut pada gambar 5.12 menggambarkan alur dari pengujian pada aplikasi manajemen ruang *meeting.*



Gambar 5.12 Bagan Pengujian Perangkat Lunak

5.2.1 Pengujian Fungsional

5.2.1.1 Pengujian Unit

Pengujian unit bertujuan untuk memastikan beberapa algoritma yang memiliki prioritas tinggi dan telah diimplementasikan sesuai dengan yang diharapkan. Pengujian unit dilakukan dengan menggunakan *basis path*. Adapun algoritma yang diuji pada pengujian unit diantaranya adalah algoritma pemilihan jam pemesanan ruang *meeting* (*select time*) dan algoritma *edit request*.

1. Pengujian Algoritma *Checking Free Room*

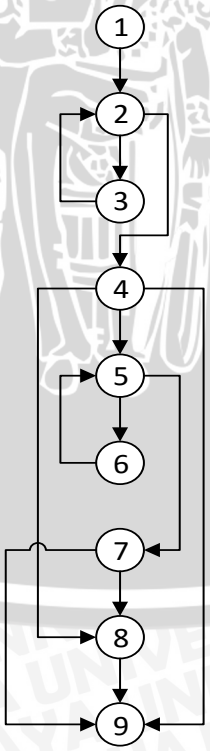
Algoritma *checking free room* adalah algoritma yang bertujuan untuk melakukan *checking* terhadap ruangan kosong yang tersedia pada salah satu gedung yang tersdia didalam sistem. Algortma ini menjadi salah satu dari komponen pemesanan ruang *meeting* untuk melihat apakah ruangana tersebut

dapat dipesan atau tidak. Gambar 5.13 menerapkan algoritma *checking free room* beserta dengan node flowgraph.

```
PUBLIC FUNCTION request_room(){
    Instantiation new class controller with class name is
    controller;                                                    1
    GET gedung in class controller to method c_get_gedung();
    GET capacity in class controller to method c_get_room();
       WHILE row = all of data in variable gedung                  2
    3    PRINT
         Value = array row building_name
       END WHILE   4
       WHILE row = all of data in variable capacity                5
    6    PRINT
         Value = array row capacity_id
       END WHILE   7
       PRINT                                                       8
       INPUT date with id name datepicker
}   9
```

Gambar 5.13 Algoritma *Checking Free Room*

Pada pembentukan node algoritma *checking free room*, banyaknya node yang terbentuk adalah 9 node. Berdasarkan pada node yang telah terbentuk, langkah selanjutnya adalah membuat flowgraph dari algoritma *checking free room* berdasarkan node-node yang telah ditentukan. Flowgraph algoritma *checking free room* dapat dilihat pada Gambar … berikut



Gambar 5.14 *Cyclometic Complexity Checking Free Room*

Berdasarkan flowgraph algoritma lihat register yang dapat dilihat pada Gambar 5.14 maka dapat dihitung nilai c*yclometic complexity* V(G) sebagai berikut,

V(G) = E – N + 2
= 13 – 9 + 2
= 4 + 2
= 6

V(G) = P + 1
= 5 + 1
= 6

Sehingga persamaan dari hasil *cyclometic complexcity* maka didapatkan jalur independen yaitu:

1. 1, 2, 3, 2, 4, 9
2. 1, 2, 3, 2, 8, 9
3. 1, 2, 4, 5, 6, 5, 7, 9
4. 1, 2, 4, 5, 6, 5, 7, 8, 9
5. 1, 2, 3, 2, 4, 5, 6, 5, 7, 9
6. 1, 2, 3, 2, 4, 5, 6, 5, 7, 8, 9

Berdasarkan 4 jalur independen yang telah didefinisikan tersebut maka dapat dibentuk kasus ujinya. Tabel 5.12 memaparkan kasusu uji algoritma *checking free room*.

Tabel 5.12 Kasus Uji Algoritma *Checking Free Room*

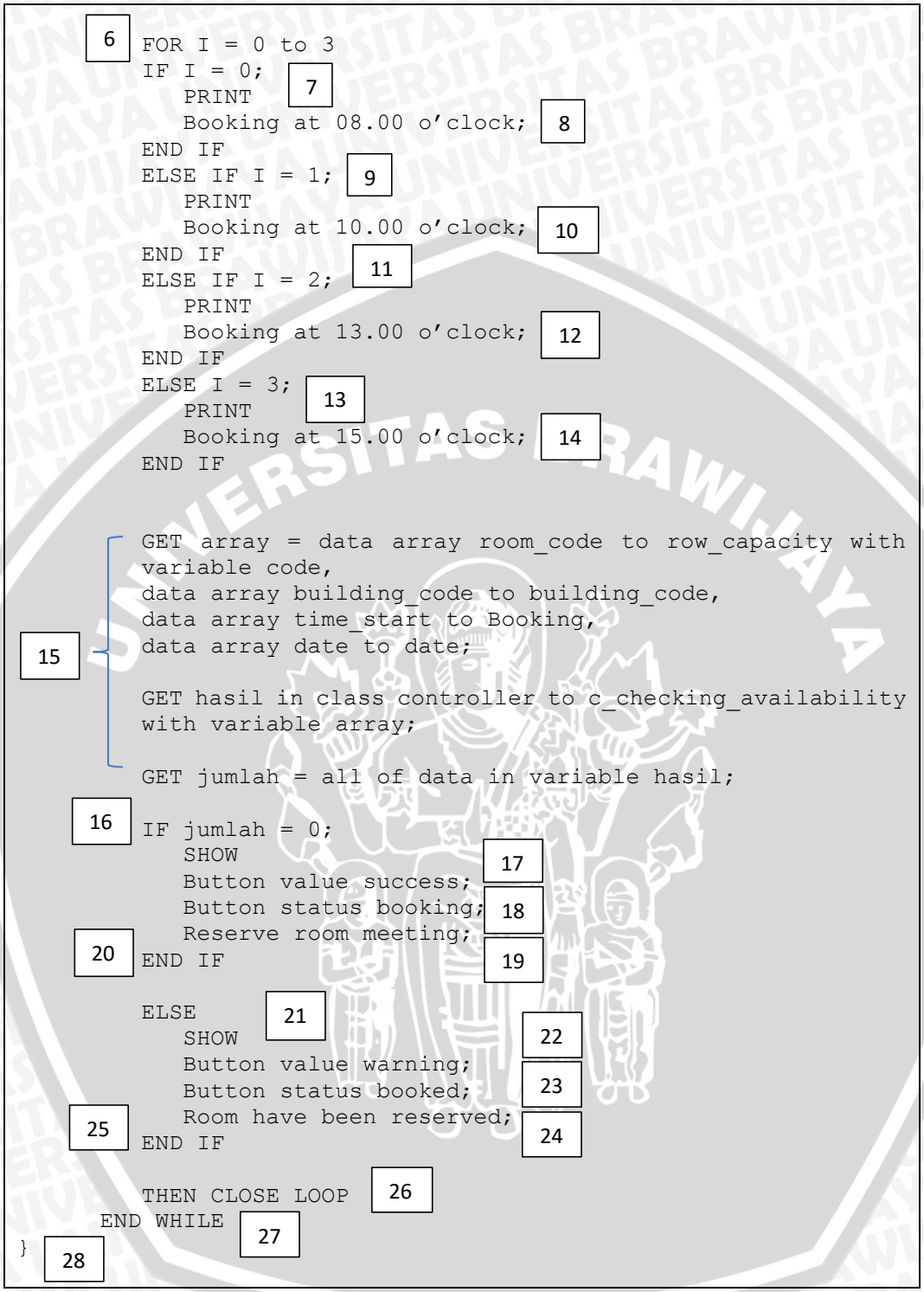
No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
1	1, 2, 3, 2, 4, 9	gedung = “char”	Sistem menampilkan seluruh daftar gedung	Sistem menampilkan seluruh daftar gedung	Valid
2	1, 2, 3, 2, 8, 9	gedung = “char”, Tanggal = “date”	Sistem menampilkan seluruh daftar gedung dan menampilkan tanggal	Sistem menampilkan seluruh daftar gedung dan menampilkan tanggal	Valid
3	1, 2, 4, 5, 6, 5, 7, 9	*capacity* = “integer”	Sistem menampilkan seluruh kapasitas ruangan	Sistem menampilkan seluruh kapasitas ruangan	Valid
4	1, 2, 4, 5, 6, 5, 7, 8, 9	*capacity* = “integer”, tanggal = “date”	Sistem menampilkan seluruh kapasitas ruangan dan	Sistem menampilkan seluruh kapasitas ruangan dan	Valid

			menampilkan tanggal	menampilkan tanggal	
5.	1, 2, 3, 2, 4, 5, 6, 5, 7, 9	gedung = "char", *capacity* = "integer"	Sistem menampilkan seluruh daftar gedung dan kapasitas ruangan	Sistem menampilkan seluruh daftar gedung dan kapasitas ruangan	Valid
6.	1, 2, 3, 2, 4, 5, 6, 5, 7, 8, 9	gedung = "char", *capacity* = "integer", tanggal = "date"	Sistem menampilkan seluruh daftar gedung, kapasitas ruangan, dan menampilkan tanggal	Sistem menampilkan seluruh daftar gedung, kapasitas ruangan, dan menampilkan tanggal	Valid

2. Pengujian Algoritma *Select Time*

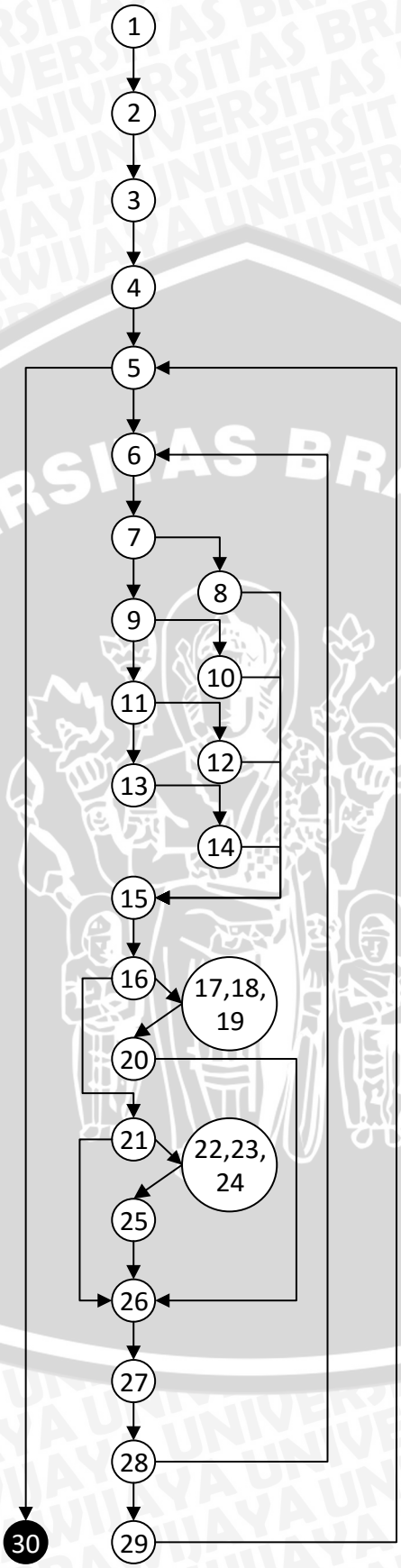
Algoritma *select time* merupakan algoritma yang bertujuan untuk menampilkan daftar ruangan yang dapat dipesan berdasarkan dengan hasil pengolahan inputan yang diberikan oleh requestor. Algoritma ini menjadi satu kesatuan dengan fitur pemesansan ruang *meeting*. Gambar 5.15 memaparkan algoritma *select time* beserta dengan node flowgraph.

```
PUBLIC FUNCTION select_time(string building_code, date date,
integer kapasitas){                                          1
   Instantiation new class controller with class name is
   controller;
   GET gedung in class controller to method c_get_gedung();
   GET capacity in class controller to method c_get_room();
   GET room_capacity in class controller to method
   c_get_room_capacity(string building_code, integer kapasitas);
   GET room in class controller to method c_room();
   Integer time = " ";
   Integer count = 4;
   Integer j = 0;
      WHILE row = all of data in variable gedung
         PRINT
         Value = array row building_name;                    2
      END WHILE

      WHILE row = all of data in variable capacity
         PRINT
         Value = array row capacity_id;                      3
      END WHILE

      PRINT
      INPUT date with id name datepicker;                    4

   5  WHILE row_capacity = all of data in variable room_capacity
         PRINT
         Value = array row_capacity;
```



Gambar 5.15 Algoritma *Select Time*

Pada pembentukan node algoritma *select time*, banyaknya node yang terbentuk adalah 28 node Berdasarkan pada node yang telah terbentuk, langkah selanjutnya adalah membuat flowgraph dari algoritma *select time* berdasarkan node-node yang telah ditentukan. Flowgraph algoritma *select time* dapat dilihat pada Gambar 5.16 berikut,



Gambar 5.16 ***Cyclometic Complexity Select Time***

Berdasarkan flowgraph algoritma lihat register yang dapat dilihat pada Gambar 5.16 maka dapat dihitung nilai c*yclometic complexity* V(G) sebagai berikut,

V(G) = E – N + 2
= 32 – 26 + 2
= 6 + 2
= 8

V(G) = P + 1
= 7 + 1
= 8

Sehingga persamaan dari hasil *cyclometic complexcity* maka didapatkan jalur independen yaitu,

1. 1, 2, 3, 4, 5, 6, 7, 8, 15, 16, (17,18,19), 20, 26, 27, 28, 29, 8, 30
2. 1, 2, 3, 4, 5, 6, 7, 8, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30
3. 1, 2, 3, 4, 5, 6, 7, 9, 10, 15, 16, (17,18,19), 20, 26, 27, 28, 29, 8, 30
4. 1, 2, 3, 4, 5, 6, 7, 9, 10, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30
5. 1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 15, 16, (17,18,19), 20, 26, 27, 28, 29, 8, 30
6. 1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30
7. 1, 2, 3, 4, 5, 6, 7, 9, 11, 13, 14, 15, 16, (17,18,19), 20, 26, 27, 28, 29, 8, 30
8. 1, 2, 3, 4, 5, 6, 7, 9, 11, 13, 14, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30

Berdasarkan 8 jalur independen yang telah didefinisikan tersebut maka dapat dibentuk kasus ujinya. Tabel 5.13 memaparkan kasusu uji algoritma *select time*.

Tabel 5.13 Kasus Uji Algortima *Select Time*

No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
1	1, 2, 3, 4, 5, 6, 7, 8, 15, 16, (17,18,19), 20, 26, 27, 28, 29, 8, 30	I = 0, book = 08.00, jumlah = 0	Sistem menampilkan daftar nama ruangan yang dapat dipesan pada jam 08.00	Sistem menampilkan daftar nama ruangan yang dapat dipesan pada jam 08.00	Valid
2	1, 2, 3, 4, 5, 6, 7, 8, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30	I = 0, book = 08.00, jumlah != 0	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 08.00	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 08.00	Valid
3	1, 2, 3, 4, 5, 6, 7, 9, 10, 15, 16, (17,18,19), 20,	I = 1, book = 10.00, jumlah = 0	Sistem menampilkan daftar nama	Sistem menampilkan daftar nama	Valid

	26, 27, 28, 29, 8, 30		ruangan yang dapat dipesan pada jam 10.00	ruangan yang dapat dipesan pada jam 10.00	
4	1, 2, 3, 4, 5, 6, 7, 9, 10, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30	I = 1, book = 10.00, jumlah != 0	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 10.00	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 10.00	Valid
5	1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 15, 16, (17,18,19), 20, 26, 27, 28, 29, 8, 30	I = 2, book = 13.00, jumlah = 0	Sistem menampilkan daftar nama ruangan yang dapat dipesan pada jam 13.00	Sistem menampilkan daftar nama ruangan yang dapat dipesan pada jam 13.00	Valid
6	1, 2, 3, 4, 5, 6, 7, 9, 11, 12, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30	I = 2, book = 13.00, jumlah != 0	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 13.00	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 13.00	Valid
7.	1, 2, 3, 4, 5, 6, 7, 9, 11, 13, 14, 15, 16, (17,18,19), 20, 26, 27, 28, 29, 8, 30	I = 3, book = 15.00, jumlah = 0	Sistem menampilkan daftar nama ruangan yang dapat dipesan pada jam 15.00	Sistem menampilkan daftar nama ruangan yang dapat dipesan pada jam 15.00	Valid
8.	1, 2, 3, 4, 5, 6, 7, 9, 11, 13, 14, 15, 16, 21, (22,23,24), 25, 26, 27, 28, 29, 8, 30	I = 3, book = 15.00, jumlah != 0	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 15.00	Sistem menampilkan daftar nama ruangan yang dapat tidak dapat dipesan pada jam 15.00	Valid

3. Pengujian Algoritma *Edit Request*

Algoritma *edit request* merupakan algoritma yang ditujukan untuk requestor yang ingin merubah keperluan dari *request meeting* yang akan berlangsung. Algoritma ini memeliki batasan yakni ketika terdapat kondisi dimana *request* tersebut belum mendapatkan *approve* oleh pihak admin, requestor dapat melakukan *edit request* berdasarkan *request* yang tersedia. Namun setelah *request meeting* telah mendapat *approve* dari pihak admin, maka secara otomatis requestor tidak dapat melakukan *edit request*. Gambar 5.17 memaparkan algoritma *select time* beserta dengan node flowgraph

```
PUBLIC FUNCTION edit_request(string request_code, string
building_code){
1   Instantiation new class controller with class name is
    controller;
    GET data in class controller to method c_get_req_bname(string
    request_code, string building_code);
    PRINT
       INPUT hidden request code;
       INPUT hidden building code;
       INPUT hidden requestor name;
       INPUT meeting title from value data array "judul_meeting";
       SELECT room_layout;
       IF data array room_layout = O Shape;  2
          PRINT
          Value selected O Shape;
          Value option U Shape;              3
          Value option Class Room Shape;
       END IF
       ELSE IF data array room_layout = U Shape;  4
          PRINT
          Value selected U Shape;
          Value option O Shape;              5
          Value option Class Room Shape;
       END IF
       ELSE data array room_layout = Class Room Shape;  6
          PRINT
          Value selected Class Room Shape;
          Value option O Shape;
          Value option U Shape;              7
       END IF

       INPUT attendees from value data array "attendees";
       SELECT facility;
       IF data array facility = Internet;  8
          PRINT
          Value selected Internet;
          Value option Intranet;             9
          Value option Intranet and Internet;
        END IF
        ELSE IF data array facility = Intranet;  10
          PRINT
          Value selected Intranet;
          Value option Internet;             11
          Value option Intranet and Internet;
        END IF
```

```
        ELSE data array facility = Intranet and Internet;
          Value selected Intranet and Internet;
          Value option Internet;
          Value option Intranet;
        END IF
        INPUT snack;
}
```

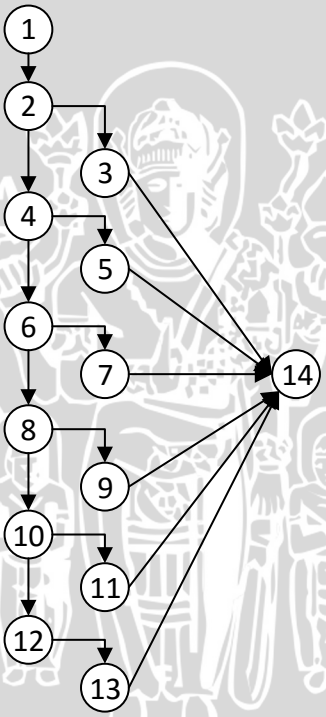
12

13

14

Gambar 5.17 Algoritma *Edit Request*

Pada pembentukan node algoritma *edit request*, banyaknya node yang terbentuk adalah 14 node Berdasarkan pada node yang telah terbentuk, langkah selanjutnya adalah membuat flowgraph dari algoritma *edit request* berdasarkan node-node yang telah ditentukan. Flowgraph algoritma *edit request* dapat dilihat pada Gambar 5.18 berikut.



Gambar 5.18 *Cyclometic Complexity Edit Request*

Berdasarkan flowgraph algoritma lihat register yang dapat dilihat pada Gambar 5.18 maka dapat dihitung nilai c*yclometic complexity* V(G) sebagai berikut,

V(G) = E – N + 2
= 18 – 14 + 2
= 4 + 2
= 6

V(G) = P + 1
= 5 + 1
= 6

Sehingga persamaan dari hasil *cyclometic complexcity* maka didapatkan jalur independen yaitu,

1. 1, 2, 3, 14
2. 1, 2, 4, 5, 14
3. 1, 2, 4, 6, 7, 14
4. 1, 2, 4, 6, 8, 9, 14
5. 1, 2, 4, 6, 8, 10, 11, 14
6. 1, 2, 4, 6, 8, 10, 12, 13, 14

Berdasarkan 6 jalur independen yang telah didefinisikan tersebut maka dapat dibentuk kasus ujinya. Tabel 5.14 memaparkan kasusu uji algoritma *edit request*.

Tabel 5.14 Kasus Uji Algortima *Edit Request*

No	Jalur	Data input	Hasil yang diharapakan	Hasil yang diperoleh	Status
1	1, 2, 3, 14	Room_layout = "O Shape"	Sistem menampilkan room_layout O Shape	Sistem menampilkan room_layout O Shape	Valid
2	1, 2, 4, 5, 14	Room_layout = "U Shape"	Sistem menampilkan room_layout U Shape	Sistem menampilkan room_layout U Shape	Valid
3	1, 2, 4, 6, 7, 14	Room_layout = "Class Shape"	Sistem menampilkan room_layout Class Shape	Sistem menampilkan room_layout Class Shape	
4	1, 2, 4, 6, 8, 9, 14	Facility = "Internet"	Sistem menampilkan facility berupa pilihan internet	Sistem menampilkan facility berupa pilihan internet	Valid
5	1, 2, 4, 6, 8, 10, 11, 14	Facility = "intranet"	Sistem menampilkan facility berupa pilihan intranet	Sistem menampilkan facility berupa pilihan intranet	Valid
6	1, 2, 4, 6, 8, 10, 12, 13, 14	Facility = "Internet dan intranet"	Sistem menampilkan facility berupa pilihan internet dan intranet	Sistem menampilkan facility berupa pilihan internet dan intranet	Valid

5.2.1.2 Pengujian Validasi

Pengujian validasi berfungsi sebagai sarana untuk mengetahui bagaimana cara berjalan setelah melewati proses pengembangan, dan dapat menyediakan beberapa fungsi yang dibutuhkan berdasarkan kebutuhan sistem. Pengujian validasi sering disebut dengan pengujian *Black box*, karena pengujian *Black Box* ini tidak berfokus pada alur algoritma, namun lebih menekankan pada kesesuaian antar kinerja sistem dengan daftar dari kebutuhan sistem. Dalam hal ini, pengujian

validasi dilakukan berdasarkan pada skenario *use case* dengan melihat kondisi yang terjadi pada masing-masing *use case*. Kondisi yang terdapat pada scenario *use case* adalah kondisi awal dan kondisi alternative. Masing-masing kondisi tersebut didefinisikan sebagai sebuah *test case* yang digunakan untuk mengetahui peluang – peluang kegiatan yang akan dilakukan oleh *stakeholder*. Berikut ini merupakan *test case* yang terbentuk dari masing-masing *use case*.

1. Memesan ruang rapat

   - *Test case* kondisi awal memesan ruang rapat

Tabel 5.15 *Test Case* Kondisi Awal Memesan Ruang Rapat

Nama Kasus Uji	Uji kondisi awal memesan ruang rapat.
Objek Uji	Uji Halaman *request room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk dapat memesan ruang meeting dengan berdasarkan inputan nama gedung, kapasitas dan tanggal .
Prosedur Uji	Aktor telah *login* dan dapat melakukan pemesanan ruangan rapat
Hasil yang Diharapkan	Sistem dapat menampilkan halaman *request room* pada kolom *today's schedule* berupa ruangan nama ruangan berdasarikan jam yang telah ditentukan.
Hasil Uji Coba	Valid

   - *Test case* kondisi alternatif memesan ruang rapat

Tabel 5.16 *Test Case* Kondisi Alternatif Memesan Ruang Rapat

Nama Kasus Uji	Uji kondisi alternatif memesan ruang rapat.
Objek Uji	Uji Halaman *request room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional melihat apakah inputan requestor sesuai.
Prosedur Uji	Aktor telah *login* dan dapat melakukan pemesanan ruangan rapat
Hasil yang Diharapkan	Sistem akan kembali pada halaman *home* untuk dilakukan *input* kembail pada form yang tersedia.
Hasil Uji Coba	Valid

2. mengubah informasi permintaan rapat

- *Test case* kondisi awal mengubah informasi rapat

Tabel 5.17 *Test Case* Kondisi Awal Mengubah Informasi Rapat

Nama Kasus Uji	Uji kondisi awal mengubah informasi rapat.
Objek Uji	Uji halaman *edit request*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk mengubah informasi rapat sesuai dengan kebutuhan.
Prosedur Uji	Aktor telah *login* dan dapat merubah permintaan ruangan dari daftar yang ada pada halaman *home*.
Hasil yang Diharapkan	Sistem memproses perintah *edit* dan informasi yang telah diperbarui disimpan kedalam sistem
Hasil Uji Coba	Valid

- *Test case* kondisi alternaif mengubah informasi rapat

Tabel 5.18 *Test Case* Kondisi Alternatif Mengubah Informasi Rapat

Nama Kasus Uji	Uji kondisi alternatif mengubah informasi rapat.
Objek Uji	Uji Halaman *edit request*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk dapat mengubah informasi .
Prosedur Uji	Aktor menekan tombol *cancel* untuk membatalkan perubahan informasi.
Hasil yang Diharapkan	Sistem akan kembali pada halaman *edit request* untuk dilakukan *input* kembail pada form yang tersedia.
Hasil Uji Coba	Valid

3. Membatalkan permintaan rapat

- *Test case* kondisi awal membatalkan permintaan rapat

Tabel 5.19 *Test Case* Kondisi Awal Membatalkan Permintaan Rapat

Nama Kasus Uji	Uji kondisi awal membatalkan permintaan rapat.
Objek Uji	Uji halaman *cancel request*.

Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan pembatalan pemesan ruangan *meeting*.
Prosedur Uji	Aktor menekan tombol batal untuk membatalkan *request*.
Hasil yang Diharapkan	Sistem akan menampilkan halaman form *cancel request* dan aktor melakukan klik pada tombol *send* untuk membatalkan *request*.
Hasil Uji Coba	Valid

4. Memberikan infromasi rapat

- *Test case* konidis awal memberikan informasi rapat

Tabel 5.20 *Test Case* Kondisi Awal Memberikan Informasi Rapat

Nama Kasus Uji	Uji kondisi awal memberikan informasi rapat.
Objek Uji	Uji tombol *send*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan pemberian informasi kepada requestor terkait dengan *request* ruangan dengan memanfaatkan teknologi *SMS Gateaway*.
Prosedur Uji	Aktor menekan tombol *send* untuk memberikan informasi *request* ruangan.
Hasil yang Diharapkan	Sistem akan mengiriman informasi kepada requestor yang telah melakukan *request* ruangan.
Hasil Uji Coba	Valid

5. Menghasilkan daftar absensi

- *Test case* kondisi awal menghasilkan daftar absensi

Tabel 5.21 *Test Case* Kondisi Awal Menghasilkan Daftar Absensi

Nama Kasus Uji	Uji kondisi awal menghasilkan daftar absensi
Objek Uji	Uji tombol *download*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk

	menghasilkan daftar absensi berupa file pdf yang dapat di unduh.
Prosedur Uji	Aktor telah *login* dan sistem dapat menampilkan daftar absensi yang dapat di unduh oleh aktor sebagai absensi peserta rapat.
Hasil yang Diharapkan	Sistem akan menampilkan halaman form *cancel request* dan aktor melakukan klik pada tombol *send* untuk membatalkan *request*.
Hasil Uji Coba	Valid

6. Manajemen pengguna

- *Test case* kondisi awal manajemen pengguna

Tabel 5.22 *Test Case* Kondisi Awal Manajemen Pengguna

Nama Kasus Uji	Uji kondisi awal manajemen pengguna
Objek Uji	Uji halaman *add new user*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional menambahkan user baru sesuai dengan hak askses yang ditentukan oleh admin.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan penambahan user baru dengan beberapa data yang tersedia dengan masing-masing hak akses yang berbeda
Hasil yang Diharapkan	Sistem akan menampilkan halaman form *add new user* dan aktor dapat menambahkan user baru.
Hasil Uji Coba	Valid
Objek Uji	Uji halaman *update user*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan *update user* yang telah terdaftar dalam sistem.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan pembaruan user berdasarkan daftar user yang tersedia dengan menekan tombol *update* pada salah satu user yang terdaftar.
Hasil yang Diharapkan	Sistem akan menampilkan *pop-up* berupa pembaruan user yang berhasil tersimpan kedalam sistem.

Hasil Uji Coba	Valid

- *Test case* kondisi alternatif manajemen pengguna

Tabel 5.23 *Test Case* Kondisi Alternatif Manajemen Pengguna

Nama Kasus Uji	Uji kondisi alternatif manajemen pengguna
Objek Uji	Uji halaman *add new user*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan pengisian data diri user baru.
Prosedur Uji	Aktor telah *login* dan mengisi beberapa data diri user baru yang akan di daftarkan.
Hasil yang Diharapkan	sistem akan melakukan *reload* halaman form penambahan *user* baru jika terjadi kekurangan pengisian data pada form yang telah tersedia.
Hasil Uji Coba	Valid
Objek Uji	Uji halaman *update user*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan pengisian perubahan informasi dari data diri berdasarkan field yang tersedia.
Prosedur Uji	Aktor telah *login* kemudian sistem menampilkan form perubahan informasi user dan aktor mengisikan salah satu dari field yang tersedia.
Hasil yang Diharapkan	Sistem menampilkan <!> *warning* bahwa form harus diisi.
Hasil Uji Coba	Valid

7. Melihat rincian permintaan rapat

- *Test case* kondisi awal melihat rincian permintaan rapat

Tabel 5.24 *Test Case* Kondisi Awal Melihat Rincian Permintaan Rapat

Nama Kasus Uji	Uji kondisi awal melihat rincian permintaan rapat
Objek Uji	Uji tombol *detail*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melihat rincian dari *request* yang dilakukan oleh requestor.

Prosedur Uji	Aktor telah *login* dan sistem dapat menampilkan rincian data dari *request* yang tersedia.
Hasil yang Diharapkan	Sistem akan menampilkan seluruh rincian dari salah satu *request* yang dipilih.
Hasil Uji Coba	Valid

8. Menerima permintaan rapat

- *Test case* kondisi awal menerima permintaan rapat

Tabel 5.25 *Test Case* Kondisi Awal Menerima Permintaan Rapat

Nama Kasus Uji	Uji kondisi awal menerima permintaan rapat
Objek Uji	Uji tombol *accept*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menerima seluruh *request* ruangan yang dilakukan oleh reqeusetor.
Prosedur Uji	Aktor telah *login* dan aktor melakukan *approve* dengan menekan tombol *accept* pada halaman *request room list*.
Hasil yang Diharapkan	Sistem akan menampilkan *request* yang telah di terima pada halaman *home* admin sebagai *request* yang telah diterima.
Hasil Uji Coba	Valid

9. Menerima pembatalan permintaan rapat

- *Test case* kondisi awal menerima pembatalan permintaan rapat

Tabel 5.26 *Test Case* Kondisi Awal Menerima Pembatalan Permintaan Rapat

Nama Kasus Uji	Uji kondisi awal menerima pembatalan permintaan rapat
Objek Uji	Uji tombol *accept*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menerima seluruh pembatalan *request* dari requestor.
Prosedur Uji	Aktor telah *login* dan aktor melakukan *approve* dari *request* pembatalan pada halaman *request cancel room list*.

Hasil yang Diharapkan	Sistem akan menerima *request* pembatalan yang dilakukan oleh requestor status *request* akan berubah menjadi *request has been cancel*.
Hasil Uji Coba	Valid

10. Manajemen gedung

- *Test case* kondisi awal manajemen gedung

Tabel 5.27 *Test Case* Kondisi Awal Manajemen Gedung

Nama Kasus Uji	Uji kondisi awal manajemen gedung
Objek Uji	Uji halaman *add new building*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menambahkan gedung baru.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan penambahan gedung baru dengan beberapa data yang tersedia.
Hasil yang Diharapkan	Sistem akan menampilkan halaman form *add new building* dan aktor dapat menambahkan gedung baru.
Hasil Uji Coba	Valid
Objek Uji	Uji halaman *update building*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan *update* gedung yang telah terdaftar dalam sistem.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan pembaruan ruangan berdasarkan daftar gedung yang tersedia dengan menekan tombol *update* pada salah satu gedung yang terdaftar.
Hasil yang Diharapkan	Sistem akan menampilkan *pop-up* berupa pembaruan informasi gedung yang berhasil tersimpan kedalam sistem.
Hasil Uji Coba	Valid

- *Test case* kondisi alternatif manajemen gedung

Tabel 5.28 *Test Case* Kondisi Alternatif Manajemen Gedung

Nama Kasus Uji	Uji kondisi awal manajemen gedung
Objek Uji	Uji halaman *add new buiding*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menambahkan gedung baru.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan penambahan gedung baru dengan beberapa data yang tersedia.
Hasil yang Diharapkan	Sistem akan menampilkan halaman form *add new building* dan aktor dapat menambahkan gedung baru.
Hasil Uji Coba	Valid
Objek Uji	Uji halaman *update building*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan *update* gedung yang telah terdaftar dalam sistem.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan pembaruan ruangan berdasarkan daftar gedung yang tersedia dengan menekan tombol *update* pada salah satu gedung yang terdaftar.
Hasil yang Diharapkan	Sistem akan menampilkan *pop-up* berupa pembaruan informasi gedung yang berhasil tersimpan kedalam sistem.
Hasil Uji Coba	Valid

11. Manajemen ruangan

- *Test case* kondisi awal manajemen ruangan

Tabel 5.29 *Test Case* Kondisi Awal Manajemen Ruangan

Nama Kasus Uji	Uji kondisi awal manajemen ruangan
Objek Uji	Uji halaman *add new room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk

	menambahkan ruangan baru berdasarkan gedung yang telah terdaftar dalam sistem.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan penambahan ruangan baru dengan beberapa data yang tersedia.
Hasil yang Diharapkan	Sistem akan menampilkan halaman form *add new room* dan aktor dapat menambahkan ruangan baru.
Hasil Uji Coba	Valid
Objek Uji	Uji halaman *update room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan *update* ruangan yang telah terdaftar dalam sistem.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan pembaruan ruangan berdasarkan daftar ruangan yang tersedia dengan menekan tombol *update* pada salah satu ruangan yang terdaftar.
Hasil yang Diharapkan	Sistem akan menampilkan *pop-up* berupa pembaruan informasi ruangan yang berhasil tersimpan kedalam sistem.
Hasil Uji Coba	Valid

- *Test case* kondisi alternatif manajemen ruangan

Tabel 5.30 *Test Case* Kondisi Alternatif Manajemen Ruangan

Nama Kasus Uji	Uji kondisi awal manajemen ruangan
Objek Uji	Uji halaman *add new room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menambahkan ruangan baru berdasarkan gedung yang telah terdaftar dalam sistem.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan penambahan ruangan baru dengan beberapa data yang tersedia.
Hasil yang Diharapkan	Sistem akan menampilkan halaman form *add new room* dan aktor dapat menambahkan ruangan baru.
Hasil Uji Coba	Valid

Objek Uji	Uji halaman *update room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melakukan *update* ruangan yang telah terdaftar dalam sistem.
Prosedur Uji	Aktor telah *login* dan sistem dapat melakukan pembaruan ruangan berdasarkan daftar ruangan yang tersedia dengan menekan tombol *update* pada salah satu ruangan yang terdaftar.
Hasil yang Diharapkan	Sistem akan menampilkan *pop-up* berupa pembaruan informasi ruangan yang berhasil tersimpan kedalam sistem.
Hasil Uji Coba	Valid

12. Melihat frekuensi penggunaan ruangan

- *Test case* kondisi awal melihat frekuensi penggunaan ruangan

Tabel 5.31 *Test Case* Kondisi Awal Melihat Frekuensi Penggunaan Ruangan

Nama Kasus Uji	Uji kondisi awal melihat frekuensi penggunaan ruangan
Objek Uji	Uji halaman form *chart*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melihat frekuensi penggunaan ruangan berdasarkan pada tahun dan bulan.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan inputan berupa bulan dan tahun.
Hasil yang Diharapkan	Sistem akan menampilkan hasil dari inputan aktor berupa diagram batang yang mengindikasikan total dari penggunaan ruangan.
Hasil Uji Coba	Valid

- *Test case* kondisi alternatif melihat frekuensi penggunaan ruangan

Tabel 5.32 *Test Case* Kondisi Alternatif Melihat Frekuensi Penggunaan Ruangan

Nama Kasus Uji	Uji kondisi alternatif melihat frekuensi penggunaan ruangan
Objek Uji	Uji halaman form *chart*.

Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk memberikan pesan keslahan pada aktor terhadap inputan untuk frekuensi penggunaan ruangan berdasarkan pada tahun dan bulan.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan inputan dari salah satu field yang tersedia yang berupa bulan dan tahun.
Hasil yang Diharapkan	Sistem akan menampilkan pesan *error* bahwa field tersebut wajib diisi untuk melihat hasil yang diinginkan.
Hasil Uji Coba	Valid

13. Melihat daftar ruangan kosong

- *Test case* kondisi awal melihat daftar ruangan kosong

Tabel 5.33 *Test Case* Kondisi Awal Melihat Daftar Ruangan Kosong

Nama Kasus Uji	Uji kondisi awal melihat daftar ruangan kosong
Objek Uji	Uji halaman *request room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menampilkan seluruh ruangan yang tersedia untuk di pesan berdsarkan pada inputan requestor.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan inputan berupa nama gedung, kapasitas ruangan dan tanggal.
Hasil yang Diharapkan	Sistem akan menampilkan hasil dari inputan aktor berupa seluruh ruangan yang tersedia dengan status ruangan kosong.
Hasil Uji Coba	Valid

- *Test case* kondisi alternatif melihat frekuensi penggunaan ruangan

Tabel 5.34 *Test Case* Kondisi Alternatif Melihat Frekuensi Penggunaan Ruangan

Nama Kasus Uji	Uji kondisi alternatif melihat daftar ruangan kosong
Objek Uji	Uji halaman *request room*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menampilkan pesan kesalahan pada aktor terhadap

	seluruh ruangan yang tersedia untuk di pesan berdsarkan pada inputan requestor.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan inputan pada salah satu field berupa nama gedung, kapasitas ruangan dan tanggal.
Hasil yang Diharapkan	Sistem akan menampilkan pesan *error* terhadap hasil dari inputan aktor.
Hasil Uji Coba	Valid

14. Menyaring daftar permintaan ruangan

- *Test case* kondisi awal menyaring daftar permintaan ruangan

Tabel 5.35 *Test Case* Kondisi Awal Menyaring Daftar Permintaan Ruangan

Nama Kasus Uji	Uji kondisi awal menyaring daftar permintaan rapat
Objek Uji	Uji halaman *home* receptionist.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional menyaring seluruh daftar permintaan rapat untuk dapat mengetahui kebutuhan requestor berdasarkan tanggal.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan inputan berupa tanggal yang merupakan range berupa tanggal.
Hasil yang Diharapkan	Sistem akan menampilkan hasil dari inputan aktor berupa *request* yang sesuai dengan tanggal.
Hasil Uji Coba	Valid

15. Mengunduh daftar absensi

- *Test case* kondisi awal mengunduh daftar absensi

Tabel 5.36 *Test Case* Kondisi Awal Mengunduh Daftar Absensi

Nama Kasus Uji	Uji kondisi awal mengunduh daftar absensi
Objek Uji	Uji halaman *home* receptionist.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk mengunduh daftar absensi dari salah satu *request* yang terdapat dalam daftar.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan klik pada tombol *download* untuk mengunduh file absensi.

Hasil yang Diharapkan	Sistem akan menampilkan lembar absensi dengan format .pdf pada *preview* didalam *browser* dan kemudian dapat dilakukan unduh dengan menekan tombol *download* pada halaman *preview* tersebut.
Hasil Uji Coba	Valid

16. Mengunggah daftar absesnsi

- *Test case* kondisi awal mengunduh daftar absensi

Tabel 5.37 *Test Case* Kondisi Awal Mengunduh Daftar Absensi

Nama Kasus Uji	Uji kondisi awal menggungah daftar absensi
Objek Uji	Uji halaman *home* receptionist.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk mengunggah daftar absensi dari salah satu *request* yang terdapat dalam daftar.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan klik pada tombol *upload* pada salah satu *request*.
Hasil yang Diharapkan	Sistem akan menampilkan hasil berupa *form* untuk mengunggah absensi yang telah di tanda tangani oleh peserta rapat. Kemudian klik tombol *save* untuk menyimpan hasil dari unggahan file absensi tersebut.
Hasil Uji Coba	Valid

17. Menutup ruangan rapat

- *Test case* kondisi awal menutup ruangan rapat

Tabel 5.38 *Test Case* Kondisi Awal Menutup Ruangan Rapat

Nama Kasus Uji	Uji kondisi awal menutup ruangan rapat
Objek Uji	Uji halaman *home* receptionist.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk menutup ruangan rapat dari salah satu *request meeting* yang terdapat dalam daftar.
Prosedur Uji	Aktor telah *login* dan aktor akan melakukan unggah file absensi untuk dapat menutup ruangan rapat tersebut.

Hasil yang Diharapkan	Sistem akan menampilkan hasil ruangan tersebut telah dapat digunakan kembali oleh requestor lainnya.
Hasil Uji Coba	Valid

18. Melihat daftar permintaan rapat yang diterima

- *Test case* kondisi awal melihat daftar permintaan rapat yang diterima

Tabel 5.39 *Test Case* Kondisi Awal Melihat Daftar Permintaan yang Diterima

Nama Kasus Uji	Uji kondisi awal melihat daftar permintaan rapat yang diterima
Objek Uji	Uji halaman *home* admin.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melihat daftar permintaan rapat yang diterima.
Prosedur Uji	Aktor telah *login* dan aktor akan melihat keseluruhan *request* requestor yang telah diterima oleh admin.
Hasil yang Diharapkan	Sistem akan menampilkan keseluruhan *request* dengan status *approve* oleh admin dan *request* tersebut dapat dilangsungkan.
Hasil Uji Coba	Valid

19. Melihat daftar permintaan rapat

- *Test case* kondisi awal melihat daftar permintaan rapat

Tabel 5.40 *Test Case* Kondisi Awal Melihat Daftar Permintaan Rapat

Nama Kasus Uji	Uji kondisi awal melihat daftar permintaan rapat
Objek Uji	Uji halaman *request room list*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melihat daftar *request* yang belum diterima oleh admin dan masuk kedalam *waiting list* sistem.
Prosedur Uji	Aktor telah *login* dan aktor akan melihat keseluruhan *request* yang belum diterima oleh admin dan sedang menunggu *accept* oleh admin.
Hasil yang Diharapkan	Sistem akan menampilkan daftar keseluruhan *request* yang belum mendapat persestujan oleh admin dan termasuk kedalam *request* didalam *waiting list* sistem.

Hasil Uji Coba	Valid

20. Melihat daftar permintaan rapat yang dibuat

- *Test case* kondisi awal melihat daftar permintaan rapat yang dibuat

Tabel 5.41 *Test Case* Kondisi Awal Melihat Daftar Permintaan yang Dibuat

Nama Kasus Uji	Uji kondisi awal melihat daftar permintaan rapat yang dibuat
Objek Uji	Uji halaman *home* requestor.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melihat dafatar permintaan rapat yang dibuat.
Prosedur Uji	Aktor telah *login* dan aktor akan keseluruhan permintaan rapat yang telah dibuat.
Hasil yang Diharapkan	Sistem akan menampilkan daftar dari *request* yang telah dibuat baik dalam status telah diterima oleh admin maupun sebaliknya beserta dengan informasi dari masing – masing *request*.
Hasil Uji Coba	Valid

21. Melihat daftar pembatalan permintaan rapat

- *Test case* kondisi awal melihat daftar pembatalan permintaan rapat

Tabel 5.42 *Test Case* Kondisi Awal Melihat Daftar Pembtalan Permintaan Rapat

Nama Kasus Uji	Uji kondisi awal melihat daftar pembatalan permintaan rapat
Objek Uji	Uji halaman *cancel request list*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk melihat daftar pembatalan permintaan rapat.
Prosedur Uji	Aktor telah *login* dan aktor akan melihat keseluruhan *request* pembatalan yang telah dibuat.
Hasil yang Diharapkan	Sistem akan menampilkan keseluruhan *request* pembatalan *meeting* dengan status telah diterima oleh admin maupun sebaliknya dengan beserta informasi pada masing – masing *request*.
Hasil Uji Coba	Valid

22. Login

- *Test case* kondisi awal login

Tabel 5.43 *Test Case* Kondisi Awal Login

Nama Kasus Uji	Uji kondisi awal login
Objek Uji	Uji halaman *index*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk *login* kedalam sistem dengan masing – masing hak akses yang diberikan.
Prosedur Uji	Aktor memasukkan username dan password berdasarkan data yang terdaftar dalam database sistem.
Hasil yang Diharapkan	Sistem akan menampilkan halaman *index* yang berupa field username dan password yang dapat dimasukkan untuk masuk kedalam halaman *home* dari masing – masing aktor.
Hasil Uji Coba	Valid

- *Test case* kondisi alternatif login

Tabel 5.44 *Test Case* Kondisi Alternatif Login

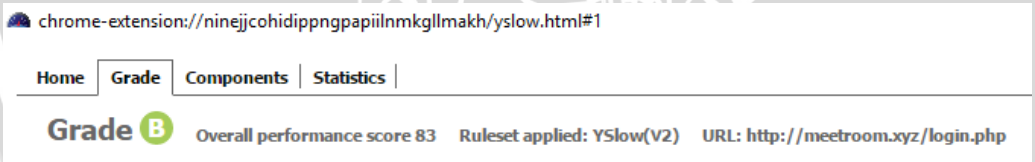
Nama Kasus Uji	Uji kondisi awal login
Objek Uji	Uji halaman *index*.
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa sistem dapat memenuhi kebutuhan fungsional untuk *login* kedalam sistem dengan masing – masing hak akses yang diberikan.
Prosedur Uji	Aktor memasukkan username dan password yang tidak terdaftar dalam sistem.
Hasil yang Diharapkan	Sistem akan menampilkan pesan *error* bahwa username atau password yang anda masukkan salah dan otomatis sistem akan melakukan *reload* halaman *index*.
Hasil Uji Coba	Valid

5.2.2 Pengujian Non-Fungsional

5.2.2.1 Pengujian Performa

Pengujiaan performa di desain untuk menguji performa dari *running time* sebuah perangkat lunak dengan beberpa konteks yang saling terintegrasi di dalamnya (Pressman, 2010). Tujuan dari pengujian performa ini adalah untuk mengetahui apakah perangkat lunak tersebut memilki kemampuan menjalankan program dengan satuan waktu tertentu. Aplikasi perangkat lunak dinilai baik ketika pengujian performanya memiliki waktu singkat untuk melakukan *load* sebuah halaman dengan beberapa konten yang ada didalam halaman aplikasi perangkat lunak tersebut. Pengujian performa sering dikaitkan dengan kebutuhan dari perangkat lunak yang mendukung aplikasi tersebut. Pada penelitian ini, peneliti memanfaatkan *tool* yang telah banyak digunakan oleh pihak develop yaitu Yslow. Yslow seperti yang sudah dijabarkan oleh peneliti sebelumnya, merupakan *tool* yang bekerja untuk menguji performa dari sebuah halaman *web* untuk mengetahui seberapa tinggi tingkat performa *web* tersebut dengan mengetahui nilai yang ditampilkan oleh Yslow.

Pengujian performa berkaitan pula dengan sebuah *test case* untuk memudahkan proses pengujian perangkat lunak. Pengujian dilakukan berdasarkan matrik uji dari Yslow yaitu *Minimum HTTP Request*. Matrik tersebut bertujuan untuk mengetahui repon dari sebuah halaman *web* berdasarkan konten yang ada didalamnya. *Test case* yang diuji yaitu *reload* halaman *login* sistem selama satu hari penuh dengan berdasarkan waktu pagi, siang, sore, dan malam. Akan dilakukan pengujian berdasarkan alamat dari *web* tersebut. Pengujian performa *test case* halaman *login* dapat dilihat pada gambar 5.19 berikut.



Gambar 5.19 Grade Yslow Keseluruhan Aplikasi

Berdasarkan pada gambar diatas hasil pengujian pada *test case load* halaman *login* dengan menggunakan *tool* Yslow. Dapat dilihat hasil akhir pengujian berdasarkan keseluruhan matrik uji Yslow mendapat grade B. Grade pada *tool* ini menujkkan tingkat performa yang cukup tinggi ketika proses *load* berlangsung dengan beberapa *environment* yang terlibat didalamnya sebagai faktor yang juga mempengaruhi hasil akhir pengujian. Matrik – matrik uji dari *tool* Yslow pun memiliki grade yang bertujuan untuk memudahkan pihak develop untuk melakukan *check* kesalahan hingga mengubah struktur dari aplikasi berdasarkan penilaian matrik uji Yslow serta *comment* yang tersedia secara otomatis setelah proses pengujian dilakukan. Sehingga pihak develop dapat lebih spesifik dalam melakukan dua hal tersebut.

Berdasarkan *test case* yang telah dirancang, peneliti hanya mengambil satu matrik uji yang digunakan sebagai pembahasan dalam pengujian performa.

Peneliti mendeskripsikan matrik *minimum HTTP request* sebagai matrik utama dalam pengujian performa. Pada pengujian menggunakan Yslow mendapat grade C. Grade tersebut dinilai cukup karena memiliki hasil yang relevan dengan konten yang terdapat didalamnya. Hasil pengujian pada matrik tersebut terdapat pada gambar 5.20 berikut.

Gambar 5.20 Grade Yslow pada Matrik *Minimum HTTP Request*

BAB 6 ANALISA DAN PEMBAHASAN HASIL

Analisa dan pembahasan hasil dilakukan bertujuan untuk mendapatkan kesimpulan dari hasil pengujian yang dilakukan oleh sistem, meliputi pengujian unit (*White Box*), pengujian integrasi, pengujian validasi dan pengujian performa. Proses yang dilakukan pada analisis dan pembahasan hasil dilakukan berdsarkan pada pengujian yang terdapat didalam sistem, yaitu analisa dan pembahasan hasil pengujian unit (*White Box)*, analisa dan pembahasan hasil pengujian fungsional, analisa dan pembahasan pengujian validasi dan analisa dan pembahasan hasil pengujian performa.

6.1 Analisa dan Pembahasan Hasil Pengujian Unit

Berdasarkan pada kasus uji yang dilaksanakan sesuai dengan prosedur pengujian basis path yang telah dijabarkan dalam sub pokok bahasan 5.2.1.1 pada algoritma *select time* dan algoritma *edit request* bahwa seluruh kasus uji memiliki hasil yang telah sesuai dengan kadidah yang berlaku serta bernilai valid. Pengujian unit dilakukan dengan menggunakan metode *cyclometic complexity* yang merupakan standart pengukuran tingkat kompleksitas dari sebuah fitur di dalam suatu aplikasi perangkat lunak. Pengukuran fitur dengan menggunakan *cyclometic complexity* ditujukan untuk mengetahui seberapa tinggi nilai yang dihasilkan agar pihak develop dapat mengetahui tingkat kerumitan pada proses pengembangan selanjutnya. Pada *sample* yang diukur masing – masing mendapatkan nilai diatas lima yang merupakan nilai yang tinggi untuk sebuah fitur dalam suatu perangkat lunak. Nilai tersebut menjadi salah satu faktor dari beberapa faktor utama untuk dapat dilakukan pengembangan lanjut tanpa melakukan perombakan besar dalam proses pengerjaannya. Dalam pengujian pada aplikasi manajemen ruang *meeting* didapatkan hasil yang cukup baik dan memenuhi standart dari pengembangan perangkat lunak. Sehingga kedepannya pihak develop tidak merasa kesulitan dalam hal pengembangan kembali aplikasi manajemen ruang *meeting*.

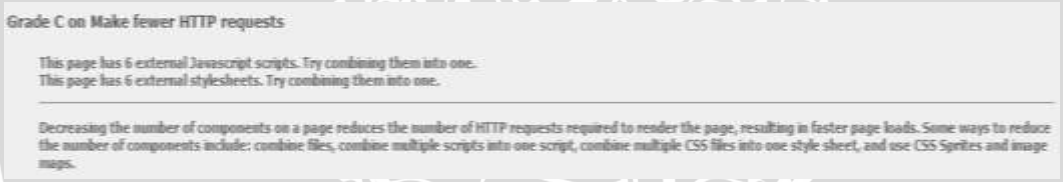
6.2 Analisa dan Pembahasan Hasil Pengujian Validasi

Berdasarkan pada kasus uji yang dilaksanakan sesuai dengan prosedur pengujian fungsional yang telah dijabarkan dalam sub pokok bahasan 5.2.1.3 maka didapatkan hasil yang valid berdasarkan pemenuhan kebutuhan dan kebutuhan yang telah terdefinisi. Analisa pada kasus uji tersebut dilakukan berdasarkan pada diagram *use case* skenario yang telah terdefinisi. Masing – masing *use case* skenario, memiliki kondisi dimana terbagi menjadi dua yaitu kondisi awal dan kondisi alternatif. Secara singkat pada kondisi awal menggambarkan aplikasi berjalan sesuai dengan yang diharapkan tidak adanya kekurangan baik dalam inputan maupun dalam proses pengolahan data tersebut hingga didapatkan hasil akhir. Sebaliknya pada kondisi alternatif yang menggambarkan aplikasi berjalan tidak sesuai dengan yang diharapkan sehingga aplikasi memaksakan melakukan proses pengolahan data inputan untuk didaptkan hasil akhir meskipun tidak sesuai dengan yang diharapkan. Kedua kondisi tersebut menjadikan prosedur pengujian

validasi pada aplikasi manajemen ruang *meeting*. Secara singkat prosedur yang dilakukan pada pengujian validasi ini, dengan melihat apakah data inputan yang diberikan pada kondisi awal sesuai dengan hasil yang diharapkan atau sebaliknya pada kondisi alternatif dapat menghasilkan sebuah peringatan berupa teks yang bertujuan untuk memberikan informasi bahwa inputan tersebut tidak sesuai dengan yang di definisikan. Berdasarkan pengujian validasi yang dilakukan hasil akhir valid pada masing – masing *use case* skenario.

6.3 Analisa dan Pembahasan Hasil Pengujian Performa

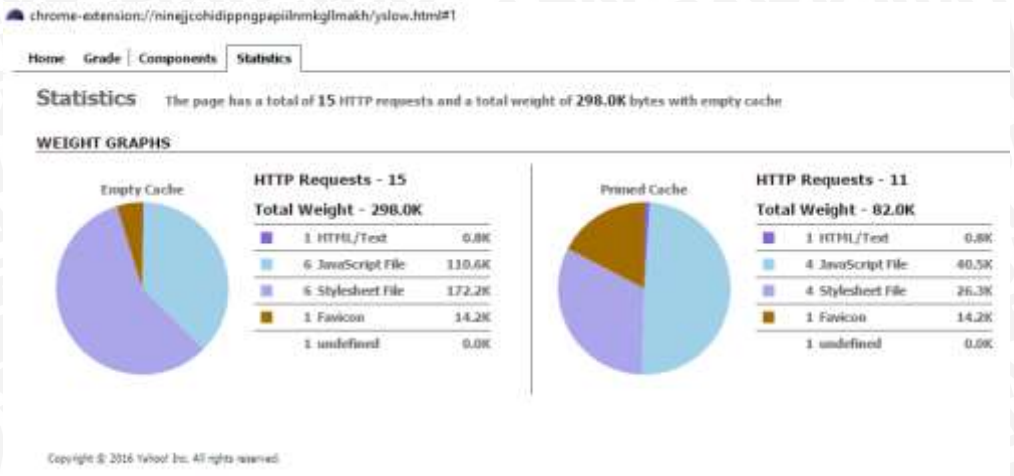
Pembahasan terkait dengan hasil pengujian performa menggunkana *tool* Yslow akan dijabarkan satu persatu berdasarkan *test case* yang telah di identifikasikan. Berdasarkan hasil pengujian terhadap matrik uji *minimum HTTP request* pada halaman *login* mendapat grade C. Grade C pada matrik tersebut bernilai antara 70 <= S < 80 dengan rincian terdapat 10 – 13 Javascript atau 8 – 9 CSS atau 13 – 16 gambar CSS. Grade tersebut menentukan seberapa baik aplikasi tersebut setelah dilakukan *load* beberapa saat. Dalam *tool* Yslow ini menjelaskan bagaimana aplikasi tersebut menjadi lebih baik setelah dilakukan pengujian. Tidak hanya memberikan grade, Yslow juga memberikan evaluasi terhadap hasil pengujian berdasarkan matik ujinya. Keterangan tersebut dapat dilihat pada gambar 6.1 berikut.



Gambar 6.1 Grade Yslow pada Matrik *Minimum HTTP Request*

Berdasarkan keterangan diatas dan nilai yang didapatkan tersebut dapat diajabarkan satu per satu. Keterangan pertama "*This page has 6 external javascript script, try combining them into one*" dapat dideskripsikan bahwa didalam satu halaman tersebut dalam satu kali *load* terdeteksi enam buah file javascript yang terdapat didalamnya dan sebaiknya untuk menggambungkan file javascript tersebut kedalam satu buah file yang dapat dipanggil oleh kelas yang mengakses halaman teresbut. Pada keterangan kedua pun tidak berbeda dengan keterangan pertama, adapun keterangan kedua "*This page has 6 external stylesheet Try combining them into one*". Keterangan kedua menggambarkan terdapat enam buah file style (CSS) yang tidak dijadikan menjadi satu buah kelas sehingga memperngaruhi proses *load* halaman tersebut. Dua hal tersebut masuk kedalam salah satu penilaian parameter grade yang didapatkan berdasarkan penjelasan grade di atas.

Berikutnya hasil dari pengujian di representasikan dalam bentuk diagram statistic. Didalam Yslow terdapat dua buah diagram yang masing-masing memilki makna yang berbeda. Hasil dari pengujian tersebut dapat dilihat pada gambar 6.2 berikut



Gambar 6.2 Statistik Hasil Pengujian Yslow

Pada gambar diatas dapat dilihat terdapat dua buah diagram yang masing-masing memberikan nilai yang berbeda. Diagram statis yang pertama adalah "*empty cache*" terdapat lima buah parameter yang masing-masing berpengaruh dalam pengujian performa. Pada aplikasi manajemen ruang *meeting* didalam diagram "*empty cache*" diidentifikasikan bahwa nilai total *weigth* adalah 298.0 K. Angka tersebut terbilang besar untuk sebuah *request HTTP* dalam sebuah aplikasi *web*. Karena browser harus melakukan pengambilan konten beserta isinya kedalam file *cache* yang kemudian akan diakses atau *load* halaman. Berbeda dengan diagram kedua yaitu "*primed cache*" yang memiliki nilai yang lebih rendah yaitu 82.0 K jauh dibandingkan dengan diagram sebelumnya.

Namun dengan adanya dua diagram tersebut dapat dilihat berapa banyak permintaan HTTP yang harus dipesan untuk melakukan satu kali *load* halaman *web.* Inti pembahasan daripada pengujian performa ini adalah aplikasi manajemen ruang *meeting* secara keseluruhan mendapat grade B yang artinya cukup baik untuk dipublikasikan kepada konsumen. Dengan grade tersebut didapatkan nilai yang berkisar antara 80 <= S < 90. Sedangkan pada grade dari matrik *minimum HTTP request* mendapat grade C dikarenakan teradapat 12 file terpisah yang seharusnya dapat di kombinasikan dalam satu buah kelas. Dan total *weight* pada satu kali *load* halaman *web* adalah 289.0 K dengan *request* HTTP sebanyak 15.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan pada hasil analisis dan perancangan sistem, implementasi dan pengujian sistem, dan analisa dan pembahasan hasil yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Proses *gathering* yang dilakukan pada analisa kebutuhan aplikasi berupa wawancara kepada pihak *stakeholder*. Hasil dari wawancara tersebut berupa proses *improvement* terhadap fitur pemesanan ruang *meeting* yang disesuaikan dengan kebutuhan. Berdasarkan hasil *improvement* pada kasus pencegahan terjadinya *crash,* pada fitur pemesanan ruang *meeting* dilakukan pemberian status dari ketersediaan ruangan yang ada. Status "*Booking*" untuk ruangan yang tersedia dan Status "*Booked*" untuk ruangan yang telah dipesan.
2. Pengembangan dengan model proses *waterfall* dimulai dari tahap penentuan kebutuhan tambahan di dalam fitur yang dilakukan pada proses wawancara selanjutnya pada proses design dilakukan perancangan dari kebutuhan yang didapatkan untuk didapatkan hasil *improvement* yang sesuai antara aplikasi dengan kebutuhan baru tersebut. Selanjutnya pada tahap implementasi menerapan hasil dari design berupa rancangan keseluruhan sistem dan terkahir dilakukan proses *testing* untuk didapatkan validasi dari kebutuhan sistem serta dengan pengujian performa sebagai penunjang dari kebutuhan non-fungsional didalam sistem.
3. Hasil daripada pengujian fungsional yang melibatkan fitur secara keseluruhan mendapatkan nilai yang baik untuk tingkat kompleksitasnya. Serta validasi 100% didapatkan pada seluruh kebutuhan yang telah terdefinisi.
4. Selain pengembangan yang dilakukan peneliti, dilakukan pengujian pula pada sisi performa dengan menggunakan *tool* Yslow. Pengujian dilakukan untuk melihat apakah aplikasi manajemen ruang *meeting* dapat memenuhi kaidah dalam penggunaan secara umum pada pihak stakeholder. Evaluasi secara keseluruhan untuk aplikasi manajemen ruang *meeting* mendapatkan grade B yang merupakan grade dengan nilai diatara 80 hingga 85 untuk seluruh matrik uji dalam Yslow.

7.2 Saran

1. Berdasarkan pada hasil pengujian performa pada matrik *minimum HTTP requset*, untuk kedepannya agar aplikasi manajemen ruang *meeting* dapat dilakukan pengujian pada matrik lainnya dengan harapan dapat meningkatkan performa dari aplikasi tersebut.
2. Aplikasi manajemen ruang *meeting* agar dapat dikembangakan dalam bentuk aplikasi *mobile* agar lebih memudahkan pengguna dalam proses *request* ruang *meeting*.

DAFTAR PUSTAKA

Akhdiyat, S., 2012. *Materi Umum Tentang Website*. Tersedia di: <http://masulum.com> [Diakses 3 Januari 2016].

Sutcliffe Alistair, 2010. *Scenario-Based Requirement Analysis*.[pdf]. Tersedia di: < http://ftp.informatik.rwth-aachen.de/ftp/pub/packages/CREWS/CREWS-98-07.pdf> [Diakses 15 Agustus 2016].

Cohen, M. A., Rogelberg, S. G., Allen, J. A., & Luong, A. (2011).Meeting design characteristics and attendee perceptions of staff/team meeting quality.Group Dynamics: Theory, Research and Practice, 15,90–104

Dharwiyanti S, W., 2003. *Pengantar Unified Modeling Language*. [pdf]. Tersedia di: <http:// ilmukompter.com/pengantarUML.pdf> [Diakses 10 Februari 2016]

Dickinger Asrtid, Stangl Brigitte., 2011. *Website Performance And Behavioral Consequences: A Formative Measurement Approach*. [pdf]. Tersedia di: < http://www.sciencedirect.com/science/article/pii/S0148296311003262> [Diakses 11 Maret 2016].

Fevzi Belli., Christof J. Budnik., Axel Hollmann., Tugkan Tugular., W. Eric Wong, 2016. "*Model-based mutation testing-Approach and case studies"*. [pdf]. Science of Computer Programming. Tersedia di: < http://ac.els-cdn.com/S0167642316000137/1-s2.0-S0167642316000137-main.pdf > [Diakses 26 Maret 2016]

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

Indira.N, 2013. "*Analysis of Yslow Performance Test tool & Emergences on Web Page Data Extraction"*. [pdf]. International Journal of Computer Science and Mobile Computing. Vol.2, Issue.5, pg.317-322. Tersedia di: < http://ijcsmc.com/docs/papers/May2013/V2I5201392.pdf> [Diakses 25 Februari 2016]

Carlos Farre Ahantbansod, Scott Barber, Dennisrea, J.D. Meier, "Performance Testing Guidance for Web Applications", World Academy of Science, Engineering and Technology, 2005. [Diakses 19 Februari 2016]

Jennifer L. Geimer., Desmond J. Leach., Justin A. DeSimone., Steven G. Rogelberg., Peter B. Warr, 2015. "*Meeting at Work: Perceived effectiveness and recommended improvement"*. [pdf]. Jurnal of Bussiness Research. Tersedia di: < http://ac.els-cdn.com/S0148296315000879/1-s2.0-S0148296315000879-main.pdf> [Diakses 26 Maret 2016]

Kumar Naresh, Z.A., 2013. *ISSN: 2331-2307 Vol.3 Issue 1, Evolving a New Software Development Life Cycle Model SDLC-2013 with Client Satisfaction,* Tersedia melalui: International Journal of Soft Computing and Engineering (IJSCE) <http://ijsce.org> [Diakses 09 Februari 2016]

Manulang.M, 1983. Dasar-dasar Manajemen, hal 15-16, Ghalia Indnesia Jakarta

Mark Alexander., Michael Thomas, 2015. "*Service life prediction and performance testing – current developments and partical appilcation"*. [pdf]. Cement and Conrete Research. Tersedia di: < http://ac.els-cdn.com/S0008884615001441/1-s2.0-S0008884615001441-main.pdf > [Diakses 26 Maret 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]

O'Brein, James A., (2005), "Pengantar Sistem Informasi", Penerbit : Salemba 4, Jakarta.

Oey Liang Lee, 1985. Pengertian Manajemen, no.1 p.15, Balai Pustaka Administrasi, Universitas Gajah Mada

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

Roger S.Pressman ,Ph,D., 2002. Software Engineering (A Practitioner's Approach). *5th*. 47-50, Boston: Burr Ridge.

Ruri Sunary, 2010. Teknologi Informasi. Tersedia di: <http://elib.unikom.ac.id> [Diakses 3 Januari 2016]

Sabale Rajendra Ganpatrao, Dr Dani., 2012. *ISSN: 2550-3021 Vol.2 Issue 7, Comparative Study of Prototype Model For Software Engginering With Sistem Development Life Cycle,* PP 21-24, Tersedia melalui: IOSR Journal of Engginering (IOSRJEN) <http://iosrjen.org> [Diakses 09 Februari 2016]

Sadaf Ateeq,Mr M., 2014. *Comparison Of Various Sdlc Models*. Tersedia di: <http://www.gjms.co.in/ > [Diakses 09 Februari 2016]

Seffa.A, Donyaee.M, 2006. *Usability Measurment and Metrics: A Consolidatde model.* Tersedia di: <http://www.psychology.concordia.ca/fac/kline/Library/sdkh06.pdf> [Diakses 10 Maret 2016]

Universitas Sriwijaya , 2010, Pengantar Sistem Informasi. Tersedia di: < http://www.unsri.ac.id/upload/arsip/I_PengatarSI.pdf> [Diaskses 10 Maret 2016].

LAMPIRAN A

INTERVIEWEE	: KARYAWAN
NAMA / KODE	: AGUS PURYONO / P
JENIS KELAMIN	: LAKI – LAKI
USIA	: 30 TAHUN
DEPARTMENT	: GENERAL AFFAIR
TEMPAT WAWANCARA	: RUANG MEETING PT. TELKOMSEL Tbk.

Interviewer	:	Selamat pagi pak, kami dari Universitas Brawijaya ingin melakukan penelitan terkait dengan pengembangan aplikasi manajemen ruang *meeting*. Apa sebelumnya sudah ada aplikasi yang serupa pak?
P	:	Hmm,,, belum ada sih mas. Kita kontrolnya ya gini gini aja manual malah cenderung rebutan buat pake ruangannya.
Interviewer	:	Begitu ya pak? Bagaimana dengan peserta rapatnya pak? Apa ada batasan dari masing – masing ruang?
P	:	Belum ada mas kalo batesan ya secukupnya dihitung sendiri apa cukup di ruangan yang dipesen. Malah kita kadang pake meja kerja kalo ruangan rapatnya dipake semua.
Interviewer	:	Baik pak kami catat sebagai daftar kebutuhannya. Untuk penyebaran informasi akan diadakan rapat dalam bentuk apa pak?
P	:	Biasanya sms mas personal ke group kadang juga pake sosmed, kadang juga dari mulut ke mulut. Ya gak ada aturan mas.
Interviewer	:	Seberapa sering pak kira kira kegiatan dalam satu hari?
P	:	Gak tentu, kadang bisa seharian ada rapat kadang juga rapat setengah hari. Tapi yang sering rapat diskusi bisa tiga sampai 4 kali rapat mas.
Interviewer		Bagaimana dengan peralatan atau kelengkapan rapat pak?
P		Kalo peralatan standart aja mas lcd, proyektor, papan tulis sama sound itu ada disetiap ruang *meeting* dikantor kita.
Interviewer		Selanjtunya untuk *human resource* yang terlibat biasanya siapa aja pak? Atau malah cenderung dari si pengguna ruang *meeting* tersebut?

P	Untuk .HR sendiri biasanya minta bantuan receptionist buat ngedata siapa yang lagi rapat di ruangan tersebut kadang juga menyiapin beberapa keperluan rapat.
Interviewer	Keperluan rapat apa pak yang biasanya disiapkan?
P	Keperluan selain dari internal ruangan biasanya snack aja sih mas. Biasanya buat rapat yang datengin petinggi petinggi gitu rapatnya pasti punya durasi yang agak panjang.
Interviewer	Baik pak. Kemudian untuk kegiatan rapat sendiri butuh pengkondisian peserta dalam bentuk presensi pak?
P	Hmmm boleh juga mas idenya karena kadang kita lupa tadi sama siapa aja rapatnya, yang kenal ya kenal yang gak inget ya gak inget sering lupanya mas sangking banyaknya rapat yang lain juga.
Interviewer	Seperti itu ya pak? Tapi selama ini ada proses pencatatan gak pak terkait dengan rapat apa yg sedang berjalan?
P	Kalo notulensi pasti ada mas cuman ya bahasannya aja bukan presensinya.
Interviewer	Baik pak, kami catat kebutuhannya. Nah begini pak pengelolaan ruang *meeting* tersebut akan kami jadikan sebuah aplikasi yang lebih memudahkan dalam hal pemesanannya. Menurut bapak sendiri apa harapan bapak untuk aplikasi yang akan kami kembangkan?
P	Ya harapannya si simple mas menjawab dari kebutuhannya yang sudah kita diskusikan jadi lebih memudahkan temen temen atau bahkan saya sendiri untuk memesan ruangan.
Interviewer	Dari kebutuhan yang kami catat, kami mengambil beberapa kebutuhan yang menjadi fungsional dalam aplikasi yang kami kembangkan seperti pemesanan berdasarkan jam kerja yang ada selanjutnya untuk proses penyampaian informasi dilakukan dengan tekhnologi SMS Gateway kemudian adanya sistem presensi untuk kemudahan rekapan rapat. Untuk aktor sendiri kami definisikan tiga pak admin, requestor dan receptionist. Apa mungkin dari bapak ada masukan?
P	Cukup mas kalo dari saya soalnya bakalan kompleks itu aplikasinya dari rangkuman yang kamu buat udah keliahtan jadinya kayak gimana. Mungkin masukan aja mas ya dari saya untuk kelola pemesannya coba ada *waiting list* yang bertujuan untuk menampung *request* dari requestor di admin berarti kan, jadinya admin bisa tahu siapa yang pesen ruangan dan mungkin di aplikasinya admin bisa *approve* dari *request* yang dibuat supaya kelolanya lebih bagus.

Interviewer	Baik pak, mungkin cukup untuk diskusi seputar dengan kebutuhan dari kelola ruang *meeting* di kantor bapak. Dari saya cukup dari bapak mungkin ada tambahan?
P	Cukup deh mas itu udah banyak loh nanti malah pusing beresinnya.
Interviewer	Seimbang dengan waktu yang diberikan kok pak mudah – mudahan bisa cepat selesai amin.
P	Ya mas semangat semoga sukses.
Interviewer	Iya pak terima kasih.

LAMPIRAN B

Category	***Rules***
Content	- Make Fewer HTTP Request - Reduce DNS Lookup - Avoid URL Redirects - Make AJAX Cacheable - Reduce The Number of DOM Elements - Avoid HTTP 404 (not found) error
Cookie	- Reduce Cookie Size - Use Cookie-Free Domains
CSS	- Put CSS at Top - Avoid CSS Expressions - Make JavaScript and CSS External - Minify JavaScript and CSS - Remove Duplicate JavaScript and CSS - Avoid AlphaImageLoader Filter
Image	- Do Not Scale Image in HTML - Make Favicon Small and Cacheable
JavaScript	- Put JavaScript at Bottom
Server	- Use a Content Delivery Network (CDN) - Avoid Empty src or href - Add Expires Headers - Compress Components with Gzip - Configure Entity Tags (ETags) - Use GET for AJAX Request

LAMPIRAN C

Rule	Weight	Points	Configs	Score Computation	(A) 90 <= S <= 100	(B) 80 <= S < 90	(C) 70 <= S < 80	(D) 60 <= S < 70	(E) 50 <= S < 60	(F) 0 <= S < 50
Make fewer HTTP requests	8	js = 3 css = 4 css images = 3	max js = 3 max css = 2 max css images = 6	(N JS - 3) * 3 (N CSS - 2) * 4 (N CSS images - 6) * 3	0 to 6 JS OR 0 to 4 CSS OR 0 to 9 CSS images	7 to 9 JS OR 5 to 7 CSS OR 10 to 12 CSS images	10 to 13 JS OR 8 to 9 CSS OR 13 to 16 CSS images	14 to 16 JS OR 10 to 12 CSS OR 17 to 19 CSS images	17 to 19 JS OR 13 to 14 CSS OR 20 to 22 CSS images	>= 20 JS OR >= 15 CSS OR >= 23 CSS images
Use a CDN	6	10	patterns = CDN hostname RegExp patterns types = js, css, image, cssimage, flash, favicon	N RegExp mismatches * 10 (ignores /favicon.ico)	0 or 1 of any type	2 of any type	3 of any type	4 of any type	5 of any type	>= 6 of any type
Avoid empty src or href	30	100	-	N empty src <img> * 100 N empty src <script> * 100 N empty href <link rel="stylesheet"> * 100	0 empty src <img> AND 0 empty src <script> AND 0 empty href <link rel="stylesheet">	-	-	-	-	>= 1 empty src <img> OR >= 1 empty src <script> OR >= 1 empty href <link rel="stylesheet">
Add Expires headers	10	11	how far = 172800s (2 days) types = js, css, image, cssimage, flash, favicon	N (unexpired or expiring in < 2 days of any type) * 11	0 unexpired or expiring in < 2 days of any type	1 unexpired or expiring in < 2 days of any type	2 unexpired or expiring in < 2 days of any type	3 unexpired or expiring in < 2 days of any type	4 unexpired or expiring in < 2 days of any type	>= 5 unexpired or expiring in < 2 days of any type
Compress components with GZip	8	11	min file size = 500 bytes types = doc, iframe, xhr, js, css	N (uncompressed or file size < 500b of any type) * 11	0 uncompressed or file size < 500b of any type	1 uncompressed or file size < 500b of any type	2 uncompressed or file size < 500b of any type	3 uncompressed or file size < 500b of any type	4 uncompressed or file size < 500b of any type	>= 5 uncompressed or file size < 500b of any type
Put CSS at top	4	10	-	1 + N CSS link tag on BODY * 10	0 CSS link tag on BODY	1 CSS link tag on BODY	2 CSS link tag on BODY	3 CSS link tag on BODY	4 CSS link tag on BODY	>= 5 CSS link tag on BODY
Put JavaScript at bottom	4	5	-	N JS on HEAD * 5 ignores injected, defered and async JS	0 to 2 JS on HEAD	3 or 4 JS on HEAD	5 or 6 JS on HEAD	7 or 8 JS on HEAD	9 or 10 JS on HEAD	>= 11 JS on HEAD
Avoid CSS expressions	3	2	-	N expressions on CSS links or inline STYLE * 2	0 to 5 expressions on CSS or inline STYLE	6 to 10 expressions on CSS or inline STYLE	11 to 15 expressions on CSS or inline STYLE	16 to 20 expressions on CSS or inline STYLE	21 to 25 expressions on CSS or inline STYLE	>= 26 expressions on CSS or inline STYLE
Make JavaScript and CSS external	4	n/a	-	none	-	-	-	-	-	-
Reduce DNS lookups	3	5	max domains = 4	N domains > 4 AND (N domains - 4) * 5	0 to 6 domains	7 or 8 domains	9 or 10 domains	11 or 12 domains	13 or 14 domains	>= 15 domains
Minify JavaScript and CSS	4	10	types = js, css	N (unminified JS or CSS external or inline) * 10	0 or 1 unminified component	2 unminified components	3 unminified components	4 unminified components	5 unminified components	>= 6 unminified components
Avoid URL redirects	4	10	-	N redirects * 10	0 or 1 redirect	2 redirects	3 redirects	4 redirects	5 redirects	>= 6 redirects
Remove duplicate JavaScript and CSS	4	5	types = js, css	N (duplicated JS or CSS) * 5	0 to 2 duplicated JS or CSS	3 or 4 duplicated JS or CSS	5 or 6 duplicated JS or CSS	7 or 8 duplicated JS or CSS	9 or 10 duplicated JS or CSS	>= 11 duplicated JS or CSS
Configure ETags	2	11	types = js, css, image, cssimage, flash, favicon	N bad etag of any type * 11	0 bad etag of any type	1 bad etag of any type	2 bad etags of any type	3 bad etags of any type	4 bad etags of any type	>= 5 bad etags of any type
Make AJAX cacheable	4	5	min cache time = 3600s	N (uncached or expiring in < 3600s) XHR * 5	0 to 2 uncacheable XHR	3 or 4 uncacheable XHR	5 or 6 uncacheable XHR	7 or 8 uncacheable XHR	9 or 10 uncacheable XHR	>= 11 uncacheable XHR
Use GET for AJAX requests	3	5	-	N XHRs not using GET * 5	0 to 2 XHRs not using GET	3 or 4 XHRs not using GET	5 or 6 XHRs not using GET	7 or 8 XHRs not using GET	9 or 10 XHRs not using GET	>= 11 XHRs not using GET
Reduce the number of DOM elements	3	10	range = 250 max dom = 900	N DOM elements > 900 AND (N DOM elements - 900) / 250 * 10	0 to 1150 DOM elements	1151 to 1400 DOM elements	1401 to 1650 DOM elements	1651 to 1900 DOM elements	1901 to 2150 DOM elements	>= 2151 DOM elements
Avoid HTTP 404 (Not Found) error	4	5	types = js, css, image, cssimage, flash, favicon, xhr	N 404 * 5	0 to 2 404	3 or 4 404	5 or 6 404	7 or 8 404	9 or 10 404	>= 11 404
Reduce cookie size	3	10	max cookie size = 1000	cookie size > 1000 AND 1 + (cookie size / 1000) * 10	0 to 1000 bytes cookies	1001 to 1900 bytes cookies	1901 to 2900 bytes cookies	2901 to 3900 bytes cookies	3901 to 4900 bytes cookies	>= 4901 bytes cookies
Use cookie-free domains	3	5	types = js, css, image, cssimage, flash, favicon	N (components of any type with cookies of any size) * 5 ignores /favicon.ico	0 to 2 components with cookie	3 or 4 components with cookie	5 or 6 components with cookie	7 or 8 components with cookie	9 or 10 components with cookie	>= 11 components with cookie
Avoid AlphaImageLoader filter	4	5	half points = 2	N alpha filters * 5 + M _hack alpha filters * 2	0 to 2 alpha filters OR 0 to 5 _hack alpha filters	3 or 4 alpha filters OR 6 to 10 _hack alpha filters	5 or 6 alpha filters OR 11 to 15 _hack alpha filters	7 or 8 alpha filters OR 16 to 20 _hack alpha filters	9 or 10 alpha filters OR 21 to 25 _hack alpha filters	>= 11 alpha filters OR >= 26 _hack alpha filters
Do not scale images in HTML	3	5	-	N (images scaled down width or height) * 5	0 to 2 scaled down images	3 or 4 scaled down images	5 or 6 scaled down images	7 or 8 scaled down images	9 or 10 scaled down images	>= 11 scaled down images
Make favicon small and cacheable	2	5	size = 2000b min cache time = 3600s	Favicon 404 not found = 5 Favicon size > 2000b = 5 No favicon expiration or expiration < 3600s = 5	Favicon is found AND Any favicon size Any expiration or not OR Favicon is not found AND Any expiration or not (usually not)	-	-	-	-	-