repository.ub.ac.id

PENGEMBANGAN SISTEM INFORMASI MUSYAWARAH DENGAN METODE ITERATIF

(Studi Kasus : Masjid Ibnu Sina Jl. Veteran, Malang)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Rizky Novriansyah
NIM: 155150401111089

PROGRAM STUDI SISTEM INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PENGESAHAN

PENGEMBANGAN SISTEM INFORMASI MUSYAWARAH DENGAN METODE ITERATIF
(STUDI KASUS : MASJID IBNU SINA JL. VETERAN, MALANG)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rizky Novriansyah
NIM: 155150401111089

Skripsi ini telah diuji dan dinyatakan lulus pada
22 Juli 2019
Telah diperiksa dan disetujui oleh:

Pembimbing I	Pembimbing II
Ismiarta Aknuranda, S.T., M.Sc., Ph.D.	Welly Purnomo, S.T., M.Kom.
NIK: 2010067407191001	NIK: 2017088101171001

Mengetahui
Ketua Jurusan Sistem Informasi

Dr. Eng. Herman Tolle, S.T., M.T.
NIP: 197408232000121001

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

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 referensi.

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, 7 Juli 2019

Rizky Novriansyah

NIM: 155150401111089

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PRAKATA

Alhamdulillah, segala puji syukur Penulis panjatkan kepada Allah SWT yang telah melimpahkan rahmat, karunia, serta hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul: "**Pengembangan Sistem Informasi Musyawarah Dengan Metode Iteratif (Studi Kasus : Masjid Ibnu Sina Jl. Veteran Malang)".**

Penulis mengucapkan terimakasih yang sebesar-besarnya kepada pihak-pihak yang telah memberikan bantuan selama pengerjaan skripsi ini dari awal hingga terselesaikannya laporan skripsi ini, diantaranya:

1. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D., Bapak Ir. Heru Nurwasito, M.Kom., Bapak Suprapto, S.T., Bapak Edy Santoso, S.Si, M.Kom. selaku Dekan, Wakil Dekan I, Wakil Dekan II, Wakil Dekan III Fakultas Ilmu Komputer Universitas Brawijaya Malang.
2. Bapak Ismiarta Aknuranda, S.T., M.Sc., Ph.D selaku dosen pembimbing I yang telah memberikan kesempatan, saran, motivasi dan waktunya untuk membimbing penulis. Serta memberikan contoh sebagai pribadi yang sabar, solutif, tekun, dan professional dalam memberikan bimbingan kepada penulis.
3. Welly Purnomo, S.T., M.Kom. selaku dosen pembimbing II yang telah memberikan kesempatan, saran, dan waktunya untuk membimbing penulis. Serta memberikan contoh sebagai pribadi yang kritis dan professional dalam memberikan bimbingan kepada penulis.
4. Bapak Dr. Eng., Herman Tolle, S.T, M.T. selaku Ketua Jurusan Sistem Informasi.
5. Bapak Yusi Tyroni Mursityo, S.Kom., M.AB. selaku Ketua Program Studi Sistem Informasi.
6. Seluruh dosen Fakultas Ilmu Komputer yang telah mendidik dan memberikan ilmu serta wawasannya selama menempuh pendidikan dan menyelesaikan skripsi ini.
7. Seluruh civitas akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberikan bantuan serta dukungan kepada penulis selama menempuh pendidikan dan menyelesaikan skripsi ini.
8. Bapak dan Ibu serta Kakak dan Adik yang tak henti mendukung dan memberikan semangat, nasehat, perhatian dan kesabarannya dalam membesarkan dan mendidik penulis.
9. Nafiani dan Mohammad Arda Dwi A., teman seperjuangan di setiap mata kuliah, setiap tugas, setiap proyek dan di setiap hal apapun itu yang tidak bisa penulis sebutkan satu persatu.

UNIVERSITAS BRAWIJAYA

10. Seluruh anggota Takmir dan Remaja Masjid Ibnu Sina yang telah memberikan ilmu serta wawasannya selama menjalani penelitian.
11. Teman-teman Kepengurusan Lembaga KBMSI periode 4 dan periode 5 yang turut serta membantu dan mendukung penulis dalam menyelesaikan skripsi, terima kasih atas pengalaman dan kekeluargaan yang diberikan selama ini.
12. Teman-teman Sistem Informasi Fakultas Ilmu Komputer angkatan 2015 atas seluruh bantuan, kerjasama, suka dan duka selama masa perkulihan di Fakultas Ilmu Komputer Universitas Brawijaya.
13. Semua pihak yang tidak dapat disebutkan satu persatu atas segala bantuan dan dukungannya baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini.

Penulis berharap skripsi yang telah disusun ini bisa memberikan sumbangsih untuk menambah pengetahuan para pembaca, dan akhir kata, dalam rangka perbaikan selanjutnya, penulis akan terbuka terhadap saran dan masukan dari semua pihak karena penulis menyadari skripsi yang telah disusun ini masih memiliki kekurangan.

Malang, 7 Juli 2019

Penulis

rizkynovriansyah82.rn@gmail.com

repository.ub.ac.id

ABSTRAK

Rizky Novriansyah, Pengembangan Sistem Informasi Musyawarah Dengan Metode Iteratif (Studi Kasus : Masjid Ibnu Sina Jl. Veteran, Malang)

Pembimbing: Ismiarta Aknuranda, S.T., M.Sc., Ph.D., dan Welly Purnomo, S.T., M.Kom.

Masjid Ibnu Sina merupakan salah satu masjid yang berada di kota Malang yang memiliki Takmir dan Remaja Masjid untuk menghidupkan dan menjalankan fungsi masjid sebagai wadah ibadah bagi umat muslim. Dengan banyaknya kegiatan yang diadakan oleh masjid, masjid sering melakukan kegiatan musyawarah untuk memaksimalkan kegiatan ibadah kerohanian untuk melayani masyarakat khususnya umat islam. Namun, terdapat beberapa permasalahan yang terjadi saat melakukan musyawarah masjid maupun kontrol pekerjaan musyawarah. Sering kali musyawarah terlupa dan sulit untuk melakukan peninjauan kembali informasi notulensi tertentu dari media yang disimpan oleh takmir masjid, yaitu buku dan sosial media. Hal tersebut menjadi terhambatnya penentuan keputusan yang musyawarah sebelumnya sudah disepakati. Hal ini juga menyebabkan multi-presepsi bagi setiap anggota musyawarah terhadap keputusan yang diberikan. Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk membangun sistem informasi musyawarah yang dapat membantu mengelola notulensi musyawarah, pekerjaan musyawarah, dan anggota Takmir dan Remaja Masjid. Sehingga dalam pengembangannya, sistem ini dibangun menggunakan Java Spring Boot untuk memudahkan pengembangan dengan menerapkan prinsip berorientasi objek. Metode pengembangan sistem yang digunakan pada penelitian ini yaitu menggunakan metode iteratif. Kemudian sistem diuji menggunakan metode validasi dan metode pengujian kompabilitas. Hasil pengujian validasi mendapatkan hasil yang valid dan kompatbilitas menunjukan sistem memiliki skor kompatibilitas yang baik pada semua peramban web.

Kata kunci: musyawarah, masjid, iteratif.

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Rizky Novriansyah, Development of Meeting Information System with Iterative Method (Case Study: Ibnu Sina Mosque, Veteran, Malang)

Supervisors*: *Ismiarta Aknuranda, S.T., M.Sc., Ph.D., and Welly Purnomo, S.T., M.Kom.

Ibnu Sina is one of the mosques that has Takmir and Remaja Masjid to revive and carry out the function of the mosque as a place for Muslims to pray. With many activities held by mosques, Takmir and Remaja Masjid often conduct meeting activities to maximize the worship activities to serve people. However, there are problems that occur when conducting meeting and control all of the mosque work. Often times meeting are forgotten and it is difficult to review a certain information from the media stored by takmir mosques, it is books and social media. This has hampered the determination of the decisions that had been agreed. Thus causing multi-perception for each member of the deliberation on the decisions given. This study aims to build an information system that can help manage meeting information. In its development, this system was built using the Java Spring Boot to facilitate development by applying the principle of the module to the feature. The system development method used in this study is an iterative method. Then the system is tested using validation methods and compatibility testing methods. The result of validation testing get valid and compatibility testing shows that system have good compatibility.

Keywords: meeting, mosque, iterative.

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

DAFTAR GAMBAR

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

BAB 1 PENDAHULUAN

1.1 Latar belakang

Masjid sudah menjadi pusat untuk melakukan peningkatan keimanan bagi setiap umat islam. Masjid berguna sebagai tempat berkumpulnya umat islam dan juga tempat perayaan dan ritual bagi umat islam (Sanusi et al., 2015). Bagi semua umat islam merupakan hal yang penting untuk membangun masjid tetapi tidak hanya untuk menjalankan ibadah salat, namun juga untuk melakukan pengabdian keagamaan. Selain tujuan sosial dan keagamaan, masjid juga menunjukan kebesaran umat islam. Untuk menjalankan kegiatan sosial dan keagamaan, diperlukannya pengurus masjid atau takmir agar dapat terlaksananya kegiatan tersebut. Takmir masjid memiliki peran untuk memaksimalkan potensi positif yang dapat diberikan oleh masjid bagi masyarakat sekitarnya. Takmir masjid dapat membantu masyarakat sekitar dengan mengadakan kegiatan kerohanian seperti salat berjamaah, menjalankan agenda bulan suci Ramadhan, buka puasa Senin-Kamis, membantu keluarga yang tidak mampu disekitar masjid, membuka pengajian rutin dan pengkajian al-qur'an. Shalat atau kegiatan ibadah di masjid harus yang terasa nyaman agar jamaah mendapat ketenangan dan kedamaian saat berdoa kepada yang maha kuasa (Abdullah, Majid and Othman, 2016). Kegiatan kerohanian tersebut merupakan salah satu alasan pentingnya kepengurusan takmir bagi kepengurusan Masjid Ibnu Sina Malang. Takmir Masjid Ibnu Sina merupakan pengurus masjid yang berusaha untuk melayani masyarakat dengan meningkatkan pelayanan masjid.

Selain terkait kegiatan kerohanian untuk masyarakat, Takmir Masjid Ibnu Sina juga mengadakan kegiatan musyawarah masjid. Musyawarah masjid dapat membantu kepengurusan Takmir Masjid untuk mengantisipasi kekurangan yang ada pada setiap kegiatan ibadah. Lemahnya kontrol kegiatan internal dapat menyebabkan meningkatnya risiko dalam kegiatan ibadah masjid (Sanusi et al., 2015). Hal tersebut akan berdampak pada kenyamanan jamaah masjid untuk melakukan ibadah di masjid. Musyawarah masjid menjadi hal yang penting, agar kegiatan ibadah terlaksana sebagaimana semestinya. Persiapan masjid dapat dilakukan pemantauan seperti kontrol dan melakukan laporan mingguan pada musyawarah yang diadakan oleh Takmir Masjid.

Musyawarah masjid merupakan salah satu cara untuk mempersiapkan kegiatan secara matang, agar kegiatan ibadah dapat secara hikmat. Pentingnya mengetahui agenda musyawarah dapat membantu Takmir Masjid dalam mengelola kumpulan pekerjaan yang akan dikerjakan dan sudah dikerjakan. Pada Masjid Ibnu Sina, terdapat musyawarah yang dilakukan saat ba'da subuh hari sabtu. Musyawarah dihadiri oleh Takmir Masjid, Remaja Masjid, dan juga masyarakat yang ingin memberi umpan balik kepada masjid. Musyawarah dibuka

UNIVERSITAS BRAWIJAYA

dengan doa oleh Amir Musyawarah sebagai pemimpin musyawarah, kemudian dilanjutkan dengan nasihat yang diberikan oleh anggota musyawarah. Lalu musyawarah dilanjutkan penyampaian laporan dari setiap anggota, usulan, dan pembahasan dari semua laporan dan usulan. Notulen musyawarah mencatat semua kegiatan musyawarah baik Amir Musyawarah, notulen, anggota musyawarah. Laporan yang diberikan pada setiap anggota merupakan laporan yang sudah dikerjaan dari pekan terakhir sampai pada musyawarah tersebut. laporan juga dapat berupa umpan balik masyarakat terhadap masjid. Kemudian usulan disampaikan untuk menanggapi pelaporan yang ada, usulan juga dapat berupa umpan balik tambahan kepada masjid. Berikutnya pembahasan terhadap laporan dan masukan akan dibahas dan dipimpin oleh Amir Musyawarah. Berdasarkan pembahasan tersebut akan muncul sebuah keputusan yang ditentukan akan ditetapkan oleh Amir musyawarah yang akan dicatat oleh notulen masjid. Setelah semua pembahasan sudah selesai. Musyawarah ditutup dan notulensi akan dikirim di grup sosial media berupa rekapan hasil musyawarah. Namun, terdapat beberapa hambatan saat berlangsungnya kegiatan musyawarah, seperti lupa terhadap keputusan yang ditetapkan sebelumnya, pembahasan yang sudah dibahas namun masih menimpulkan ambiguitas, dan memiliki dua media pencatatan hasil musyawarah pada buku serta media sosial yang menyebabkan perbedaan isi dari hasil musyawarah. Hambatan-hambatan tersebut memiliki permasalahan utama yang sama yaitu diakibatkan karena adanya kerancuan informasi notulensi yang berbeda pada buku serta notulen media sosial dan sulitnya mencari informasi berdasarkan konteks tertentu.

Untuk mengatasi permasalahan tersebut maka dibutuhkan sistem informasi musyawarah. Sistem informasi dapat mempermudah kegiatan musyawarah Masjid Ibnu Sina untuk melakukan pencatatan notulensi musyawarah dan pencarian data notulensi tertentu. Notulensi musyawarah yang tercatat dapat berupa agenda pembahasan musyawarah, pelaporan kegiatan, pembahasan keputusan, pengambilan keputusan, dan juga tanggung jawab terhadap keputusan yang diberikan (Demir and Bulut, 2018). Dengan menggunakan sistem informasi, notulensi dapat dikelompokan dan dicari untuk mempermudah takmir masjid untuk mengetahui informasi secara cepat. Sistem informasi juga dapat dimaksimalkan untuk melakukan pemantuan dan manajemen pekerjaan dari Masjid Ibnu Sina. Sistem informasi dapat mengurangi kesalahan dalam melakukan pekerjaan tertentu yang sudah ditetapkan. Oleh karena itu solusi menggunakan sistem informasi diharapkan dapat membantu takmir masjid dalam melakukan manajemen masjid.

Pada skripsi ini dikembangan sistem informasi dengan kemampuan utama untuk melakukan pengelolaan notulensi, pekerjaan, dan pencarian notulensi untuk membantu kegiatan musyawarah dan manajemen pekerjaan Masjid Ibnu Sina. Untuk mengembangkan sistem informasi tersebut diperlukan metode SDLC

(*System Development Life Cycle*) tertentu untuk mempermudah peneliti melakukan proses pengembangan sistem. Metode yang digunakan merupakan metode yang cepat dan dikembangkan secara modular untuk memastikan pengembangan sistem dapat dikembangkan lebih lanjut. Maka, untuk mengembangkan sistem informasi tersebut dipilih metode iteratif yang memiliki kecocokan dengan kebutuhan pengembangan sistem informasi. Metode tersebut terdapat 4 fase yaitu analisis permasalahan, analisis persyaratan, implementasi kode, kemudian pengujian dan evaluasi. Metode analisis persyaratan akan dilakukan dengan melakukan observasi kegiatan masjid untuk mendapatkan kebutuhan secara jelas. Kemudahan dengan menggunakan metode iteratif ini untuk melihat kemajuan dengan melibatkan pengguna untuk mencoba sistem yang dikembangkan, dapat dikembangkan dengan cepat dengan mengimplementasi fitur tertentu terlebih dahulu dengan user, dan lebih mudah dalam merubah kebutuhan karena keterlibatan user lebih intens saat sistem sedang dievaluasi.

1.2 Rumusan masalah

Sistem informasi musyawarah merupakan aplikasi yang dapat membantu anggota musyawarah masjid dalam meningkatkan kemudahan pencatatan, penelusuran informasi notulensi, pekerjaan yang dimiliki oleh Masjid Ibnu Sina. Rumusan masalah yang dapat dikaji berdasarkan hal tersebut adalah:

1. Bagaimana persyaratan sebuah sistem untuk memenuhi kebutuhan pencatatan dan penelusuran notulensi?
2. Bagaimana rancangan dan implementasi dari sistem dari sistem informasi musyawarah?
3. Bagaimana hasil pengujian sistem informasi musyawarah masjid?

1.3 Tujuan

Berdasarkan permasalahan yang telah dijabarkan pada latar belakang, maka dapat didefinisikan tujuan penelitian ini sebagai berikut:

1. Untuk mengetahui persyaratan sistem kebutuhan pencatatan dan penelusuran catatan masjid pada sistem informasi musyawarah.
2. Untuk mengetahui rancangan dan implementasi dari sistem informasi musyawarah.
3. Untuk mengetahui bagaimana hasil pengujian pada sistem informasi musyawarah.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

1.4 Manfaat

Manfaat yang diharapkan didapatkan dari penelitian ini antara lain:

1. Memberikan wawasan baik bagi peneliti maupun pembaca proses kegiatan dakwah dan pengambilan keputusan dari musyawarah Masjid Ibnu Sina.
2. Sebagai masukan bagi Masjid Ibnu Sina dalam menerapkan teknologi informasi yang lebih baik sehingga dapat meningkatkan efisiensi kinerja takmir masjid.
3. Sebagai bahan referensi untuk penelitian selanjutnya dengan topik yang berhubungan dengan aplikasi dengan metode iteratif.

1.5 Batasan masalah

Penelitian ini memiliki batasan masalah sebagai berikut:

1. Data yang digunakan merupakan data hasil observasi dan wawancara kepada anggota musyawarah Masjid Ibnu Sina.
2. Aplikasi dikembangkan menggunakan pendekatan website.
3. Menggunakan metode iteratif dalam kegiatan *System Development Life Cycle* (SDLC).
4. Desain sistem yang dilakukan adalah desain berbasis Object Oriented, menggunakan bahasa *UML.*
5. Objek masjid yang dilakukan penelitian yaitu Masjid Ibnu Sina yang beralamat di Jalan Veteran, Malang.

1.6 Sistematika pembahasan

Untuk memberikan gambaran dan kemudahan bagi pembaca dalam memahami isi, penulis memberikan sistematika pembahasan yang terbagi dalam lima bab. Sistematika tersebut terdiri dari:

BAB 1 PENDAHULUAN

Bab ini berisi latar belakang penulisan, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah serta sistematika penulisan.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisikan dasar teori untuk mendukung penyelesaian masalah dan menjelaskan landasan teori dari poin-poin penting yang digunakan dalam penelitian ini. Teori yang didapat berasal dari teori sebelumnya yang berhubungan dengan judul penelitian.

BAB 3 METODOLOGI PENELITIAN

Bab ini berisi tentang metode serta tahapan yang digunakan dalam penelitian. Tahapan yang digunakan terdiri dari tahap analisis permasalahan, analisis persyaratan sistem, iterasi luar dan dalam, evaluasi, hingga penarikan kesimpulan dan pemberian saran.

BAB 4 HASIL PENELITIAN

Bab ini membahas tentang proses penelitian dalam menerapkan proses metodologi. Proses penelitian terdapat pemodelan proses bisnis, analisis permasalahan, penerapan metode iteratif dalam pengembangan sistem. Penerapan pengembangan metode iteratif terdapat tahapan dalam melakukan pemodelan, perancangan, implementasi, dan integrasi dalam sistem. Pengujian juga akan dilakukan pada setiap proses iterasi yang dijalankan menggunakan metode pengujian *black box.*

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari setiap tahap penelitian yang telah dilakukan beserta saran untuk penelitian selanjutnya yang berhubungan dengan penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Penelitian Sebelumnya

Penelitian sebelumnya ini menjadi dasar dalam penerapan musyawarah dan penerapan implementasi secara pembangunan sistem yang dapat menunjang penelitian. Salah satu dari jurnal yang digunakan sebagai dasar penelitian ini yaitu "*Perceived Meeting Effectiveness The Role of Design Characteristics*" (Leach et al., 2009). Penelitian berikutnya juga membahas tentang pengaruh manajemen meeting yang dapat membangun meeting yang baik, jurnal tersebut berjudul "A New Model for Respected Meetings" (Demir and Bulut, 2018). Penelitian lainnya yang memiliki implementasi metode yang serupa terdapat pada penelitian "Pembangunan Sistem Informasi Manajemen Akademik Sekolah Dasar (Studi pada SDN WATES Kabupaten Kediri)" (Faraday, 2018). Penelitian berikutnya yaitu "Pembangunan Aplikasi Penilaian Ujian Skripsi Berbasis Android dengan Menggunakan Metode Prototyping (Studi Kasus: Fakultas Ilmu Komputer Universitas Brawijaya)" (Nurwansyah, 2018).

Dalam manajemen musyawarah terdapat beberapa pertimbangan untuk mendapatkan hasil musyawarah yang efektif dan efisien, terdapat tipe-tipe musyawarah yang dapat diterapkan dalam melakukan musyawarah yang efisien. Pada penelitian "*Perceived Meeting Effectiveness The Role of Design Characteristics*" oleh Warr (2009) menjelaskan bagaimana tipe *meeting* dapat mempengaruhi keefektifan pembahasan musyawarah. Penelitian tersebut menjabarkan 2 kasus studi dalam melakukan *meeting*. Dua kasus studi tersebut yaitu *meeting* yang dilakukan secara berkala dan juga kasus yang dilaksanakan pada hari tertentu. Pada penelitian tersebut juga menggabarkan dan melakukan kalkulasi untuk menghitung besarnya efektifitas dari dua kasus tersebut. Penelitian tersebut dapat dijadikan dasar dalam melakukan penelitian pada Masjid Ibnu Sina untuk dapat diusulkan dan diimplementasi secara sistem.

Penerapan *meeting* atau musyawarah pada Masjid Ibnu Sina setiap anggota musyawarah harus memiliki dasar dalam menyikapi agenda musyawarah. Anggota musyawarah harus memahami pentingnya proses musyawarah, baik sebelum musyawarah, saat musyawarah, dan setelah musyawarah. Takmir masjid harus memastikan bahwa kondisi musyawarah baik sebelum dan sesudah musyawarah harus mencapai tujuan yang disepakati. Penelitian pada "A New Model for Respected Meetings" oleh Demir (2018) diteliti di Iraq menjabarkan terdapat 3 tingkat level meeting dalam mencapai meeting yang dihargai. Tingkatan tersebut dibagi atas *Pre-In-Post Meeting*. *Pre-Meeting* adalah kondisi meeting belum dilaksanakan. lalu terdapat 6 dimensi yang saling mempengaruhi ditiap tingkatnya. Pada *pre-meeting* terdapat 2 dimensi yang bepengaruh untuk memaksimalkan *pre-meeting* yaitu *intention* dan *competence.* Kemudian pada tingkat *in-meeting* terdapat dimensi *patience, responsibility, dan coherence*. Lalu pada tingkat berikutnya yaitu *post-*

meeting terdapat dimensi *Respect to Decision*. Berdasarkan penelitian tersebut anggota musyawarah Masjid Ibnu Sina dapat melakukan pemahaman pentingnya 6 dimensi tersebut dan setiap proses tingkatan musyawarah.

Pada penelitian berikutnya ini berisi teknik dasar teori dalam penerapan sistem yang akan dikembangkan untuk mendukung terbentuknya sistem informasi musyawarah. Pengembangan yang serupa menggunakan sistem MVC (model – view – controller) pada penelitian yang berjudul "Pembangunan Sistem Informasi Manajemen Akademik Sekolah Dasar (Studi pada SDN WATES Kabupaten Kediri)" yang dilakukan oleh Faraday (2018). Penelitian ini membahas terkait sistem manajemen menggunakan metode *System Development Life Cycle Agile* untuk membantu SDN WATES dalam melakukan distribusi informasi informasi nilai rapot anak yang diinput oleh guru sekolah dan diberikan ke orang tua melalui website. Pendeketan Agile merupakan salah satu tahapan SDLC yang serupa dengan tahapan Iteratif. Pada penelitian tersebut juga menggunakan pendekatan *Object Oriented* seperti yang akan digunakan pada pengembangan sistem informasi musyawarah.

Penelitian lainnya yaitu "Pembangunan Aplikasi Penilaian Ujian Skripsi Berbasis Android dengan Menggunakan Metode Prototyping (Studi Kasus: Fakultas Ilmu Komputer Universitas Brawijaya)", penelitian ini menggunakan pendekatan prototyping dan juga menggunakan pendekatan *Object-Oriented* dalam pengembangan sistem. Penelitian tersebut diteliti oleh Nurwansyah (2018) yang penelitian tersebut bertujuan untuk membantu Fakultas Ilmu Komputer Universitas Brawijaya dalam melakukan penilaian ujian skripsi berlangsung. Pencatatan kegiatan tersebut merupakan kegiatan civitas akademik yang digunakan pada perangkat *mobile*.

Dari beberapa penelitian yang telah dilakukan tersebut, maka peneliti menggunakan beberapa hasil penelitian tersebut untuk dijadikan sebagai dasar dan panduan dalam melakukan penelitian. Metode manajemen musyawarah akan dijadikan dasar untuk melakukan perbaikan proses yang sedang berlangsung saat diteliti pada musyawarah masjid. Metode penelitian lainnya akan dijadikan sebagai dasar metode pembangunan sistem. Penelitian ini menggunakan metode *iterative* dan menggunakan *Unified Modelling Language* sebagai bahasa pemodelan sistem. Sistem akan dibentuk sesuai dengan kebutuhan yang diperlukan oleh musyawarah Masjid Ibnu Sina khususnya pada penerapan manajemen notulensi musyawarah dan manajemen pekerjaan.

2.2 Landasan Teori

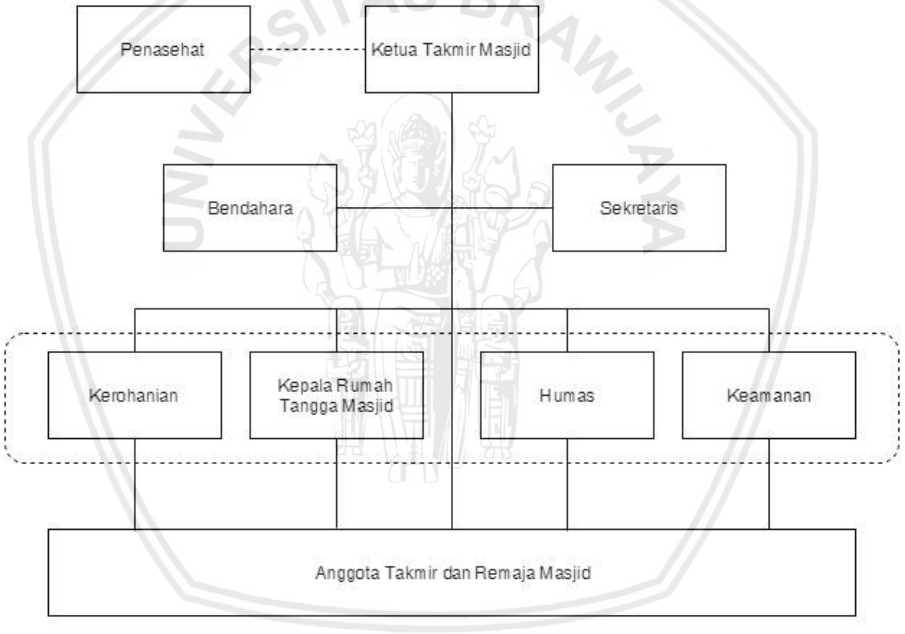
Untuk mendukung penelitian ini, maka perlu dikemukakan hal atau teori yang berkaitan dengan permasalahan dan ruang lingkup pembahasan sebagai landasan dalam penelitian ini.

2.2.1 Musyawarah Masjid

Musyawarah merupakan kegiatan pembahasan bersama dengan maksud mencapai keputusan atas penyelesaian masalah. Efektifnya sebuah musyawarah dapat dilihat dengan terbantunya anggota musyawarah menyelesaikan pekerjaannya dan organisasi (Allen, Lehmann-Willenbrock and Landowski, 2014). Musyawarah akan membantu anggota dan organisasi dalam menjalankan pekerjaan dan tujuan dari organisasi tersebut. Terdapat 3 bagian dalam pelaksanaan musyawarah yaitu *pre-meeting*, *in-meeting*, dan *post-meeting* (Demir and Bulut, 2018). Pada penelitian *A New Model for Respected Meeting* (2018), terdapat dampak yang diberikan antara pre-meeting dengan niat dan kompetensi anggota; in-meeting dengan kesabaran, coherence, dan tanggung jawab; dan post-meeting dengan menghargai keputusan musyawarah. Penelitian tersebut juga menyimpulkan bahwa sebesar 63,5% prosedur musyawarah mempengaruhi keputusan musyawarah. Perlakuan yang diberikan kepada anggota musyawarah disetiap bagian tersebut dapat memberikan efek kepada anggota untuk menjadi lebih mudah memberikan tanggapan atau membuat lebih sulit. Selain itu, Lemahnya kontrol internal dapat menyebabkan meningkatnya potensi permasalahan dalam memanajemen masjid (Sanusi et al., 2015). Selain masjid dijadikan tempat berdoa, masjid juga dijadikan tempat untuk mendapatkan ilmu pengetahuan dan menjadi tempat untuk berkumpulnya umat muslim untuk mengadakan kegiatan islami seperti pernikahan serta Ramadhan. Oleh karena itu pentingnya musyawarah akan membantu berjalannya organisasi. Musyawarah akan membantu memaksimalkan kegiatan yang akan diadakan oleh organisasi. Organisasi dapat membangun sistem yang tepat dapat membantu organisasi menjadi lebih efektif dan lebih efisien (Sanusi et al., 2015). Kegiatan musyawarah khusus juga dapat membantu kegiatan dapat berjalan lebih efisien. Penelitian berjudul *Percieve Meeting Efectiveness The Role of Design* (2009) menjelaskan terdapat dua kasus jenis musyawarah. Terdapat musyawarah yang memiliki pendekatan agenda dan musyawarah yang memiliki pendekatan waktu tertentu saja. Musyawarah tersebut mempunyai beberapa keunggulan dan kelemahan. Pada musyawarah pendekatan agenda, memiliki keunggunalan dapat menyatukan setiap anggota yang memiliki tugas yang berbeda. Namun pendekatan ini juga mengakibatkan pelaporan kegiatan dapat berdampak bias karena waktu dan focus pembahasan yang berbeda. Kemudian pada musyawarah yang memiliki pendekatan waktu tertentu memiliki keunggulan dalam pelaporan dan pembahasan musyawarah karena fokus yang dibahas hanya pada topik tertentu, tetapi hal tersebut dapat mengakibatkan anggota musyawarah dengan berbeda fokus lebih pasif. Pentingnya keikutsertaan anggota musyawarah sangat penting pada kedua tipe musyawarah tersebut memiliki efek langsung terhadap efektifitas dan juga hubungan antar anggota (2009). Oleh karena itu pentingnya tipe musyawarah juga dapat membantu efektifnya keberlangsungan organisasi.

2.2.2 Takmir Masjid Ibnu Sina

Masjid Ibnu Sina adalah salah satu masjid yang berada dimalang berdiri pada tahun 1998. Masjid ini beralamat Jl. Veteran RW. 06 Klojen Kota Malang. Masjid Ibnu Sina memiliki kepengurusan takmir dan remaja masjid untuk membangun masjid yang nyaman bagi masyarakat sekitar dan dapat dimanfaatkan sebagai wadah bersosialisasi bagi seluruh elemen masyarakat untuk berkumpul. Takmir Masjid Ibnu Sina diketuai oleh Bapak Noorrachmad Priyang atau yang akrab di sapa dengan Pak Bowo. Lalu terdapat sekretaris takmir yaitu Ismiarta Aknuranda dan terdapat bendahara takmir yaitu Nodhi Dwi Purwoko Mangku dan Ahadian Febie. Kemudian terdapat pengurus harian yaitu: Muhammad Rifqi (Bidang Kerohanian), Mega Joehar Evandri (Bidang Kepala Rumah Tangga Masjid), Nodhi Dwi Purwoko (Hubungan Masyarakat), Bapak Ainur dan Bapak Aris (Bidang Keamanan). Remaja Masjid atau Remas memiliki tanggung jawab dalam membantu Takmir Masjid dalam mengelola kegiatan dan pekerjaan masjid seperti azan, membersihkan masjid, dan menjaga masjid. Remas terdapat ketua dan anggota masjid yang diketuai oleh Muklis.



Gambar 2.1 Struktur Organisasi Takmir dan Remaja Masjid

2.2.3 Sistem Informasi

Sistem informasi adalah alat berupa kombinasi manusia, *software*, jaringan dan komunikasi, kumpulan data, maupun aturan dan prosedur yang dibuat, diterima, diubah, dan disebarluaskan berupa informasi yang diterapkan di organisasi (O'Brien and Marakas, 2010). Semakin berkembangnya teknologi dimasyarakat, manusia sudah mulai mengandalkan teknologi untuk berkomunikasi satu dan lainnya

menggunakan berbagai media seperti *hardware*, *software, network,* dan kumpulan data.

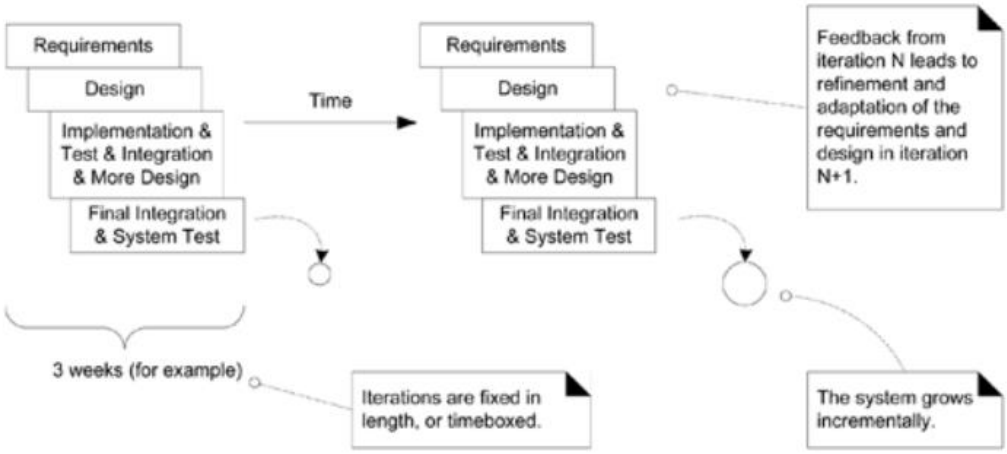
Sistem informasi mempunyai komponen subsistem yang merupakan komponen pendukung yang lebih kecil untuk membentuk sistem informasi. Komponen tersebut dapat berupa input, proses, output. Data input merupakan data yang dimasukan oleh manusia untuk menyimpan, mengubah, menghapus sehingga sistem informasi akan memprosesnya menjadi informasi yang dapat diberikan kepada masyarakat. Kemudian komponen proses, merupakan komponen logika untuk menghitung atau menkalkulasi data yang tersimpan menjadi informasi. Lalu komponen output, komponen ini merupakan komponen untuk menampilkan data yang disajikan dalam bentuk informasi yang sudah diproses sebelumnya.

2.2.4 System Development Life Cycle

System Development Life Cycle (SDLC) merupakan salah satu pendekatan dalam pengembangan sistem informasi. SDLC juga digunakan di organisasi pada umumnya yang menerapkan analisis dan desain, yang dapat dilihat sebagai tahapan, serta menerapkan proses yang bersifat terus menerus(O'Brien and Marakas, 2010). Selain itu, SDLC juga penting pada saat melakukan kegiatan maintenance software. Pada *software engineering*, banyak tipe SDLC yang dapat diterapkan pada proses pengembangan. Metodologi SDLC merupakan sebuah *framework* untuk melakukan perencanaan dan kontrol terhadap sistem informasi pada proses pengembangan (Kute and Thorat, 2014). Aktivitas dasar yang dilakukan pada pengembangan *software* yaitu pendefinisian persyaratan, desain sistem, implementasi, dan sistem testing.

2.2.4.1 Iterative Model

Model SDLC (*System Development Life Cycle*) yang dipakai dalam penelitian ini adalah model *Iterative*. Metode iteratif ini merupakan metode yang memiliki siklus berulang. Metode ini membantu mengasumsi kebutuhan terlebih dahulu untuk didefinisikan secara detail pada proses pemodelan. Metode ini memecah proses pengembangan menjadi sekumpulan proses singkat yang disebut sebagai proses iterasi (Larman, 2004). Pada setiap proses iterasi terdapat analisis persyaratan, desain dan perancangan, implementasi, dan testing.



Gambar 2.2 *Iteratif Development*

Sumber: Larman (2004)

Siklus iteratif ini dapat terdiri dari beberapa iterasi, iterasi ini akan mendapatkan masukan dan akan di adaptasi pada sistem. Sistem akan terus berkembang seiring waktu, iterasi ke iterasi, penerapan ini dikenal sebagai dengan pengembangan *iterative* dan *incremental*. Karena masukan lalu pengadaptasian akan mengevolusi spesifikasi dan desain yang dikenal sebagai pengembangan *iterative* dan *evolutionary* (Larman, 2004).

2.2.5 BPMN (Business Process Modeling Notation)

Business Process Modeling Notation (BPMN) merupakan sebuah standar yang dibuat oleh OMG (Object Management Group, 2011) untuk pemodelan proses bisnis. Tujuan utama dari BPMN adalah untuk membantu notasi yang lebih mudah dipahami bagi semua aktor. Sisi bisnis akan membantu menganalisis dan dibentuk sebuah draft untuk mengetahui prosesnya. Sedangkan dari sisi teknikal, pengembang bertanggnug jawab untuk mengimplementasi proses tersebut. tujuan dari BPMN ini adalah untuk mengstandarisasi pemodelan proses dan notasi dari berbagai pemodelan. BPMN menyediakan seperangkat bahasa notasi grafis yang terstandarisasi yang sudah dikenalkan sejak tahun 2002. Dalam Busines Process Diagram terdapat elemen pemodelan berupa notasi. Jenis jenis notasi dalam BPMN terdapat sebagai berikut:

1. *Swimlane*

   Swimlane adalah notasi grafis yang berbentuk persegi panjang yang digunakan untuk menandakan serangkaian aktivitas satu dengan aktivitas lainnya. Terdapat 2 macam *swimlane* yang terdapat pada BPMN. Terdapat *swimlane pool* dan *swimlane lane*. *Pool* merepresentasikan tanggung jawab aktivitas dalam sebuah organisasi, *Pool* dapat memodelkan sebuah proses secara

mendetail, dan *pool* juga dapat dibentuk tanpa penjabaran proses yang mendetail. Proses yang tidak mendetail tersebut akan diasumsikan sebagai *blackbox*. Sedangkan *Lane* merepresentasikan pembagian *pool* yang dibagi berdasarkan aktor, peran, maupun sistem. *Lane* menentukan siapa yang akan menyelesaikan proses dalam *swimline. Line* akan digunakan untuk mengorganisir dan menkategorikan aktivitas (Object Management Group, 2011).

2. *Text Annotation*

   Text annotation digunakan untuk memberikan informasi mengenai object atau elemen di sebuah proses, tanpa memberi dampak pada alur proses bisnis. *Text annotation* dapat merepresentasikan data atau catatan yang mendeskripsikan data atau catatan yang mendeskripsikan atau mengatur proses atau task (Object Management Group, 2011).

3. *Gateway*

   Gateway digunakan untuk mengatur bagaimana alur proses berlangsung melalui *sequence flow*. Alur-alur tersebut dapat bertemu dan berpisah pada proses tertentu. *Gateway* akan membantu untuk melakukan branching, forking, merging, dan joining pada proses bpmn. *Gateway* mengontrol alur, dengan cara menanamkan autran atau kondisi pada flow yang keluar dari *gateway*. *Gateway* dapat memiliki banyak input maupun output berupa flow (Object Management Group, 2011).

4. Data

   Saat mengeksekusi proses bisnis, dimungkinkan terdapat data yang ditambahkan atau dibaca, baik selama proses berlangsung maupun setelah proses tersebut berakhir. Pada BPMN, data dapat dimodelkan dengan beberapa jenis objek data, yaitu data objek dan data store (Object Management Group, 2011). Data akan digabarkan sebagai informasi pada aktivitas, yang akan diolah. Data objek juga dapat direpresentasikan sebaai satu buah objek maupun banyak objek.

5. *Activity*

   Activity atau notasi aktivitas adalah pekerjaan yang dilakukan perusahaan atau organisasi dalam sebuah proses bisnis. Penggunaan notasi aktivitas pada alur proses bisnis digunakan bersamaan dengan notasi lainnya untuk mendeskripsikan lebih rinci mengenai kegiatan di alur proses. Tanpa adanya notasi aktivitas, tidak aka nada proses bisnis karena tidak ada sesuatu yang bisa diselesaikan. Tipe aktivitas yang dapat menjadi bagian sebuah proses diantaranya proses, sub-proses, dan task(Object Management Group, 2011)

6. *Event*

Event adalah sesuatu yang terjadi pada sekumpulan alur proses. Notasi ini akan berfungsi sebagai *trigger* atau hasil pada sekumpulan alur proses. Dalam BPMN, terdapat tiga jenis dasar *event*, yaitu *start event, intermediate event,* dan *end event*. Ketiga jenis *event* ini dapat menangkap (catch), melempar (throw), dan *event* (sesuatu yang terjadi) (Object Management Group, 2011).

7. *Connection Object*

   Connection object berfungsi untuk membuat struktur proses bisnis beraturan dengan cara membuat alur bagi objek-objek yang ada pada proses. Sebuah connection object dapat terhubung dari dan ke objek apa saja yang ada, baik di dalam satu *pool* yang sama maupun lintas *pool* (Object Management Group, 2011). Terdapat *connection object* berupa sequence flow dan message flow. Sequence flow berfungsi sebagai alur proses sebelum dan sesudah dari sebuah aktivitas. Message flow menggambarkan alur informasi dari dua partisipan yang memiliki sifat kolaborasi.

2.2.6 UML (Unified Modeling Language)

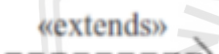
Unified Modelling Language (UML) adalah bahasa sandar dalam melakukan penulisan rancangan software. UML dapat di visualisasi, dispesifikasi, dan pendokumentasian artefak dari perangkat lunak yang dikembangkan (Pressman, 2010). UML menyediakan kebutuhan teknologi untuk mendukung pembelajaran *object-oriented software engineering.* UML memiliki diagram-diagram yang digunakan untuk mendesain suatu sistem atau perangkat lunak, diantaranya yaitu: Use Case Diagram, Activity Diagram, Sequence Diagram, serta Class Diagram.

2.2.6.1 Use Case Diagram

Use case diagram adalah narrative atau contoh yang menjelaskan sebuah fungsi sistem atau fitur dari sudur pandang pengguna. Use case di tentukan berdasarkan pengguna dan sistem dasar dalam penjabaran requiremen (Pressman, 2010). Pengertian lain *Use case diagram* juga sebagai sebuah kebiasaan yang sesuai dengan urutan transaksi yang berjalan dan dijalankan oleh aktor dalam sistem yang terukur dan bernilai kepada aktor (Booch et al., 2007). *Use case* memiliki keuntungan untuk mengilustrasikan sistem, menverifikasi kebutuhan yang tercatatat, mudah dipahami dengan adanya terminology pengguna (IBM, 2007). Berdasarkan pengertian tersebut diatas, *Use case diagram* adalah diagram yang menggambarkan kegunaan atau fungsi yang dapat dilakukan oleh sistem dengan menunjukan bagaimana interaksi antara pengguna dan sistem terkait fungsi apa saja yang bisa digunakan oleh pengguna.

Didalam diagram *use case* terdapat 5 elemen penyusun dari diagram *use case* diantaranya

Tabel 2.1 Simbol *Use case Diagram*

No	Simbol	Deskripsi
1	*Use case* 	*Use case* merupakan sekumpulan proses aksi yang dapat dilakukan oleh sistem.
2	Aktor 	Aktor adalah tipe peranayang dijalankanaoleh sebuah entitas untukaberinteraksi denganasistem, sehingga actor merupakan bukan sebuah sistem. Aktor dapat direpresentasikan oleh manusia, perangkatakeras eksternal atau yang lain.
3	Association	*Association* merupakan sebuah koneksi antara aktor dan *use case*. *Association* mengindikasikan bahwa *aktor* dapat melakukan suatu *use case*. Dalam UML, *association* berarti bahwa sebuah aktor terlibar dalam sebuah *use case.*
4	Extend 	*Extend* merupakan sebuah hubungan yang memungkinkan kita untuk memodifikasi tingkah laku sebuah *use case* dasar dengan menambahkan sebuah *use case* tambahan.
5	Include 	*Include* menunjukan bahwa *behaviour* sebuah *included use case* termasuk bagian dari sebuah *use case* dasar. Tujuannya untuk penggunaan kembali sebuah *use case* yang digunakan berulang kali dalam *use case* lain.

Sumber: Sukamto & Shalahudin (2014)

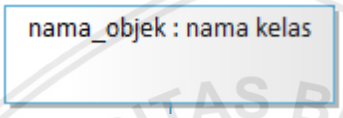
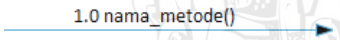
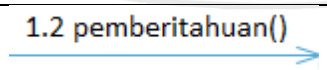
2.2.6.2 Sequence diagram

Sequence diagram atau diagram sekuen menggambarkan interaksi antar objek didalam dan di sekitar sistem (termasuk pengguna, tampilan, dan sebagainya) berupa pesan atau *message* yang digambarkan terhadap waktu (Pressman, 2010). Saat sebuah proses diidentifikasi sebagai sebuah usecase, proses tersebut dapat dirancang menjadi sekuen diagram. Diagram sekuen biasa menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari *event* untuk menghasilkan keluaran tertentu. *Message* yang digambarkan pada diagram sekuen nantinya akan dipetakan menjadi fungsi/metoda dari *class* pada tahapan desain selanjutnya. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan simbol khusus untuk objek *boundary, controller* dan *persistent entit*.

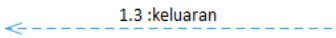
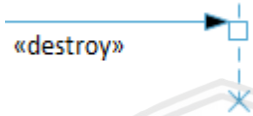
repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

Diagram sekuen merupakan diagram interaksi yang menekankan pada saat permintaan pesan yang menunjukkan serangkaian peranan dan pesan yang dikirim dan diterima oleh bagian yang ada pada peranan tersebut. Diagram sekuen digunakan untuk menggambarkan *dynamic view* dari sistem (Wesley, 2015). Simbol-simbol yang digunakan pada diagram *sequence* ditunjukkan pada Tabel 2.2.

Tabel 2.2 Simbol-simbol Diagram *Sequence*

Nama	Simbol	Deskripsi
Lifeline		Menunjukkan keberadaan objek selama periode waktu.
Objek		Menyatakan objek yang berinteraksi menggunakan pesan atau *message.*
Activation Bar		Menunjukkan periode waktu selama sebuah objek melakukan kegiatan, baik secara langsung maupun melalui prosedur perintah.
Pesan tipe call (*synchronous*)		Menyatakan suatu objek memanggil fungsi yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki fungsi, karena ini memanggil sebuah fungsi maka fungsi yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe *send* (*asynchronous*)		Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya

Tabel 2.2 Simbol-simbol Diagram *Sequence*

Nama	Simbol	Deskripsi
Pesan tipe *return*		Menyatakan bahwa suatu objek yang telah menjalankan suatu fungsi menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe *destroy*		Objek dapat diakhiri selama interaksi yang berlangsung. *Lifeline* objek berakhir apabila menerima pesan *destroy.* Bila ada pesan tipe *create* maka sebaiknya ada pesan tipe *destroy.*

Sumber: Wesley (2015)

2.2.6.3 Class Diagram

Class Diagram digunakan untuk menunjukan apa saja *class* atau objek yang berkaitan dengan system dan hubungan antara *class* untuk saling bertukar informasi seperti apa. Suatu *class diagram* merepresentasikan struktur dari sebuah *class* yang ada di dalam sistem. Saat fase analisis, class diagram digunakan untuk mengidiikasi batasan dan tanggung jawab dari entitas yang akan mendukung sistem. Pada tahap pemodelan, class diagram digunakan untuk mengetahui informasi struktur dari class yang akan dibangun menjadi arsitektur sebuah sistem (Booch et al., 2007). Elemen yang paling penting dalam sebuah *class diagram* adalah *class* dan hubungannya. Tabel 2.3 menjelaskan simbol dari *class diagram*.

Tabel 2.3 Simbol *Class Diagram*

No	Simbol	Deskripsi
1	Class 	*Class* merupakan sebuah *template* objek yang dibuat. *Class* mendefnisikan atribut, operasi dan *instance*.
2	*Association*	Relasi yang menunjukan hubungan antara dua *class*.

No	Simbol	Deskripsi
3	*Generalization*	Relasi yang menunjukan pewarisan dari *class* utama (*parent*) ke *class* anak (*child)*
4	*Dependency*	Relasi yang menunjukan suatu *class* bergantung pada *class* yang lain
5	Agregasi / *Aggregation* 	Relasi yang menunjukan sebuah elemen yang terdiri dari beberapa komponen kecil

Sumber: Sukamto & Shalahudin (2014)

Dalam pemodelan *class diagram*, terdapat tiga bentuk berbeda bersasarkan tingkat kerinciannya. Tingkat pemodelan *class diagram* melibatkan tiga tahap dalam pengembangan perangkat lunak yaitu definisi kebutuhan, analisis, dan perancangan sistem. Ketiga tahap tersebut antara lain (Kendall, 2007):

1. *Domain-Level Class Diagram*
   Pada tahap *domain-level class diagram, class diagram* hanya menampilkan nama class saja. Tujuan dari *class diagram* pada tahap ini adalah untuk menyediakan istilah umum yang digunakan dalam proyek pengembangan perangkat lunak sehingga para anggota pengembangan dapat memahami dengan jelas proyek yang dikerjakan.
2. *Analysis-Level Class Diagram*
   Pada tahap *analysis-level class diagram,* ditambahkan atribut pada tiap *class* di *domain-level class diagram*. *Analysis-level class diagram* tidak perlu menunjukan operasi pada *class* karena hal tersebut masuk dalam fase perancangan.
3. *Design-Level Class Diagram*
   Pada tahap *design-level class diagram*, *class diagram* digambarkan secara lengkap operasi dan atribut sesuai dengan standar *class diagram* pada notasi *unified modelling language* (UML).

2.2.7 Database

Basis data adalah suatu kumpulan data yang berhubungan secara logika dan secara deskripsi dari data-data yang dirancang untuk memenuhi kebutuhan informasi dalam suatu organisasi. Basis data menawarkan keuntungan penyimpanan data dengan format yang independen dan fleksibel (Connolly & Begg, 2010).

Database merupakan kumpulan informasi yang disimpan dalam sebuah komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program

komputer untuk memperoleh informasi dari basis data (*database*) tersebut (Fathansyah, 2012).

Penerapan database dalam suatu informasi disebut dengan database System. Database digunakan untuk menyimpan informasi atau data yang terintegrasi dengan baik di dalam komputer. *Database* adalah sekumpulan data yang sudah disusun sedemikian rupa dengan ketentuan atau aturan tertentu yang saling berelasi sehingga memudahkan pengguna dalam mengelolanya juga memudahkan memperoleh informasi. Selain itu ada pula yang mendefinisikan database sebagai kumpulan *file*, tabel, atau arsip yang saling terhubung yang disimpan di dalam media elektronik.

2.2.8 Java

Java menurut definisi dari Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan (Shalahuddin and S., 2010). Java 2 adalah generasi kedua dari Java *platform.* Java berdiri di atas sebuah mesin *interpreter* yang diberi nama JVM atau *Java Virtual Machine.* JVM inilah ang akan membaca *bytecode* dalam *file* .class dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu, bahasa Java disebut sebagai bahasa pemrograman yang portabel karena dapat dialankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM.

Java memiliki beberapa versi *library* atau teknologi yang disebut juga sebagai edisi dari bahasa pemrograman Java. Tiga edisi utama dari *library* tersebut adalah Micro, Standard, dan Enterprise. J2ME (Java2 Micro Edition) merupakan edisi *library* yang dirancang untuk membuat aplikasi *desktop* atau applet pada web *browser*. J2EE (Java2 Enterprise Edition) merupakan edisi *library* yang dirancang untuk membuat sebuah aplikasi *enterprise* yang memerlukan antarmuka dengan sumber data (*datasource*) atau dapat pula dikatakan bahwa J2EE adalah kelompok yang lebih besar dengan J2SE di dalamnya.

2.2.8.1 Framework Spring Boot

Spring merupakan framework *open source* berbasis java yang menyediakan infrastruktur secara kompreherensif untuk membangun *java enterprise* dengan mudah dan cepat. Framework ini dikembangkan oleh Rod Johnson pada tahun 2003. Spring membantu pembangunan sistem secara modular karena sifatnya reusable dan menerapkan Aspect Oriented Programming. Selain itu spring juga memanfaatkan struktur MVC (*Model-View-Controller*) untuk melakukan pengelompokan fungsi model, view, dan controller. Proses berjalanannya MVC pada spring diilustrasikan dalam Gambar 2.3. Setiap request dari user akan diterima oleh Front controller yang akan diteruskan ke Controller. Lalu Controller akan menerima request tersebut lalu diproses dan memberi respon ke Front Controller berupa model. Model akan

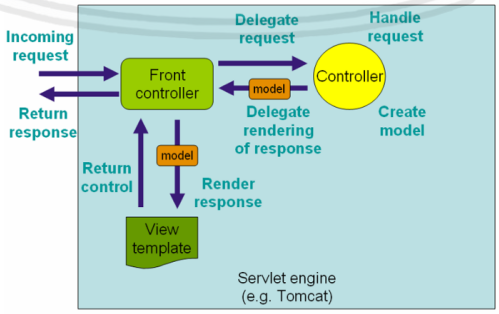
menyimpan data dan akan ditampilkan ke view template. Penerapan sistem pada Spring menggunakan Notasi untuk memberi tanda dari sebuat atribut atau method yang terdapat pada sistem.

Tabel 2.4 Contoh Anotasi Java Spring Boot

Notasi	Deskripsi
@Autowired	Melakukan injeksi pada variable tersebut secara otomatis.
@Configuration	Memiliki fungsi untuk mendifinisikan beans dari konfigurasi file yang sudah disimpan.
@RestController	Digunakan sebagai penanda java class yang mengembalikan sebuah objek model dan bukan sebuah view.
@Controller	Digunakan sebagai penanda java class yang merupakan sebuah controller, penggunaan anotasi ini akan membuat java class menjadi Spring controller
@SpringBootApplication	Merupakan sebuah anotasi yang digunakan sebagai dasar project aplikasi spring boot

Sumber : Spring Boot (2019)

Spring Boot adalah sebuah framework bersifat open source yang dikhususkan untuk pengembangan aplikasi web dengan bahasa pemrograman Java. Spring Boot merupakan pengembangan dari framework Spring. Spring boot juga membantu mempercepat proses pembangunan aplikasi, anotasi dapat membantu mempercepat automasi yang dikonfigurasi oleh spring boot agar dapat lebih berfokus pada pengembangan fitur bisnis dan lebih sedikit pada infrastruktur (Spring Boot, 2019).



Gambar 2.3 Proses MVC Spring

Sumber: Spring Fundamental, Filkom (2018)

2.2.9 MySQL

Menurut Rudianto (2011) “MySQL adalah salah satu jenis database server yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunkan database sebagai sumber dan pengolahan datanya”.

MySQL dikembangkan oleh perusahaan Swedia bernama MySQL AB yang pada saat ini bernama Tcx DataKonsult AB sekitar tahun 1994-1995, namun cikal bakal kodenya sudah ada sejak tahun 1979. Awalnya Tcx merupakan perusahaan pengembangan software dan konsultan database, dan saat ini MySQL sudah diambil alih oleh Oracle Corp.

Kepopuleran MySQL antara lain karena MySQL menggunkan SQL sebagai bahasa dasar untuk mengakses databasenya sehingga mudah untuk digunakan, kinerja query cepat, dan mencukupi untuk kebutuhan database perusahaan – perusahaan yang berskala kecil sampai menengah, MySQL juga bersifat Open Source (tidak berbayar).

MySQL merupakan database pertama kali yang didukung oleh bahasa pemrograman script untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan software pembangun aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web. Umumnya pengembangan aplikasinya menggunakan Bahasa pemrograman script PHP. MySQL didistribusikan dengan lisensi Open Source GPL (General Public License) mulai versi 3.23 pada bulan Juni 2000.

2.3 Pengujian *Black-Box*

Pengujian perangkat lunak mengacu pada proses evaluasi perangkat lunak dengan maksud untuk mengetahui kesalahan di dalamnya. Pengujian perangkat lunak adalah teknik yang ditujukan untuk mengevaluasi atribut atau kemampuan suatu program menentukan bahwa program tersebut memenuhi kualitasnya. Dalam istilah yang paling sederhana, pengujian berarti mengamati eksekusi sistem perangkat lunak untuk memvalidasi apakah berperilaku seperti sesuai yang dirancangkan dan mengidentifikasi malfungsi potensial.

Pengujian *black-box* merupakan pengujian perangkat lunak berdasarkan persyaratan keluaran dan tanpa pengetahuan tentang struktur internal atau pengkodean dalam program. Tujuannya adalah untuk menguji seberapa baik komponen sesuai dengan persyaratan yang diterbitkan untuk komponen tersebut. Pengujian black-box sedikit atau tidak berkaitan dengan struktur logis internal sistem, hanya memeriksa aspek fundamental sistem. Ini memastikan bahwa input diterima dengan benar dan output diproduksi dengan benar.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

2.3.1 Pengujian Validasi

Pengujian validasi merupakan pengujian yang berfokus pada hasil dari sistem yang terlihat oleh pengguna. Pengujian ini dibangun untuk membuktikan sistem telah memenuhi kebutuhan pengguna yang telah didefinisikan. Pengujian validasi dilakukan dengan cara mendemonstrasikan perangkat lunak yang dikembangkan (Pressman, 2010).

Pengujian validasi dapat menggunakan *use case scenario* sebagai panduan skenario dalam melakukan testing. *Use case scenario* dapat membantu untuk menemukan kesalahan alur atau sistem yang terdapat pada perangkat lunak. Setelah pengujian ini selesai dilakukan akan didapatkan dua hasil yaitu (Pressman, 2010):

1. Sistem telah sesuai dengan spesifikasi dan diterima oleh *stakeholder.*
2. Kekurangan pada perangkat lunak yang dikembangkan akan terlihat dan kemudian akan dicatat.

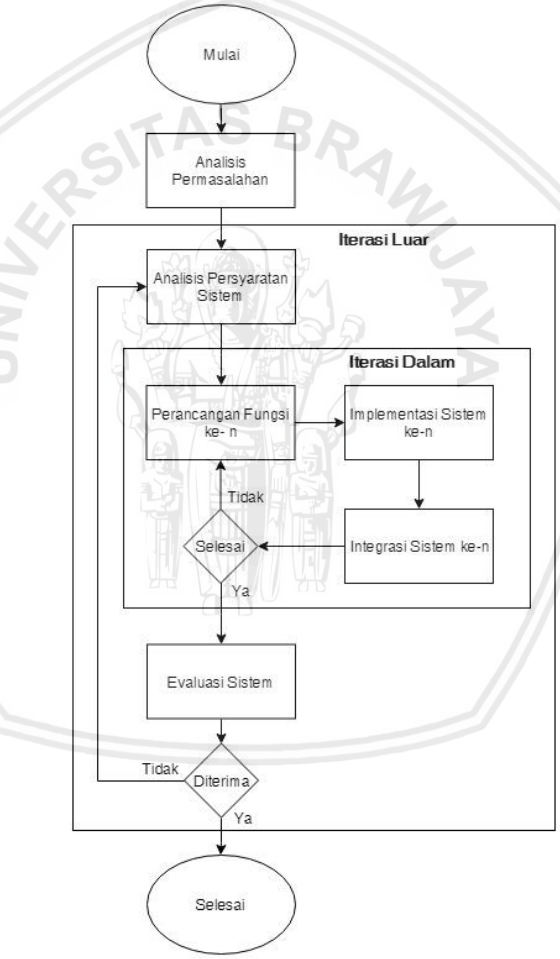
2.3.2 Pengujian Kompatibilitas Peramban

Pengujian kompatibilitas peramban adalah salah satu metode pengujian yang dapat dilakukan untuk menguji antarmuka pengguna pada aplikasi situs web. Pengujian ini berguna untuk mengetahui aplikasi memiliki masalah ketika aplikasi berjalan pada perangkat yang berbeda atau dengan kecepatan koneksi yang berbeda. Pengujian ini dilakukan dengan melakukan konfigurasi pada server yang disediakan. Hal ini dilakukan untuk mengetahui kesalahan pada eksekusi aplikasi pada beberapa konfigurasi yang berbeda (Pressman, 2010).

BAB 3 METODOLOGI

3.1 Metodologi Penelitian

Penelitian yang dilakukan merupakan penelitian implementatif yang diterapkan dengan metode iteratif. Pendekatan untuk pengembangan sistem dengan metode iteratif diterapkan dengan berorientasi objek (OO). Langkah-langkah utama yang dilakukan adalah analisis permasalahan, analisis persyaratan sistem, perancangan sistem, implementasi sistem, dan evaluasi sistem. Iterasi tertentu dijalankan secara berkala selama kebuthuan sistem belum terpenuhi. Setiap proses iterasi disesuaikan dengan kebutuhan dari pengguna sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.



Gambar 3.1. Alur Metodologi Penelitian

3.2 Analisis Permasalahan

Pada tahapan ini dilakukan pengumpulan permasalahan dan pemahaman terhadap kegiatan musyawarah Masjid Ibnu Sina. Pengambilan data pada tahap analisis permasalahan dilakukan dengan pendekatan wawancara dan observasi. Wawancara dilakukan kepada anggota musyawarah, baik Takmir maupun Remaja Masjid. Kemudian observasi dilakukan dengan mengikuti kegiatan musyawarah mingguan yang ada di Masjid Ibnu Sina untuk mendapatkan informasi yang mendalam.

3.3 Implementasi Proses Iterasi

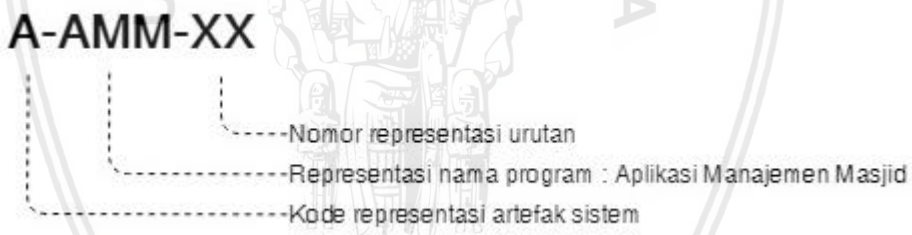
Terdapat 2 iterasi yang digunakan dalam menerapkan metode pengembangan iteratif, yaitu iterasi luar dan iterasi dalam. Proses Iterasi Luar merupakan proses penggalian persyaratan yang menjadikan sebagai dasar memulainya proses iterasi dalam, kemudian sistem yang dikembangkan tersebut lalu dievaluasi. Jumlah iterasi luar ditentukan dengan mengelompokkan proses proses berdasarkan kepentingan tertentu agar proses iterasi dapat berjalan sesuai dengan rencana. Jumlah proses iterasi ini dapat ditentukan berdasarkan prioritas pekerjaan. Prioritas seperti inti dari sistem seperti arsitektur sistem, karena inti tersebut dapat menjadi penunjang dari sistem lainnya. Selain itu, prioritas proses iterasi juga dapat ditentukan seberapa penting sistem tersebut membantu kepentingan bisnis, serta prioritas juga dapat diprioritaskan dari fungsi yang memiliki peluang yang berisiko tinggi. Setelah mengetahui permasalahan dan solusi fitur, proses iterasi luar kemudian dilakukan dengan penggalian persyaratan pada tahap analisis persyaratan sistem. Analisis persyaratan dilakukan untuk mengidentifikasi persyaratan sistem kepada pihak-pihak yang terlibat pada sistem. Proses tersebut digambarkan menggunakan *Business Process Model Notation* (*BPMN*). Persyaratan sistem tersebut kemudian dievaluasi yang berbentuk usulan perbaikan sistem atau usulan penambahan sistem. Kemudian setelah mengetahui model proses sistem yang diusulkan, Iterasi Dalam dimulai. Iterasi dalam terdapat beberapa proses yang setiap prosesnya menggunakan pendekatan berorientasi object. Proses tersebut dimodelkan menggunakan OOAD (*Object Oriented Analysis Design*) dan diterapkan menggunakan OOP (*Object Oriented Programming*). Setelah melakukan pengembangan tiap fiturnya menggunakan iterasi dalam, sistem dievaluasi menggunakan metode *black-box testing* yaitu pengujian validasi dan pengujian kompatibilitas peramban.

BAB 4 HASIL PENELITIAN

Bab ini membahas tentang bagaimana artefak sistem dimodelkan. Bab ini terdapat proses bisnis yang terjadi pada musyawarah, analisis persyaratan, pemodelan usecase, proses iterasi dalam, dan evaluasi sistem. Analisis persyaratan terdapat identifikasi tipe pemangku kepentingan, identifikasi kebutuhan pemangku kepentingan, identifikasi pengguna, identifikasi fitur, persyaratan fungsional, dan persyaratan nonfungsional pada sistem. Kemudian pada pemodelan *use case*, terdapat deskripsi aktor, *use case* diagram, dan spesifikasi usecase. Lalu, pada proses iterasi dalam terdapat perancangan, struktur artefak sistem, dan implementasi. Perancangan terdiri dari pemodelan *sequence diagram*, *class diagram,* pemodelan data, dan perancangan antarmuka pengguna. Pada implementasi terdiri dari implementasi desain antarmuka, implementasi data definition language, dan implementasi fungsi. Lalu terdapat evaluasi sistem berupa rencana pengujian validasi dan pengujian kompatibilitas.

4.1 Aturan Penomoran

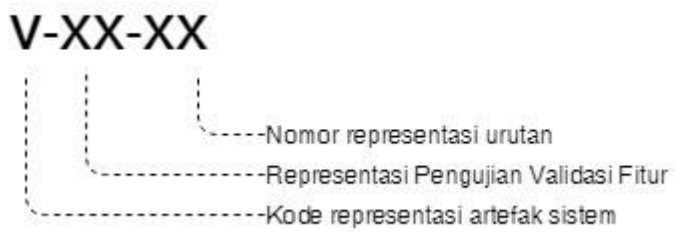
Aturan penomoran digunakan sebagai identitas untuk memudahkan *traceablitity* dokumen. Aturan yang digunakan peneliti sebagai berikut:



Kode Representasi Artefak yang terdapat pada sistem didefinisikan sebagai berikut:

1. **PBA** : proses bisnis saat ini (*as-is*).
2. **PBT** : proses bisnis usulan (*to-be*).
3. **KB** : artefak kebutuhan.
4. **IFT** : artefak identifikasi fitur.
5. **FT** : artefak fitur.
6. **FTF** : artefak kebutuhan fungsional.
7. **FTNF** : artefak kebutuhan nonfungsional.
8. **UC** : artefak use-case.

Aturan penomoran validasi mempunyai kode khusus yang pada tengah kode merupakan representasi pengujian validasi fitur Aturan yang digunakan. Aturan penomoran validasi digunakan sebagai berikut:



Kode Representasi Artefak yang terdapat pada sistem didefinisikan sebagai berikut:

1. **MA** : pengujian validasi mengelola anggota.
2. **MP** : pengujian validasi mengelola pekerjaan masjid.
3. **MN** : pengujian validasi melihat notulensi.
4. **MIN** : pengujian validasi mengelola notulensi.
5. **MCN** : pengujian validasi mencari notulensi.
6. **MVP** : pengujian validasi menverifikasi status pekerjaan.
7. **MVN** : pengujian validasi menverifikasi notulensi.

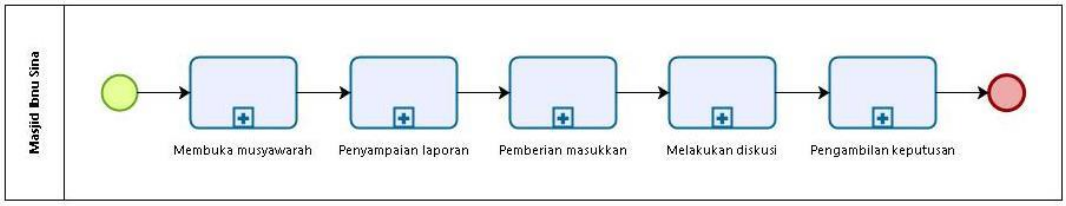
4.2 Deskripsi Proses Bisnis

Pemodelan proses bisnis ini berdasarkan kegiatan yang dijalankan oleh Masjid Ibnu Sina dalam menjalankan kegiatan musyawarah masjid. Pemodelan bisnis yang diteliti terdapat dua macam proses bisnis yaitu proses bisnis berlaku saat ini (*as-is*) dan proses bisnis usulan (*to-be*) dengan menggunakan sistem informasi. Terdapat 2 perbaikan proses bisnis dan 2 usulan tambahan untuk membantu proses bisnis yang terdapat pada proses bisnis perbaikan.

4.2.1 Deskripsi Proses Bisnis *As-Is*

Proses bisnis yang berjalan saat ini (*as-is*) pada penelitian ini merupakan proses bisnis yang sedang berlangsung pada Masjid Ibnu Sina. Proses bisnis pada penelitian ini didapatkan melalui mewawancarai salah satu Takmir Masjid untuk mengetahui proses musyawarah yang terjadi pada Masjid Ibnu Sina. Pada proses wawancara awal dijelaskan terdapat 4 aktivitas yang terjadi pada saat musyawarah

yaitu pelaporan, pengusulan, pembahasan dan pengambilan keputusan. Proses ini terjadi secara berurutan dan bergantian.



Gambar 4.1 Analisis Musyawarah Masjid

Untuk mengetahui lebih jauh peneliti mengikuti kegiatan musyawarah takmir masjid untuk menggali kebutuhan secara langsung bagaimana proses musyawarah berlangsung. Pemodelan proses kegiatan musyawarah digambarkan dengan diagram BPMN untuk mempermudah pemangku kepentingan memahami hasil analisis proses bisnis saat ini (*as-is*).

4.2.2 Deskripsi Proses Bisnis *To-Be*

Proses bisnis *to-be* yang dimodelkan pada penelitian ini adalah proses bisnis usulan yang dikembangkan dari proses bisnis *as-is*. Proses bisnis *to-be* merupakan pengembangan proses yang terdapat pada *as-is* dengan melakukan pemotongan rantai proses bisnis menggunakan solusi IT. Proses bisnis *to-be* juga dapat berupa penambahan proses bisnis baru untuk menunjang pekerjaan tertentu pada proses bisnis *to-be* lainnya. Proses ini dimodelkan menggunakan BPMN (*Business Process Modeling Notation*) untuk mengetahui perubahan proses yang diusulkan. Hasil dari proses bisnis ini dijadikan sebagai persyaratan sistem yang akan dikembangkan. Proses bisnis *to-be* dijabarkan pada setiap iterasi luar saat proses bisnis usulan dikembangkan.

4.3 Analisis Permasalahan

Analisis permasalahan dilakukan untuk memahami permasalahan yang terjadi pada kegiatan musyawarah masjid, dan juga masalah yang harus diselesaikan oleh pemangku kepentingan. Permasalahan ini berdasarkan hasil observasi dan hasil wawancara kepada anggota musyawarah masjid. Hasil analisis permasalahan yang dijelaskan pada Tabel 4.1.

Tabel 4.1 Pernyataan Masalah

Masalah	● Terdapat dua tugas yang memiliki peran yang sama yaitu pencatatan menggunakan buku dan sosial media. ● Penelusuran notulen tidak dapat ditelusuri sesuai dengan kategori tertentu pada buku maupun sosial media. ● Pencarian notulensi notulensi masjid belum dilakukan berdasarkan atribut. ● Belum terdapat verifikasi baik pada pekerjaan maupun notulensi.
Mempengaruhi	Takmir dan Remas
Dampak	● Dapat terjadi perbedaan informasi yang terdapat pada buku dan sosial media ● Sulitnya melakukan pencarian notulensi ● Sulit untuk melakukan revisi pada notulensi karena terdapat dua media dalam membuat notulensi.
Solusi	Menyediakan sistem yang dapat digunakan untuk melakukan pencatatan notulensi, pencarian berdasarkan kata kunci tertentu, dan persetujuan data notulensi dan pekerjaan yang dibuat

4.4 Iterasi Luar Ke-1

Proses iterasi luar ke-1 dijadikan sebagai proses pertama dalam melakukan penggalian kebutuhan pertama kali pada Masjid Ibnu Sina. Iterasi pertama ini menjadi persyaratan utama yang dikembangkan pertama kali dalam proses iterasi. Berdasarkan solusi yang sudah didefinisikan, iterasi ini dikembangkan untuk mengelola keanggotaan takmir, mengelola pekerjaan, dan mengelola notulensi.

4.4.1 Pemodelan Proses Bisnis *As-Is*

4.4.1.1 Proses Bisnis *As-Is* Mengelola Keanggotaan Musyawarah (PBA-AMM-01)

Takmir Masjid Ibnu Sina tidak sepenuhnya melakukan pengesahan secara tertulis untuk menjadi anggota. Takmir Masjid tidak secara sering melakukan pencatatan anggota kecuali pada saat keperluan tertentu yang membutuhkan informasi anggota.

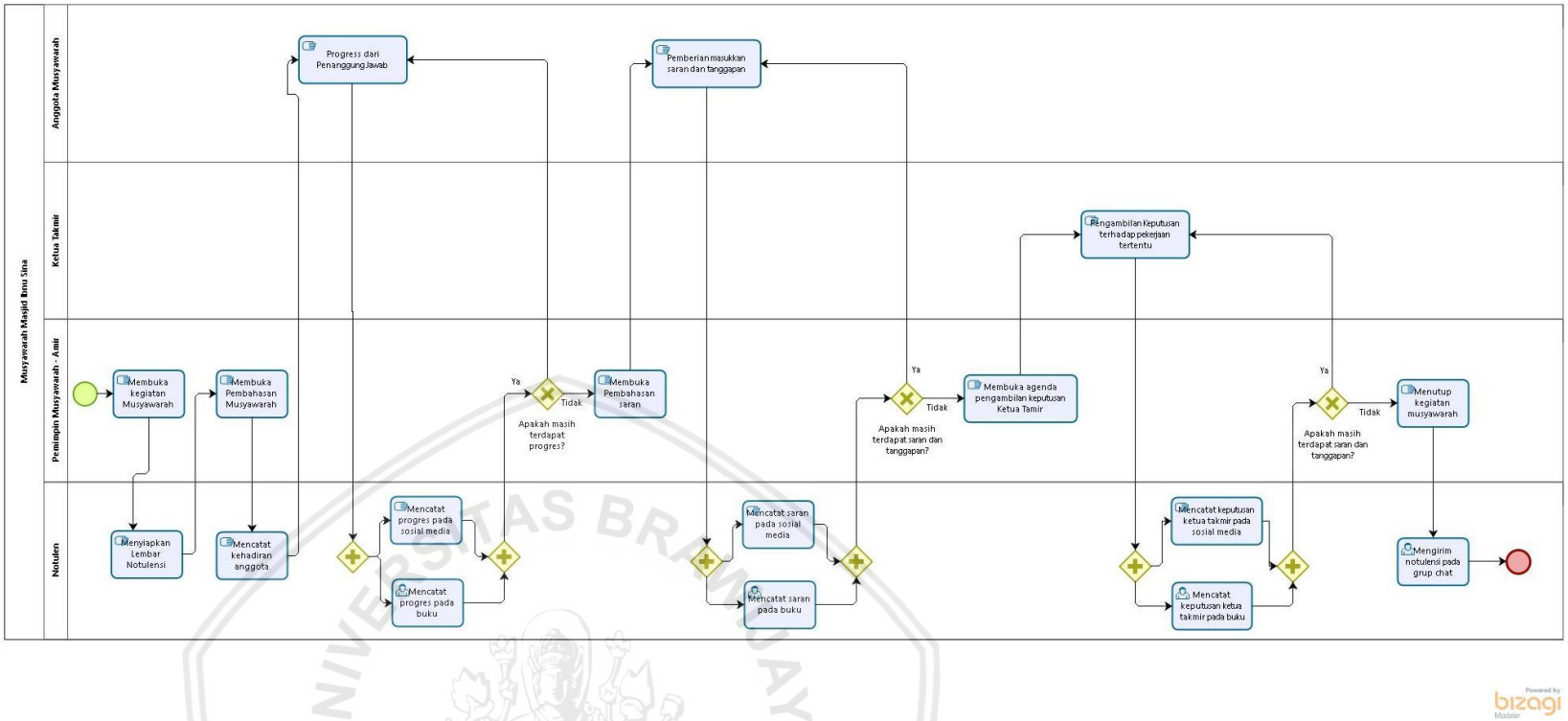
4.4.1.2 Proses Bisnis *As-Is* Mengelola Pekerjaan Masjid (PBA-AMM-02)

Pengelolaan pekerjaan masjid mempunyai sifat yang sama dengan pengelolaan keanggotaan musyawarah. Takmir tidak memiliki catatan khusus dalam melakukan pemantauan dan meihat informasi pekerjaan yang sedang berjalan. Penjabaran progres pekerjaan masjid dilakukan pada saat musyawarah masjid. Selain itu musyawarah juga dapat memastikan pekerjaan yang sedang berjalan.

4.4.1.3 Proses Bisnis *As-Is* Musyawarah Masjid (PBA-AMM-03)

Kegiatan musyawarah masjid merupakan proses penyampaian laporan mingguan yang diadakan oleh Takmir dan Remas Masjid. Kegiatan ini berlangsung setiap ba'da subuh hari sabtu yang dilakukan selama 1-2 jam. Kegiatan ini dimulai dengan dibuka oleh Amir Musyawarah. Amir Musyawarah berbeda setiap minggunya berdasarkan urutan yang sudah ditetapkan. lalu Amir Musyawarah membuka musyawarah dengan salam dan penyampaian dakwah singkat yang diberikan oleh anggota musyawarah. Kemudian Amir Musyawarah membuka pembahasan pertama yaitu laporan dari setiap anggota secara bergiliran. Setiap anggota diperbolehkan untuk menyampaikan pekerjaan apa saja yang telah dikerjakan, kendala, dan masukan dari jamaah masjid lainnya terhadap masjid. Kemudian Amir Musyawarah masjid membuka pembahasan kedua yaitu saran terhadap laporan. Saran dilakukan seperti penjabaran laporan, yaitu dilakukan secara bergilir tetapi dapat menanggapi laporan anggota lain yang sudah disampaikan sebelumnya. Setiap laporan dan saran akan dicatat oleh Amir Musyawarah pada buku notulensi dan juga *chat* grup. Saran diberikan untuk mendapat sudut pandang tambahan untuk Amir Musyawarah agar mendapat keputusan secara mufakat. Setelah semua saran diberikan oleh anggota musyawarah, Amir Musyawarah membuka agenda keputusan. Semua pembahasan akan dibahas oleh notulen untuk memastikan bahwa laporan tidak terlewat. Keputusan dapat berupa pekerjaan tambahan atau arahan tertentu dalam pekerjaan. Setelah musyawarah diselesaikan, Amir Musyawarah masjid akan menutup dengan doa penutup majelis. Lalu notulen yang dicatat melalui grup *chat* akan di rekap dan dikirim pada grup. Proses ini digambarkan dalam Gambar 4.2.

Berdasarkan observasi yang dilakukan dapat dianalisis proses kegiatan musyawarah masjid terdapat 5 fase yang berjalan saat musyawarah yaitu pembuka musyawarah, pelaporan pekerjaan masjid, tanggapan serta saran, pengambilan keputusan ketua takmir, dan penutup. Fase pembuka musyawarah merupakan awal dari kegiatan musyawarah untuk memulai kegiatan musyawarah yang dipimpin oleh Amir Musyawarah. Pada fase pembuka notulen juga mulai mempersiapkan buku untuk pencatatan tertulis dan membuka grup sosial media untuk pencatatan *online*. Fase pelaporan pekerjaan masjid merupakan kegiatan seluruh anggota musyawarah



Gambar 4.2 Proses Bisnis *As-Is* Musyawarah Masjid

menyampaikan laporan atau capaian pada pekerjaan tertentu yang sudah dikerjakan maupun hambatan selama kegiatan. Laporan diberikan oleh masing masing penanggung jawab urutan dari laporan bersifat bebas. Fase berikutnya merupakan fase tanggapan dan saran. Fase ini dari setiap Takmir diperbolehkan untuk memberi saran dan tanggapan kepada penanggung jawab tertentu, saran dan tanggapan diperbolehkan untuk disampaikan saat Amir Musyawarah memberi waktu kepada anggota yang ingin memberikan tanggapan dan saran. Kemudian terdapat fase pengambilang keputusan, pengambilan keputusan pekerjaan ditentukan oleh ketua takmir untuk langkah pekerjaan barikutnya maupun tambahan pekerjaan baru bagi penanggung jawab. Lalu terdapat fase penutup yaitu ditutupnya kegiatan musyawarah oleh Amir Musyawarah.

Terdapat beberapa proses yang dapat dievaluasi dalam kegiatan musyawarah masjid. Proses penjabaran laporan pekerjaan hanya dilakukan untuk menyampaikan laporan, kesalahan terdapat pada laporan pada musyawarah sebelumnya tidak di perhatikan untuk memastikan keputusan sebelumnya sudah diselesaikan atau belum. Hal ini menyebabkan ketidak efektifnya tujuan musyawarah untuk menjabarkan selesainya pekerjaan yang dilakukan. Permasalahan berikutnya terdapat pada duplikasi proses notulensi yang dilakukan saat musyawarah. Notulensi dikerjakan oleh dua orang yang berbeda untuk mencatat pada buku dan pada grup *chat*. Hal tersebut merupakan proses yang menggunakan sumber daya yang berlebih dalam mencatat notulensi. Lalu hal tersebut juga dapat meningkatkan kerancuan notulensi karena memiliki perbedaan informasi yang dicatat pada buku maupun grup *chat*.

4.4.2 Pemodelan Proses Bisnis *To-Be*

4.4.2.1 Proses Bisnis *To-Be* Mengelola Keanggotaan Musyawarah (PBT-AMM-01)

Proses mengelola keanggotaan musyawarah juga merupakan proses bisnis usulan tambahan, untuk mendukung proses bisnis *to-be* mengelola pekerjaan dan proses bisnis manajemen notulensi. Dengan adanya proses bisnis ini, aktor dapat mendaftarkan keanggotaannya untuk diberikan batasan sistem yang akan disesuaikan pada perannya. Keanggotaan juga dapat menunjang proses bisnis notulensi untuk mendata anggota yang hadir serta menunjuk Amir Musyawarah masjid yang memimpin. Selain itu proses bisnis ini juga membantu dalam mengelola pekerjaan yang diberikan tanggung jawab berdasarkan masing masing anggota musyawarah.

Proses ini dimulai oleh aktor sekretaris dan ketua takmir dalam melakukan melihat anggota yang sudah terdaftar pada sistem. Kemudian sekretaris dan ketua takmir dapat menambah anggota dengan menambahkan formulir tambah anggota. Apabila sudah menambah anggota, sistem menyimpan data kemudian ditampilkan pada sistem. Untuk anggota yang sudah terdaftar, sekretaris dan ketua dapat

melakukan manipulasi anggota seperti mengedit maupun menghapus anggota musyawarah. Proses ini digambarkan dalam Gambar 4.3.

4.4.2.2 Proses Bisnis *To-Be* Mengelola Pekerjaan Masjid (PBT-AMM-02)

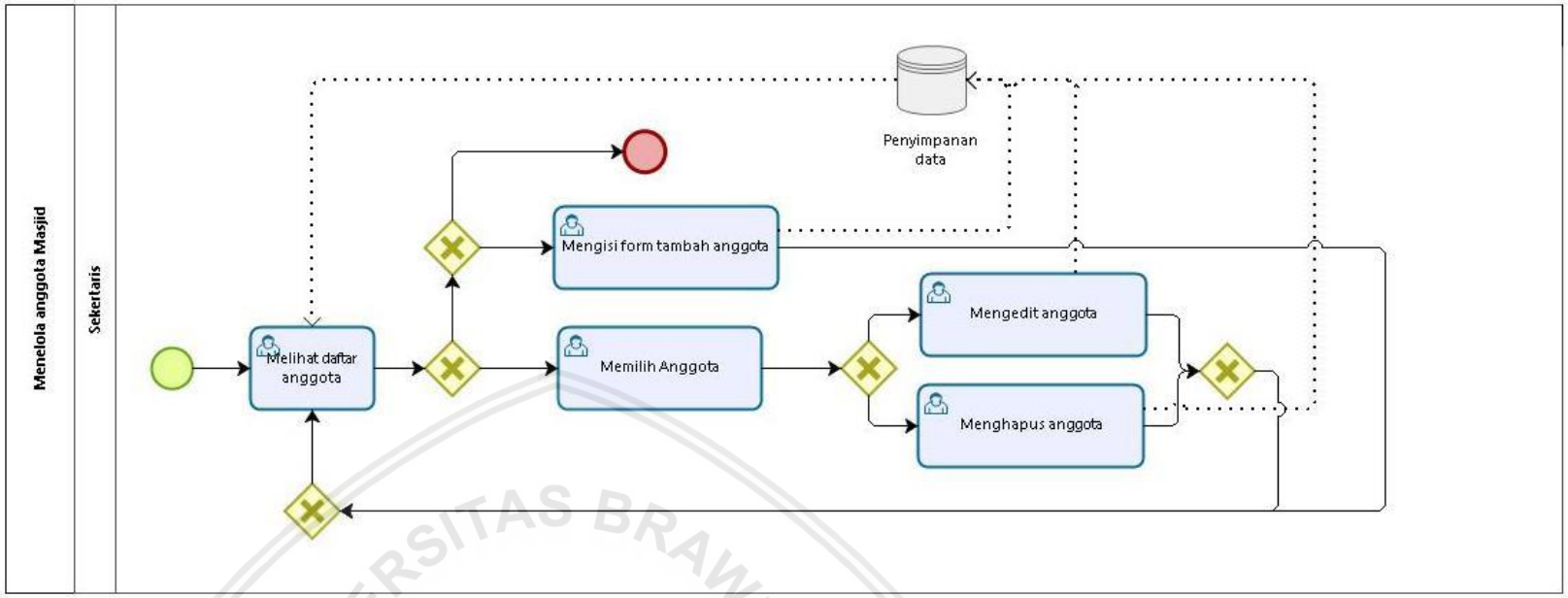
Proses bisnis mengelola pekerjaan masjid merupakan proses bisnis usulan tambahan untuk menunjang persyaratan proses pemantauan manajemen notulensi pada setiap poin pekerjaan. Selain itu untuk mendukung musyawarah secara khusus, pembahasan melalui pekerjaan masjid akan lebih detail dan efektif. Contoh dalam pembahasan khusus seperti persiapan bulan Ramadhan akan lebih mendetail dalam proses pengelolaan pekerjaan.

Proses ini dimulai Takmir atau Remaja masjid dapat memilih untuk menambahkan pekerjaan. Pada saat menambahkan pekerjaan, ketua takmir harus menverifikasi pekerjaan tersebut untuk memastikan pekerjaan merupakan pekerjaan penting untuk kemakmuran jamaah. Kemudian apabila sudah terdapat pekerjaan yang terverifikasi, pekerjaan dapat dikelola, baik dalam mengedit, menghapus, menambah informasi, maupun menutup pekerjaan menjadi status selesai. Proses ini digambarkan dalam Gambar 4.4.

4.4.2.3 Proses Bisnis *To-Be* Musyawarah Masjid (PBT-AMM-03)

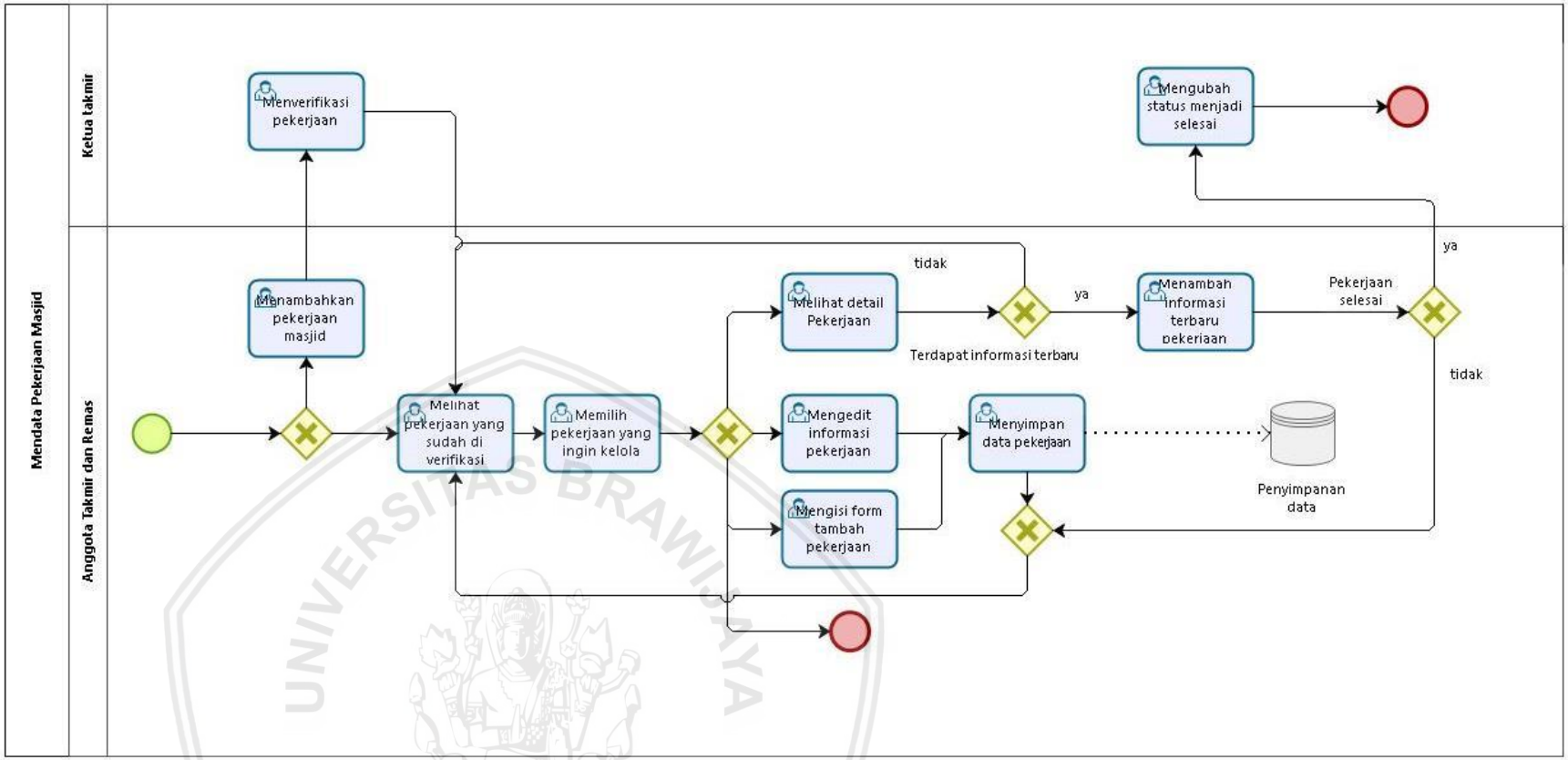
Dalam Gambar 4.5 merupakan proses bisnis usulan dalam melakukan kegiatan musyawarah masjid. Proses ini merupakan perbaikan proses bisnis *as-is* yang dianalisis dan disolusikan dengan pendekatan IT dan studi literatur musyawarah. Proses bisnis ini memiliki beberapa aktor yaitu Notulen, Amir Musyawarah, Ketua Amir Musyawarah, Anggota Musyawarah. Proses ini dimulai dengan dibukanya agenda musyawarah oleh Amir Musyawarah masjid. lalu notulen membuka aplikasi untuk menambah notulensi baru yang akan disimpan. Amir Musyawarah kemudian membuka pembahasan laporan. Saat dibukanya pembahasan notulen mencatat kehadiran anggota musyawarah yang hadir. Pencatatan kehadiran digunakan untuk memastikan pada penelusuran notulensi terdapat saksi yang memastikan bahwa laporan benar. Lalu laporan dimulai dari masing-masing penanggung jawab secara bergiliran yang setiap laporannya dicatat oleh notulen musyawarah. Setelah semua melakukan laporan, dilanjutkan dengan agenda pemberian saran dan tanggapan terhadap laporan yang sudah disampaikan. Setelah semua laporan tersampaikan agenda berikutnya adalah pengambilan keputusan yang dilakukan oleh ketua takmir masjid untuk memutuskan kegiatan tambahan atau arahan yang akan dilakukan. Setiap keputusan dimulai dengan penjelasan laporan yang dicatat oleh notulen yang kemudian diputuskan oleh ketua masjid. notulen mencatat semua keputusan dari setiap laporan yang ada. Setelah semua keputusan disampaikan, Amir Musyawarah masjid menutup musyawarah masjid, lalu notulen menyimpan notulensi.

repository.ub.ac.id



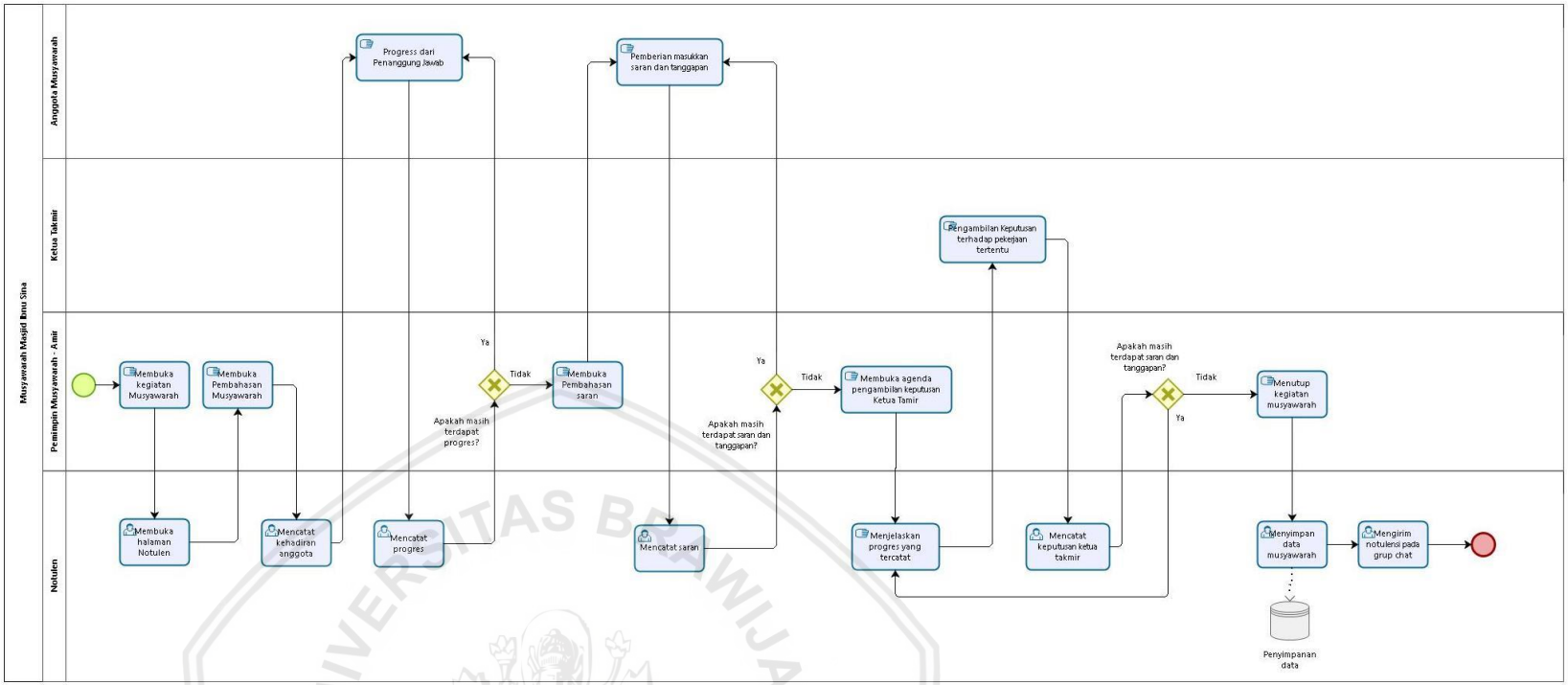
Gambar 4.3 Proses Bisnis *to-be* Pengelolaan Anggota Masjid

repository.ub.ac.id



Gambar 4.4 Proses Bisnis *to-be* Mengelola Pekerjaan Masjid

repository.ub.ac.id



Gambar 4.5 Proses Bisnis *to-be* Musyawarah Masjid

Notulensi yang tersimpan diperiksa oleh seluruh anggota musyawarah dan memberikan masukan untuk perubahan notulensi. Amir Musyawarah masjid akan mendapat hak untuk melakukan verifikasi notulensi setelah perubahan yang diperbaiki.

4.4.3 Analisis Perbaikan Proses Bisnis

Analisis perbaikan proses bisnis dilakukan untuk membandingkan proses bisnis *as-is* dan proses bisnis *to-be* yang sudah dirancang. Proses ini akan mempermudah untuk mengetahui proses bisnis yang dilakukan perbaikan. Terdapat proses bisnis usulan yang ditambahkan dan diubah berdasarkan persyaratan aktor menjalankan kegiatan musyawarah. Analisis ini dijabarkan pada Tabel 4.3.

4.4.4 Analisis Persyaratan

4.4.4.1 Identifikasi Tipe Pemangku Kepentingan

Hasil identifikasi tipe pemangku kepentinkan berupa daftar karakteristik pemangku kepentingan yang ada pada Masjid Ibnu Sina. Hasil identifikasi ini akan digunakan sebagai informasi untuk menganalisis kegiatan yang akan dilakukan oleh pengguna untuk berinteraksi pada sistem.

Tabel 4.2 Hasil Identifikasi Tipe Pemangku Kepentingan

Tipe Pemangku Kepentingan	Deskripsi	Pemangku Kepentingan
Pengguna	Orang yang menggunakan sistem secara langsung dan berperan sebagai aktor dalam *use case*	- Ketua Takmir - Sekretaris Takmir - Takmir dan Remas - Notulen - Amir Musyawarah

repository.ub.ac.id

Tabel 4.3 Analisis Perbedaan Proses Bisnis As-Is dan To-Be serta Waktu yang Dibutuhkan

No.	Aktor *As-Is*	Proses Bisnis *As-Is*	Aktor *To-Be*	Proses Bisnis *To-Be*	Waktu Proses	Perkiraan Waktu Proses Baru	Keterangan
1.	Pemimpin Musyawarah - Amir Musyawarah, Ketua Takmir, Anggota Takmir dan Remas, Notulen	Pencatatan Pembahasan Musyawarah Masjid menggunakan buku dan grup *chat* (PBA-AMM-03)	Pemimpin Musyawarah - Amir Musyawarah, Ketua Takmir, Anggota Takmir dan Remas, Notulen	Pencatatan Pembahasan Musyawarah Masjid menggunakan sistem (PBT-AMM-03)	15 menit	10 menit	Diubah
2.	Ketua Takmir, Anggota Takmir dan Remas	Pencatatan pekerjaan masjid. (PBA-AMM-02)	Ketua Takmir, Anggota Takmir dan Remas	Pencatatan Pekerjaan Masjid. (PBT-AMM-02)	-	20 detik	Diubah
3.	Ketua takmir dan Sekretaris	Pencatatan anggota takmir. (PBA-AMM-01)	Sekretaris	Pencatatan keanggotaan takmir. (PBT-AMM-01)	-	20 detik	Diubah

4.4.4.2 Identifikasi Kebutuhan Pemangku Kepentingan dan Pengguna

Identifikasi kebutuhan pemangku kepentingan dan pengguna menjelaskan sekumpulan pernyataan yang berhubungan dengan permasalahan yang ada pada tabel pernyataan masalah. Hasil identifikasi kebutuhan pemangku kepentingan dan pengguna digunakan sebagai dasar informasi pada tahap identifikasi fitur. Hasil identifikasi kebutuhan pemangku kepentingan dan pengguna dijelaskan pada Tabel 4.4.

Tabel 4.4 Hasil Identifikasi Kebutuhan Pengguna

Kode Kebutuhan	Kebutuhan Pengguna	Pemangku Kepentingan	Situasi Saat Ini	Solusi Yang Ditawarkan
KB-AMM-01	Sistem harus menyediakan layanan untuk mengelola data anggota	Ketua Takmir dan Sekretaris Takmir	Proses penyimpanan data anggota tidak disimpan secara administratif	Menyediakan sistem untuk menyimpan data anggota aktif pada sistem yang dapat di manajemen oleh ketua dan sekretaris takmir
KB-AMM-02	Sistem harus menyediakan layanan untuk mengelola data pekerjaan masjid	Takmir dan Remas	Proses pendataan pekerjaan yang berjalan tidak disimpan secara administratif	Menyediakan sistem untuk menyimpan data pekerjaan yang dapat digunakan untuk menjadi sarana pelaporan kegiatan yang berjalan
KB-AMM-03	Sistem harus menyediakan layanan untuk melihat data notulensi	Takmir dan Remas	Proses melihat notulensi musyawarah melalui buku dan grup sosial media	Menyediakan sistem untuk melihat data notulensi seperti laporan,

	musyawarah masjid			masukan, dan keputusan
KB-AMM-04	Sistem harus menyediakan layanan untuk mengelola notulensi musyawarah masjid	Notulen dan Amir Musyawarah	Proses mengelola notulensi berjalan pada buku dan grup sosial media	Menyediakan sistem untuk mengelola data notulensi untuk dicatat pelaporan, usulan, dan pembahasan

4.4.4.3 Identifikasi Pengguna

Identifikasi pengguna bertujuan untuk mengetahui peran-peran yang akan digunakan pada sistem. Identifikasi pengguna merupakan pengkategorian pengguna berdasarkan hasil analisis pemangku kepentingan. Identifikasi pengguna akan digunakan sebagai aktor dalam *use case* diagram.

Tabel 4.5 Hasil Identifikasi Pengguna

Tipe Pemangku Kepentingan	Tipe Pengguna	Deskripsi
Pengguna	Ketua Takmir	Seseorang yang memiliki tanggung jawab terhadap pengelolaan anggota musyawarah masjid
	Sekretaris Takmir	Seseorang yang memiliki tanggung jawab terhadap pengelolaan anggota musyawarah masjid
	Notulen	Seseorang yang memiliki tanggung jawab terhadap pencatatan notulensi saat musyawarah yang berlangsung
	Amir Musyawarah	Seseorang yang memiliki tanggung jawab terhadap berlangsungnya musyawarah masjid dan pengambilan keputusan musyawarah masjid
	Takmir dan Remas	Seseorang yang memiliki bagian dalam pelaksanaan musyawarah masjid

4.4.4.4 Identifikasi Fitur

Identifikasi fitur pada penelitian ini merupakan solusi yang ditawarkan dalam menyelesaikan permasalahan yang didefinisikan pada analisis permasalahan. Pada Tabel 4.6 Hasil Identifikasi Fitur terdapat kode fitur dan deskripsi singkat fitur yang akan menggambarkan terdapat layanan yang terdapat pada sistem. Hasil identifikasi fitur digunakan sebagai informasi dalam melakukan identifikasi persyaratan fungsional dan nonfungsional pada sistem informasi.

Tabel 4.6 Hasil Identifikasi Fitur

Kode Fitur	Deskripsi
FT-AMM-01	Sistem dapat mengenali identitas pengguna dan membatasi akses pengguna sesuai peran dan tanggungjawab.
FT-AMM-02	Sistem mampu untuk mengelola keanggotaan takmir masjid. sistem dapat menambah anggota, menghapus anggota, mengedit informasi anggota, dan menampilkan keanggotaan takmir masjid.
FT-AMM-03	Sistem mampu untuk mengelola informasi pekerjaan masjid. Sistem dapat menambah pekerjaan masjid, menambah informasi pekerjaan masjid, menghapus pekerjaan masjid, mengedit pekerjaan masjid, dan menampilkan informasi pekerjaan masjid.
FT-AMM-04	Sistem mampu untuk menampilkan data detail notulensi.
FT-AMM-05	Sistem mampu untuk mencatat, mengubah dan menghapus notulensi.

4.4.4.5 Persyaratan Fungsional

Persyaratan fungsional merupakan penjabaran kondisi yang harus dipenuhi dalam fitur yang sudah didefiniskan. Persyaratan fungsional merupakan penjabaran detail dari fitur, penjabaran ini akan memudahkan peneliti dalam memenuhi rangka kemampuan fitur. Hubungan antara fungsional dan fitur dijelaskan pada Tabel 4.7.

Tabel 4.7 Hasil Identifikasi Persyaratan Fungsional

Kode Persyaratan Fungsional	Deskripsi	Kode Fitur
FTF-AMM-01	Sistem mampu menampilkan halaman otentikasi pengguna.	FT-AMM-01
FTF-AMM-02	Sistem mampu mengotentikasi pengguna berdasarkan data yang tersimpan dalam sistem.	FT-AMM-01
FTF-AMM-03	Sistem mampu mengganti keterangan *session* berdasarkan pengguna yang sedang aktif atau masuk pada sistem.	FT-AMM-01
FTF-AMM-04	Sistem mampu menghentikan *session* pengguna yang sedang aktif.	FT-AMM-01
FTF-AMM-05	Sistem mampu menampilkan semua anggota takmir masjid.	FT-AMM-02
FTF-AMM-06	Sistem mampu menambah anggota takmir masjid.	FT-AMM-02
FTF-AMM-07	Sistem mampu mengedit anggota takmir masjid.	FT-AMM-02
FTF-AMM-08	Sistem mampu menghapus anggota takmir masjid.	FT-AMM-02
FTF-AMM-09	Sistem mampu menampilkan semua pekerjaan masjid.	FT-AMM-03
FTF-AMM-10	Sistem mampu menampilkan detail pekerjaan masjid.	FT-AMM-03
FTF-AMM-11	Sistem mampu menambah informasi detail pekerjaan masjid.	FT-AMM-03
FTF-AMM-12	Sistem mampu menambah pekerjaan masjid.	FT-AMM-03
FTF-AMM-13	Sistem mampu mengedit pekerjaan masjid.	FT-AMM-03
FTF-AMM-14	Sistem mampu menghapus pekerjaan masjid.	FT-AMM-03
FTF-AMM-15	Sistem mampu menambah notulensi musyawarah baru.	FT-AMM-05
FTF-AMM-16	Sistem mampu menyimpan notulensi musyawarah.	FT-AMM-05
FTF-AMM-17	Sistem mampu menampilkan daftar notulensi musyawarah.	FT-AMM-05

Kode Persyaratan Fungsional	Deskripsi	Kode Fitur
FTF-AMM-18	Sistem mampu menampilkan detail informasi notulensi musyawarah.	FT-AMM-04
FTF-AMM-19	Sistem mampu menghapus notulensi musyawarah.	FT-AMM-04
FTF-AMM-20	Sistem mampu mengubah informasi notulensi musyawarah.	FT-AMM-05

4.4.4.6 Persyaratan Nonfungsional

Persyaratan nonfungsional merupakan hasil identifikasi yang tidak secara langsung berhubungan dengan fitur utama. Persyaratan nonfungsional akan digunakan untuk mendukung berjalannya sistem yang akan digunakan. Persyaratan nonfungsional memiliki kode fitur, kode persyaratan, dan deskripsi persyaratan nonfungsional. Hasil identifikasi persyaratan nonfungsional dijelaskan pada Tabel 4.6.

Tabel 4.8 Hasil Identifikasi Persyaratan Nonfungsional

Kode Persyaratan Nonfungsional	Deskripsi	Kode Fitur
FTNF-AMM-01	Sistem dapat berjalan pada beberapa versi aplikasi peramban tanpa ada kendala.	FT-AMM-01
		FT-AMM-02
		FT-AMM-03
		FT-AMM-04
		FT-AMM-05

4.4.5 Pemodelan *Use Case*

Pemodelan *use case* dilakukan untuk menggambarkan peran aktor memiliki kemampuan dalam melakukan aktivitas atau kegiatan saat menggunakan sistem. Saat melakukan pemodelan *use case* terdapat identifikasi aktor, identifikasi *use case*, dan spesifikasi *use case*. Identifikasi aktor digunakan untuk mengetahui kelompok sistem yang terdapat pada sistem informasi berdasarkan karakteristik yang dimiliki oleh aktor. Identifikasi *use case* dilakukan untuk menentukan tujuan aktor dalam menggunakan sistem informasi berdasarkan *use case* yang ada. Sedangkan spesifikasi *use case* adalah penjelasan alur *use case* dengan mendefinisikan kondisi yang

dibutuhkan sebelum dijalankan, alur *use case* saat berjalan, dan hasil dari *use case* berjalan.

4.4.5.1 Deskripsi Aktor

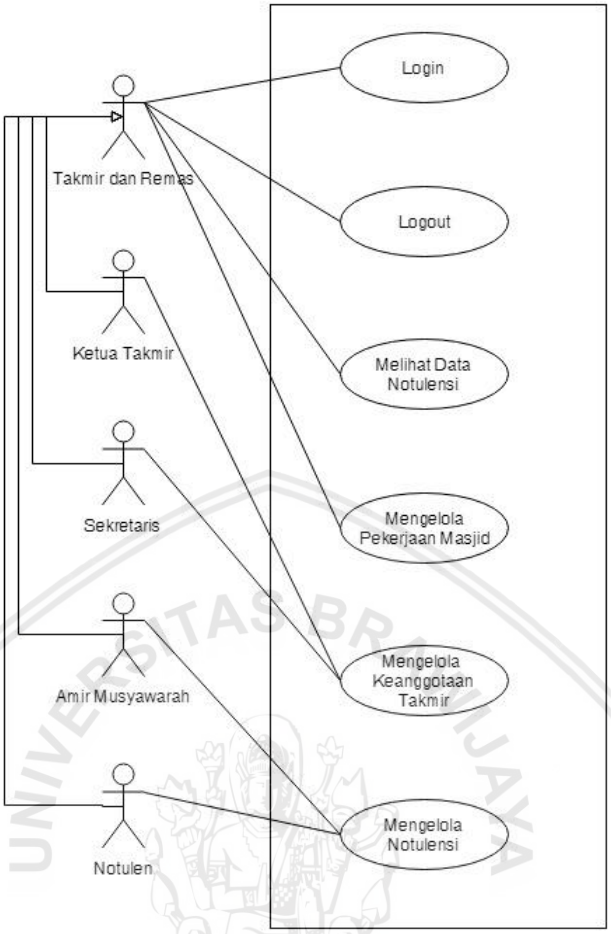
Deskripsi aktor merupakan penjelasan secara singkat aktor-aktor yang sudah diidentifikasi dalam *use case* diagram. Deskripsi aktor berisi tanggung jawab dalam menggunakan sistem. Deskripsi ini dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil Identifikasi Aktor

No	Nama Aktor	Deskripsi Aktor
1.	Takmir dan Remas	Aktor Anggota takmir yang diperankan oleh anggota takmir memiliki batasan untuk melakukan melihat notulensi, pengelolaan pekerjaan masjid, dan melakukan login dan logout.
2.	Ketua Takmir dan Sekretaris Takmir	Aktor Ketua takmir dan Sekretaris Takmir masjid memiliki batasan untuk menambah, menghapus, mengedit anggota takmir. Sekretaris Takmir juga memiliki kemampuan fungsi yang sama seperti anggota Takmir dan Remas
4.	Amir Musyawarah dan Notulen	Aktor Amir Musyawarah dan Notulen musyawarah memiliki batasan untuk mengelola notulensi sesuai dengan notulen yang dibuat oleh aktor notulen. Notulen juga memiliki kemampuan fungsi yang sama seperti anggota Takmir dan Remas

4.4.5.2 *Use Case* Diagram

Use case diagram memberikan gambaran hubungan antara aktor dengan kemampuan persyaratan fungsional pada sistem. Dalam Gambar 4.6 ditunjukan pengelompokan dari setiap aktor dengan use case yang terdapat pada sistem informasi.



Gambar 4.6 Use Case Diagram

Kemudian, hubungan antara bisnis *to-be* dengan *use case* dapat kita hubungkan dengan *use case* yang didefinisikan. Tujuan dari penghubungan ini untuk menunjukan hubungan antara pemodelan proses bisnis dan *use case* yang dibuat.

Tabel 4.10 Hubungan Proses Bisnis *To-Be* dan *Use Case* Diagram

No	Proses Bisnis *To-Be*	Aktivitas	*Use Case*
1.	Kegiatan Musyawarah Masjid (PBT-AMM-03)	Menampilkan data notulensi yang tersimpan. Mencatat, mengubah, dan menghapus dari notulensi yang tersimpan	Melihat data Notulensi. (UC-AMM-03)
			Mengelola Notulensi. (UC-AMM-06)

No	Proses Bisnis *To-Be*	Aktivitas	*Use Case*
3.	Mengelola Pekerjaan Masjid. (PBT-AMM-02)	Menambah pekerjaan masjid, mengedit pekerjaan masjid, menghapus dan menampilkan detail data pekerjaan yang tersimpan.	Mengelola Pekerjaan Masjid. (UC-AMM-04)
4.	Mengelola keanggotaan takmir. (PBT-AMM-01)	Menambah anggota masjid, mengedit pekerjaan masjid, menghapus dan menampilkan anggota yang terdaftar .	Mengelola keanggotaan takmir. (UC-AMM-05)

Setelah menghubungkan *use case* dan proses bisnis *to-be*, kemudian peneliti menghubungkan *use case* dan pengguna yang bertujuan untuk memahami kemampuan pengguna dalam penggunaan sistem. Hubungan ini dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil Identifikasi Use Case Berdasarkan Pengguna

No	Kode	*Use case*	Pengguna
1.	UC-AMM-01	Login	Anggota Takmir dan Remas.
2.	UC-AMM-02	Logout	Anggota Takmir dan Remas.
3.	UC-AMM-03	Melihat Data Notulensi	Anggota Takmir dan Remas.
4.	UC-AMM-04	Mengelola Pekerjaan Masjid	Anggota Takmir dan Remas.
5.	UC-AMM-05	Mengelola keanggotaan Masjid	Ketua Takmir dan Sekretaris Takmir.
6.	UC-AMM-06	Mengelola notulensi	Notulen dan Amir Musyawarah.

Lalu, dilakukan identifikasi hubungan dengan *use case* dengan fitur yang sudah dijabarkan sebelumnya. Hal ini bertujuan untuk menunjukan hasil pemodelan *use case* sesuai dengan hasil analisis persyaratan yang dijelaskan. Hubungan *use case* dengan fitur ditunjukan pada Tabel 4.12.

Tabel 4.12 Hasil Identifikasi Hubungan Use Case dengan Kode Fitur

No	*Use Case*	Fitur
1.	Login	FT-AMM-01
2.	Logout	FT-AMM-01
3.	Mengelola Keanggotaan.	FT-AMM-02
4.	Mengelola Pekerjaan Masjid.	FT-AMM-03
5.	Melihat Data Notulensi	FT-AMM-04
6.	Mengelola Notulensi	FT-AMM-05

4.4.5.3 Spesifikasi *Use Case*

Spesifikasi *use case* berisi penjelasan mengenai setiap *use case* yang didefinisikan. Setiap spesifikasi use case terdapat tabel yang berisi *brief description*, aktor, *pre-condition*, *post-condition*, *basic flow*, *alternative flow* dan *sub flow*. *Brief description* berisi informasi dasar yang menunjukan kemampuan *use case*. Kemudian pada kolom aktor terdapat informasi aktor yang memiliki peran yang dapat melakukan *use case* tersebut. pada kolom *pre-condition* terdapat informasi yang dibutuhkan sebelum memulai *basic flow* tersebut, *pre-condition* sudah harus terpenuhi dalam memulai *basic flow*. *Basic flow* merupakan tahapan yang dilakukan oleh aktor dalam menjalankan *use case* tersebut. *Basic flow* merupakan langkah dalam mendapatkan tujuan utama dari masing masing use case. Tujuan utama dari masing-masing definisikan dalam kolom *post condition*. Kolom *post-condition* berisi informasi tujuan utama dari *use case* tersebut, kolom ini menunjukan hasil dari *use case* tersebut. kemudian terdapat *alternative-flow* merupakan *error-handling* saat basic flow tidak terpenuhi dalam sistem. *Sub-flow* merupakan penyederhanaan alur normal *use case* yang terlalu terperinci.

(a) Spesifikasi *Use Case Login*

Spesifikasi *use case* login menjabarkan tujuan bagaimana aktor masuk atau teridentifikasi kedalam sistem. Aktor akan di identifikasi yang kemudian akan disesuaikan dengan hak ases sesuai dengan tujuannya. Spesifikasi ini berisi urutan kegiatan yang dilakukan oleh aktor saat menggunakan sistem. Penjelasan spesfikasi ini dijabarkan dalam Tabel 4.13.

Tabel 4.13 Spesifikasi *Use Case Login*

Use case code	UC-AMM-01
Brief Description	*Use case login* menjelaskan bagaimana aktor pengguna masuk ke dalam sistem sesuai hak akses dan identitas yang sudah dikenali sistem untuk keamanan informasi.
Aktor	Pengguna
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Data aktor sudah tersimpan dalam sistem.
Post-condition	- Identitas aktor teridentifikasi oleh sistem. - Aktor berhasil masuk ke dalam sistem. - Sistem menampilkan informasi sesuai hak akses yang dimiliki oleh aktor.
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih masuk ke dalam sistem. **{isi data}** 2. Aktor mengisi *username* dan *password.* 3. Aktor mengirimkan *username* dan *password.* **{cek isian}** **{sistem mengautentikasi dan mengautorisasi}** 4. Aktor masuk kedalam dashboard. **{*use case* selesai}** 5. *Use case* selesai.
Alternative Flow	**A1. Menangani *Username* atau *Password* Kosong** Pada **{cek isian}** apabila Pengguna tidak mengisi salah satu dari kedua isian tersebut, maka sistem akan menampilkan pesan *error* bahwa *username* dan *password* kosong dan wajib disisi, kemudian kembali ke **{isi data}**. **A2. Menangani *Username* atau *Password* Salah**

	Pada **{mengidentifikasi pengguna}** apabila *username* dan *password* pengguna salah atau tidak diidentifikasi oleh sistem, maka sistem akan menampilkan pesan *error* bahwa *username* dan *password* salah, kemudian *use case* selesai.

(b) Spesifikasi *Use Case Logout*

Spesifikasi *use case* logout menjabarkan tujuan bagaimana aktor keluar atau menghapus identifikasi yang sudah dimasukan pada login. Aktor yang teridentifikasi akan keluar lalu identifikasi peran akan dicabut sehingga hak akses akan ditiadakan pada aktor tersebut. spesifikasi ini berisi urutan kegiatan yang dilakukan oleh aktor dalam melakukan logout pada sistem. Penjelasan spesifikasi ini dijelaskan pada Tabel 4.14.

Tabel 4.14 Spesifikasi *Use Case Logout*

Use case code	UC-AMM-02
Brief Description	*Use case logout* menjelaskan bagaimana aktor pengguna mengakhiri sesinya pada sistem. Sehingga sistem tidak akan menggunakan akun yang tersimpan saat membuka halaman karena tidak terdapat adanya sesi yang disimpan.
Aktor	Pengguna
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sesi aktor masih disimpan pada sistem
Post-condition	- Aktor keluar dari system - Sistem tidak menyimpan sesi aktor
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih menu keluar. **{Sistem menghapus autorisasi actor}** **{*use case* selesai}** 2. *Use case* selesai.
Alternative Flow	-

(c) Spesifikasi *Use Case* Mengelola Keanggotaan Masjid

Spesifikasi keanggotaan berisi langkah yang harus dilakukan dalam pengelolaan anggota masjid. pengelolaan masjid dapat berupa melihat daftar anggota, menambah anggota, mengedit anggota, dan menghapus anggota pada sistem. Spesifikasi ini memiliki dapat dijalankan pada aktor Ketua Takmir dan Sekretaris Takmir. Spesifikasi ini dijelaskan pada Tabel 4.15.

Tabel 4.15 Spesifikasi *Use Case* Mengelola Keanggotan Masjid

Use case code	UC-AMM-05
Brief Description	*Use case* Mengelola keanggotaan takmir masjid menjelaskan bagaimana aktor Ketua Takmir, Sekretaris Takmir menampilkan daftar anggota masjid, menampilkan detail anggota takmir masjid, menambah anggota takmir masjid, mengubah anggota takmir masjid, dan menghapus anggota takmir masjid
Aktor	Ketua Takmir atau Sekretaris Takmir
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Ketua Takmir atau Anggota Takmir telah berhasil masuk ke dalam sistem.
Post-condition	- Sistem berhasil menampilkan, menyimpan, mengedit, menghapus data anggota takmir masjid
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih untuk membuka daftar takmir. **{Menampilkan daftar anggota takmir}** 2. Aktor membuka formulir tambah takmir. **{Menampilkan formulir tambah takmir}** 3. Aktor mengisi formulir tambah anggota takmir. **{Isi data takmir}** 4. Aktor melakukan pengisian formulir nama, jabatan.

	5. Aktor menyimpan data anggota takmir **{Sistem menyimpan anggota baru}** 6. Sistem akan menampilkan daftar takmir 7. Jika Aktor ingin melakukan pengeditan data, Aktor dapat menjalankan sub-flow S1 8. Jika Aktor ingin melakukan penghapusan anggota, Aktor dapat menjalankan sub-flow S2. 9. Sistem akan menampilkan daftar takmir **{*use case* selesai}** 10. *Use case* selesai.
Alternative Flow	**A1. Sistem Gagal Menyimpan data anggota baru** Pada **{Sistem menyimpan anggota baru}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{Isi data takmir}.** **A2. Sistem Gagal Menyimpan edit data anggota** Pada **{Sistem menyimpan edit data anggota}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{edit data takmir}.** **A3. Sistem Gagal Menghapus data anggota** Pada **{Sistem menghapus data anggota}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{Menampilkan daftar anggota takmir}.**
Sub Flow	**S1. Sistem mengedit anggota** *Sub flow* ini digunakan untuk mengedit data anggota. 1. Aktor memilih anggota takmir yang ingin diubah **{edit data takmir}** 2. Aktor mengubah data takmir 3. Aktor menyimpan data takmir yang diubah **{Sistem menyimpan edit data anggota}** 4. *Use case* selesai.

	S2. Sistem menghapus anggota *Sub flow* ini digunakan untuk menghapus data pengguna. 1. Aktor memilih anggota yang ingin dihapus 2. Aktor menghapus anggota **{Sistem menghapus data anggota}** 3. *Use case* selesai.

(d) Spesifikasi *Use Case* Mengelola Pekerjaan Masjid

Spesifikasi *use case* mengelola pekerjaan masjid menjelaskan bagaimana aktor dapat mengelola pekerjaan masjid. pengelolaan pekerjaan masjid dapat berupa melihat pekerjaan, menambah pekerjaan, dan menghapus pekerjaan yang terdapat pada takmir masjid. pekerjaan masjid ini dapat dijalankan oleh anggota masjid. penjabaran ini berisi tahapan dalam melakukan pengelolaan pekerjaan masjid. Spesifikasi ini dijelaskan pada Tabel 4.16.

Tabel 4.16 Spesifikasi *Use Case* Mengelola Pekerjaan Masjid

Use case code	UC-AMM-04
Brief Description	*Use case* Mengelola pekerjaan masjid menjelaskan bagaimana aktor Takmir dan Remas melakukan pengelolaan data pekerjaan masjid seperti menampilkan daftar pekerjaan masjid, menampilkan detail pekerjaan masjid, menambah pekerjaan masjid, mengubah pekerjaan masjid, dan menghapus pekerjaan masjid
Aktor	Takmir dan Remas
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Takmir dan Remas telah berhasil masuk ke dalam sistem.
Post-condition	- Sistem berhasil menampilkan, menyimpan, mengedit, menghapus data pekerjaan masjid
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih untuk membuka daftar pekerjaan.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

	{Menampilkan daftar pekerjaan} 2. Aktor membuka formulir tambah pekerjaan. **{Menampilkan formulir pekerjaan}** 3. Aktor mengisi formulir tambah pekerjaan. **{Isi data}** 4. Aktor melakukan pengisian formulir nama, detail pekerjaan, dan penanggung jawab. 5. Aktor melakukan penyimpanan pekerjaan baru. **{Sistem menyimpan pekerjaan}** 6. Jika Aktor ingin menyimpan laporan pekerjaan, Aktor dapat menjalankan *sub flow* S1. 7. Jika Aktor ingin mengubah data pekerjaan, Aktor dapat menjalankan *sub flow* S2. 8. Jika Aktor ingin pekerjaan dihapus, Aktor dapat menjalankan *sub flow* S3. 9. *Use case* selesai.
Alternative Flow	**A1. Sistem Gagal Menyimpan data Pekerjaan Baru** Pada **{Sistem menyimpan pekerjaan}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{ Menampilkan formulir notulensi}.** **A2. Sistem Gagal Menyimpan Update Laporan Pekerjaan** Pada **{Sistem menyimpan update laporan pekerjaan terbaru}** apabila sistem gagal melakukan penyimpanan laporan pekerjaan maka alur *alternative* ini akan kembali ke **{Sistem menampilkan detail notulensi}.** **A3. Sistem Gagal Menyimpan Update Laporan Pekerjaan** Pada **{Sistem menyimpan status pekerjaan selesai}** apabila sistem gagal melakukan penyimpanan status pekerjaan selesai maka alur *alternative* ini akan kembali ke **{Sistem menampilkan detail notulensi}.**
Sub Flow	**S1. Sistem menyimpan laporan pekerjaan.** *Sub flow* ini digunakan untuk menyimpan data notulensi. 1. Aktor melihat detail pekerjaan.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

	{Sistem menampilkan detail notulensi} 2. Aktor menambahkan laporan pekerjaan. 3. Aktor mengirim laporan pekerjaan. **{Sistem menyimpan update laporan pekerjaan terbaru}** 4. *Use case* selesai. **S2. Sistem mengedit pekerjaan.** *Sub flow* ini digunakan untuk mengubah data notulensi. 1. Aktor membuka detail salah satu pekerjaan. **{Sistem menampilkan data pekerjaan}** 2. Aktor memilih edit pekerjaan **{isi data}** 3. Aktor mengubah data pekerjaan 4. Aktor menyimpan pekerjaan. **{Sistem menyimpan data pekerjaan}** 5. *Use case* selesai. **S3. Sistem menghapus pekerjaan** *Sub flow* ini digunakan untuk menghapus data pekerjaan. 1. Aktor membuka detail salah satu pekerjaan. **{Sistem menampilkan data pekerjaan}** 2. Aktor menghapus pekerjaan. **{Sistem menghapus data pekerjaan}** 3. *Use case* selesai.

(e) Spesifikasi *Use Case* Melihat Data Notulensi

Spesifikasi *use case* melihat data notulensi merupakan penjabaran langkah-langkah yang dilakukan dalam melihat data notulensi. Langkah melihat data notulensi terdapat melihat notulensi dan menambah komentar pada notulensi. Melihat Data notulensi ini dapat dijalankan oleh aktor Takmir dan Remas. Spesifikasi ini dijabarkan pada Tabel 4.17.

Tabel 4.17 Spesifikasi *Use Case* Melihat Data Notulensi

Use case code	UC-AMM-03
Brief Description	*Use case* Melihat Data Notulensi menjelaskan bagaimana aktor Takmir dan Remas melihat data notulensi seperti menampilkan daftar notulensi, menampilkan detail notulensi, dan memberikan komentar pada notulensi
Aktor	Takmir atau Remas
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Takmir atau Remas telah berhasil masuk ke dalam sistem.
Post-condition	- Sistem berhasil menampilkan data Notulensi
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih untuk membuka daftar notulensi. 2. Aktor membuka notulensi yang tersimpan. **{Sistem menampilkan detail notulensi}** 3. Jika Aktor ingin memberi komentar pada notulensi, Aktor dapat menjalankan *sub flow* S1. **{*use case* selesai}** 4. *Use case* selesai.
Alternative Flow	**A1. Sistem Gagal Menyimpan komentar Notulensi** Pada **{Sistem menyimpan komentar}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{Menampilkan detail notulensi}**.
Sub Flow	**S1. Sistem memberi komentar notulensi** *Sub flow* ini digunakan untuk memberi komentar pada notulensi . *1.* Aktor memberi komentar pada notulensi. *2.* Aktor menyimpan komentar pada notulensi.

	{Sistem menyimpan komentar} 3. *Use case* selesai.

(f) Spesifikasi *Use Case* Mengelola Notulensi

Spesifikasi *use case* mengelola notulensi ini merupakan penjelasan langkah dalam menambah, mengubah, dan menghapus notulensi. Mengelola notulensi ini dipisahkan dalam *use case* melihat data notulensi karena menambah, mengubah, dan menghapus notulensi hanya bisa diubah oleh Notulen dan Amir Musyawarah. Langkah dalam mengelola notulensi dijelaskan pada Tabel 4.18.

Tabel 4.18 Spesifikasi *Use Case* Mengelola Notulensi

Use case code	UC-AMM-06
Brief Description	*Use case* Mengelola Notulensi menjelaskan bagaimana aktor Notulen dan Amir Musyawarah mengelola notulensi
Aktor	Notulen atau Amir Musyawarah
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Amir Musyawarah atau Notulen telah berhasil masuk ke dalam sistem.
Post-condition	- Sistem berhasil menyimpan, mengubah, dan menghapus data notulensi
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih untuk membuka daftar notulensi. 2. Aktor membuka formulir penambahan notulensi. **{Menampilkan formulir notulensi}** 3. Aktor melakukan pemilihan notulen, Amir Musyawarah, dan peserta musyawarah yang hadir. **{Isi data}** 4. Aktor melakukan pencatatan pelaporan. 5. Aktor melakukan pencatatan usulan musyawarah.

	6. Aktor melakukan pencatatan keputusan ketua takmir. **{Menampilkan notulensi}** 7. Aktor menyimpan notulensi. **{Sistem menyimpan data notulensi}** 8. Aktor membuka notulensi yang tersimpan. **{Sistem menampilkan detail notulensi}** 9. Jika Aktor ingin memberi mengubah isi notulensi, Aktor dapat menjalankan *sub flow* S1. 10. Jika Aktor ingin menghapus notulensi, Aktor dapat menjalankan *sub flow* S2. **{*use case* selesai}** 11. *Use case* selesai.
Alternative Flow	**A1. Sistem Gagal Menyimpan data Notulensi** Pada **{Sistem *menyimpan data notulensi*}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{Menampilkan notulensi}**.
Sub Flow	**S1. Sistem mengubah isi notulensi** *Sub flow* ini digunakan untuk mengubah isi notulensi. 1. Aktor membuka notulensi **{Sistem menampilkan data notulensi}** 2. Aktor membuka formulir pengubahan notulensi **{Menampilkan formulir notulensi}** 3. Aktor mengubah isi notulensi **{Menampilkan notulensi}** 4. Aktor menyimpan data **{Sistem menyimpan data notulensi}** 5. *Use case* selesai. **S2. Sistem menghapus notulensi** *Sub flow* ini digunakan untuk menghapus data notulensi. 1. Aktor membuka notulensi

	{Sistem menampilkan data notulensi} 2. Aktor menghapus notulensi **{Sistem menghapus data notulensi}** 3. *Use case* selesai.

4.4.6 Proses Iterasi Dalam

Proses iterasi dalam merupakan proses yang dibutuhkan dalam sisi pengembangan sistem. Proses iterasi dalam dikembangkan dan diimplementasi berdasarkan identifikasi persyaratan yang sudah didefinisikan. Proses ini akan menerapkan proses iterasi, yang dilakukan secara berkala baik pada perancangan sistem dan implementasi. Pada sisi perancangan terdapat analisis sequence diagram, class diagram, pemodelan data. Kemudian pada sisi implementasi terdapat implementasi antarmuka, implementasi class diagram, implementasi data definition language, implementasi fungsi. Perancangan dan implementasi dijalankan sebanyak 4 kali iterasi dalam yaitu dimulai dari mengelola keanggotaan, mengelola pekerjaan masjid, mengelola notulensi, dan mengedit notulensi.

4.4.6.1 Perancangan

Perancangan merupakan penjabaran rancangan mekanisme sistem akan berjalan. Perancangan pada iterasi ini akan dirancang dengan menggunakan UML (*Unified Modelling Language)* sebagai dasar implementasi pembangunan saat menerapkan mekanisme OOP (*Object Oriented Programming*).

(a) Sequence Diagram

Proses analisis sequence diagram merupakan penerapan hasil identifikasi alur berdasarkan bisnis proses *to-be*. Sequence diagram digunakan untuk melihat bagaimana interaksi aktor menggunakan sistem dan alur data yang dijalankan pada sistem. Pada iterasi dalam ini menjelaskan analisis sequence mengelola keanggotaan, mengelola pekerjaan masjid, mengelola notulensi, dan mengedit notulensi.

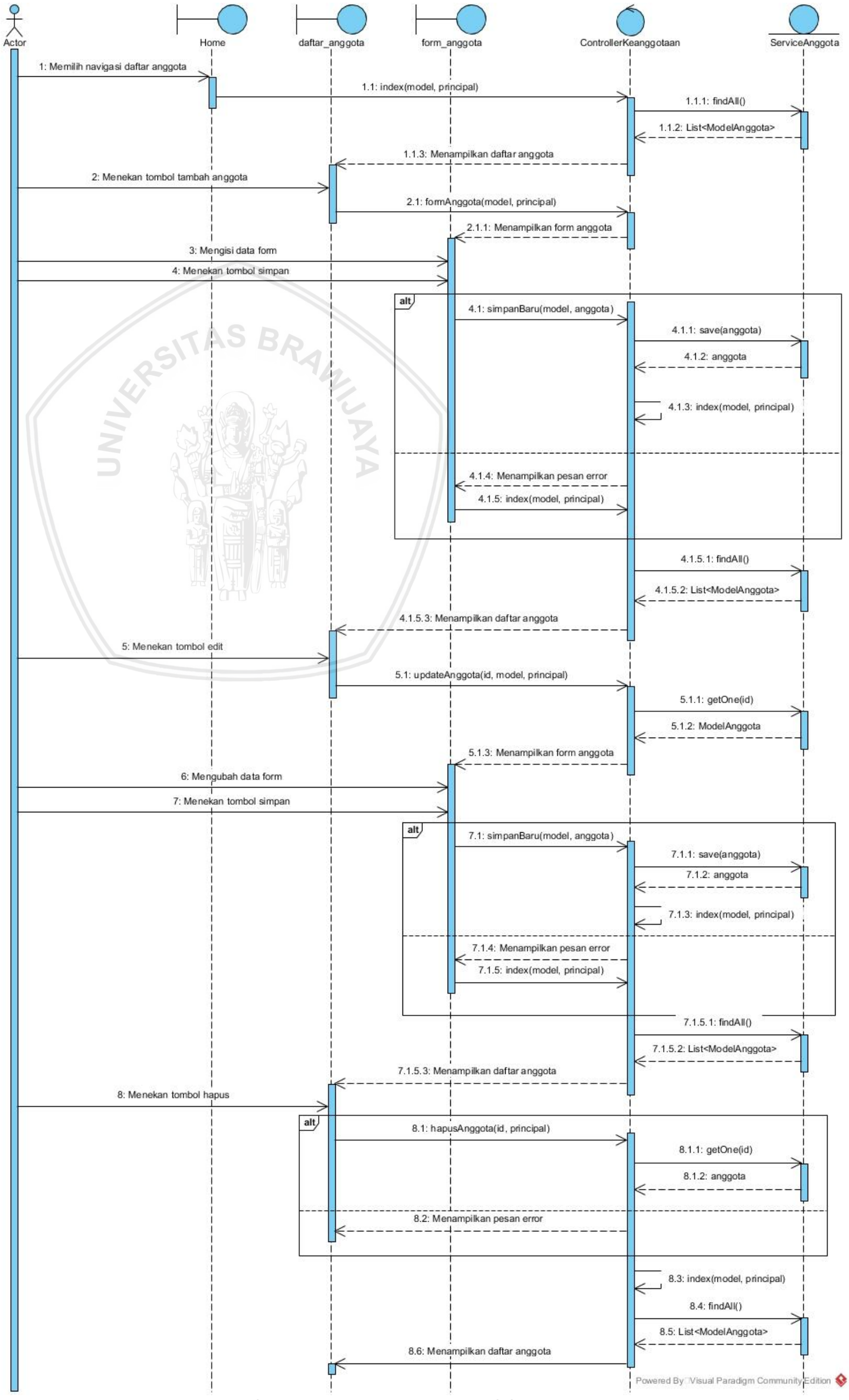
Dalam Gambar 4.7 menjelaskan alur mekanisme interaksi sequensial mengelola keanggotaan. Terdapat ControllerKeanggotaan, Home, daftar_anggota, form_anggota, dan Service Anggota. Fungsi dari ControllerKeanggotaan adalah untuk menangani permintaan aktor dengan boundary Home untuk mengakses fungsi pada sistem. boundary daftar_anggota merupakan tampilan untuk menampilkan daftar anggota yang terdaftar pada sistem. Form_anggota berisi tampilan untuk mengisi tampilan untuk memasukan data diri anggota yang berbentuk formulir. Dan ServiceAnggota merupakan service untuk melakukan manipulasi objek yang akan disimpan.

Kemudian dalam Gambar 4.8 menjelaskan alur mekanisme interaksi sequensial mengelola pekerjaan masjid. Pada ControllerPekerjaan berfungsi untuk menangani permintaan aktor dengan boundary home saat mengakses sistem. Halaman daftar_pekerjaan merupakan tampilan untuk menampilkan informasi dasar dari pekerjaan yang disimpan. Pada daftar pekerjaan tersebut akan terdapat fungsi untuk mengakses controller untuk menampilkan boundary detail pekerjaan. Dari detail pekerjaan tersebut akan ditampilkan informasi secara sepesifik terkait pekerjaan yang dibuka.sedangkan form_pekerjan merupakan halaman form pengeditan pekerjaan dan form penambahan pekerjaan yang dikelola. Informasi tersebut diakses menggunakan entitas ServiceAnggota, ServicePekerjaan, dan ServiceDetailProgres. Pada entitas ServiceAnggota merupakan service yang dapat berinteraksi dengan objek anggota yang tersimpan pada sistem. ServicePekerjaan juga akan memiliki fungsi untuk melakukan interaksi dengan objek pekerjaan. Dan ServiceDetailProgres akan memiliki interaksi dengan objek DetailProgres.

Dalam Gambar 4.9 menjelaskan alur mekanisme interaksi sequensial dalam mengelola notulensi musyawarah. Contoller yang akan digunakan sebagai media interaksi data adalah ControllerNotulensi. Controller ini akan menjadi media antara service dan boundary yang terkait. Controller tersebut memiliki hubungan dengan boundary home, daftar_notulensi, form_notulensi, dan detail_notulensi. ControllerNotulensi juga memiliki hubungan dengan beberapa service diantaranya ServiceAnggota, ServicePekerjaan, ServiceNotulensi, ServiceKomentarNotulensi. Pada boundary daftar_notulensi terdapat informasi notulensi yang tersimpan pada sistem yang dapat diakses. Masing-masing notulensi tersebut jika dibuka akan menampilkan halaman detail_notulensi yang akan berisi informasi laporan, keputusan, dan catatan tambahan pada notulensi. Sedangkan untuk menambah notulensi baru, boundary yang digunakan adalah form_notulensi. ServiceNotulensi dan ServiceKomentarNotulensi berfungsi untuk melakukan manipulasi objek dari objek notulensi dan komentar notulensi.

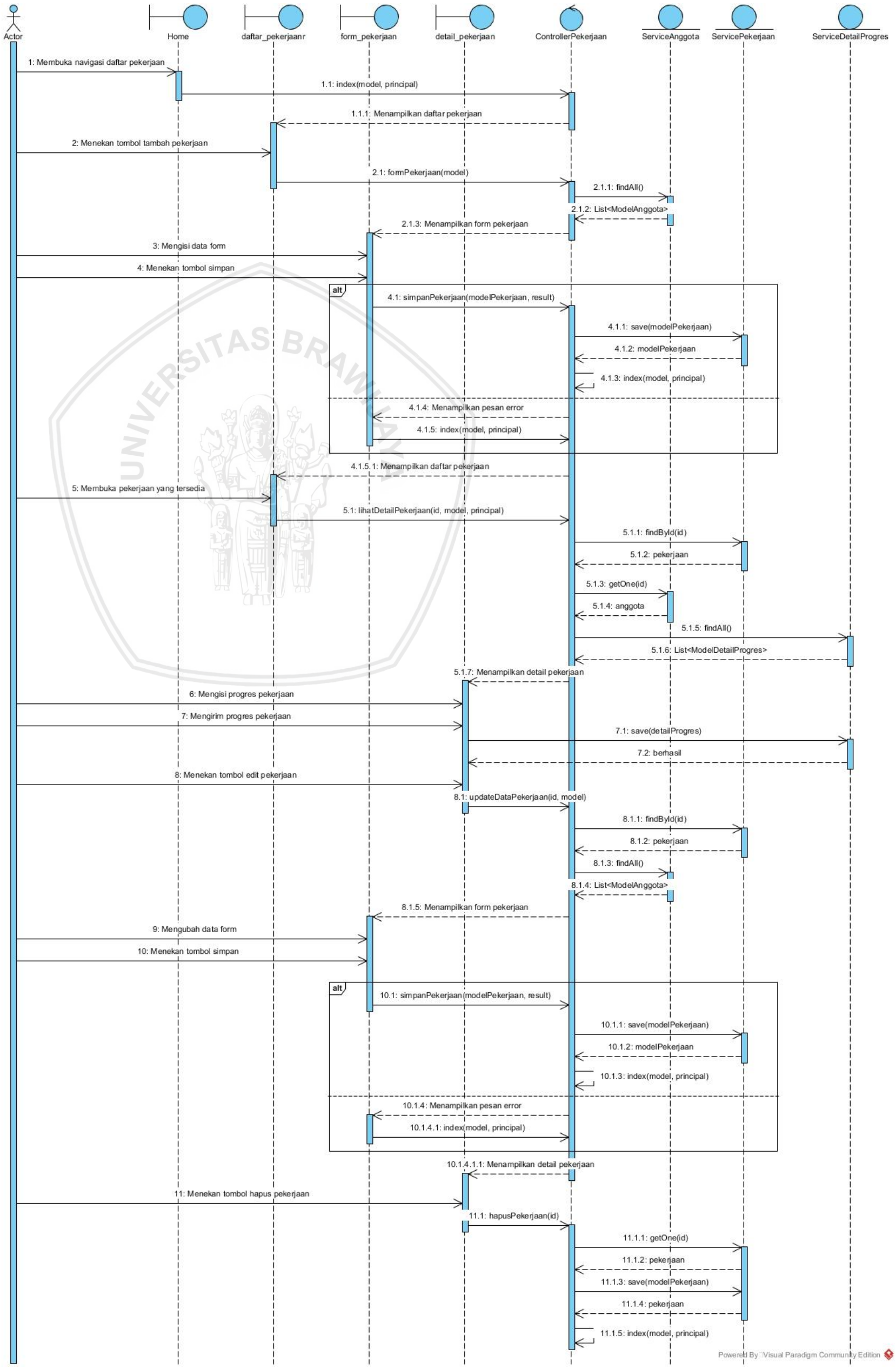
Dalam Gambar 4.10 menjelaskan alur mekanisme interaksi sequensial dalam melakukan pengubahan data notulensi. ControllerNotulensi tetap digunakan sebagai interaksi data dalam proses manipulasi. Dalam proses pengeditan notulensi boundary yang digunakan merupakan form_edit_notulensi. Dan boundary serta service yang digunakan pada sequence ini masih tetap sama kecuali form_notulensi.

repository.ub.ac.id



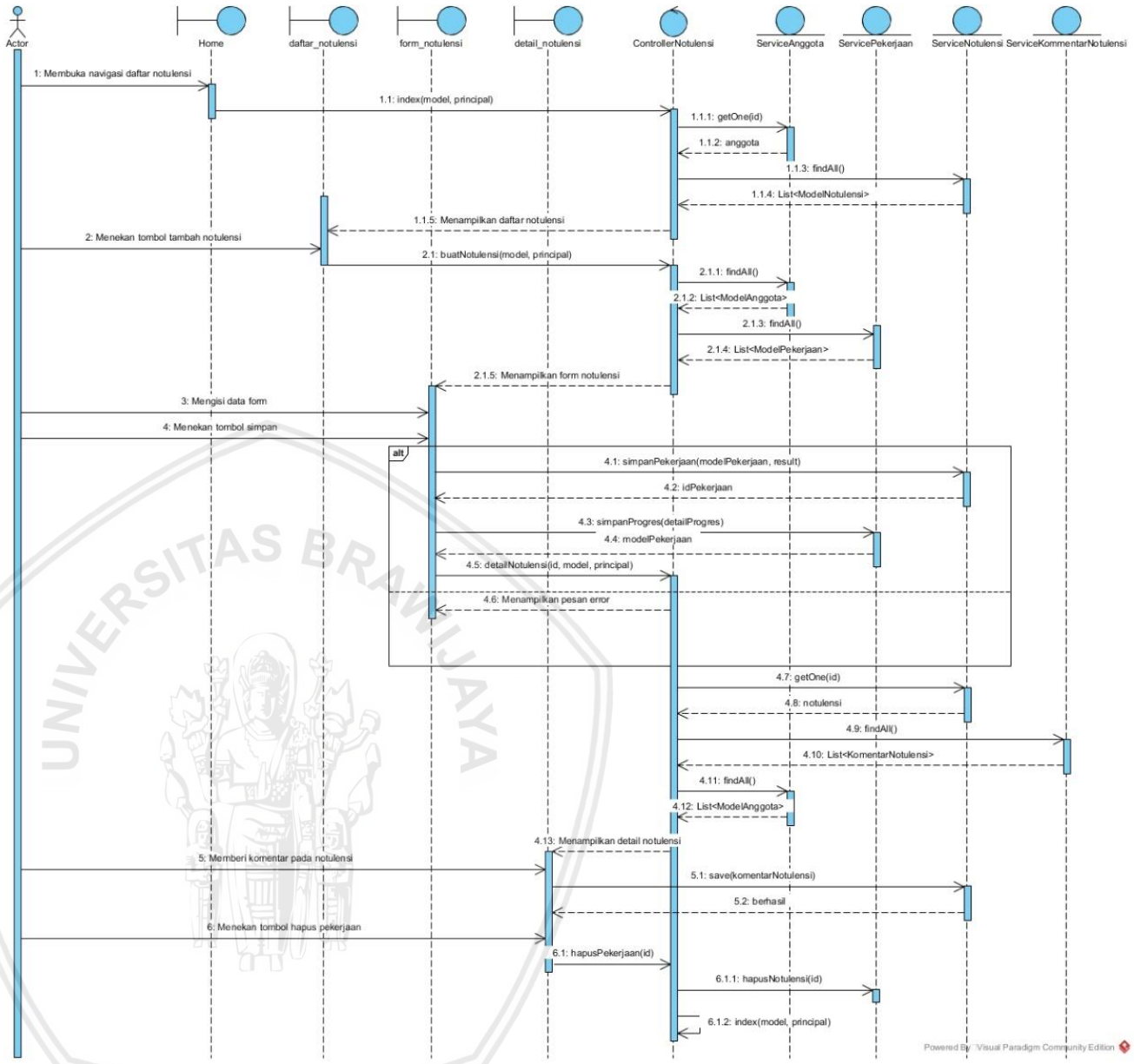
Gambar 4.7 Sequence Diagram Mengelola Keanggotaan

repository.ub.ac.id

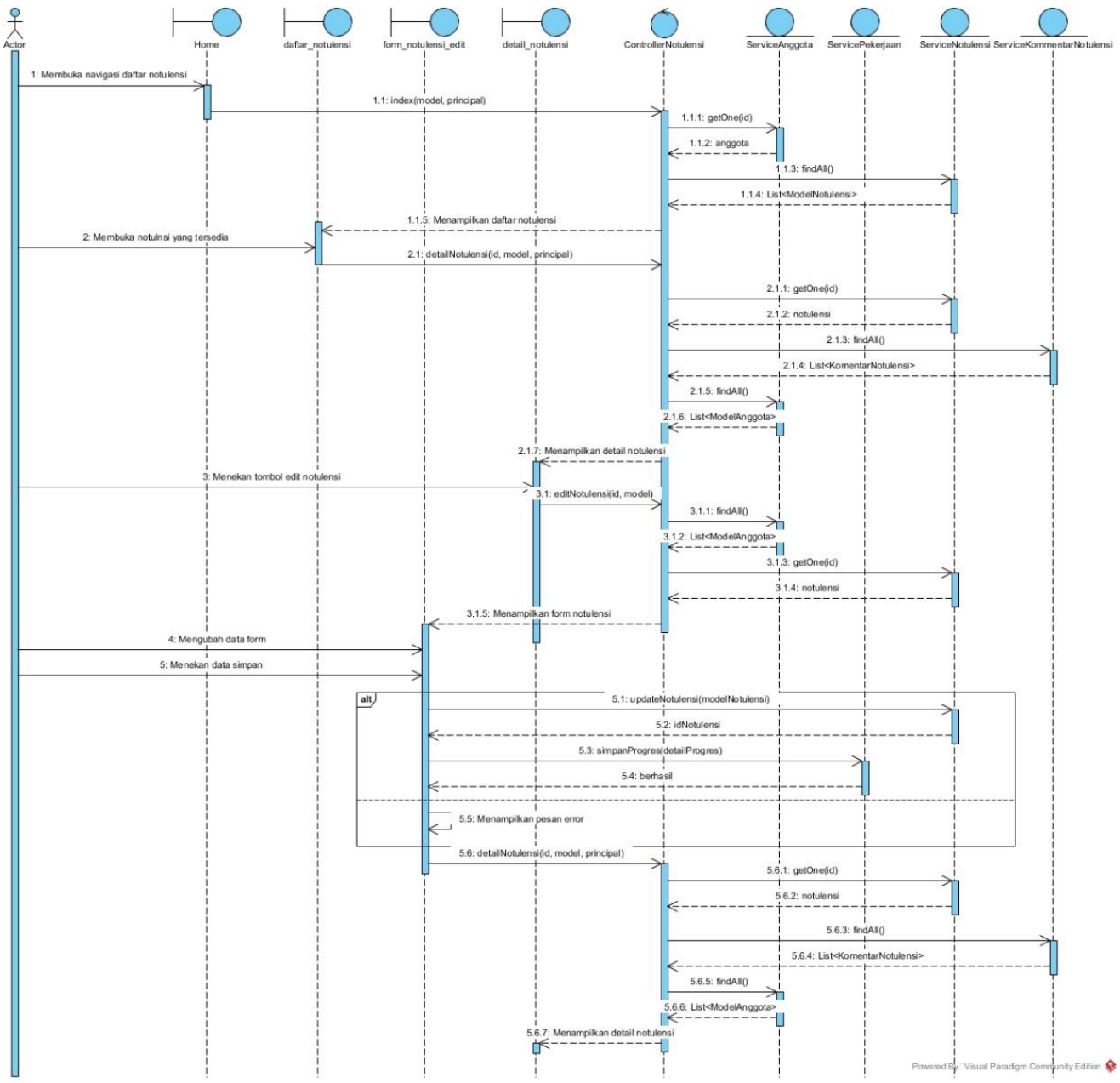


Gambar 4.8 Sequence Diagram Mengelola Pekerjaan Masjid

repository.ub.ac.id



Gambar 4.9 Sequence Diagram Mengelola Notulensi

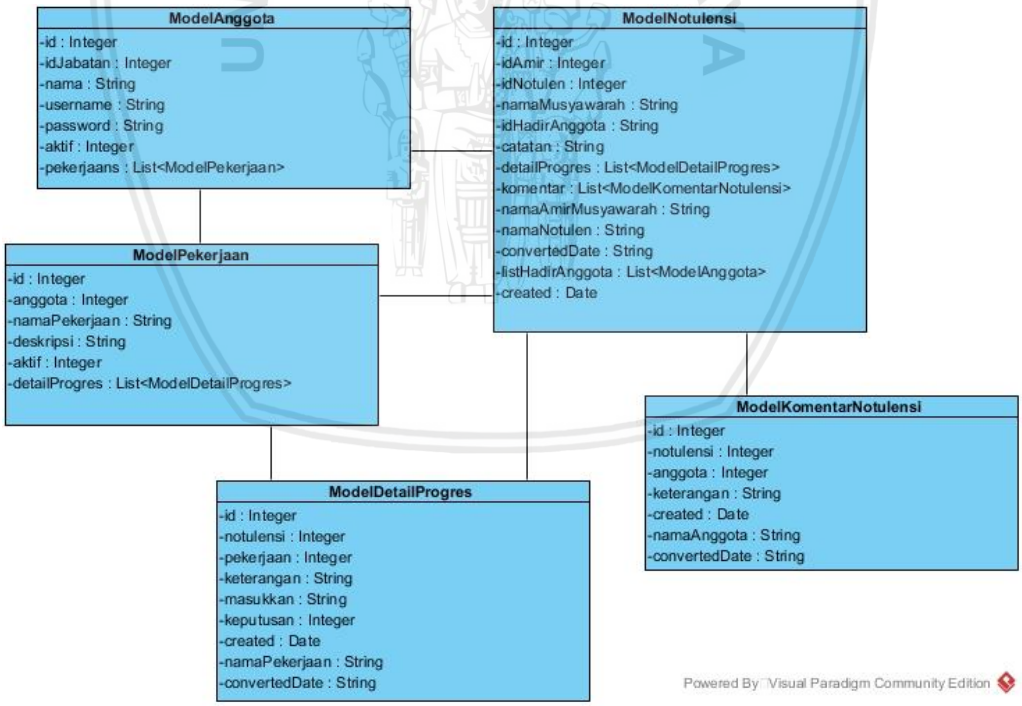


Gambar 4.10 Sequence Diagram Mengubah Notulensi

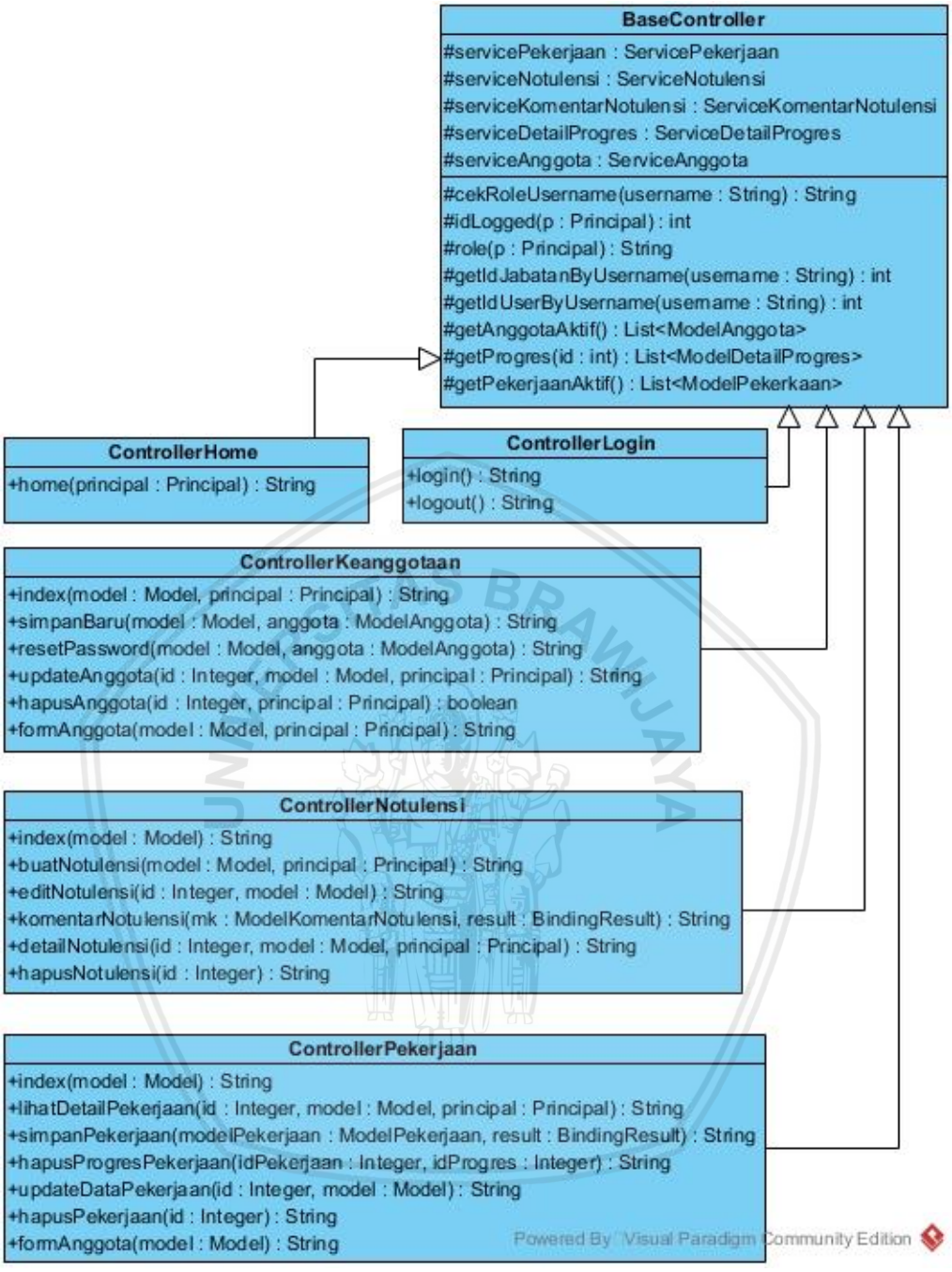
(b) Class Diagram

Diagram kelas yang digambarkan akan dibentuk berdasarkan diagram sekuensial. Class diagram dimodelkan berdasarkan fitur dengan menerapkan konsep MVC (Model-View-Controller). Diagram kelas digolongkan menjadi 2 jenis diagram, diantaranya logical class dan model class.

Kelas logical diagram dalam Gambar 4.12 merupakan hubungan *class* diagram pada sistem informasi musyawarah. Pada kelas diagram tersebut terdapat Controller Pekerjaan, ControllerNotulensi, ControllerKeanggotaan, ControllerLogin, ControllerHome, dan BaseController. BaseController digunakan sebagai *super* dari *controller* lainnya untuk menurunkan atribut dan fungsi yang dapat digunakan bersama. Kelas ControllerLogin digunakan untuk keperluan mengakses fungsi masuk dan keluar dari sistem. Kelas ControllerHome digunakan sebagai fungsi halaman pertama setelah sistem menerima masukan *login* dari pengguna. Kelas ControllerKeanggotaan digunakan untuk keperluan mengakses fungsi sistem dan tampilan mengelola keanggotaan. Kemudian kelas ControllerNotulensi juga berfungsi sebagai media akses fungsi sistem notulensi dan tampilan notulensi. Kelas ControllerPekerjaan juga berfungsi sama seperti ControllerAnggota dan ControllerNotulensi tetapi mengakses fungsi dan tampilan dari mengelola pekerjaan.



Gambar 4.11 Class Diagram Domain Model Sistem Informasi Musyawarah



Gambar 4.12 Class Diagram Logical Sistem Informasi Musyawarah

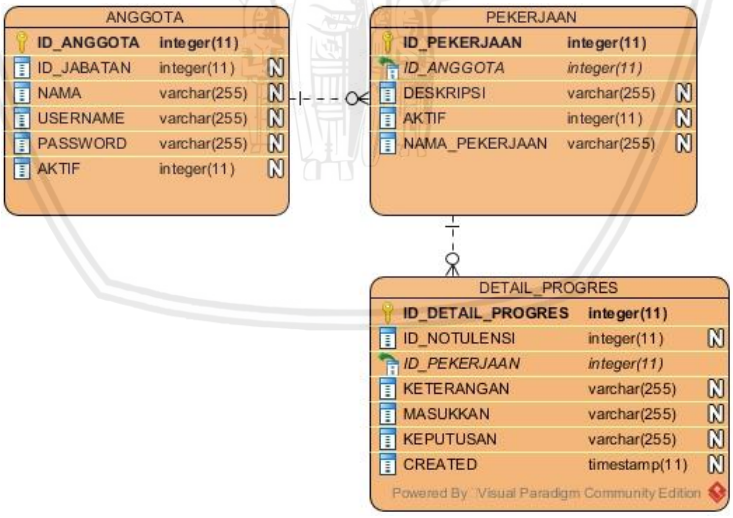
Kelas model diagram mengelola keanggotaan dalam Gambar 4.11 merupakan kelas yang digunakan untuk acuan pemodelan data dalam sistem informasi musyawarah. terdapat 5 model yang dirancang yaitu ModelAnggota, ModelPekerjaan, ModelDetailProgres, ModelKomentarNotulensi, ModelNotulensi. Kelas ModelAnggota dijadikan sebagai acuan pemodelan data pengelolaan anggota. Kelas ModelPekerjaan terhubung dengan ModelAnggota dan ModelDetailProgres

dijadikan sebagai acuan pemodelan data pengelolaan pekerjaan. Sedangkan ModelNotulensi terhubung dengan kelas model lainnya dijadikan sebagai acuan untuk pemodelan data pengelolaan notulensi.

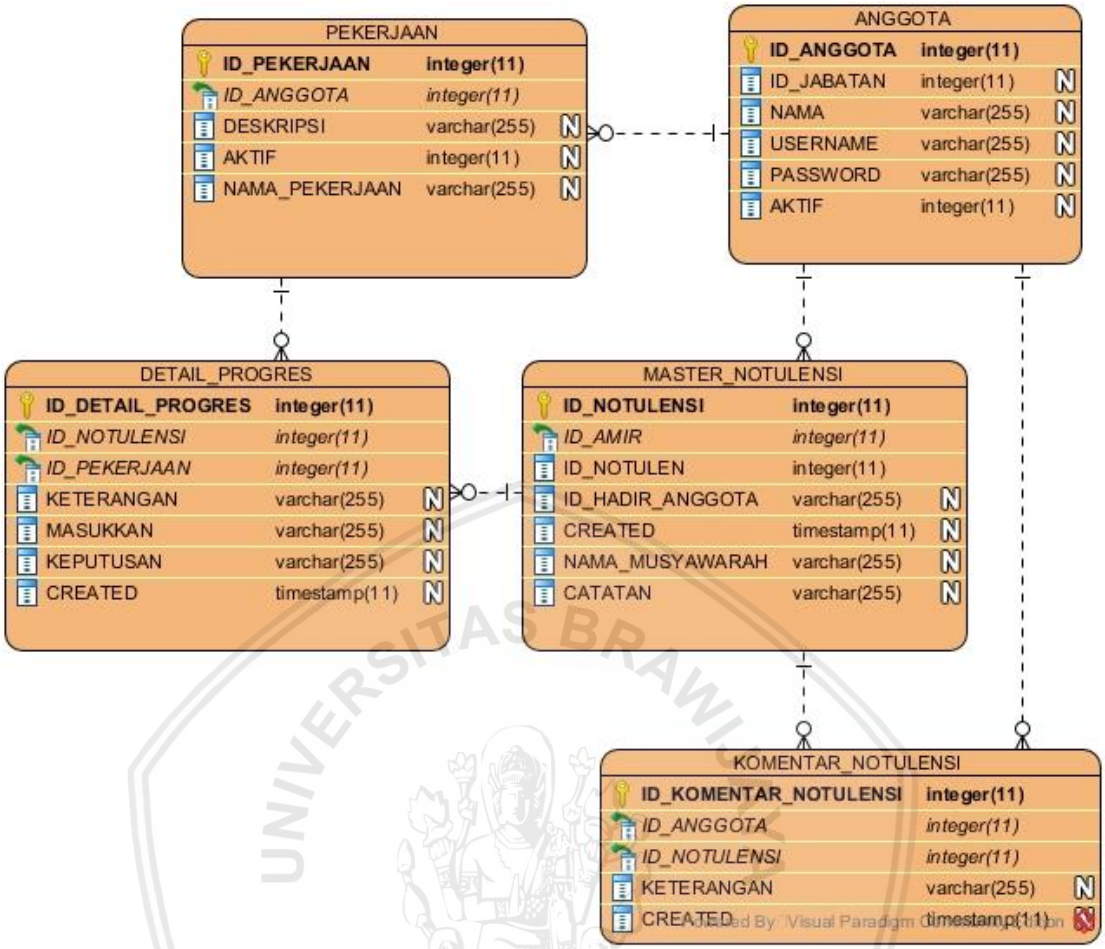
(c) Pemodelan Data

Data dimodelkan dalam bentuk PDM (*Physical Data Modeling*) yang dikembangkan dari tiap pemodelan data. Pemodelan data dimulai dengan pemodelan data pada keanggotaan yang digambarkan dalam Gambar 4.13. lalu saat pemodelan data anggota sudah terdefinisi, pemodelan data dikembangkan pada pemodelan data mengelola pekerjaan masid yang digambarkan dalam Gambar 4.14. kemudian PDM dikembangkan kembali yang digambarkan dalam Gambar 4.15 untuk pemodelan data notulensi.

Gambar 4.13 Pemodelan Data Mengelola Keanggotaan



Gambar 4.14 Pemodelan Data Mengelola Pekerjaan Masjid



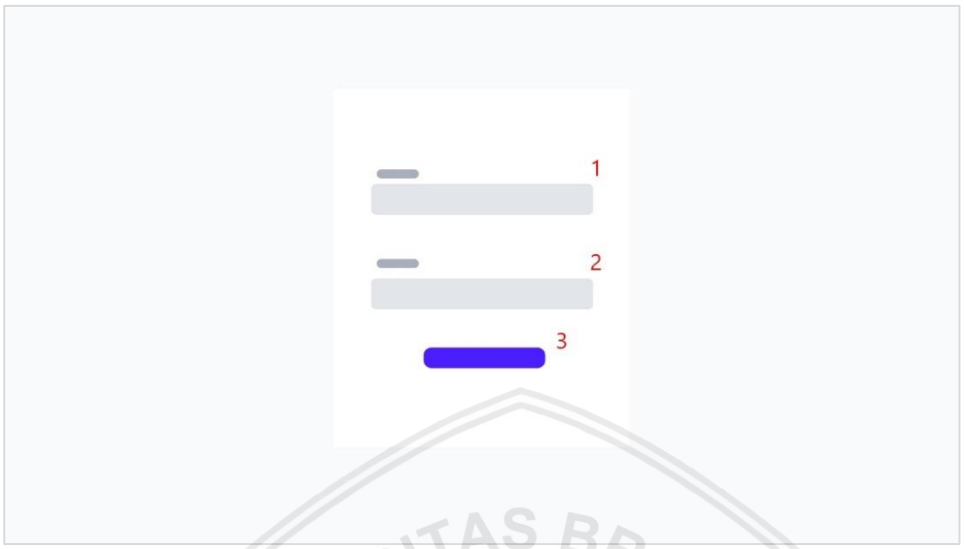
Gambar 4.15 Pemodelan Data Mengelola Notulensi

(d) Perancangan Antarmuka Pengguna

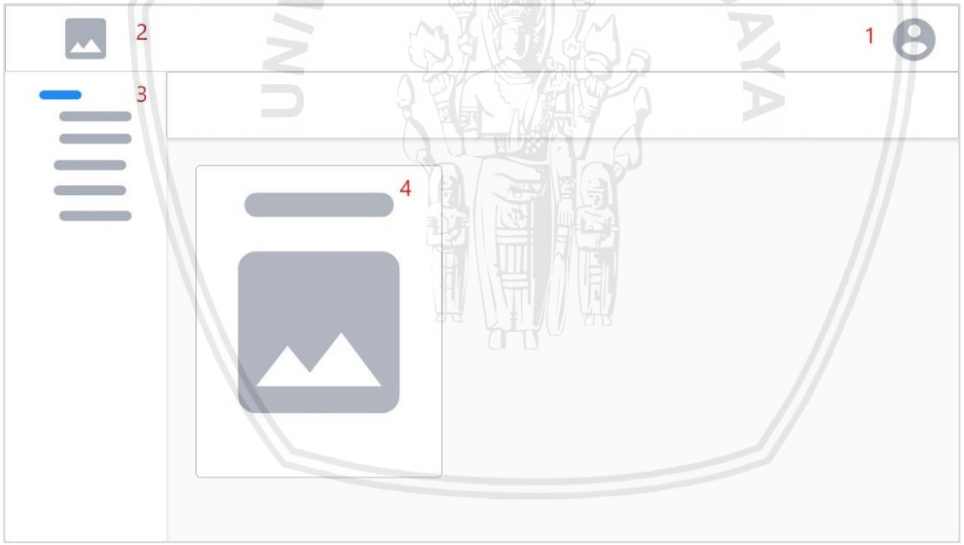
Perancangan antarmuka pengguna terdapat desain yang akan dijadikan panduan untuk melakukan implementasi dalam bentuk kode pada suatu halaman. Pada perancangan antarmuka halaman login Gambar 4.16 terdapat 2 input berupa username (1) dan password (2). Kemudian terdapat tombol login (3) berfungsi sebagai trigger untuk memproses masukan username dan password.

Pada perancangan antarmuka halaman masuk Gambar 4.17 terdapat gambar pengguna (1), logo masjid (2), dan navigasi (3). Gambar pengguna (1) digunakan sebagai interasi user dengan fungsi logout pada sistem, akan tampil menu logout saat user menerkan gambar tersebut. logo masjid (2) memiliki fungsi untuk kembali ke halaman selamat datang saat pengguna masuk. Gambar pengguna (1), logo masjid (2), dan navigasi (3) juga terdapat pada halaman lainnya yang dapat digunakan seperti

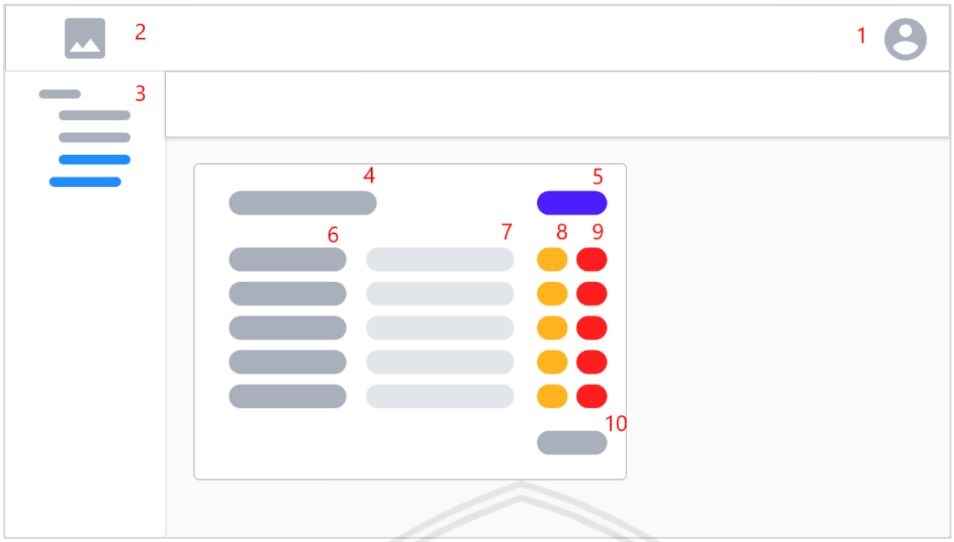
pada halaman masuk. Pada bagian nomor (4) merupakan informasi selamat datang kepada pengguna.



Gambar 4.16 Login

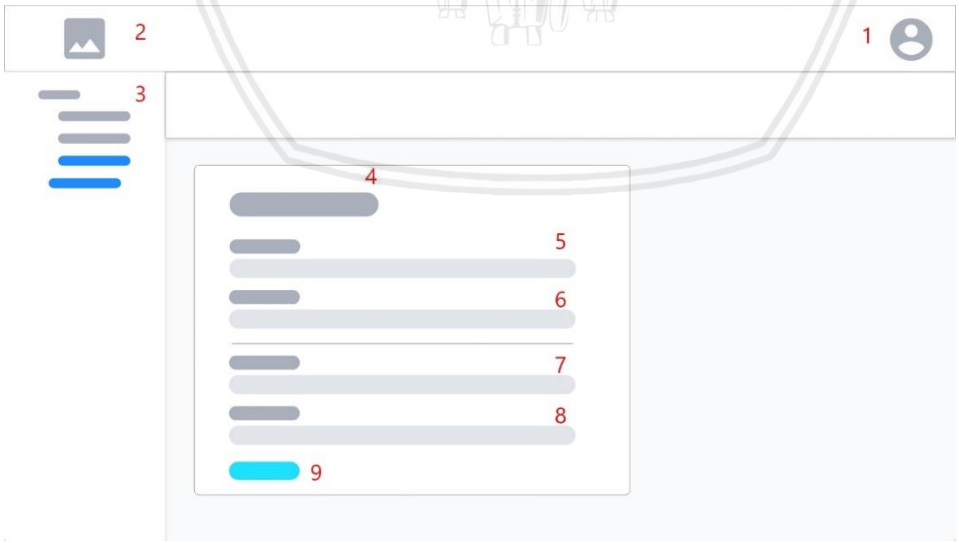


Gambar 4.17 Halaman masuk



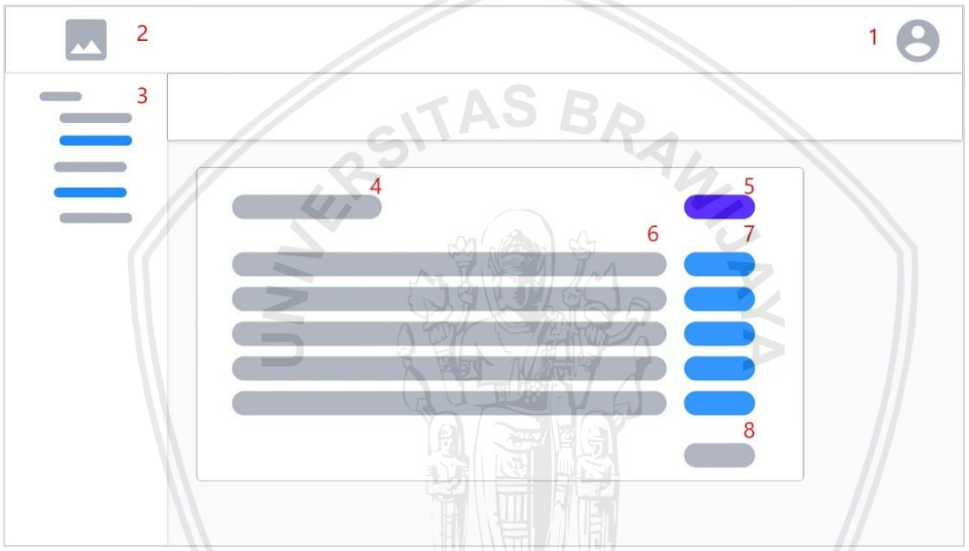
Gambar 4.18 Daftar Anggota

Pada perancangan antarmuka daftar anggota Gambar 4.18, terdapat informasi sebagai berikut: judul 'Daftar anggota' (4), tombol menambah anggota (5), jabatan anggota (6), nama anggota (7), tombol edit anggota (8), hapus anggota (9), serta daftar sebelum dan sesudah (10). Informasi nama anggota (6) akan menampilkan anggota yang memiliki status aktif pada sistem, apabila anggota tidak aktif maka data tidak akan tampil pada sistem. Pada tombol tambah anggota (5) memiliki fungsi untuk pindah ke halaman formulir tambah anggota untuk menambahkan anggota baru. Tombol edit anggota (8) memiliki fungsi untuk berpindah dari halaman daftar anggota ke halaman edit anggota, serta tombol hapus (9) akan berfungsi untuk menghapus anggota dari daftar.



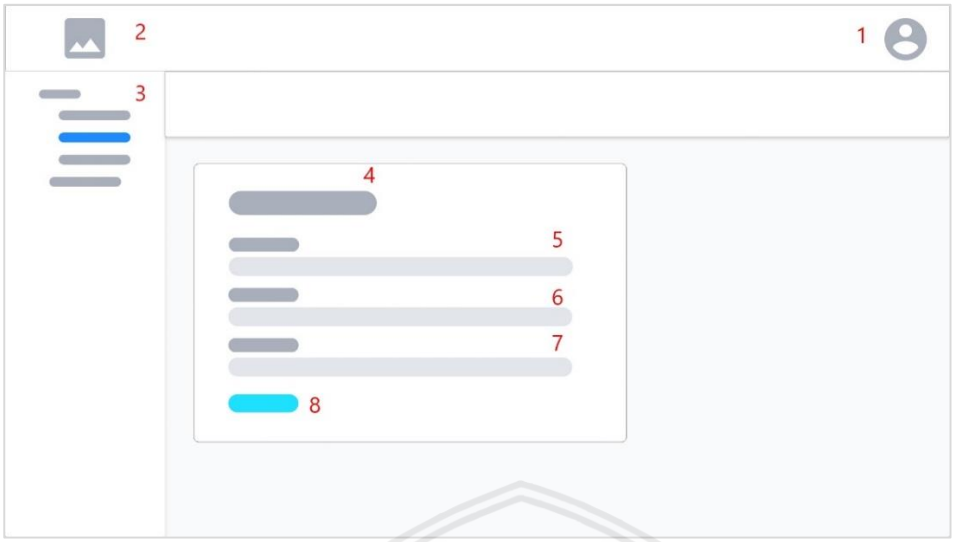
Gambar 4.19 Formulir Membuat Anggota

Pada perancangan antarmuka formulir membuat anggota Gambar 4.19, terdapat informasi sebagai berikut: judul 'Simpan Anggota' (4), masukan nama lengkap (5), pilih jabatan anggota (6), username anggota (7), password anggota (8) dan tombol simpan (9). Pada masukan nama lengkap, pengguna diwajibkan untuk mengisi nama lengkap pribadinya pada sistem. Lalu pengguna juga memilih jabatan yang terdedia pada pilih jabatan anggota (6), pilihan tersebut tersedia ketua takmir, sekretaris takmir, anggota takmir, dan anggota remaja masjid. kemudian pengguna juga mengisi masukan username (7) dan password (8) yang nanti akan digunakan untuk login kedalam sistem. Kemudian menekan tombol simpan (9) yang kemudian sistem akan menyimpan data diri anggota tersebut.



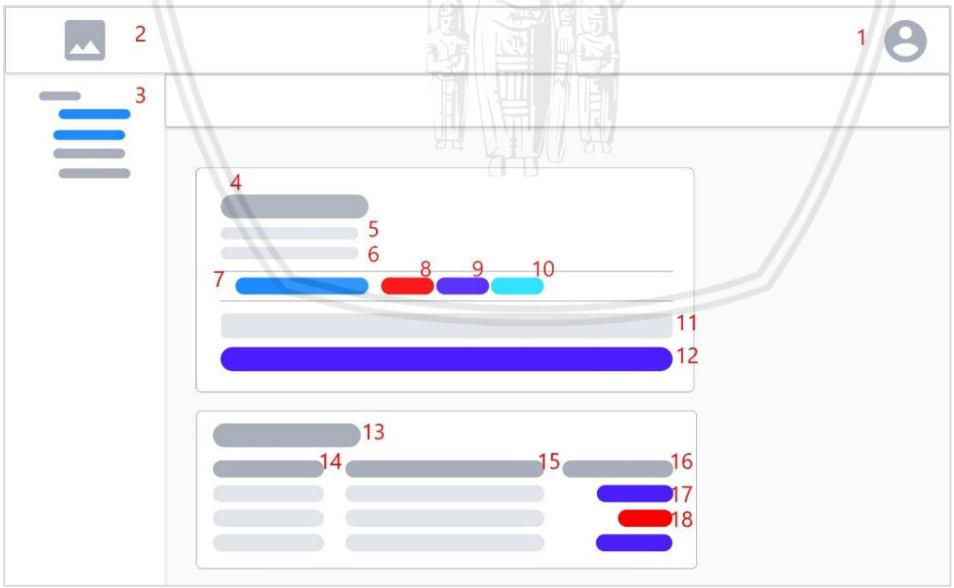
Gambar 4.20 Daftar Pekerjaan

Pada perancangan antarmuka daftar pekerjaan Gambar 4.20, terdapat informasi sebagai berikut: judul 'Daftar Pekerjaan' (4), tombol menambah pekerjaan (5), nama pekerjaan (6), lihat detail pekerjaan (7), serta daftar sebelum dan sesudah (8). Tombol tambah pekerjaan (5) memiliki fungsi untuk berpindah halaman dari daftar pekerjaan ke halaman formulir tambah pekerjaan. Nama pekerjaan (6) akan ditampilkan pada tabel, namun pekerjaan yang tampil pada sistem hanya pekerjaan yang masih dalam kondisi aktif. Tombol lihat detail pekerjaan (7) memiliki fungsi untuk berpindah halaman ke detail pekerjaan yang dipilih.



Gambar 4.21 Formulir Membuat Pekerjaan

Pada perancangan antarmuka formulir membuat pekerjaan Gambar 4.21, terdapat informasi sebagai berikut: judul 'Simpan Pekerjaan' (4), masukan nama pekerjaan (5), deskripsi singkat pekerjaan (6), penanggung jawab (7), tombol simpan pekerjaan (8). Pada masukan nama pekerjaan (5), pengguna dapat memasukan nama pekerjaan yang ingin disimpan dalam sistem. Kemudian pengguna memasukan deskripsi pekerjaan (6) pada formulir tersbut, dan memilih penanggung jawab pada list penanggung jawab (7). Setelah data formulir sudah masukan pengguna dapat menekan tombol simpan pekerjaan (8).



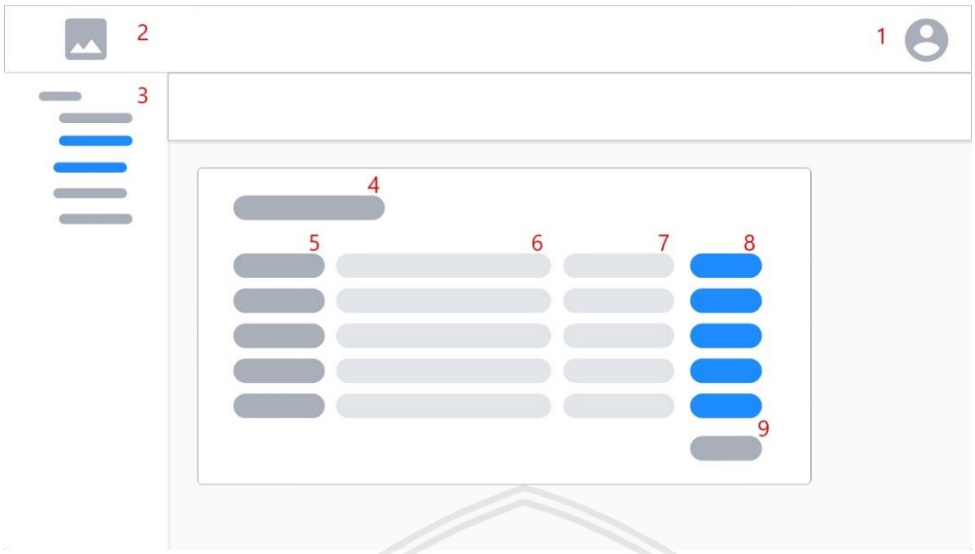
Gambar 4.22 Detail Pekerjaan

Pada perancangan antarmuka detail pekerjaan Gambar 4.22, terdapat informasi sebagai berikut: judul pekerjaan(4), deskripsi pekerjaan(5), penanggung jawab pekerjaan (6), status pekerjaan (7), hapus pekerjaan(8), edit pekerjaan(9), pekerjaan selesai (10), masukan laporan pekerjaan (11), tombol kirim laporan pekerjaan (12), judul 'Data Progres' (13), tanggal laporan (14), keterangan laporan (15), aksi (16), lihat notulensi (17), hapus progres (18). Pada judul pekerjaan (4) merupakan informasi judul yang sedang dibuka saat membuka halaman detail pekerjaan. Deskripsi pekerjaan (5) merupakan informasi deskripsi pekerjaan yang dimasukan pada saat melakukan pembuatan pekerjaan baru. Penanggung jawab pekerjaan (6) merupakan informasi nama anggota yang bertugas sebagai penanggung jawab dari pekerjaan tersebut. Status pekerjaan merupakan informasi status perkerjaan, status pekerjaan dapat berstatus belum diverifikasi, sedang berjalan dan sudah selesai. Hapus pekerjaan (8) merupakan sebuah tombol untuk menghapus pekerjaan dari sistem. Edit pekerjaan (9) merupakan tombol untuk berpindah ke halaman edit pekerjaan tersebut. Pekerjaan selesai (10) memiliki fungsi untuk mengubah status pekerjaan yang sedang berjalan menjadi pekerjaan selesai. Masukan laporan pekerjaan (11) merupakan masukan informasi laporan terbaru apabila penanggung jawab melaporkan laporan secara pribadi. Informasi laporan tersebut akan tampil pada keterangan laporan (15) dan juga tanggal laporan (14) dikirim. Pada laporan tersebut akan memiliki aksi hapus (18) karena laporan tersebut tidak terdapat pada notulensi. Pengiriman laporan tersebut dengan menekan tombol kirim laporan pekerjaan (12). Sedangkan tombol lihat notulensi (17) merupakan tombol untuk melihat notulensi yang punya hubungan dengan keterangan yang ada pada daftar data laporan.

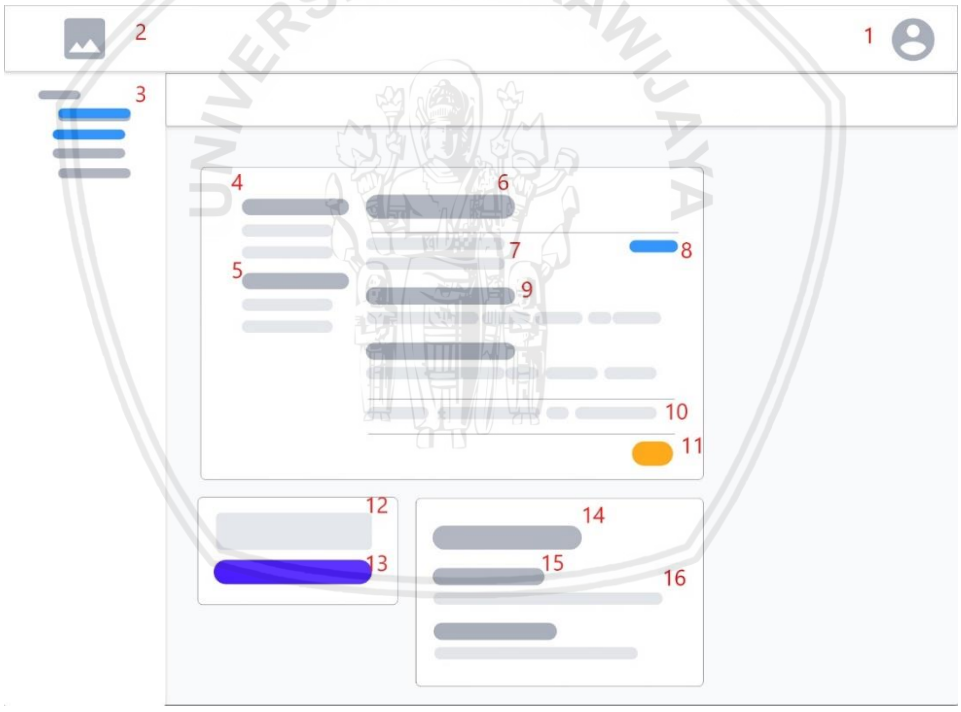
Pada perancangan antarmuka formulir buat notulensi Gambar 4.43, terdapat informasi sebagai berikut: judul 'Simpan Pekerjaan' (4), masukan nama pekerjaan (5), deskripsi singkat pekerjaan (6), penanggung jawab (7), tombol simpan pekerjaan (8). Pada masukan nama pekerjaan (5), pengguna dapat memasukan nama pekerjaan yang ingin disimpan dalam sistem. Kemudian pengguna memasukan deskripsi pekerjaan (6) pada formulir tersbut, dan memilih penanggung jawab pada list penanggung jawab (7). Setelah data formulir sudah masukan pengguna dapat menekan tombol simpan pekerjaan (8).

Pada perancangan antarmuka daftar notulensi Gambar 4.24, terdapat informasi sebagai berikut: judul 'Daftar Notulensi' (4), tanggal notulensi (5), nama musyawarah (6), amir musyawarah (7), lihat notulensi (8), serta daftar sebelum dan sesudah (9). Notulensi akan diurutkan berdasarkan tanggal notulensi (5) yang tersimpan. Nama musyawarah (6) dan amir musyawarah masjid juga akan ditampilkan pada tabel tersebut. Tombol lihat notulensi (8) akan berfungsi untuk membuka detail dari notulensi yang tersimpan.

Gambar 4.23 Formulir Membuat Notulensi



Gambar 4.24 Daftar Notulensi



Gambar 4.25 Detail Notulensi

Pada perancangan antarmuka detail notulensi Gambar 4.25, terdapat informasi sebagai berikut: judul 'Pembahasan Pekerjaan' (4), judul 'Daftar hadir' (5), nama musyawarah (6), nama amir dan notulen msuyawarah (7), tanggal notulensi (8), pelaporan notulensi berdasarkan pekerjaan (9), catatan tambahan (10), tombol edit

(11), masukan komentar (12), tombol kirim komentar (13), judul 'Komentar' (14), nama anggota pengirim komentar (15), dan komentar dari anggota(16). Pada perancangan ini terdapat informasi notulensi dan kolom komentar untuk memberi tanggapan terhadap notulensi. Informasi daftar pekerjaan yang masuk ke dalam notulensi akan ditampilkan pada pembahasan pekerjaan (4), list tersebut dapat mengakses detail pekerjaan sesuai dengan pekerjaan yang dipilih. Kemudian daftar hadir anggota akan ditampilkan pada daftar hadir (5), daftar hadir akan menampilkan nama-nama anggota yang hadir pada notulensi musyawarah. Pada judul musyawarah (6) akan ditampilkan diatas nama amir dan notulen yang tercatat pada notulensi (7). Tanggal notulensi (8) juga ditampilkan pada *layout* tersebut. lalu terdapat pelaporan dan keputusan pekerjaan (9) yang telah dicatat selama musyawarah. Terdapat catatan tambahan (10) yang juga ditampilkan setelah pelaporan dan keputusan pekerjaan. Terdapat tombol edit notulensi (11) yang berfungsi untuk mengakses halaman edit notulensi pada sistem. Pengguna dapat memberikan masukan atau komentar terhadap notulensi pada masukan komentar (12) dan mengirimnya dengan menekan tombol kirim komentar (13). Kemudian komentar akan tampil pada tabel komentar (14) dan nama anggota pengirim komentar (15) akan tampil beserta komentarnya pada komentar anggota (16).

4.4.6.2 Struktur Artefak Sistem

Struktur artefak sistem merupakan informasi artefak dalam sistem informasi yang dikembangkan. Terdapat 2 *package* utama yaitu *package* view dan *package* java. *Package* view merupakan *package* yang dijadikan sebagai tampilan pada sistem, sedangkan *package* java merupakan *package* logika sistem dengan menggunakan bahasa *java*. Struktur artefak sistem diilustrasikan dalam Gambar 4.29.

Package view berisikan tampilan yang diimplementasi dalam bentuk kode html yang akan digunakan pada sistem. Terdapat folder view dari setiap fungsi utama pada sistem, yaitu folder anggota, pekerjaan, dan notulensi. Pada folder view utama terdapat view layout.html, login.html, home.html. Pada folder view anggota terdapat daftar_anggota.html dan form_anggota.html. Kemudian, pada folder view pekerjaan terpadat daftar_pekerjaan.html, detail_pekerjaan.html dan form_pekerjaan.html. Pada folder view notulensi terdapat daftar_notulensi.html, detail_notulensi.html, form_notulensi.html dan form_notulensi_edit.html.

Package java terdapat lima folder yang dipisahkan berdasarkan fungsinya, terdapat folder config, folder service, folder rest, folder controller, dan folder model. Folder controller memiliki *class* sebagai fungsi untuk dapat mengakses halaman yang terdapat pada sistem. Terdapat enam controller yang terdapat pada package yaitu BaseController.java, ControllerHome.java, ControllerKeanggotaan.java, ControllerLogin.java, ControllerNotulensi.java, ControllerPekerjaan.java. Folder model memiliki class objek yang digunakan untuk menjadi perantara, saat melakukan

interaksi data pada objek. Terdapat lima model yang terdapat pada sistem yaitu ModelAnggota.java, ModelPekerjaan.java, ModelNotulensi.java, ModelDetailProgres.java, dan ModelKomentarNotulensi.java. Folder service merupakan interface yang implement fungsi java spring untuk digunakan. terdapat lima service yang terdapat pada sistem yaitu ServiceAnggota.java, ServicePekerjaan.java, ServiceNotulensi.java, ServiceDetailProgres.java, dan ServiceKomentarNotulensi.java. Pada folder config memiliki class untuk melakukan konfigurasi sistem untuk melakukan *authentication*. Terdapat 3 kelas java yaitu MvcConfig.java, MyUserDetailsService.java, dan WebSecurityConfig.java.

4.4.6.3 Implementasi

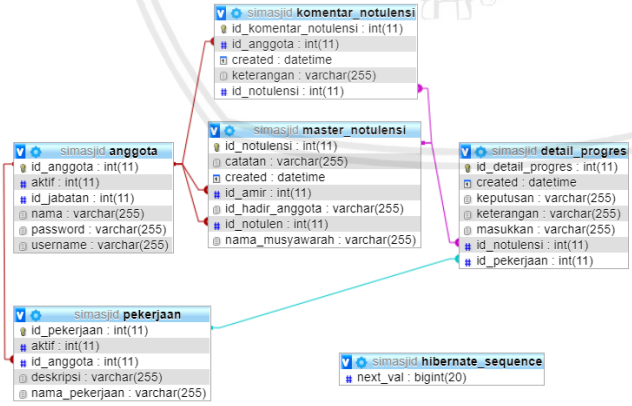
Pada tahap implementasi dijabarkan pengimplementasian sistem secara antarmuka, class diagram, data definition language, dan implementasi fungsi. Proses ini menjelaskan bahwa proses pengimplementasian dilakukan pada akhir iterasi dalam yaitu proses iterasi dalam 3.

(a) Implementasi Desain Antarmuka

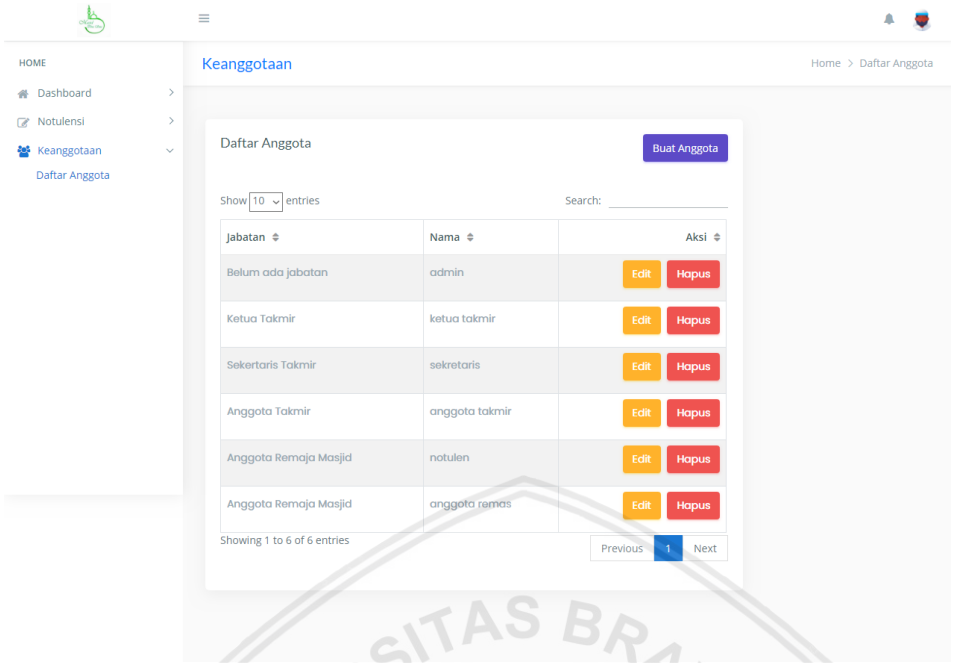
Implementasi desain antarmuka terdapat tampilan yang mewakilkan penggabaran bentuk sistem. Tampilan desain antarmuka dikembangkan berdasarkan perancangan antarmuka yang sudah didesain sebelumnya. Terdapat tampilan pengelolaan anggota dalam Gambar 4.27, pengelolaan pekerjaan dalam Gambar 4.28, dan daftar notulensi dalam Gambar 4.30.

(b) Implementasi Data Definition Language

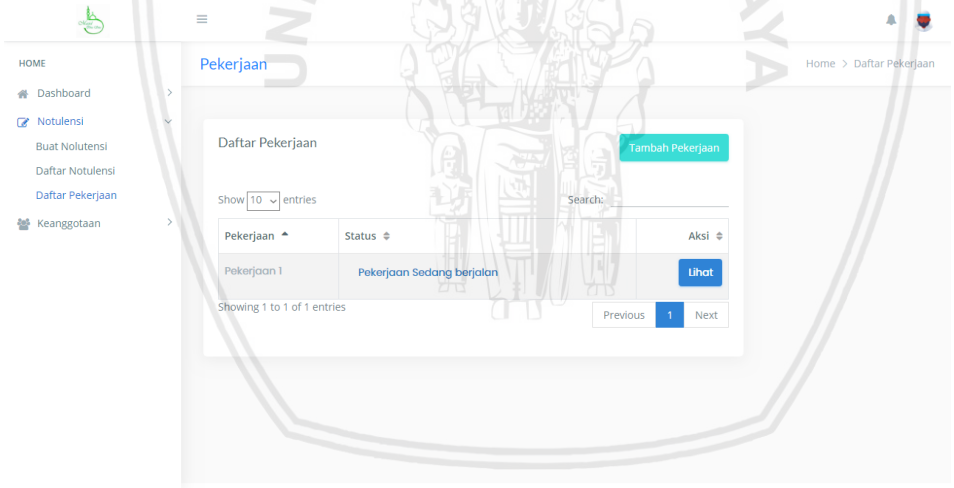
Dalam Gambar 4.26 merupakan implementasi Data Definition Language berdasarkan perancangan sebelumnya.



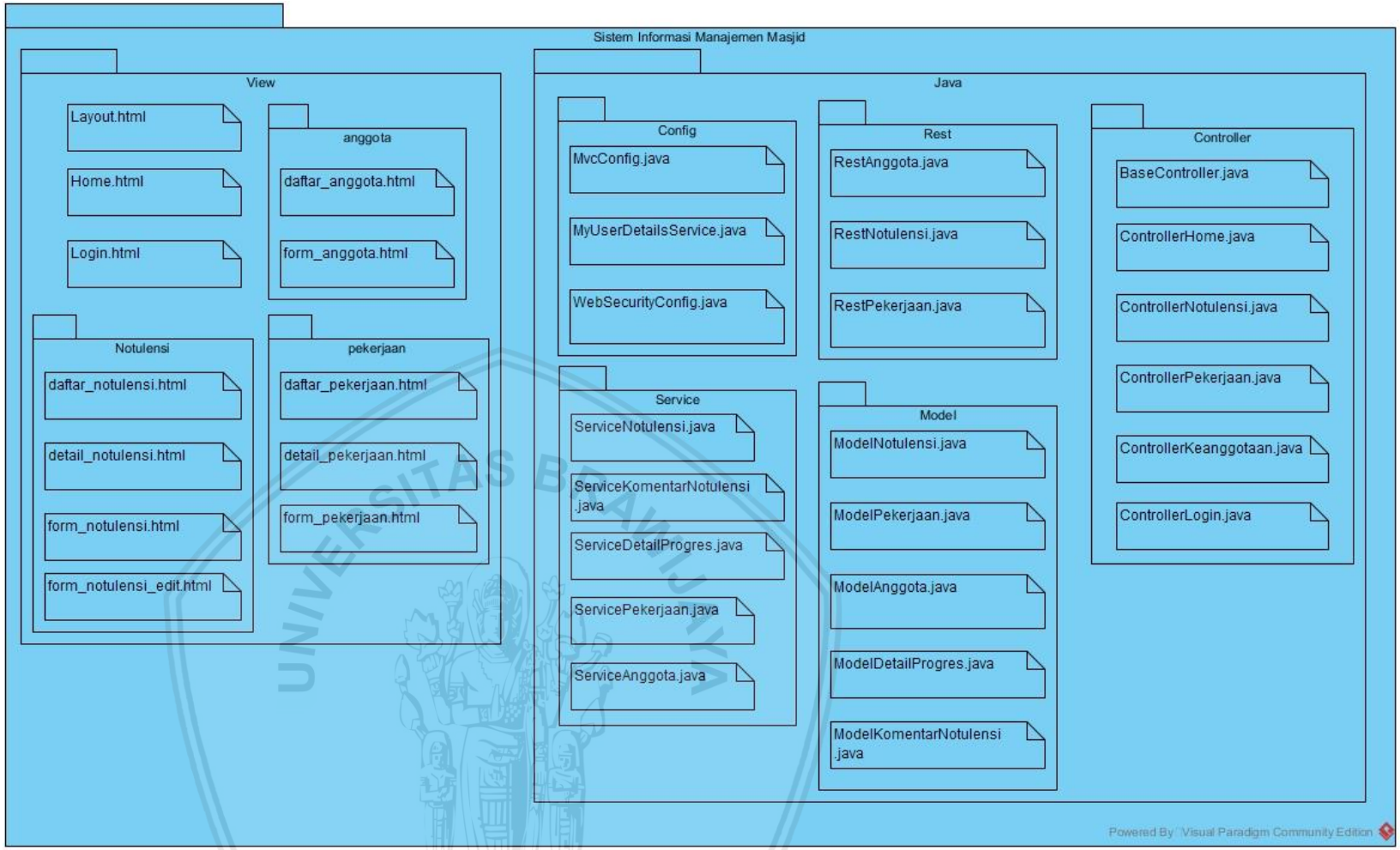
Gambar 4.26 Implementasi *Data Definition Language*



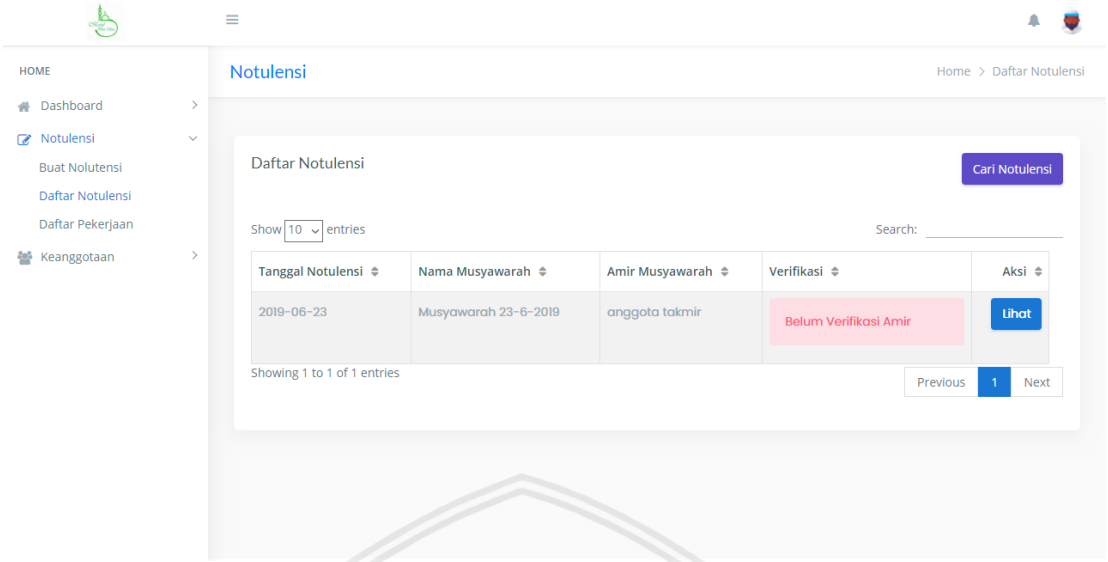
Gambar 4.27 Implementasi Antarmuka Daftar Anggota



Gambar 4.28 Implementasi Antarmuka Daftar Pekerjaan



Gambar 4.29 Struktur Artefak Sistem



Gambar 4.30 Implementasi Antarmuka Daftar Notulensi

(c) Implementasi Fungsi

Berikut ini merupakan implementasi fungsi controller keanggotaan pada sistem. Terdapat lima halaman yang dapat digunakan diakses pada controller keanggotaan. Controller pertama merupakan halaman '/anggota', halaman tersebut berisikan daftar seluruh anggota yang terdapat pada sistem. Pada halaman 'anggota/form' 'anggota/update/{id}' berisikan fungsi untuk menampilkan formulir pendaftaran anggota. Kemudian pada halaman '/anggota/simpan' merupakan proses untuk menjalankan penyimpanan anggota menggunakan method post pada halaman sebelumnya. Lalu pada halaman '/anggota/hapus/{id}' merupakan fungsi untuk menghapus anggota yang tersimpan.

```
package com.skripsi.simasjid.controller;

import com.skripsi.simasjid.model.ModelAnggota;

import com.skripsi.simasjid.services.ServiceAnggota2;

import org.springframework.beans.factory.annotation.Autowired;

import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import
org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import java.util.ArrayList;
import java.util.List;

@Controller
public class ControllerKeanggotaan {

 @Autowired
 private ServiceAnggota2 serviceAnggota2;

 private PasswordEncoder passwordEncoder;

 @RequestMapping(value = "/anggota")
 public String index(Model model){
    List<ModelAnggota> maList = serviceAnggota2.findAll();
    List<ModelAnggota> maUsed = new ArrayList<>();
    for (ModelAnggota ma: maList) {
       if (ma.getAktif() == 1){
         maUsed.add(ma);
        }
     }
     model.addAttribute("anggotas", maUsed);
     return "anggota/daftar_anggota";
 }

 @RequestMapping(value = "/anggota/simpan",
    method = RequestMethod.POST)
  public String simpanBaru(Model model, ModelAnggota anggota){
      anggota.setAktif(1);
      if (anggota.getId() == null){
         String password = anggota.getPassword();
         String encodedPassword;
    encodedPassword = new BCryptPasswordEncoder().encode(password);
         anggota.setPassword(encodedPassword);
      }
```

```
     model.addAttribute("anggotas", serviceAnggota2.save(anggota));
     return "redirect:/anggota";
 }

 @RequestMapping(value = "/anggota/resetpassword",
    method = RequestMethod.POST)
 public String resetPassword(Model model, ModelAnggota anggota){
      anggota.setAktif(1);
      String password = anggota.getPassword();
      String encodedPassword;
      encodedPassword = new
BCryptPasswordEncoder().encode(password);
      anggota.setPassword(encodedPassword);
      model.addAttribute("anggotas",
serviceAnggota2.save(anggota));
      return "redirect:/anggota";
 }

 @RequestMapping(value = "/anggota/update/{id}",
    method = RequestMethod.GET)
  public String updateAnggota(@PathVariable Integer id, Model
model){
      model.addAttribute("formbaru", '0');
      model.addAttribute("anggota", serviceAnggota2.getOne(id));
      return "anggota/form_anggota";
  }

 @RequestMapping(value = "/anggota/hapus/{id}",
     method = RequestMethod.GET)
 public String hapusAnggota(@PathVariable Integer id){
      System.out.println("Hapus "+id);
      ModelAnggota ma = serviceAnggota2.getOne(id);
      ma.setAktif(0);
      serviceAnggota2.save(ma);
      return "redirect:/anggota";
 }
```

```
 @RequestMapping(value = "/anggota/form")
 public String formAnggota(Model model){
     model.addAttribute("formbaru", 1);
     model.addAttribute("anggota", new ModelAnggota());
     return "anggota/form_anggota";
 }
}
```

Berikut ini merupakan implementasi fungsi controller pekerjaan pada sistem. Terdapat delapan halaman yang dapat digunakan diakses pada controller pekerjaan. Controller pertama merupakan halaman '/pekerjaan, halaman tersebut berisikan daftar seluruh pekerjaan yang terdapat pada sistem. Pada halaman '/pekerjaan/detail/{id}' merupakan halaman untuk menampilkan notulensi yang sudah disimpan. Pada halaman 'pekerjaan/form' dan 'pekerjaan/update/{id}' berisikan fungsi untuk menampilkan formulir pekerjaan untuk ditambah atau di edit. Kemudian pada halaman '/pekerjaan/simpan' merupakan proses untuk menjalankan penyimpanan pekerjaan menggunakan method post pada halaman sebelumnya. Lalu pada halaman '/pekerjaan/hapus/{id}' merupakan fungsi untuk menghapus anggota yang tersimpan. Halaman 'pekerjaan/buatprogres' merupakan fungsi untuk menambahkan laporan terakhir pada halaman detail progress. Halaman 'pekerjaan/setselesai' merupakan proses untuk mengganti status pekerjaan menjadi selesai.

```
package com.skripsi.simasjid.controller;
import com.skripsi.simasjid.model.ModelAnggota;
import com.skripsi.simasjid.model.ModelDetailProgres;
import com.skripsi.simasjid.model.ModelPekerjaan;
import com.skripsi.simasjid.services.ServiceAnggota2;
import com.skripsi.simasjid.services.ServiceDetailProgres;
import com.skripsi.simasjid.services.ServicePekerjaan;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
@Controller
public class ControllerPekerjaan {
 @Autowired
 private ServicePekerjaan servicePekerjaan;
 @Autowired
 private ServiceDetailProgres serviceDetailProgres;
 @Autowired
 private ServiceAnggota2 serviceAnggota2;
 /*Iterasi luar 1*/

 @RequestMapping("/pekerjaan")
 public String index(Model model) {
    return "pekerjaan/daftar_pekerjaan";
 }

 @RequestMapping(value = "/pekerjaan/detail/{id}",
    method = RequestMethod.GET)
 public String lihatDetailPekerjaan(@PathVariable Integer id, Model
model) {
   Optional<ModelPekerjaan>
     modelPekerjaan = servicePekerjaan.findById(id);
   String pic = serviceAnggota2.getOne(
     modelPekerjaan.get().getAnggota()).getNama();
   model.addAttribute("namaPekerjaan",
     modelPekerjaan.get().getNamaPekerjaan());
   model.addAttribute("deskripsiPekerjaan",
     modelPekerjaan.get().getDeskripsi());
   model.addAttribute("statusPekerjaan",
     modelPekerjaan.get().getIdStatus());
   model.addAttribute("pic", pic);
   model.addAttribute("idPekerjaan", id);
```


repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

```
    model.addAttribute("statusaktif",
modelPekerjaan.get().getAktif());
    try {
      List<ModelDetailProgres>
        mdpList = serviceDetailProgres.findAll();
      List<ModelDetailProgres> mdpById = new ArrayList<>();
      for (ModelDetailProgres mdp : mdpList) {
         if (mdp.getPekerjaan() == id) {
             mdp.setConvertedDate(mdp.getCreated());
             mdpById.add(mdp);
         }
      }
         model.addAttribute("progress", mdpById);
      } catch (Exception e) {
      System.out.println("ERROR : " + e);
      List<ModelDetailProgres> mdpList = new ArrayList<>();
      model.addAttribute("progress", mdpList);
    }
    return "pekerjaan/detail_pekerjaan";
  }

  @RequestMapping(value = "/pekerjaan/simpan",
     method = RequestMethod.POST)
  public String simpanPekerjaan(@ModelAttribute("ModelPekerjaan")
    ModelPekerjaan modelPekerjaan, BindingResult result) {
    modelPekerjaan.setIdStatus("0");
    modelPekerjaan.setAktif(1);
    servicePekerjaan.save(modelPekerjaan);
    return "redirect:/pekerjaan";
  }

  @RequestMapping(value = "/pekerjaan/hapusprogres/{idPekerjaan}
      /{idProgres}", method = RequestMethod.GET)
  public String tambahUpdateProgres(@PathVariable Integer
      idPekerjaan, @PathVariable Integer idProgres) {
        serviceDetailProgres.deleteById(idProgres);
```

```
        return "redirect:/pekerjaan/detail/" + idPekerjaan;
 }

 @RequestMapping(value = "/pekerjaan/edit/{id}",
     method = RequestMethod.GET)
  public String updateDataPekerjaan(@PathVariable Integer id,
    Model model) {
     model.addAttribute("pekerjaan",
servicePekerjaan.findById(id));
     model.addAttribute("anggotas", getAnggotaAktif());
     return "pekerjaan/form_pekerjaan";
  }

 @RequestMapping(value = "/pekerjaan/hapus/{id}",
    method = RequestMethod.GET)
 public String hapusPekerjaan(@PathVariable Integer id) {
     ModelPekerjaan mp = servicePekerjaan.getOne(id);
     mp.setAktif(0);
     servicePekerjaan.save(mp);
     return "redirect:/pekerjaan";
 }

 @RequestMapping(value = "/pekerjaan/form")
 public String formAnggota(Model model) {
     model.addAttribute("anggotas", getAnggotaAktif());
     model.addAttribute("pekerjaan", new ModelPekerjaan());
     return "pekerjaan/form_pekerjaan";
 }
 private List<ModelAnggota> getAnggotaAktif() {
     List<ModelAnggota> maList = serviceAnggota2.findAll();
     List<ModelAnggota> maUsed = new ArrayList<>();
     for (ModelAnggota ma : maList) {
        if (ma.getAktif() == 1) {
           maUsed.add(ma);
        }
     }
```

```
        return maUsed;
    }
}
```

Berikut ini merupakan implementasi fungsi controller notulensi pada sistem. Terdapat delapan halaman yang dapat digunakan diakses pada controller pekerjaan. Controller pertama merupakan halaman '/notulensi', merupaka halaman berisikan daftar seluruh notulensi yang terdapat pada sistem. Pada halaman '/notulensi/buat' dan '/notulensi/update/{id}' berisikan fungsi untuk menampilkan formulir notulensi untuk ditambah atau diedit. Kemudian pada halaman '/notulensi/simpan' merupakan proses untuk menjalankan penyimpanan notulensi menggunakan *method post* pada halaman sebelumnya. Pada halaman '/notulensi/detail/{id}' merupakan halaman untuk menampilkan notulensi yang sudah disimpan. Lalu pada halaman '/pekerjaan/hapus/{id}' merupakan fungsi untuk menghapus anggota yang tersimpan. Halaman 'pekerjaan/buatprogres' merupakan fungsi untuk menambahkan laporan terakhir pada halaman detail laporan. Halaman 'pekerjaan/setselesai' merupakan proses untuk mengganti status pekerjaan menjadi selesai.

```
package com.skripsi.simasjid.controller;
import com.skripsi.simasjid.model.*;
import com.skripsi.simasjid.services.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import java.util.*;

@Controller
public class ControllerNotulensi {

 @Autowired
 private ServicePekerjaan servicePekerjaan;

 @Autowired
 private ServiceNotulensi serviceNotulensi;
```

```
@Autowired
private ServiceKomentarNotulensi serviceKomentarNotulensi;

@Autowired
private ServiceDetailProgres serviceDetailProgres;

@Autowired
private ServiceAnggota2 serviceAnggota2;

/*Iterasi dalam 1*/
@RequestMapping("/notulensi")
public String index(Model model){
    List<ModelNotulensi> data = serviceNotulensi.findAll();
    for (ModelNotulensi mn: data) {
        System.out.println("MA get by id : "+mn.getIdAmir());
        ModelAnggota ma = serviceAnggota2.getOne(mn.getIdAmir());
        mn.setNamaAmirMusyawarah(ma.getNama());
        mn.setConvertedDate(mn.getCreated());
    }
    model.addAttribute("notulensis",data);
    return "notulensi/daftar_notulensi";
}

@RequestMapping("/notulensi/buat")
public String buatNotulensi(Model model){
    model.addAttribute("anggotas", getAnggotaAktif());
    model.addAttribute("pekerjaans",getPekerjaanAktif());
    return "notulensi/form_notulensi";
}

@RequestMapping("/notulensi/edit/{id}")
public String editNotulensi(@PathVariable Integer id, Model
model){
    model.addAttribute("anggotas", serviceAnggota2.findAll());
    ModelNotulensi mn = serviceNotulensi.getOne(id);
```


repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

```
      System.out.println("Cek id mn : "+mn.getId());
      model.addAttribute("notulensi", mn);
      model.addAttribute("progress", getProgres(id));
      String [] idAnggotaHadir = mn.getIdHadirAnggota().split(",");
      List<ModelAnggota> listAnggota = getAnggotaAktif();
      List<ModelAnggota> listAnggotaUsed = new ArrayList<>();
      for (ModelAnggota ma: listAnggota) {
        for (String tempIdAnggota : idAnggotaHadir) {
           if
(ma.getId().toString().equalsIgnoreCase(tempIdAnggota)){
               listAnggotaUsed.add(ma);
           }
        }
      }
      mn.setListHadirAnggota(listAnggotaUsed);
      model.addAttribute("listAnggotaHadir", listAnggotaUsed);
      return "notulensi/form_notulensi_edit";
  }

  @RequestMapping(value = "/notulensi/simpankomentar"
    , method = RequestMethod.POST)
  public String komentarNotulensi(
    @ModelAttribute("ModelKomentarNotulensi")
    ModelKomentarNotulensi mk, BindingResult result){
      System.out.println("id notulensi "+mk.getNotulensi());
      System.out.println("id anggota "+mk.getAnggota());
      serviceKomentarNotulensi.save(mk);
      return "notulensi/detail/"+mk.getNotulensi();
  }

  @RequestMapping(value = "/notulensi/detail/{id}",
    method = RequestMethod.GET)
  public String detailNotulensi(@PathVariable Integer id, Model
model){
      ModelNotulensi mn = serviceNotulensi.getOne(id);
      mn.setNamaAmirMusyawarah(serviceAnggota2.getOne(
```

```
        mn.getIdAmir()).getNama());
    mn.setNamaNotulen(serviceAnggota2.getOne(
       mn.getIdNotulen()).getNama());
    mn.setConvertedDate(mn.getCreated());
    System.out.println("Cek id mn : "+mn.getId());
    model.addAttribute("notulensi", mn);
    model.addAttribute("progress", getProgres(id));
    List<ModelKomentarNotulensi> listKomentar;
    listKomentar = serviceKomentarNotulensi.findAll();
    List<ModelKomentarNotulensi> listKomentarUsed;
    listKomentarUsed = new ArrayList<>();
    for (ModelKomentarNotulensi mk: listKomentar) {
       if (mk.getNotulensi() == id){
          mk.setNamaAnggota(serviceAnggota2.getOne(
             mk.getAnggota()).getNama());
          mk.setConvertedDate(mk.getCreated());
          listKomentarUsed.add(mk);
       }
    }
    model.addAttribute("komentars", listKomentarUsed);
    String [] idAnggotaHadir = mn.getIdHadirAnggota().split(",");
    List<ModelAnggota> listAnggota = getAnggotaAktif();
    List<ModelAnggota> listAnggotaUsed = new ArrayList<>();
    for (ModelAnggota ma: listAnggota) {
      for (String tempIdAnggota : idAnggotaHadir) {
         if
(ma.getId().toString().equalsIgnoreCase(tempIdAnggota)){
            listAnggotaUsed.add(ma);
         }
      }
    }
    mn.setListHadirAnggota(listAnggotaUsed);
    model.addAttribute("listAnggotaHadir", listAnggotaUsed);
    return "notulensi/detail_notulensi";
 }
```

```
@RequestMapping("/notulensi/hapus/{id}")
public String hapusNotulensi(@PathVariable Integer id){
    serviceNotulensi.deleteById(id);
    return "redirect:/notulensi";
}
private List<ModelAnggota> getAnggotaAktif(){
   List<ModelAnggota> maList = serviceAnggota2.findAll();
   List<ModelAnggota> maUsed = new ArrayList<>();
   for (ModelAnggota ma: maList) {
      if (ma.getAktif() == 1){
         maUsed.add(ma);
      }
   }
   return maUsed;
}

private List<ModelDetailProgres> getProgres(int id){
   List<ModelDetailProgres> listDetailProgres;
   listDetailProgres = serviceDetailProgres.findAll();
   List<ModelDetailProgres> listDpUsed = new ArrayList<>();
   for (ModelDetailProgres mdp: listDetailProgres) {
      if (mdp.getNotulensi() == id){
         mdp.setNamaPekerjaan(servicePekerjaan.getOne(
           mdp.getPekerjaan()).getNamaPekerjaan());
         listDpUsed.add(mdp);
      }
   }
   return listDpUsed;
}

private List<ModelPekerjaan> getPekerjaanAktif() {
   List<ModelPekerjaan> modelPekerjaanList;
   modelPekerjaanList = servicePekerjaan.findAll();
   List<ModelPekerjaan> data = new ArrayList<>();
   for (ModelPekerjaan mp : modelPekerjaanList) {
```


repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

```
        if (mp.getAktif() == 1){
            data.add(mp);
        }
    }
    return data;
 }
}
```

Berikut ini salah satu contoh dari model yang terdapat pada sistem. Model ini merupakan ModelNotulensi.java yang direpresentasikan sebagai objek bagi notulensi saat dilakukan penyimpanan. Terdapat id bertipe integer, idAmir bertipe integer idNotulen bertipe integer, idStatus bertipe integer, namaMusyawarah bertipe String, idHadirAnggota bertipe String, catatan bertipe String, dan created bertipe Date. Terdapat list yang dihubungkan dengan model lainnya seperti detailProgres yang bertipe *generic* List<ModelDetailProgres> dan komentarNotulensi bertipe *generic* List<ModelKomentarNotulensi>. Kemudian terdapat variable Transient yang berguna untuk menyimpan nilai sementara tanpa mengubah struktur model utama, terdapat variable namaAmirMusyawarah bertipe String, namaNotulen bertipe String, convertedDate bertipe String, keyword bertipe String, dan listHadirAnggota bertipe *generic* List<ModelAnggota>. Variabel created dibuat secara otomatis menggunakan notasi @Prepersist.

```
package com.skripsi.simasjid.model;
        import javax.persistence.*;
        import java.text.SimpleDateFormat;
        import java.util.Date;
        import java.util.List;
@Entity
@Table(name = "master_notulensi")
public class ModelNotulensi {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_notulensi")
    private Integer id;

    @Column(name = "id_amir")
    private Integer idAmir;
```

```
    @Column(name = "id_notulen")
    private Integer idNotulen;

    @Column(name = "nama_musyawarah")
    private String namaMusyawarah;

    @Column(name = "id_hadir_anggota")
    private String idHadirAnggota;

    @Column(name = "catatan")
    private String catatan;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id_notulensi", referencedColumnName =
"id_notulensi")
    private List<ModelDetailProgres> detailProgres;

    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id_notulensi", referencedColumnName =
"id_notulensi")
    private List<ModelKomentarNotulensi> komentarNotulensi;

    @Transient
    private String namaAmirMusyawarah;

    @Transient
    private String namaNotulen;

    @Transient
    private String convertedDate;

    @Transient
    private String keyword;

    @Transient
```

```
    private List<ModelAnggota> listHadirAnggota;


    private Date created;


    @PrePersist
    protected void onCreate() {
        created = new Date();
    }
}
```

Berikut ini merupakan salah satu contoh dari baris kode RestPekerjaan.java yang berfungsi menerima data menggunakan *javascript*. Pada rest tersebut terdapat variable serviceNotulensi yang memiliki autowired dengan ServiceNotulensi, serviceKomentarNotulensi yang memiliki autowired dengan ServiceKomentarNotulensi, servicePekerjaan yang memiliki autowired dengan ServicePekerjaan. Kemudian terdapat beberapa *method* yang dapat dijalankan pada class ini. Fungsi getAll memiliki fungsi untuk mengembalikan semua notulensi yang tersimpan. Kemudian terdapat fungsi simpanNotulensi dan updateNotulensi yang berparameter modelNotulensi untuk menyimpan data notulensi. Terdapat hapusId yang berparameter integer yang berfungsi sebagai menghapus notulensi. Terdapat simpanKomentarNotulensi berparameter komentarNotulensi untuk menyimpan komentar yang dikirim. Terdapat cariNotulensi yang berfungsi untuk mengakses fungsi getListingNotulensi, getListingNotulensi akan menyaring notulensi berdasarkan parameter tertentu yang dimasukan.

```
package com.skripsi.simasjid.rest;


import com.skripsi.simasjid.model.ModelDetailProgres;
import com.skripsi.simasjid.model.ModelKomentarNotulensi;
import com.skripsi.simasjid.model.ModelNotulensi;
import com.skripsi.simasjid.services.ServiceKomentarNotulensi;
import com.skripsi.simasjid.services.ServiceNotulensi;
import com.skripsi.simasjid.services.ServicePekerjaan;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

@RestController
@RequestMapping("/rest/notulensi")
public class RestNotulensi {

    @Autowired
    ServiceNotulensi serviceNotulensi;

    @Autowired
    ServiceKomentarNotulensi serviceKomentarNotulensi;

    @Autowired
    ServicePekerjaan servicePekerjaan;

    @GetMapping("/all")
    public List<ModelNotulensi> getAll() {
        return serviceNotulensi.findAll();
    }

    @RequestMapping(value = "/simpan", method = RequestMethod.POST,
consumes = MediaType.APPLICATION_JSON_VALUE)
    public String simpanNotulensi(@RequestBody ModelNotulensi
modelNotulensi) {
        System.out.println("body : " +
modelNotulensi.getIdNotulen());
        serviceNotulensi.save(modelNotulensi);
        return "" + modelNotulensi.getId();
    }

    @RequestMapping(value = "/update", method = RequestMethod.POST,
consumes = MediaType.APPLICATION_JSON_VALUE)
```

```
    public String updateNotulensi(@RequestBody ModelNotulensi
modelNotulensi) {
        ModelNotulensi tempMn =
serviceNotulensi.getOne(modelNotulensi.getId());
        modelNotulensi.setCreated(tempMn.getCreated());

modelNotulensi.setKomentarNotulensi(tempMn.getKomentarNotulensi());
        serviceNotulensi.save(modelNotulensi);
        return "" + modelNotulensi.getId();
    }


    @GetMapping("/hapus/{id}")
    public String getId(@PathVariable("id") final Integer id) {
        try {
            serviceNotulensi.deleteById(id);
            return "Berhasil";
        } catch (Exception e) {
            return "Gagal";
        }
    }


    @RequestMapping(value = "/simpanKomentar", method =
RequestMethod.POST, consumes = MediaType.APPLICATION_JSON_VALUE)
    public String simpanKomentarNotulensi(@RequestBody
ModelKomentarNotulensi komentarNotulensi) {
        serviceKomentarNotulensi.save(komentarNotulensi);
        return "berhasil";
    }


    @RequestMapping(value =
"/cari/{tanggal}/{pekerjaan}/{keyword}", method =
RequestMethod.GET)
    public List<ModelNotulensi> cariNotulensi(Model model,
@PathVariable String tanggal, @PathVariable String pekerjaan,
@PathVariable String keyword) {
        return getListingNotulensi(tanggal, pekerjaan, keyword);
    }
```

```
    private     List<ModelNotulensi>     getListingNotulensi(String
tanggal, String pekerjaan, String keywords) {
        Date awal = new Date();
        Date akhir = new Date();
        ;
        String[] keyword = {""};
        boolean issetDate = false;
        boolean issetDateAkhir = false;
        boolean issetPekerjaan = false;
        boolean issetKeywords = false;

        System.out.println("Tanggal : " + tanggal);
        System.out.println("Pekerjaan : " + pekerjaan);
        System.out.println("Keyword : " + keywords);

        if (!tanggal.equalsIgnoreCase("-")) {
            System.out.println("Cek tanggal ");
            if (tanggal.contains("-")) {
                System.out.println("duo tanggal ");
                String[] tanggals = tanggal.split("-");
                try {
                    awal                    =                    new
SimpleDateFormat("dd_MM_yyyy").parse(tanggals[0]);
                    akhir                   =                    new
SimpleDateFormat("dd_MM_yyyy").parse(tanggals[1]);
                    issetDateAkhir = true;
                } catch (ParseException e) {
                    e.printStackTrace();
                }
            } else {
                System.out.println("single tanggal ");
                try {
                    awal                    =                    new
SimpleDateFormat("dd_MM_yyyy").parse(tanggal);
                    System.out.println("Tanggal     :     "     +
awal.toString());
                } catch (ParseException e) {
```

```
                    e.printStackTrace();
                }
            }
            issetDate = true;
        }
        if (!pekerjaan.equalsIgnoreCase("-")) {
            issetPekerjaan = true;
        }
        if (!keywords.equalsIgnoreCase("-")) {
            keyword = keywords.split(" ");
            issetKeywords = true;
        }

        List<ModelNotulensi> modelNotulensiList =
serviceNotulensi.findAll();
        List<ModelNotulensi> notulensiFilter = new
ArrayList<ModelNotulensi>();
        List<ModelDetailProgres> tempDetaiProgres;

        for (ModelNotulensi nf : modelNotulensiList) {
            boolean addNotulen = false;

            if (issetKeywords) {

                String tempCatatan = nf.getCatatan().toLowerCase();
                for (String key : keyword) {
                    if (tempCatatan.contains(key)) {
                        addNotulen = true;
                    }
                }

                tempDetaiProgres = nf.getDetailProgres();
                for (ModelDetailProgres mdp : tempDetaiProgres) {
                    String keterangan;
                    try {
```

```
                        keterangan                                          =
mdp.getKeterangan().toLowerCase();
                    } catch (Exception e) {
                        keterangan = "";
                    }
                    String masukan;
                    try {
                        masukan = mdp.getMasukan().toLowerCase();
                    } catch (Exception e) {
                        masukan = "";
                    }
                    String keputusan;
                    try {
                        keputusan                                           =
mdp.getKeputusan().toLowerCase();
                    } catch (Exception e) {
                        keputusan = "";
                    }
                    String              namaPekerjaan                       =
servicePekerjaan.getOne(mdp.getPekerjaan()).getNamaPekerjaan();
                    try {
                        namaPekerjaan                                       =
namaPekerjaan.toLowerCase();
                    } catch (Exception e) {
                        namaPekerjaan = "";
                    }

                    for (String key : keyword) {
                        if       (keterangan.contains(key)       ||
masukan.contains(key)      ||      keputusan.contains(key)      ||
namaPekerjaan.contains(key)) {
                            addNotulen = true;
                            if (nf.getKeyword() == null) {
                                nf.setKeyword(key);
                            } else {
                                if (!nf.getKeyword().contains(key))
{
```

```
                                    nf.setKeyword(nf.getKeyword() +
" " + key);
                                }
                            }
                        }
                    }
                }
            }

            if (issetPekerjaan) {
                tempDetaiProgres = nf.getDetailProgres();
                for (ModelDetailProgres mdp : tempDetaiProgres) {
                    /*System.out.println("Cek   id   pekerjaan   :
"+pekerjaan);
                    System.out.println("Cek id pekerjaan notulen :
"+mdp.getPekerjaan().toString());*/
                    if
(mdp.getPekerjaan().toString().equalsIgnoreCase(pekerjaan)) {
                        addNotulen = true;
                    }
                }
            }

            if (issetDate) {
                Calendar calCreated = Calendar.getInstance();
                Calendar calAwal = Calendar.getInstance();
                Calendar calAkhir = Calendar.getInstance();
                if (issetDateAkhir) {
                    calCreated.setTime(nf.getCreated());
                    calAwal.setTime(nf.getCreated());
                    calAkhir.setTime(nf.getCreated());
                    boolean              sameDayAwal              =
calCreated.get(Calendar.DAY_OF_YEAR)                             ==
calAwal.get(Calendar.DAY_OF_YEAR) &&
                            calCreated.get(Calendar.YEAR)        ==
calAwal.get(Calendar.YEAR);
```

```
                    boolean             sameDayAkhir              =
calCreated.get(Calendar.DAY_OF_YEAR)                              ==
calAkhir.get(Calendar.DAY_OF_YEAR) &&
                            calCreated.get(Calendar.YEAR)         ==
calAkhir.get(Calendar.YEAR);
                    if       (nf.getCreated().before(awal)        ||
nf.getCreated().after(akhir)) {
                        if (sameDayAwal || sameDayAkhir) {
                            addNotulen = true;
                        } else {
                            addNotulen = false;
                        }
                    } else {
                        addNotulen = true;
                    }
                } else {
                    calCreated.setTime(nf.getCreated());
                    calAwal.setTime(awal);
                    boolean             sameDay               =
calCreated.get(Calendar.DAY_OF_YEAR)                              ==
calAwal.get(Calendar.DAY_OF_YEAR) &&
                            calCreated.get(Calendar.YEAR)         ==
calAwal.get(Calendar.YEAR);
                    if (sameDay) {
                        addNotulen = true;
                    } else {
                        addNotulen = false;
                    }
                }

            }
            if (addNotulen) {
                nf.setConvertedDateCari(nf.getCreated());
                notulensiFilter.add(nf);
                System.out.println("Add        :        "        +
nf.getNamaMusyawarah());
            }
        }
```

```
        return notulensiFilter;
    }


}
```

4.4.6.4 Integrasi Sistem

Proses integrasi sistem mengalami perkembangan pada setiap iterasi dalam dimulai pada proses inisialisasi projek sistem. Proses inisialisasi ini dijalankan untuk memulai untuk mengembangkan sistem untuk tahap implementasi. Implementasi dimulai dengan tahapan paling dasar yaitu keanggotaan. Sistem pengelolaan dikerjakan terlebih dahulu karena mempunyai dasar sebagai *assignee* pada pekerjaan maupun notulensi. Konsep pengelolaan anggota harus dirancang secara global untuk memastikan bahwa pengelolaan anggota digunakan sebagai modul yang akan di integrasi dengan sistem lainnya. Setelah konsep pengelolaan anggota selesai dikerjakan, kemudian proses pengimplementasian berikutnya yaitu pekerjaan. Pada tahap notulensi diperlukannya informasi pekerjaan yang akan dikerjakan oleh tiap anggota, oleh karena itu implementasi pengelolaan pekerjaan menjadi tugas berikutnya yang harus dikerjakan. Saat melakukan konsep pengelolaan pekerjaan, konsep harus dipastikan dapat terintegrasi dengan modul pengelola anggota baik secara hak akses sistem maupun hubungan data dari modul pengelolaan pekerjaan. Kemudian fitur yang dikembangkan berikutnya adalah notulensi. Fitur notulensi akan diintegrasikan dengan modul pengelolaan pekerjaan dan pengelolaan keanggotaan sehingga notulensi juga harus dipersiapkan untuk mampu terintegrasi dengan modul lainnya.

4.4.7 Evaluasi Sistem

4.4.7.1 Pengujian Validasi

Pengujian validasi dilakukan untuk memastikan bahwa sistem dapat berjalan sesuai dengan persyaratan dan ekspetasi pengguna. Pengujian ini akan dilakukan berdasarkan scenario yang didapatkan dari beberapa alur *use case*. Kemudian kasus uji tersebut akan ditentukan berdasarkan scenario uji yang ditentukan. Pengujian ini dilakukan untuk pada usecase mengelola anggota, mengelola pekerjaan, melihat notulensi, dan mengelola notulensi.

(a) Pengujian Validasi Mengelola Anggota

Pengujian validasi mengelola anggota menjelaskan pengujian sistem yang dapat dilakukan pengguna saat melakukan pengelolaan anggota musyawarah. Skenario ini akan diuji kasuskan seperti pada Tabel 4.19.

Tabel 4.19 Matriks Skenario Mengelola Anggota

Kode Pengujian	Alur Awal	Alur Alternatif
V-MA-01	Basic Flow	-
V-MA-02	Basic Flow	A4. Gagal menyimpan data anggota
V-MA-03	Basic Flow	-
V-MA-04	Basic Flow	-
V-MA-05	Basic Flow	-

(i) Pengujian Validasi Fungsi Mengelola Data Anggota

Tabel 4.20 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-06. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MA-01. Pengujian ini menunjukan hasil valid pada fungsi mengelola data anggota.

Tabel 4.20 Rencana Pengujian dan Kasus Uji Fungsi Mengelola Data Anggota

Kode Pengujian	V-MA-01
Kode Persyaratan	FTF-AMM-06
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan menyimpan data anggota pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua takmir atau sekretaris 2. Penguji memilih fungsi menambah anggota 3. Penguji memasukan data yang akan disimpan pada sistem 4. Penguji menyimpan data

Hasil yang diharapkan	Sistem dapat menyimpan data anggota dan data anggota terdapat pada daftar anggota
Hasil pengujian	Sistem berhasil menyimpan data yang diinputkan dan menampilkan data pada sistem
Status pengujian	Valid
Masukan dan saran Takmir dan Remas	

(ii) Pengujian Validasi Fungsi Mengelola Data Anggota Data Kosong

Tabel 4.21 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-06 namun sistem akan menampilkan pesan *required* saat data tidak dimasukan. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MA-02. Pengujian ini menunjukan hasil valid pada alternatif fungsi mengelola data anggota.

Tabel 4.21 Rencana Pengujian dan Kasus Uji Fungsi Mengelola Data Anggota Gagal menyimpan data anggota

Kode Pengujian	V-MA-02
Kode Persyaratan	FTF-AMM-06
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat menampilkan pesan error, pada saat melakukan penyimpanan data namun tidak terdapat isi pada masukan.
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua takmir atau sekretaris masjid 2. Penguji memilih fungsi menambah anggota 3. Penguji tidak memasukan data yang akan disimpan pada sistem 4. Penguji menyimpan data
Hasil yang diharapkan	Sistem akan menampilkan pesan *required* pada input

Hasil pengujian	Form akan menampilkan pesan untuk mengisi data saat masukan tidak terdapat data masukan
Status pengujian	Valid
Masukan dan saran	

(b) Pengujian Validasi Mengelola Data Pekerjaan Masjid

Pengujian validasi mengelola data pekerjaan menjelaskan pengujian sistem yang dapat dilakukan pengguna saat melakukan pengelolaan pekerjaan masjid. Skenario ini akan diuji kasuskan seperti pada Tabel 4.22.

Tabel 4.22 Matriks Skenario Mengelola Data Pekerjaan

Nomor Skenario	Alur Awal	Alur Alternatif
V-MP-01	Basic Flow	-
V-MP-02	Basic Flow	S1. Mengedit pekerjaan
V-MP-03	Basic Flow	-
V-MP-04	Basic Flow	-
V-MP-05	Basic Flow	-
V-MP-06	Basic Flow	S2. Menghapus pekerjaan

(i) Pengujian Validasi Fungsi Mengelola Data Pekerjaan Masjid

Tabel 4.23 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-12. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MP-01. Pengujian ini menunjukan hasil valid pada fungsi mengelola pekerjaan masjid.

Tabel 4.23 Rencana Pengujian dan Kasus Uji Fungsi Pengelolaan Pekerjaan

Kode Pengujian	VA-MP-01
Kode Persyaratan	FTF-AMM-12
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan menyimpan data pekerjaan pada sistem.

Prosedur Uji	1. Penguji teridentifikasi sebagai anggota takmir dan remas 2. Penguji memilih fungsi menambah pekerjaan 3. Penguji memasukan data yang akan disimpan pada sistem 4. Penguji menyimpan data
Hasil yang diharapkan	Sistem dapat menyimpan data pekerjaan dan data anggota terdapat pada daftar pekerjaan
Hasil pengujian	Sistem berhasil menyimpan data yang diinputkan dan menampilkan data pada sistem
Status pengujian	Valid
Masukan dan saran	

(ii) Pengujian Validasi Fungsi Mengedit pekerjaan

Tabel 4.24 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-13. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MP-01. Pengujian ini menunjukan hasil valid pada fungsi mengedit pekerjaan.

Tabel 4.24 Rencana Pengujian dan Kasus Uji Fungsi Mengedit pekerjaan

Kode Pengujian	VA-MP-02
Kode Persyaratan	FTF-AMM-13
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan mengubah data pekerjaan pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai anggota takmir dan remas 2. Penguji membuka salah satu pekerjaan 3. Penguji memilih fitur edit

	4. Penguji mengubah data menjadi data yang akan disimpan pada sistem 5. Penguji menyimpan data
Hasil yang diharapkan	Sistem berhasil menyimpan data yang diubah
Hasil pengujian	Sistem berhasil menyimpan data yang diinputkan dan menampilkan data pada sistem
Status pengujian	Valid
Masukan dan saran	

(c) Pengujian Validasi Melihat Notulensi

Pengujian validasi melihat notulensi menjelaskan pengujian sistem yang dapat dilakukan pengguna saat melakukan melihat dan mengomentari notulensi masjid. Skenario ini akan diuji kasuskan seperti pada Tabel 4.25.

Tabel 4.25 Tabel Matriks

Nomor Skenario	Alur Awal	Alur Alternatif
V-MN-01	Basic Flow	-
V-MN-02	Basic Flow	S1. Mengomentari notulensi

(i) Pengujian Validasi Fungsi Melihat Notulensi

Tabel 4.26 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-18. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MN-01. Pengujian ini menunjukan hasil valid pada fungsi melihat notulensi.

Tabel 4.26 Rencana Pengujian dan Kasus Uji Fungsi Melihat Notulensi

Kode Pengujian	V-MN-01
Kode Persyaratan	FTF-AMM-18
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan

	melihat detail notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai Takmir dan Remas 2. Penguji memilih Daftar Notulensi 3. Penguji membuka salah satu notulensi
Hasil yang diharapkan	Sistem dapat menampilkan data notulensi yang tersimpan
Hasil pengujian	Sistem berhasil menampilkan data notulensi
Status pengujian	Valid
Masukan dan saran	

(ii) Pengujian Validasi Fungsi Menambah komentar Notulensi

Tabel 4.27 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-18. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MN-02. Pengujian ini menunjukan hasil valid pada fungsi menambah komentar notulensi.

Tabel 4.27 Rencana Pengujian dan Kasus Uji Fungsi Komentar Notulensi

Kode Pengujian	V-MN-02
Kode Persyaratan	FTF-AMM-18
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan menambah komentar pada notulensi yang tersimpan.
Prosedur Uji	1. Penguji teridentifikasi sebagai Takmir dan Remas 2. Penguji memilih Daftar Notulensi 3. Penguji membuka salah satu notulensi 4. Penguji memasukkan komentar 5. Penguji mengirim komentar

Hasil yang diharapkan	Sistem dapat menyimpan komentar yang dimasukan
Hasil pengujian	Sistem berhasil menyimpan komentar yang dimasukan
Status pengujian	Valid
Masukan dan saran	

(d) Pengujian Validasi Mengelola Notulensi

Pengujian validasi mengelola notulensi menjelaskan pengujian sistem yang dapat dilakukan pengguna saat melakukan pengelolaan notulensi masjid. Skenario ini akan diuji kasuskan seperti pada Tabel 4.28.

Tabel 4.28 Tabel Matriks

Nomor Skenario	Alur Awal	Alur Alternatif
V-MIN-01	Basic Flow	-
V-MIN-02	Basic Flow	S1. Mengubah isi Notulensi
V-MIN-03	Basic Flow	S2. Menghapus Notulensi

(i) Pengujian Validasi Fungsi Membuat Notulensi

Tabel 4.29 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-15. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MIN-01. Pengujian ini menunjukan hasil valid pada fungsi mengelola notulensi.

Tabel 4.29 Rencana Pengujian dan Kasus Uji Fungsi Mengelola Notulensi

Kode Pengujian	V-MIN-01
Kode Persyaratan	FTF-AMM-15
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan menambah data notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai notulensi

	2. Penguji memilih fungsi menambah notulensi 3. Penguji memasukan data nama musyawarah, data amir musyawarah, data kehadiran anggota musyawarah yang akan disimpan pada sistem 4. Penguji menambah pekerjaan yang akan di progress 5. Penguji memasukan data laporan, masukan, keputusan, dan catatan tambahan musyawarah 6. Penguji menyimpan data
Hasil yang diharapkan	Sistem dapat menyimpan data yang dimasukan
Hasil pengujian	Sistem berhasil menyimpan data yang dimasukan dan menampilkan data pada sistem
Status pengujian	Valid
Masukan dan saran	

(ii) Pengujian Validasi Fungsi Mengubah Isi Notulensi

Tabel 4.30 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-20. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MIN-03. Pengujian ini menunjukan hasil valid pada fungsi mengubah isi notulensi.

Tabel 4.30 Rencana Pengujian dan Kasus Uji Fungsi Mengubah Isi Notulensi

Kode Pengujian	V-MIN-02
Kode Persyaratan	FTF-AMM-20
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat mengubah isi notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua atau sekretaris masjid

	2. Penguji membuka detail notulensi 3. Penguji memilih fitur edit notulensi 4. Penguji melakukan perubahan data laporan, masukan, keputusan, dan catatan tambahan musyawarah 5. Penguji menyimpan data
Hasil yang diharapkan	Sistem dapat mengubah data notulensi yang sudah tersimpan
Hasil pengujian	Sistem berhasil mengubah data notulensi yang sudah tersimpan
Status pengujian	Valid
Masukan dan saran	

(iii) Pengujian Validasi Fungsi Menghapus Notulensi

Tabel 4.31 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-19. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MIN-04. Pengujian ini menunjukan hasil valid pada fungsi mengapus notulensi.

Tabel 4.31 Rencana Pengujian dan Kasus Uji Fungsi Menghapus Notulensi

Kode Pengujian	V-MIN-03
Kode Persyaratan	FTF-AMM-19
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan menghapus data notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai Notulen atau Amir Musyawarah 2. Penguji membuka detail notulensi 3. Penguji memilih fitur menghapus notulensi
Hasil yang diharapkan	Sistem berhasil menghapus data notulensi

Hasil pengujian	Sistem berhasil menghapus data pada sistem
Status pengujian	Valid
Masukan dan saran	

4.4.7.2 Hasil Pengujian Validasi

Pengujian validasi diuji oleh Takmir dan Remas untuk menunjukan kebutuhan sistem sudah terimplementasi dan diterima oleh Takmir dan Remas. Pengujian validasi pada mengelola anggota, mengelola data pekerjaan, melihat notulensi, dan mengelola notulensi menunjukan hasil yang valid. Terdapat masukan dan tanggapan dari penguji untuk menambahkan fitur tertentu untuk dikembangkan lebih lanjut. Pada validasi mengelola anggota, fitur ini terdapat masukan untuk menambah biodata, nomor telpon rumah, hp, email, target selama menjadi remaja masjid, taat dan patuh aturan masjid. Masukan pada saat melakukan validasi mengelola data pekerjaan penambahan penomoran urutan dari pekerjaan, kemudian diberi fitur untuk menambahkan batasan tanggal berakhirnya pekerjaan serta pemberitahuan untuk menampilkan pekerjaan yang sudah melewati tanggal. Pada pengujian validasi melihat notulensi dan mengelola notulensi belum terdapat masukan tetapi sudah divalidasi oleh penguji. Pada hasil pengujian tersebut dapat dinyatakan proses pengujian validasi proses iterasi luar ke-1 dinyatakan valid dan kebutuhan fungsional telah terpenuhi.

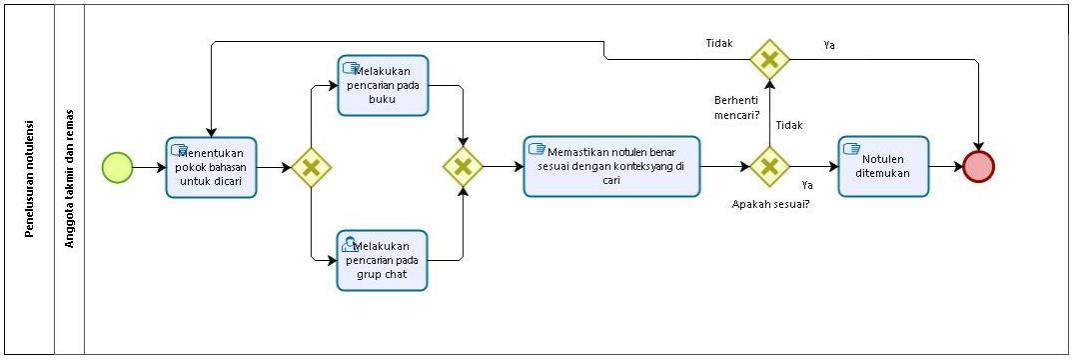
4.5 Iterasi Luar Ke-2

Proses iterasi luar ke-2 dijadikan sebagai proses pengembangan kedua dalam melakukan penggalian kebutuhan setelah evaluasi sistem pada iterasi pertama. Terdapat beberapa solusi yang dikembangkan berikutnya yaitu mencari notulensi, verifikasi pekerjaan dan verifikasi notulensi.

4.5.1 Pemodelan Proses Bisnis *As-Is*

4.5.1.1 Proses Bisnis *As-is* Mencari Notulensi (PBA-AMM-04)

Kegiatan penelusuran notulensi merupakan kegiatan pencarian informasi yang terdapat pada buku dan grup *chat* Takmir dan Remas. Proses ini dilakukan untuk memastikan informasi yang didapat sesuai dengan kesepakatan yang diputuskan dalam musyawarah masjid. proses ini dimulai dengan menentukan bahasan topik permasalahan yang ingin dicari. Lalu anggota akan mencari pada buku notulensi dan grup *chat*. Jika informasi masih belum ditemukan maka informasi akan terus dicari sampai terdapat informasi yang sesuai.



Gambar 4.31 Proses Bisnis *As-is* Pencarian Notulensi Musyawarah

Berdasarkan wawancara kepada takmir dapat dianalisis bahwa proses ini merupakan proses yang dapat dilakukan dengan singkat. Proses ini memiliki 2 sumber informasi yang dapat digunakan yaitu buku notulensi dan juga grup *chat*. Terdapat permasalahan yang dapat dianalisis dari proses tersebut. proses tersebut menggunakan 2 sumber informasi yang berbeda, yang menyebabkan anggota musyawarah harus mencari dari kedua-duanya yang memakan waktu. Selain itu, proses tersebut juga dimungkinkan perbedaan informasi yang didapatkan dari 2 sumber tersebut sehingga dimungkinkannya ambiguitas informasi yang didapat. Proses pencarian ini tentunya akan lebih cepat menggunakan grup *chat* dengan kata kunci tertentu, tetapi informasi tersebut tidak akan secara mendetail karena keterbatasan informasi yang dapat dimasukan dalam grup *chat*.

4.5.1.2 Proses Bisnis *As-is* Verifikasi Pekerjaan Masjid (PBA-AMM-05)

Kegiatan verifikasi pekerjaan masjid berjalan secara langsung tanpa adanya pencatatan perkegiatan pada masjid. kegiatan pekerjaan masjid berjalan pada saat pekerjaan masjid disahkan oleh amir musyawarah masjid pada saat keputusan musyawarah masjid. pekerjaan tersebut berjalan selama kegiatan yang ditentukan saat keputusan belum terpenuhi, dan pekerjaan dianggap selesai pada saat pekerjaan sudah menyelesaikan tanggung jawabnya oleh anggota yang bertanggung jawab.

4.5.1.3 Proses Bisnis *As-is* Verifikasi Notulensi Masjid (PBA-AMM-06)

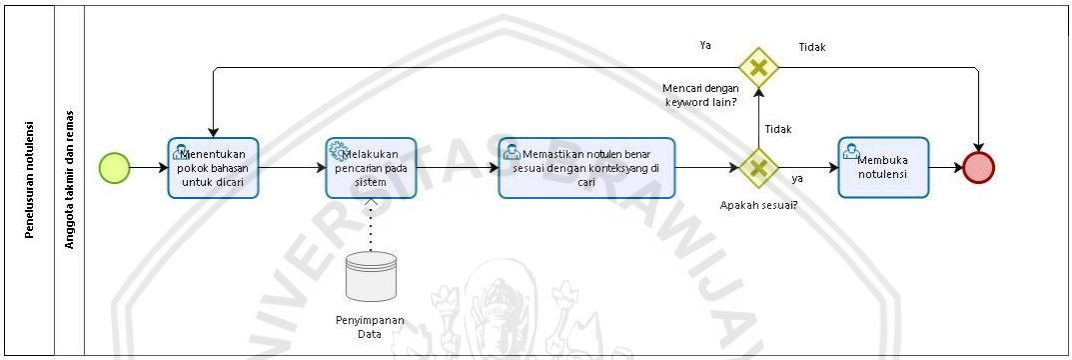
Kegiatan verifikasi notulensi masjid dilakukan dilaporkan melalui grup untuk memastikan notulensi diterima oleh seluruh anggota musyawarah yang hadir. Verifikasi ini tidak secara langsung memberikan status bahwa notulensi dianggap

sudah, tetapi notulensi dianggap terverifikasi apabila tidak ada tanggapan oleh anggota musyawarah yang hadir

4.5.2 Pemodelan Proses Bisnis *To-Be*

4.5.2.1 Proses Bisnis *To-Be* Mencari Notulensi (PBT-AMM-04)

Proses bisnis mencari notulensi merupakan proses bisnis usulan perbaikan dari proses bisnis as-is mencari notulensi. Proses bisnis ini melibatkan sistem untuk membantu actor dalam mencari notulensi berdasarkan parameter yang diinginkan. Pencarian notulensi ini menggunakan atribut yang terekam pada penyimpanan data.

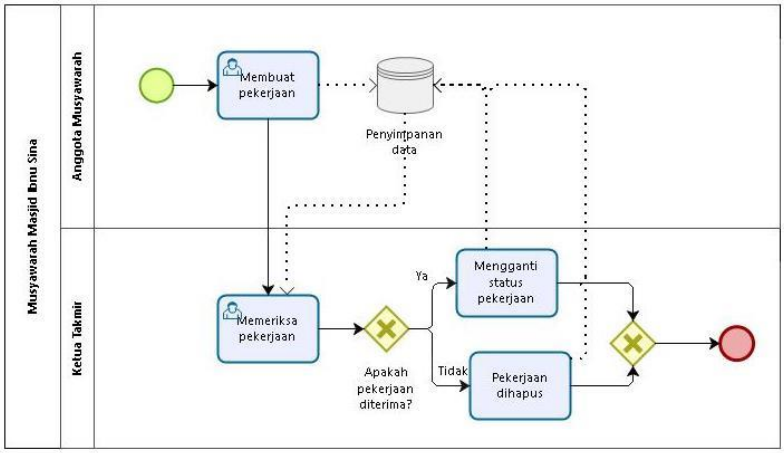


Gambar 4.32 Proses Bisnis *To-be* Pencarian Notulensi Musyawarah

Proses bisnis ini dimulai pada saat anggota membuka sistem kemudian memilih fitur filter untuk pencarian. Actor memasukan atribut yang diinginkan untuk dicari. Apabila sistem tidak terdapat notulensi yang diinginkan, apabila pencarian sudah selesai maka actor dapat menghetikan proses pencarian notulensi.

4.5.2.2 Proses Bisnis *To-Be* Menverifikasi Status Pekerjaan (PBT-AMM-05)

Proses menverifikasi status pekerjaan merupakan proses dalam mengganti status pekerjaan. Status pekerjaan akan dijadikan sebagai tanda pekerjaan dalam pengerjaan pekerjaan. Status pekerjaan dapat berupa aktif, dihapus, dan berjalan. Proses ini akan membantu aktor dalam memastikan pekerjaan yang akan dan sedang dijalan sesuai dengan aturan.

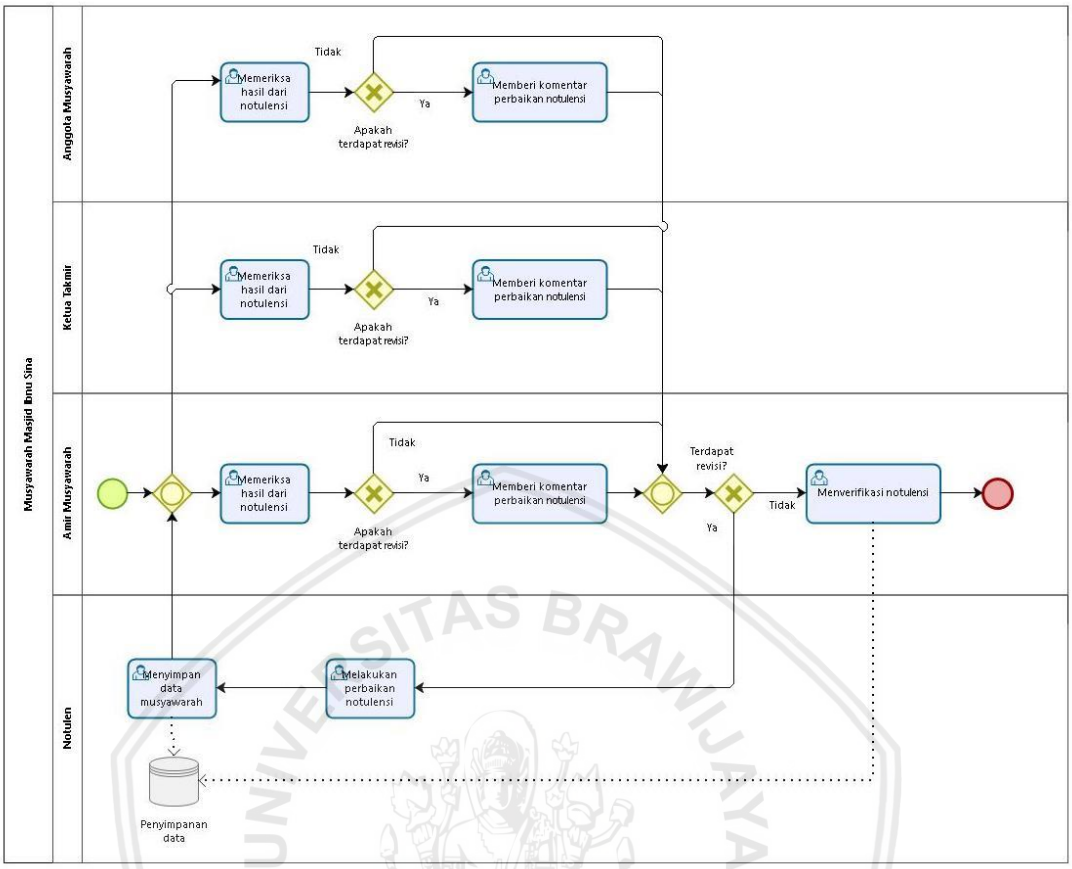


Gambar 4.33 Proses Bisnis *To-be* Menverifikasi Pekerjaan Masjid

Proses ini dimulai pada saat anggota musyawarah membuat pekerjaan dan disimpan pada sistem. Kemudian ketua takmir menggunakan akunnya dapat melihat pekerjaan-pekerjaan yang masih belum terverifikasi dan kemudian membuat keputusan untuk menverifikasi pekerjaan atau mengganti statusnya. Status yang diubah akan disimpan pada sistem yang kemudian proses bisnis ini berhenti.

4.5.2.3 Proses Bisnis *To-Be* Menverifikasi Notulesi (PBT-AMM-06)

Proses menverifikasi notulensi merupakan proses bisnis usulan, untuk memenuhi persyaratan *stakeholder* dan juga membantu memastikan bahwa notulensi yang di inputkan oleh notulen valid terhadap hasil musyawarah. Proses bisnis *to-be* menverifikasi notulensi dijalankan saat proses bisnis kegiatan musyawarah telah selesai.



Gambar 4.34 Proses Bisnis *To-Be* Menverifikasi Notulensi

Proses ini dimulai saat sistem sudah merekam notulensi dan Amir Musyawarah masjid melakukan pemeriksaan hasil musyawarah tersebut. apabila terdapat hasil musyawarah yang ingin direvisi maka notulen akan melakukan perbaikan dan kembali ke proses memeriksa notulensi. Pemeriksaan notulensi juga dapat dilakukan oleh ketua takmir atau anggota musyawarah lainnya pada sistem. Setelah pemriksaan telah selesai, maka sistem akan di verifikasi dan proses bisnis berhenti.

4.5.3 Analisis Perbaikan Proses Bisnis

Analisis perbaikan proses bisnis dilakukan untuk membandingkan proses bisnis *as-is* dan proses bisnis *to-be* yang sudah dirancang. Proses ini akan mempermudah untuk mengetahui proses bisnis yang dilakukan perbaikan. Terdapat proses bisnis usulan yang ditambahkan dan diubah berdasarkan persyaratan aktor menjalankan kegiatan musyawarah. Analisis ini dijabarkan pada Tabel 4.33.

4.5.4 Analisis Persyaratan

4.5.4.1 Identifikasi Tipe Pemangku Kepentingan

Hasil identifikasi tipe pemangku kepentinkan berupa daftar karakteristik pemangku kepentingan yang ada pada Masjid Ibnu Sina. Hasil identifikasi ini akan digunakan sebagai informasi untuk menganalisis kegiatan yang akan dilakukan oleh pengguna untuk berinteraksi pada sistem.

Tabel 4.32 Hasil Identifikasi Tipe Pemangku Kepentingan

Tipe Pemangku Kepentingan	Deskripsi	Pemangku Kepentingan
Pengguna	Orang yang menggunakan sistem secara langsung dan berperan sebagai aktor dalam *use case*	- Ketua Takmir - Sekretaris Takmir - Takmir dan remas - Notulen - Amir Musyawarah

4.5.4.2 Identifikasi Kebutuhan Pemangku Kepentingan dan Pengguna

Identifikasi kebutuhan pemangku kepentingan dan pengguna menjelaskan sekumpulan pernyataan yang berhubungan dengan permasalahan yang ada pada tabel pernyataan masalah. Hasil identifikasi kebutuhan pemangku kepentingan dan pengguna pada Tabel 4.4 terdapat beberapa perubahan yang dijelaskan pada Tabel 4.34.

repository.ub.ac.id

Tabel 4.33 Analisis Perbedaan Proses Bisnis As-Is dan To-Be serta Waktu yang Dibutuhkan

No.	Aktor *As-Is*	Proses Bisnis *As-Is*	Aktor *To-Be*	Proses Bisnis *To-Be*	Waktu Lama	Perkiraan Waktu Baru	Keterangan
1.	Takmir dan Remas	Mencari Notulensi (PBA-AMM-04)	Takmir dan Remas	Mencari Notulensi (PBT-AMM-04)	15 menit	10 menit	Diubah
2.	Ketua Takmir	Menverifikasi Pekerjaan Masjid (PBA-AMM-05)	Ketua Takmir	Menverifikasi Pekerjaan Masjid (PBT-AMM-05)	30 detik	20 detik	Diubah
3.	Amir Musyawarah	Menverifikasi Notulensi Masjid (PBA-AMM-06)	Amir Musyawarah	Menverifikasi Notulensi Masjid (PBT-AMM-06)	30 detik	20 detik	Diubah

Tabel 4.34 Hasil Identifikasi Kebutuhan Pengguna

Kode Kebutuhan	Kebutuhan Pengguna	Pemangku Kepentingan	Situasi Saat Ini	Solusi Yang Ditawarkan
KB-AMM-01	Sistem harus menyediakan layanan untuk mengelola data anggota	Ketua Takmir dan Sekretaris Takmir	Proses penyimpanan data anggota tidak disimpan secara administratif	Menyediakan sistem untuk menyimpan data anggota aktif pada sistem yang dapat di manajemen oleh ketua dan sekretaris takmir
KB-AMM-02	Sistem harus menyediakan layanan untuk mengelola data pekerjaan masjid	Takmir dan Remas	Proses pendataan pekerjaan yang berjalan tidak disimpan secara administratif	Menyediakan sistem untuk menyimpan data pekerjaan yang dapat digunakan untuk menjadi sarana pelaporan kegiatan yang berjalan
KB-AMM-03	Sistem harus menyediakan layanan untuk melihat data notulensi musyawarah masjid	Takmir dan Remas	Proses melihat notulensi musyawarah melalui buku dan grup sosial media	Menyediakan sistem untuk melihat data notulensi seperti laporan, masukan, dan keputusan
KB-AMM-04	Sistem harus menyediakan layanan untuk mengelola notulensi	Notulen	Proses mengelola notulensi berjalan pada	Menyediakan sistem untuk mengelola data notulensi untuk dicatat pelaporan,

	musyawarah masjid		buku dan grup sosial media	usulan, dan pembahasan
KB-AMM-05	Mencari Notulensi	Takmir dan Remas	Proses pencarian notulensi dilakukan pada buku notulensi dan pencarian pada grup media sosial	Menyediakan sistem untuk mencari data notulensi berdasarkan kata tertentu yang sudah tersimpan pada sistem
KB-AMM-06	Menverifikasi Status Pekerjaan	Ketua Takmir	Proses verifikasi pekerjaan hanya pada saat melakukan musyawarah	Menyediakan sistem untuk menverifikasi pekerjaan masjid yang dapat digunakan sebagai pekerjaan sudah dikerjakan maupun belum berjalan
KB-AMM-07	Menverifikasi Notulensi	Amir Musyawarah	Proses verifikasi notulensi hanya pada grup sosial media	Menyediakan sistem untuk melakukan verifikasi notulensi untuk memastikan notulensi sudah sesuai dengan pembahasan musyawarah masjid

4.5.4.3 Identifikasi Pengguna

Identifikasi pengguna bertujuan untuk mengetahui peran-peran yang akan digunakan pada sistem. Identifikasi pengguna merupakan pengkategorian pengguna berdasarkan hasil analisis pemangku kepentingan. Identifikasi pengguna akan digunakan sebagai aktor dalam *use case* diagram.

Tabel 4.35 Hasil Identifikasi Pengguna

Tipe Pemangku Kepentingan	Tipe Pengguna	Deskripsi
Pengguna	Kepala Takmir	Seseorang yang memiliki tanggung jawab terhadap pengelolaan anggota musyawarah masjid dan menyetujui pekerjaan masjid
	Sekretaris Takmir	Seseorang yang memiliki tanggung jawab terhadap pengelolaan anggota musyawarah masjid
	Notulen	Seseorang yang memiliki tanggung jawab terhadap pencatatan notulensi saat musyawarah yang berlangsung
	Amir Musyawarah	Seseorang yang memiliki tanggung jawab terhadap berlangsungnya musyawarah masjid, pengambilan keputusan musyawarah masjid, dan menyetujui notulrensi yang dicatat
	Takmir dan Remas	Seseorang yang memiliki bagian dalam pelaksanaan musyawarah masjid

4.5.4.4 Identifikasi Fitur

Identifikasi fitur pada bagian ini merupakan solusi penambahan dari hasil penggalian permasalahan pada saat melakukan evaluasi tahap 1. Pada Tabel 4.36 terdapat kode fitur baru dan deskripsi singkat fitur yang akan menggambarkan layanan yang terdapat pada sistem. Hasil identifikasi fitur tersebut digunakan sebagai informasi dalam melakukan identifikasi persyaratan fungsional dan nonfungsional pada sistem informasi.

Tabel 4.36 Penambahan Identifikasi Fitur

Kode Fitur	Deskripsi
FT-AMM-06	Sistem mampu untuk pencarian informasi dari notulensi yang telah di simpan.
FT-AMM-07	Sistem mampu untuk menverifikasi status pekerjaan yang telah di simpan.
FT-AMM-08	Sistem mampu untuk menverifikasi informasi notulensi yang telah di simpan.

4.5.4.5 Identifikasi Fungsional

Persyaratan fungsional ini merupakan penjabaran kondisi yang harus dipenuhi dalam fitur tambahan pada identifikasi pada iterasi ini yang sudah didefinisikan. Persyaratan fungsional merupakan penjabaran detail dari fitur, penjabaran ini akan memudahkan peneliti dalam memenuhi rangka kemampuan fitur. Hubungan antara fungsional dan fitur dihubungkan sesuai dengan Tabel 4.37.

Tabel 4.37 Penambahan Identifikasi Fungsional

Kode Persyaratan Fungsional	Deskripsi	Kode Fitur
FTF-AMM-21	Sistem mampu menampilkan daftar notulensi musyawarah berdasarkan kata kunci tertentu.	FT-AMM-06
FTF-AMM-22	Sistem mampu menverifikasi pekerjaan baru masjid	FT-AMM-07
FTF-AMM-23	Sistem mampu mengubah status pekerjaan	FT-AMM-07
FTF-AMM-24	Sistem mampu mengubah status notulensi dari belum terverifikasi menjadi terverifikasi	FT-AMM-08
FTF-AMM-25	Sistem mampu mengubah status notulensi dari terverifikasi menjadi belum terverifikasi.	FT-AMM-08

4.5.4.6 Identifikasi Nonfungsional

Persyaratan nonfungsional ini merupakan hasil identifikasi yang tidak secara langsung berhubungan dengan fitur utama. Persyaratan nonfungsional akan digunakan untuk mendukung berjalannya sistem yang akan digunakan. Persyaratan nonfungsional memiliki kode fitur, kode persyaratan, dan deskripsi persyaratan

nonfungsional. Persyaratan nonfungsional dijelaskan pada Tabel 4.38**Error! Not a valid bookmark self-reference.**.

Tabel 4.38 Penambahan Identifikasi Nonfungsional

Kode Persyaratan Nonfungsional	Deskripsi	Kode Fitur
FTNF-AMM-01	Sistem dapat berjalan pada beberapa versi aplikasi peramban tanpa ada kendala.	FT-AMM-06
		FT-AMM-07
		FT-AMM-08

4.5.5 Pemodelan *Use Case*

Pemodelan *use case* kembali dilakukan untuk menggambarkan peran aktor memiliki kemampuan tambahan pada iterasi luar ke-2. Langkah yang digunakan sama seperti pada saat iterasi pertama yaitu mengidentifikasi aktor, mengidentifikasi use case diagram, identifikasi use case, dan spesifikasi use case.

4.5.5.1 Deskripsi Aktor

Deskripsi actor pada tahap ini merupakan penambahan informasi dan penjelasan secara singkat aktor-aktor yang sudah diidentifikasi dalam *use case* diagram. Deskripsi aktor berisi tanggung jawab dalam menggunakan sistem. Deskripsi ini dapat dilihat pada Tabel 4.39

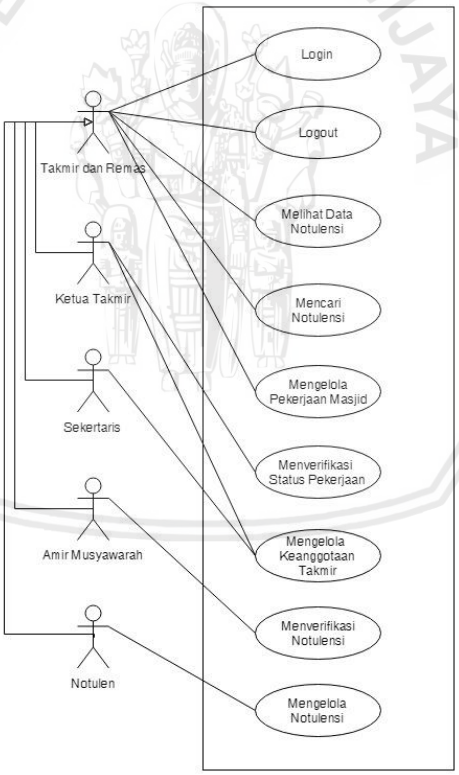
Tabel 4.39 Perubahaan Informasi Deskripsi Aktor

No	Nama Aktor	Deskripsi Aktor
1.	Takmir dan Remas	Aktor Takmir dan Remas yang diperankan oleh anggota Takmir maupun Remasmemiliki batasan untuk melihat notulensi, mencari notulensi, dan mengelola pekerjaan masjid.
2.	Ketua Takmir	Aktor Ketua takmir masjid yang di perankan oleh ketua takmir memiliki batasan untuk menverifikasi pekerjaan dan melakukan mengelola anggota Takmir dan Remas. Ketua Takmir juga memiliki kemampuan fungsi yang sama seperti Takmir dan Remas
3.	Sekretaris Takmir	Aktor Sekretaris Takmir yang di perankan oleh Sekretaris Takmir memiliki batasan untuk mengelola keanggotaan Takmir. Sekretaris juga memiliki kemampuan fungsi yang sama seperti Takmir dan Remas

No	Nama Aktor	Deskripsi Aktor
4.	Amir Musyawarah	Aktor Amir Musyawarah yang di perankan oleh Amir Musyawarah memiliki batasan untuk menverifikasi notulensi yang dibuat oleh notulen. Amir Musyawarah juga memiliki kemampuan fungsi yang sama seperti Takmir dan Remas
5.	Notulen	Aktor notulen diperankan oleh notulen musyawarah pada salah satu takmir. Notulen memiliki batasan untuk mengelola notulensi sesuai dengan notulen yang dibuat oleh aktor notulen. Notulen juga memiliki kemampuan fungsi yang sama seperti Takmir dan Remas

4.5.5.2 *Use Case* Diagram

Use case diagram pada iterasi ini merupakan pengembangan use case diagram dalam Gambar 4.6 menjadi Gambar 4.35. *Use case* dikembangkan pada mencari notulensi, menverifikasi status pekerjaan, dan menverifikasi notulensi.



Gambar 4.35 Pengembangan Use Case

Kemudian, hubungan antara bisnis *to-be* dengan *use case* dapat kita hubungkan dengan use case yang didefinisikan. Tujuan dari penghubungan ini untuk menunjukan hubungan antara pemodelan proses bisnis dan *use case* yang dibuat.

Tabel 4.40 Hubungan Proses Bisnis *To-Be* dan *Use Case* Diagram

No	Proses Bisnis To-Be	Aktivitas	*Use Case*
1.	Penelusuran Notulensi. (PBT-AMM-04)	Melakukan pencarian terhadap notulensi menggunakan kata kunci tertentu	Mencari Notulensi
2.	Menverifikasi status Pekerjaan. (PBT-AMM-05)	Melakukan verifikasi pekerjaan yang sudah dibuat	Menverifikasi status pekerjaan
3.	Menverifikasi Notulensi. (PBT-AMM-06)	Melakukan verifikasi notulensi yang sudah dibuat	Menverifikasi Notulensi

Setelah menghubungkan use case dan proses bisnis to-be, kemudian peneliti menghubungkan use case dan pengguna yang bertujuan untuk memahami kemampuan pengguna dalam penggunaan sistem. Hubungan ini dapat dilihan pada Tabel 4.41.

Tabel 4.41 Hasil Identifikasi Use Case Berdasarkan Pengguna

No	Kode	*Use case*	Pengguna
1.	UC-AMM-07	Pencarian notulensi	Takmir dan Remas.
2.	UC-AMM-08	Menverifikasi status pekerjaan	Ketua Takmir.
3.	UC-AMM-09	Menverifikasi notulensi	Amir Musyawarah.

Lalu dilakukan identifikasi hubungan dengan use case dengan fitur yang sudah dijabarkan sebelumnya. Hal ini bertujuan untuk menunjukan hasil pemodelan use case sesuai dengan hasil analisis persyaratan yang dijelaskan. Hubungan use case dengan fitur ditunjukan pada Tabel 4.42.

Tabel 4.42 Penambahan Hubungan Use Case dengan Fitur

No	*Use Case*	Fitur
1.	Pencarian Notulensi	FT-AMM-06
2.	Menverifikasi Status Pekerjaan	FT-AMM-07
3.	Menverifikasi Notulensi	FT-AMM-08

4.5.5.3 Spesifikasi *Use Case*

Spesifikasi *use case* berisi penjelasan mengenai setiap *use case* yang didefinisikan. Setiap spesifikasi use case terdapat tabel yang berisi *brief description*, aktor, *pre-condition*, *post-condition*, *basic flow*, *alternative flow* dan *sub flow*. *Brief description* berisi informasi dasar yang menunjukan kemampuan *use case*. Kemudian pada kolom aktor terdapat informasi aktor yang memiliki peran yang dapat melakukan *use case* tersebut. pada kolom *pre-condition* terdapat informasi yang dibutuhkan sebelum memulai *basic flow* tersebut, *pre-condition* sudah harus terpenuhi dalam memulai *basic flow*. *Basic flow* merupakan tahapan yang dilakukan oleh aktor dalam menjalankan *use case* tersebut. *Basic flow* merupakan langkah dalam mendapatkan tujuan utama dari masing masing use case. Tujuan utama dari masing-masing definisikan dalam kolom *post condition*. Kolom *post-condition* berisi informasi tujuan utama dari *use case* tersebut, kolom ini menunjukan hasil dari *use case* tersebut. kemudian terdapat *alternative-flow* merupakan *error-handling* saat basic flow tidak terpenuhi dalam sistem. *Sub-flow* merupakan penyederhanaan alur normal *use case* yang terlalu terperinci.

(a) Spesifikasi *Use Case* Pencarian Notulensi

Spesifikasi use case pencarian notulensi merupakan penjabaran detail use case dalam melakukan pencarian notulensi. Pencarian notulensi akan menggunakan atribut yang diisi oleh actor yaitu Takmir dan Remas. Atribut akan menjadi parameter sebagai informasi yang akan dicari oleh actor pada notulensi yang ada pada sistem. Penjelasan spesifikasi ini dijelaskan mendetail pada Tabel 4.43.

Tabel 4.43 Spesifikasi *Use Case* Pencarian Notulensi

Use case code	UC-AMM-07
Brief Description	*Use case* Mencari Notulensi menjelaskan bagaimana aktor Takmir dan Remas melakukan pencarian terhadap notulensi yang ada pada sistem.

Aktor	Takmir dan Remas
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Aktor telah berhasil masuk ke dalam sistem.
Post-condition	- Menampilkan notulensi yang sesuai dengan pencarian
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih untuk membuka daftar notulensi. 2. Aktor memilih memilih filter notulensi. **{Menampilkan formulir filter}** 3. Aktor mengisi atribut filter. 4. Aktor melakukan pencarian notulensi. **{Menampilkan notulensi sesuai filter}** **{*use case* selesai}** 5. *Use case* selesai.
Alternative Flow	**A1. Menangani notulensi tidak terdapat pada sistem** Pada **{ Menampilkan notulensi sesuai filter }** apabila tidak terdapat notulensi sesuai dengan filter, maka sistem akan menampilkan pesan tidak terdapat notulensi.

(b) Spesifikasi *Use Case* Menverifikasi Status Pekerjaan

Spesifikasi use case menverifikasi status pekerjaan merupakan penjabaran alur dalam melakukan verifikasi pekerjaan yang sudah dibuat. Use case ini hanya dapat dijalankan oleh ketua takmir sebagai penanggung jawab tertinggi terhadap pekerjaan yang ada pada masjid. Detail alur use case menverifikasi status pekerjaan dijabarkan pada Tabel 4.44.

Tabel 4.44 Spesifikasi *Use Case* Menverifikasi Status Pekerjaan

Use case code	UC-AMM-08

Brief Description	*Use case* Menverifikasi Status Pekerjaan menjelaskan bagaimana aktor Ketua Takmir melakukan perubahan status terverifikasi maupun belum terverifikasi terhadap pekerjaan yang ada pada sistem.
Aktor	Ketua Takmir
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Aktor telah berhasil masuk ke dalam sistem.
Post-condition	- Status Pekerjaan terverifikasi atau belum terverifikasi
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih untuk membuka halaman pekerjaan. 2. Aktor membuka salah satu pekerjaan masjid. **{Menampilkan detail pekerjaan}** 3. Aktor melakukan verifikasi atau pekerjaan selesai. **{Sistem menampilkan konfirmasi pilihan}** 4. Aktor mengkonfirmasi pilihan. **{Sistem menyimpan status pekerjaan}** **{*use case* selesai}** 5. *Use case* selesai.
Alternative Flow	**A1. Menangani konfirmasi dengan pilihan tidak** Pada **{Sistem *menampilkan konfirmasi pilihan*}** apabila Ketua takmir memilih pilihan tidak pada konfirmasi pilihan maka alur *alternative* ini akan kembali ke **{Menampilkan notulensi}**. **A2. Sistem Gagal Merubah Status** Pada **{Sistem *menyimpan status notulensi*}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{Menampilkan notulensi}**.

(c) Spesifikasi *Use Case* Menverifikasi Notulensi

Spesifikasi use case menverifikasi notulensi merupakan penjabaran alur dalam menjalankan verifikasi notulensi yang dapat dijalankan oleh Amir Musyawarah. Amir Musyawarah akan melakukan verifikasi terhadap notulensi yang sudah dibuat oleh notulen. Spesifikasi ini dijabarkan pada Tabel 4.45.

Tabel 4.45 Spesifikasi *Use Case* Menverifikasi Notulensi

Use case code	UC-AMM-09
Brief Description	*Use case* Menverifikasi Notulensi menjelaskan bagaimana aktor Amir Musyawarah melakukan perubahan status terverifikasi maupun belum terverifikasi terhadap notulensi yang ada pada sistem.
Aktor	Amir Musyawarah
Pre-condition	- Perangkat yang digunakan untuk mengakses sistem terhubung dengan internet. - Sistem terhubung dengan *server.* - Aktor telah berhasil masuk ke dalam sistem.
Post-condition	- Status notulensi terverifikasi atau belum terverifikasi
Basic Flow	**{*use case* dimulai}** 1. *Use case* dimulai ketika aktor pengguna memilih untuk membuka detail notulensi. **{Menampilkan notulensi}** 2. Aktor melakukan verifikasi atau tidak verifikasi. **{Sistem *menampilkan konfirmasi pilihan*}** 3. Aktor mengkonfirmasi pilihan. **{Sistem *menyimpan status notulensi*}** **{*use case* selesai}** 4. *Use case* selesai.
Alternative Flow	**A1. Menangani konfirmasi dengan pilihan tidak** Pada **{Sistem *menampilkan konfirmasi pilihan*}** apabila Ketua takmir memilih pilihan tidak pada konfirmasi pilihan

	maka alur *alternative* ini akan kembali ke **{Menampilkan notulensi}**. **A2. Sistem Gagal Merubah Status** Pada **{Sistem *menyimpan status notulensi*}** apabila sistem gagal melakukan penyimpanan status maka alur *alternative* ini akan kembali ke **{Menampilkan notulensi}**.

4.5.6 Proses Iterasi Dalam

4.5.6.1 Perancangan

Perancangan merupakan penjabaran rancangan mekanisme sistem akan berjalan. Perancangan pada iterasi ini akan dirancang dengan menggunakan UML (*Unified Modelling Language)* sebagai dasar implementasi pembangunan saat menerapkan mekanisme OOP (*Object Oriented Programming*).

(a) *Sequence* Diagram

Proses analisis sequence diagram merupakan penerapan hasil identifikasi alur berdasarkan bisnis proses to-be. Sequence diagram digunakan untuk melihat bagaimana interaksi aktor menggunakan sistem dan alur data yang dijalankan pada sistem. Pada iterasi dalam ini menjelaskan analisis sequence mencari notulensi, menverfikasi pekerjaan masjid, dan menverifikasi notulensi.

Dalam Gambar 4.37 menjelaskan alur mekanisme interaksi sequensial dalam mencari notulensi musyawarah. Controller yang akan digunakan sebagai media interaksi data adalah ControllerNotulensi. Controller ini dikembangkan dari iterasi luar ke-1 dengan menambahkan method membuka haman formCariNotulensi. Controller tersebut akan menampilkan boundary formCariNotulensi dan menggunakan ServiceAnggota, ServiceNotulensi, dan ServicePekerjaan untuk mengakses data yang akan ditampilkan.

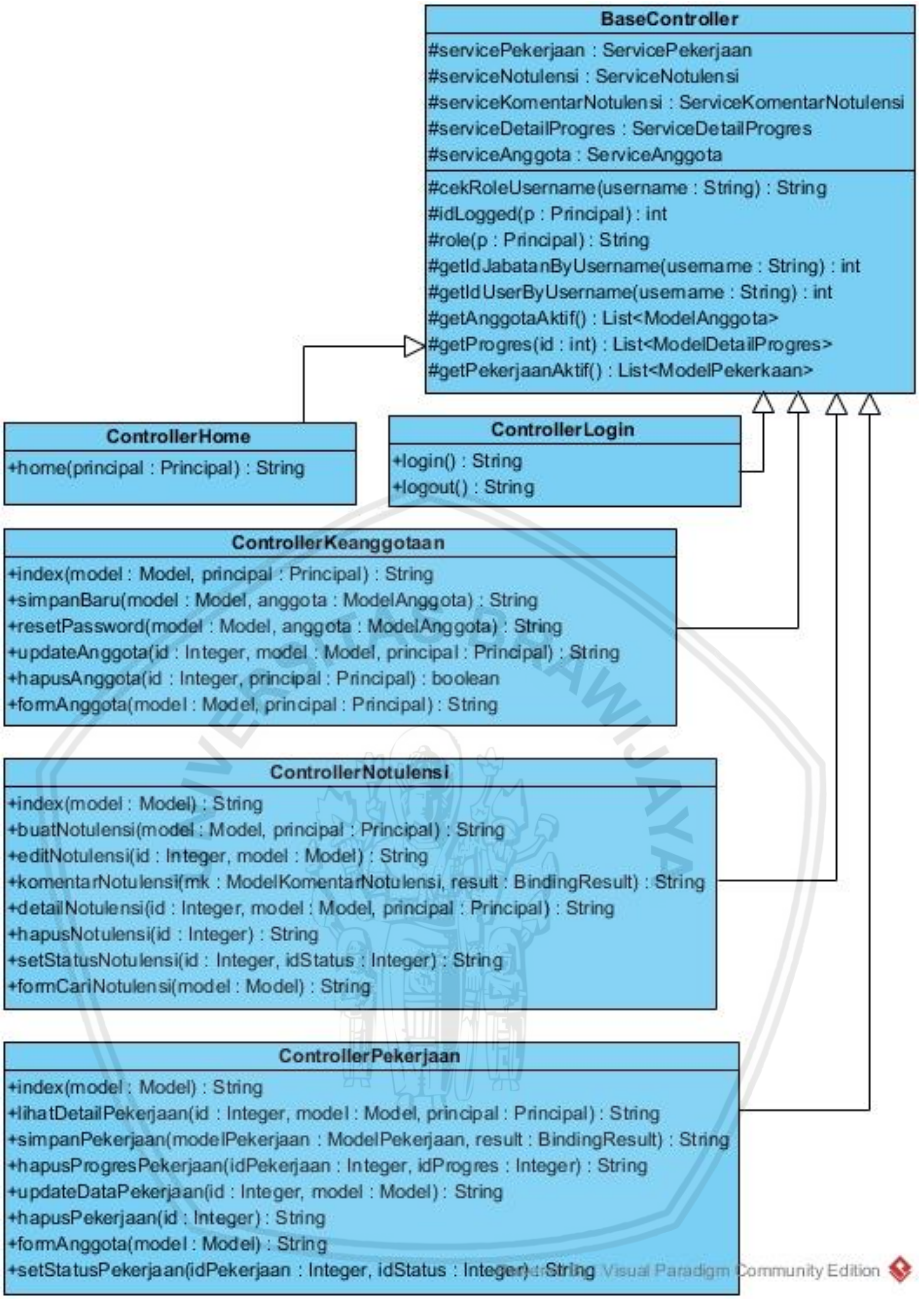
Kemudian dalam Gambar 4.39 menjelaskan alur mekanisme interaksi sequensial menverifikasi pekerjaan masjid. ControllerPekerjaan akan berfungsi untuk menangani permintaan aktor untuk menverifikasi pekerjaan masjid. Halaman pekerjaan yang dapat melakukan interaksi verifikasi pekerjaan yaitu pada boundary detail_pekerjaan. Pada boundary tersebut akan men-*trigger* controller untuk mengubah status sesuai dengan status tertentu.

Lalu dalam Gambar 4.38 menjelaskan alur mekanisme interaksi sequensial dalam menverifikasi notulensi musyawarah. Controller yang akan digunakan sebagai media interaksi data adalah ControllerNotulensi. Controller ini juga akan menerima masukan untuk mengubah status verifikasi yang akan di-*trigger* pada halaman detail notulensi.

(b) Class Diagram

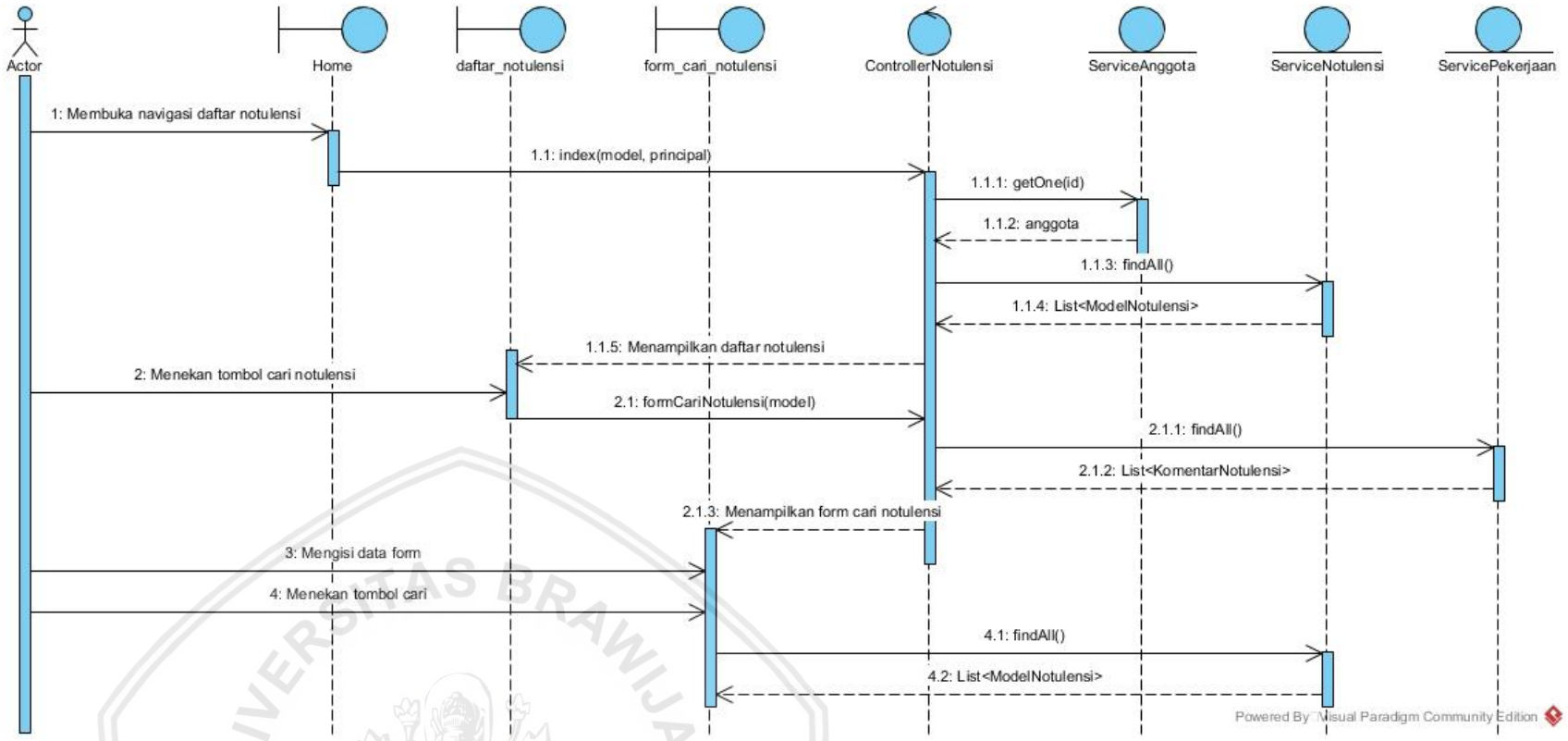
Diagram kelas yang digambarkan akan dibentuk berdasarkan diagram kelas sekuensial. Class diagram dimodelkan berdasarkan fitur dengan menerapkan konsep MVC (Model-View-Controller). Diagram Kelas digolongkan menjadi 2 jenis diagram, yaitu diagram *logical class* dan *model class*. *Logical class* digambarkan untuk mengetahui struktur dari *controller* pada sistem. Sedangkan *model class* digambarkan untuk mengetahui kebutuhan sistem dari sisi kebutuhan data.

Kelas logical diagram dalam Gambar 4.36 merupakan hubungan *class* diagram pada sistem informasi musyawarah. Pada kelas diagram tersebut terdapat Controller Pekerjaan, ControllerNotulensi, ControllerKeanggotaan, ControllerLogin, ControllerHome, dan BaseController. BaseController digunakan sebagai *super* dari *controller* lainnya untuk menurunkan atribut dan fungsi yang dapat digunakan bersama. Kelas ControllerLogin digunakan untuk keperluan mengakses fungsi masuk dan keluar dari sistem. Kelas ControllerHome digunakan sebagai fungsi halaman pertama setelah sistem menerima masukan *login* dari pengguna. Kelas ControllerKeanggotaan digunakan untuk keperluan mengakses fungsi sistem dan tampilan mengelola keanggotaan. Kemudian kelas ControllerNotulensi juga berfungsi sebagai media akses fungsi sistem notulensi dan tampilan notulensi. Kelas ControllerPekerjaan juga berfungsi sama seperti ControllerAnggota dan ControllerNotulensi tetapi mengakses fungsi dan tampilan dari mengelola pekerjaan. Terdapat penambahan fungsi pada iterasi luar ke-2 yaitu pada ControllerNotulensi dan ControllerPekerjaan. Penambahan fungsi pada ControllerNotulensi yaitu fungsi setStatusNotulensi dan formCariNotulensi. Pada ControllerPekerjaan terdapat penambahan fungsi setStatusPekerjaan.

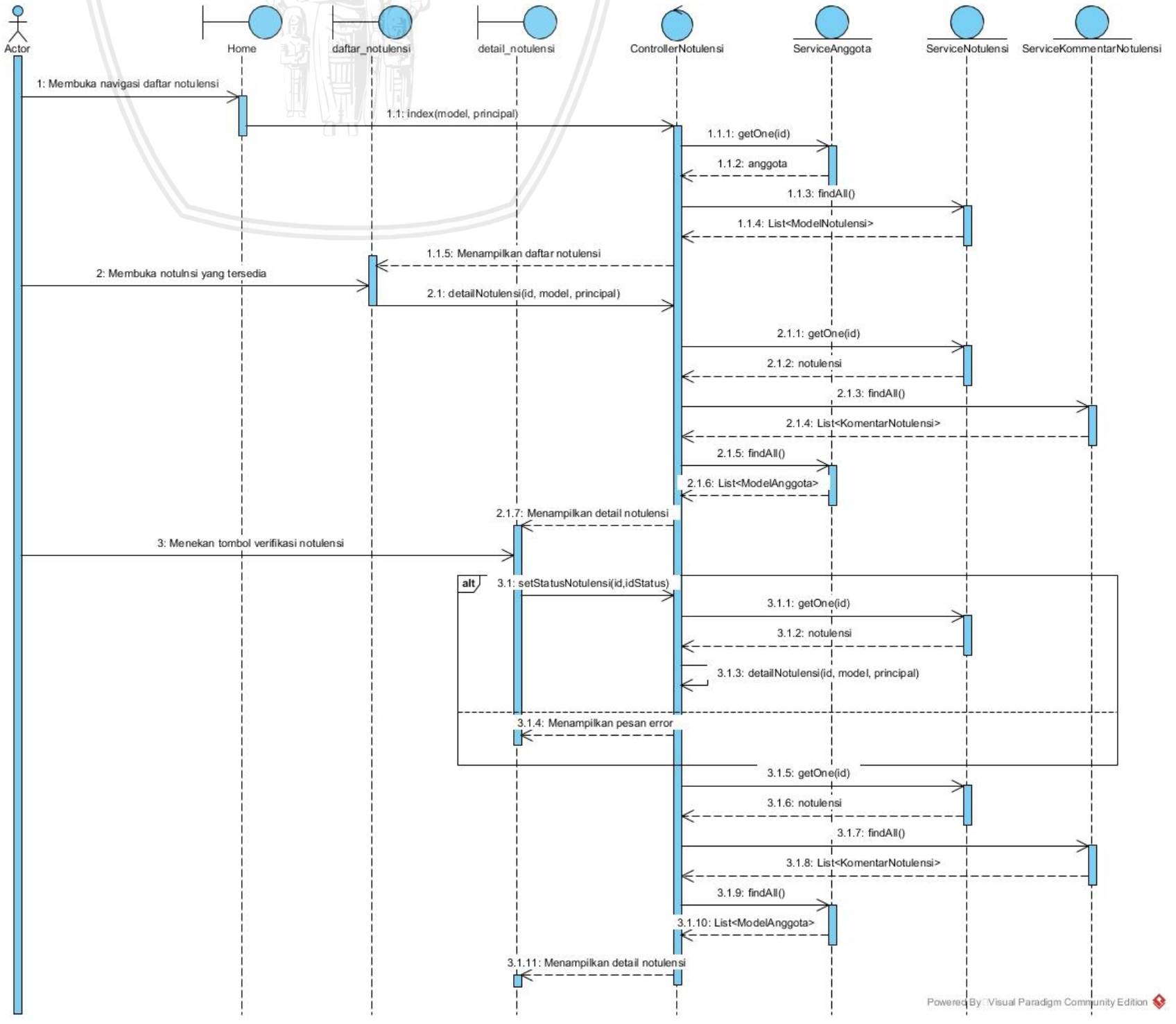


Gambar 4.36 Class Diagram Logical Sistem Informasi Musyawarah

repository.ub.ac.id

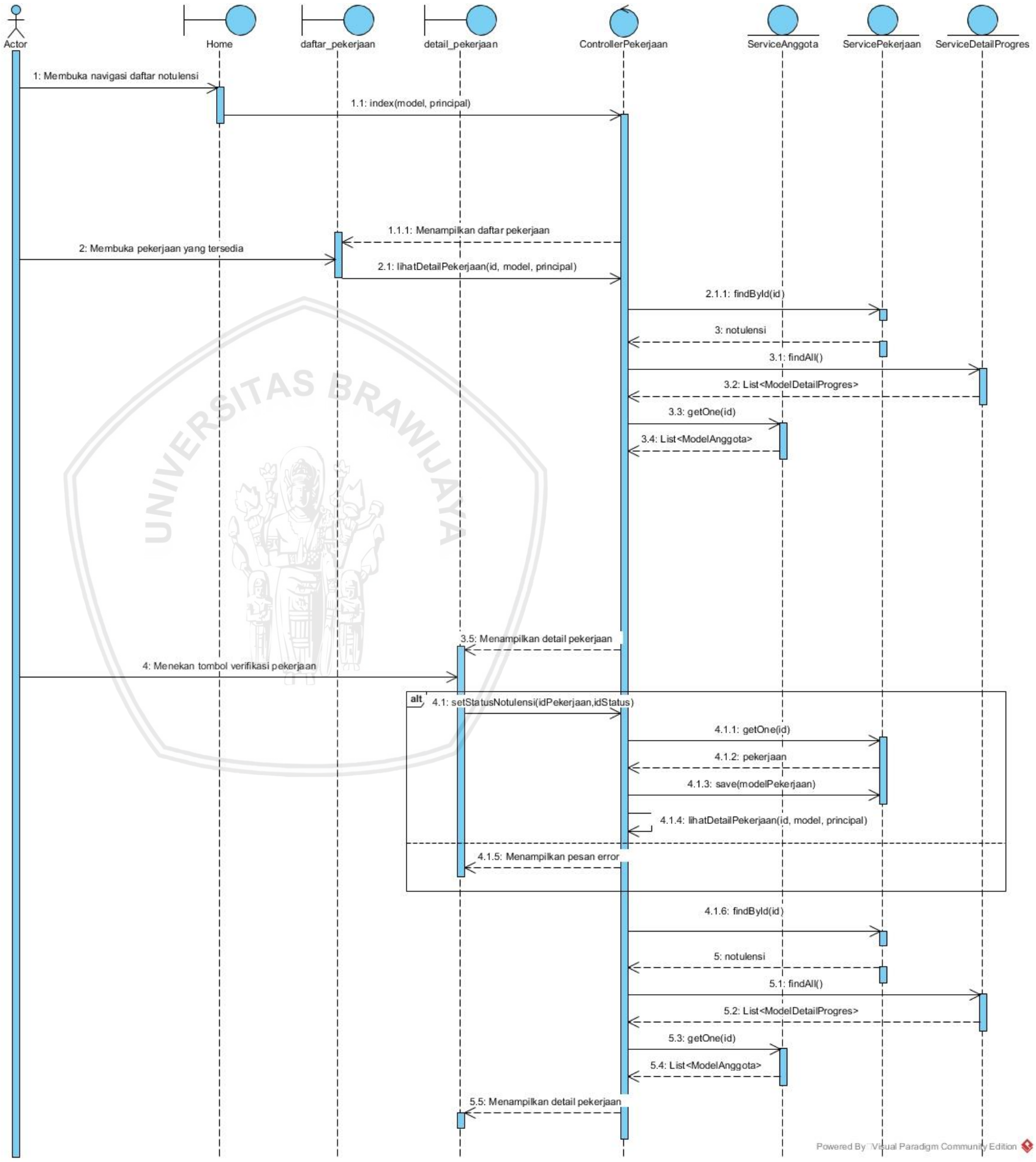


Gambar 4.37 *Sequence* Diagram Mencari Notulensi

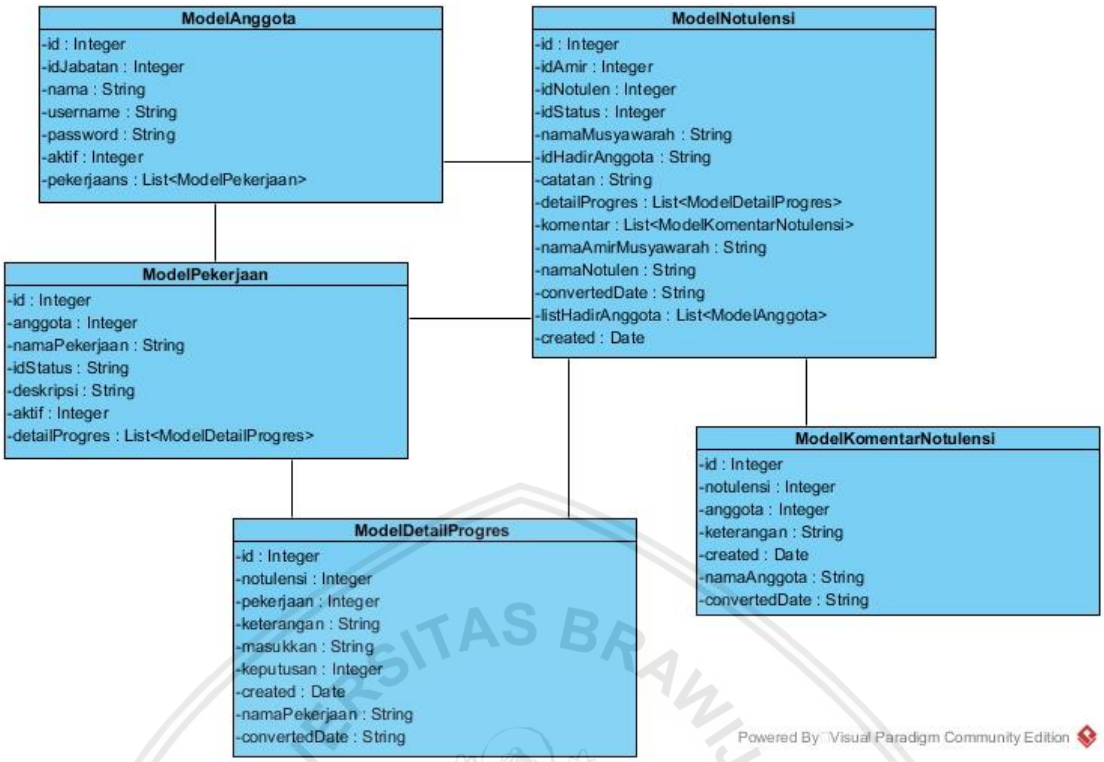


Gambar 4.38 *Sequence* Diagram Memverifikasi Notulensi

repository.ub.ac.id



Gambar 4.39 *Sequence* Diagram Memverifikasi Pekerjaan Masjid



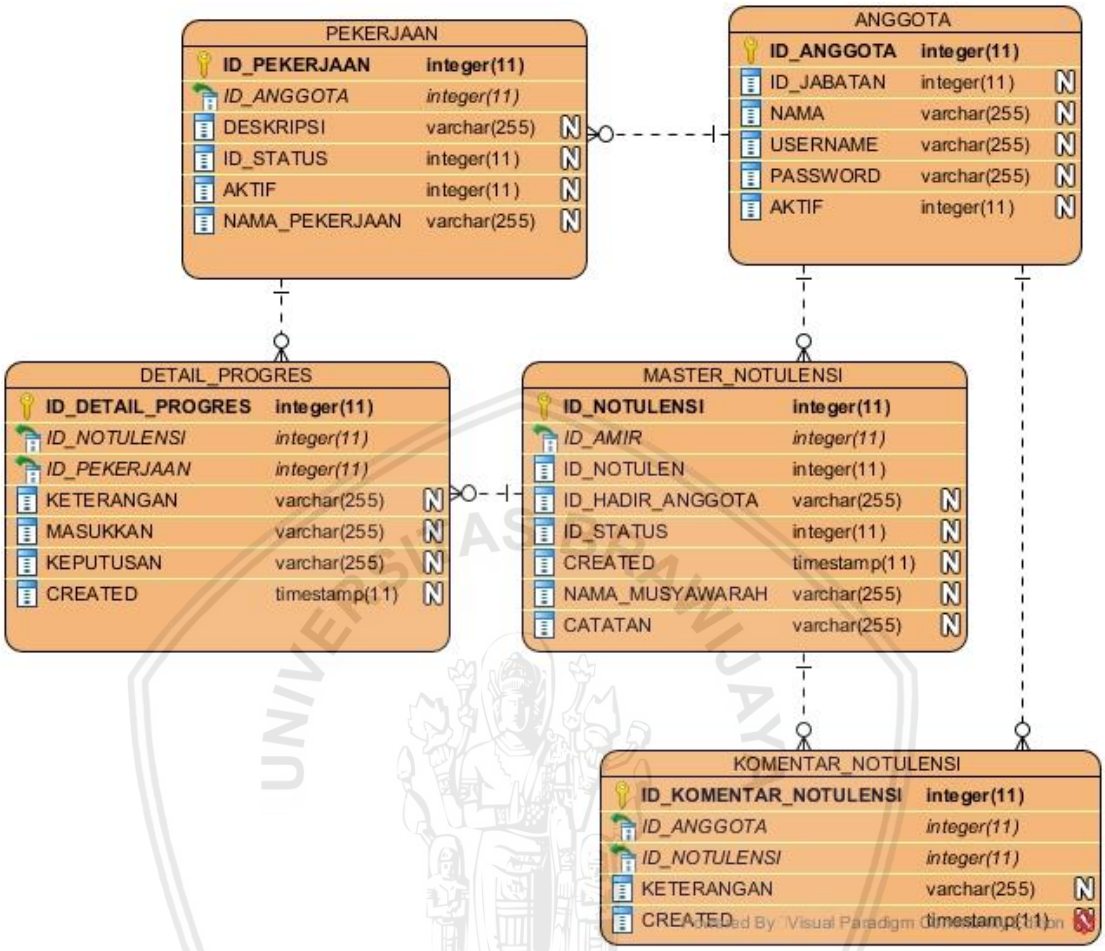
Gambar 4.40 Class Diagram Domain Model Sistem Informasi Musyawarah

Kelas model diagram mengelola keanggotaan dalam Gambar 4.40 merupakan kelas yang digunakan untuk acuan pemodelan data dalam sistem informasi musyawarah. terdapat 5 model yang dirancang yaitu ModelAnggota, ModelPekerjaan, ModelDetailProgres, ModelKomentarNotulensi, ModelNotulensi. Kelas ModelAnggota dijadikan sebagai acuan pemodelan data pengelolaan anggota. Kelas ModelPekerjaan terhubung dengan ModelAnggota dan ModelDetailProgres dijadikan sebagai acuan pemodelan data pengelolaan pekerjaan. Sedangkan ModelNotulensi terhubung dengan kelas model lainnya dijadikan sebagai acuan untuk pemodelan data pengelolaan notulensi. Pada diagram model Gambar 4.40 terdapat penambahan pada idStatus pada ModelPekerjaan dan ModelNotulensi yang akan digunakan sebagai penanda verifikasi.

(c) Pemodelan Data

Pemodelan data tidak terdapat penambahan class berdasarkan iterasi sebelumnya. Untuk melakukan pencarian notulensi, verifikasi pekerjaan, dan verifikasi notulensi sudah diatasi pada iterasi sebelumnya. Gambar 4.41 merupakan pemodelan yang sudah dikembangkan pada iterasi luar ke-1. Pada entitas MASTER_NOTULENSI dan PEKERJAAN terdapat penambahan ID_STATUS yang

digunakan sebagai penanda bahwa pekerjaan maupun notulensi diverifikasi atau tidak.



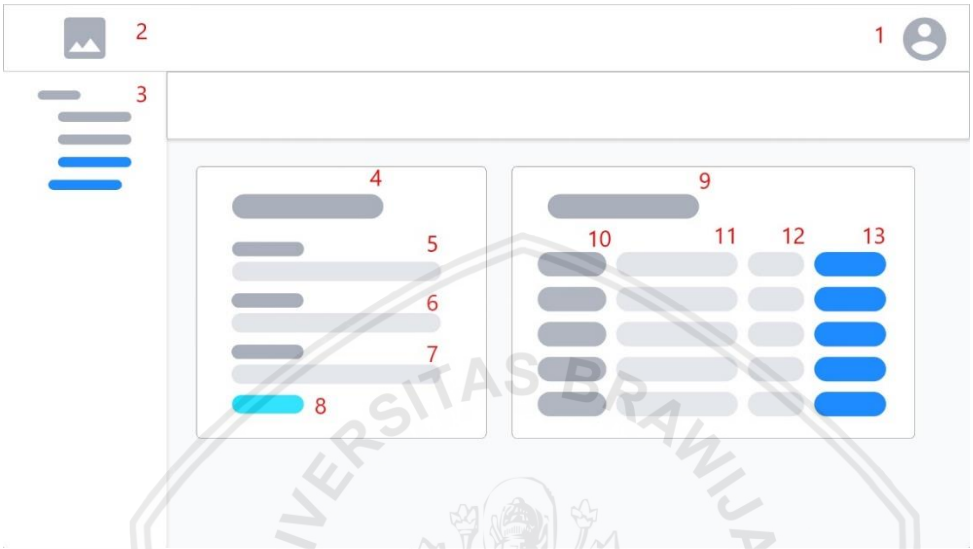
Gambar 4.41 Pemodelan Data Sistem Informasi Musyawarah

(d) Perancangan Antarmuka

Perancangan antarmuka pada iterasi luar ke-2 ini memiliki penambahan komponen pada halaman detail pekerjaan, daftar notulensi dan detail notulensi yang diilustrasikan dalam Gambar 4.43, Gambar 4.44, dan Gambar 4.45. Kemudian terdapat penambahan halaman cari notulensi yang diilustrakan dalam Gambar 4.42.

Pada perancangan antarmuka cari notulensi Gambar 4.42, terdapat informasi sebagai berikut: judul 'Cari Notulensi' (4), Masukan tanggal notulensi(5), pilihan pekerjaan(6), *keyword* pencarian(7), tombol mencari (8), judul 'Data Notulensi'(9), tanggal notulensi yang dicari(10), nama musyawarah yang dicari(11), keyword yang terdapat pada isi notulensis](12), dan tombol lihat notulensi(13). Pengguna dapat memasukan input berupa tanggal(5), memilih pekerjaan yang akan dicari(6), *keyword*

yang ingin ditemukan dalam notulensi musyawarah(7). Kemudian pengguna akan menekan tombol cari(8) yang kemudian sistem akan mencari dan menampilkan dalam bentuk daftar. Daftar tersebut akan menampilkan tanggal(10), nama musyawarah(11), keyword yang terdapat pada notulensi(12), dan pengguna dapat menekan tombol lihat notulensi(13) yang akan membuka notulensi berdasarkan musyawarah tersebut.

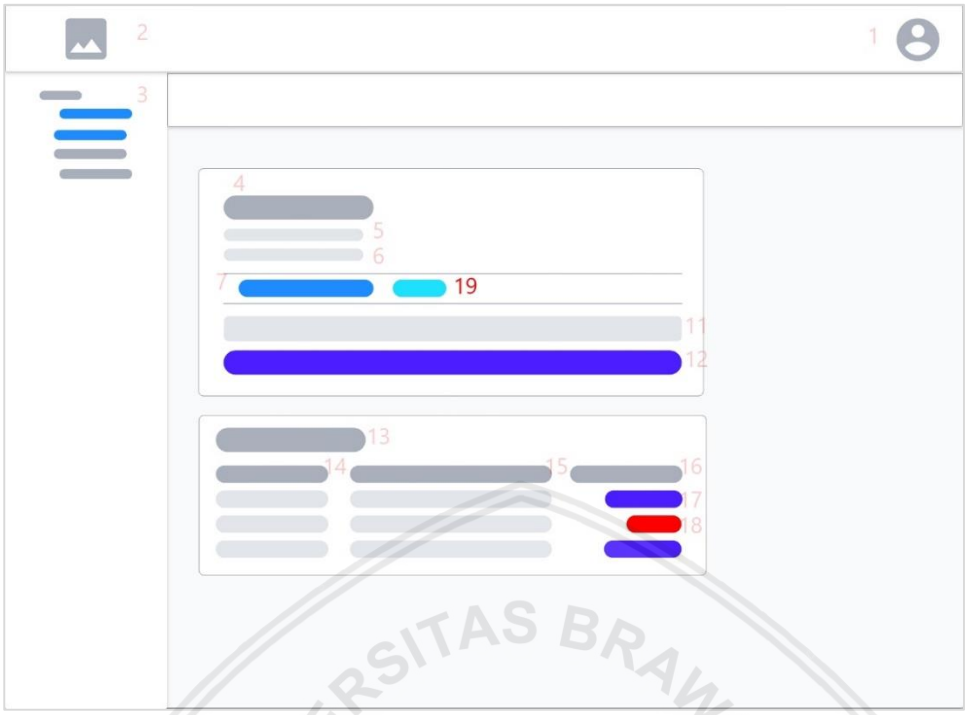


Gambar 4.42 Cari Notulensi

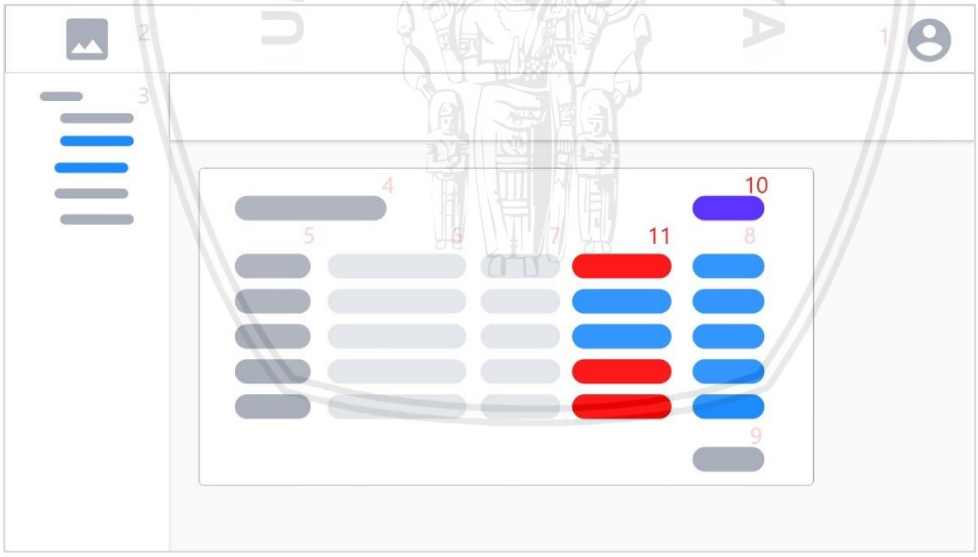
Pada perancangan antarmuka detail pekerjaan Gambar 4.43, terdapat penambahan komponen verifikasi pekerjaan (19). Tombol verifikasi pekerjaan tersebut berfungsi untuk membantu pengguna mengubah status menjadi pekerjaan sedang berjalan.

Pada perancangan antarmuka daftar notulensi Gambar 4.44, terdapat penambahan tombol cari notulensi(10) dan komponen status notulensi (11). Cari notulens (10) tersebut berfungsi untuk membantu pengguna berpindah halaman untuk mengakses halaman cari notulensi. Status notulensi (11) berisi informasi status terverifikasi atau belum terverifikasi amir. Status akan berwarna merah menunjukan bahwa notulensi belum terverifikasi, dan biru menunjukan status sudah diverifikasi.

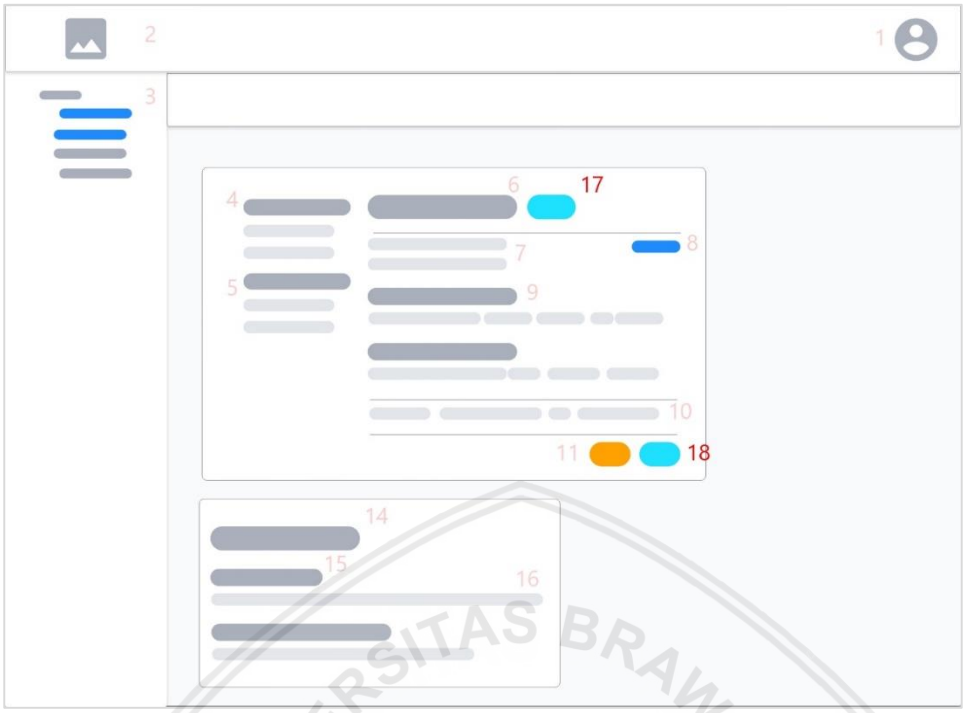
Pada perancangan antarmuka detail notulensi Gambar 4.45, terdapat penambahan komponen informasi terverifikasi (17) dan tombol verifikasi (18). Informasi terverifikasi (17) akan muncul apabila amir musyawarah sudah melakukan verifikasi dengan menekan tombol verifikasi (18). Dengan aksi tersebut, menambah komentar akan dihilangkan dan tombol edit notulensi (11) akan dihilangkan.



Gambar 4.43 Pembaharuan Detail Pekerjaan



Gambar 4.44 Pembaharuan Daftar Notulensi



Gambar 4.45 Pembaharuan Detail Notulensi

4.5.6.2 Struktur Artefak Sistem

Struktur artefak sistem merupakan informasi artefak dalam sistem informasi yang dikembangkan. Terdapat 2 *package* utama yaitu *package* view dan *package* java. *Package* view merupakan *package* yang dijadikan sebagai tampilan pada sistem, sedangkan *package* java merupakan *package* logika sistem dengan menggunakan bahasa *java*. Struktur artefak sistem pada iterasi ini diilustrasikan dalam Gambar 4.47.

Package view berisikan tampilan yang diimplementasi dalam bentuk kode html yang akan digunakan pada sistem. Terdapat folder view dari setiap fungsi utama pada sistem, yaitu folder anggota, pekerjaan, dan notulensi. Pada folder view utama terdapat view layout.html, login.html, home.html. Pada folder view anggota terdapat daftar_anggota.html dan form_anggota.html. Kemudian, pada folder view pekerjaan terpadat daftar_pekerjaan.html, detail_pekerjaan.html dan form_pekerjaan.html. Pada folder *view* notulensi terdapat daftar_notulensi.html, detail_notulensi.html, form_notulensi.html, form_notulensi_edit.html, dan penambahan *view* yaitu form_cari_notulensi.html.

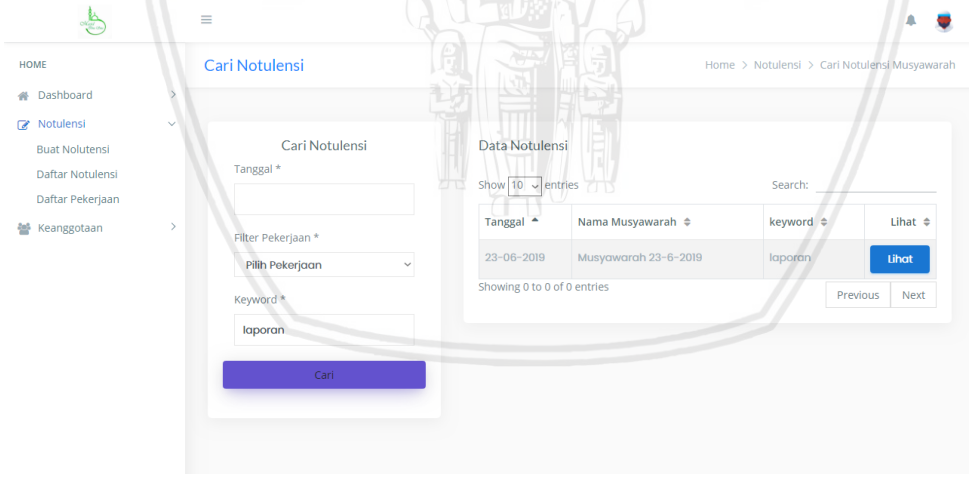
Package java terdapat lima folder yang dipisahkan berdasarkan fungsinya, terdapat folder config, folder service, folder rest, folder controller, dan folder model. Folder controller memiliki *class* sebagai fungsi untuk dapat mengakses halaman yang terdapat pada sistem. Terdapat enam controller yang terdapat pada package yaitu

BaseController.java, ControllerHome.java, ControllerKeanggotaan.java, ControllerLogin.java, ControllerNotulensi.java, ControllerPekerjaan.java. Folder model memiliki class objek yang digunakan untuk menjadi perantara, saat melakukan interaksi data pada objek. Terdapat lima model yang terdapat pada sistem yaitu ModelAnggota.java, ModelPekerjaan.java, ModelNotulensi.java, ModelDetailProgres.java, dan ModelKomentarNotulensi.java. Folder service merupakan interface yang implement fungsi java spring untuk digunakan. terdapat lima service yang terdapat pada sistem yaitu ServiceAnggota.java, ServicePekerjaan.java, ServiceNotulensi.java, ServiceDetailProgres.java, dan ServiceKomentarNotulensi.java. Pada folder config memiliki class untuk melakukan konfigurasi sistem untuk melakukan *authentication*. Terdapat 3 kelas java yaitu MvcConfig.java, MyUserDetailsService.java, dan WebSecurityConfig.java.

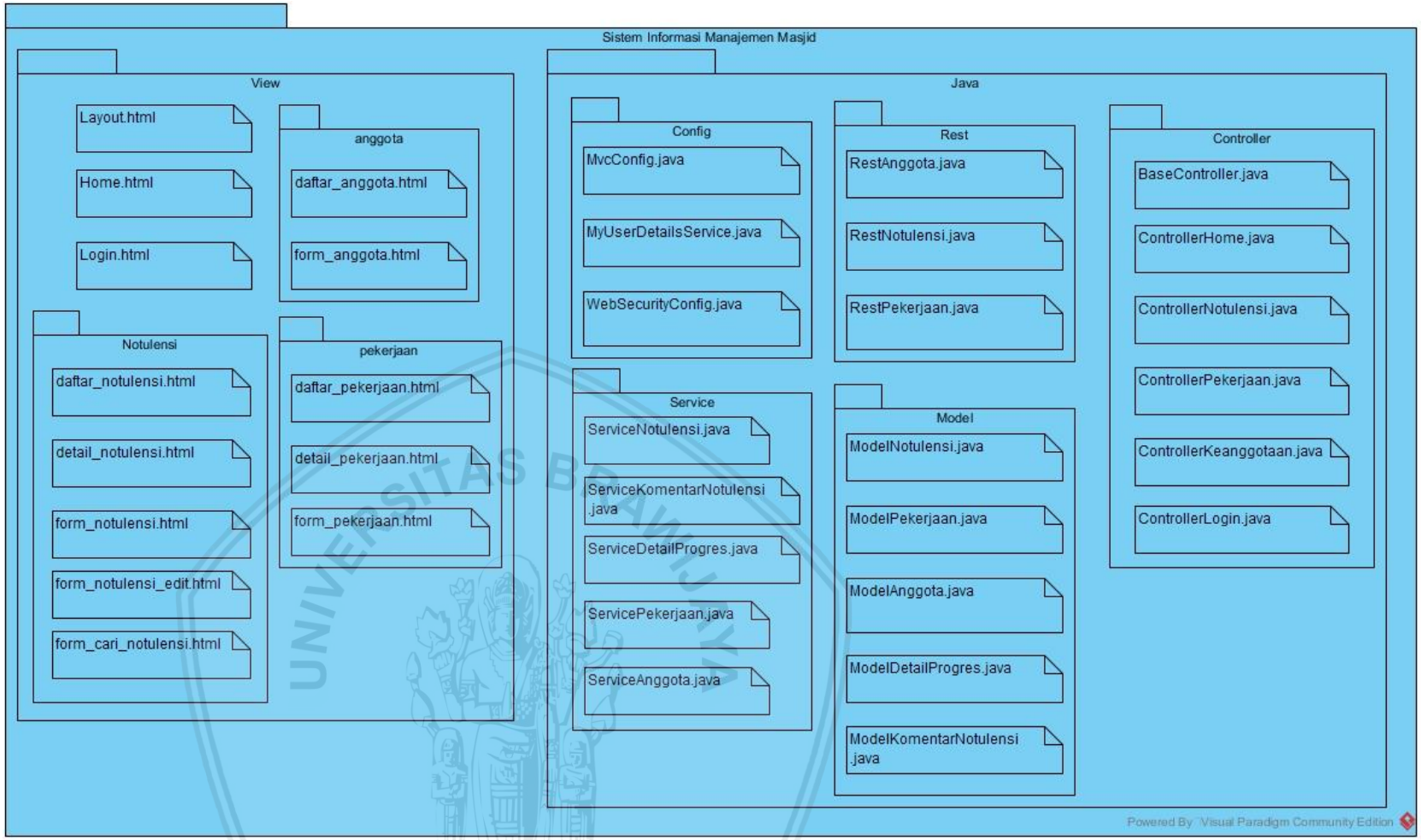
4.5.6.3 Implementasi

(a) Implementasi Desain Antarmuka

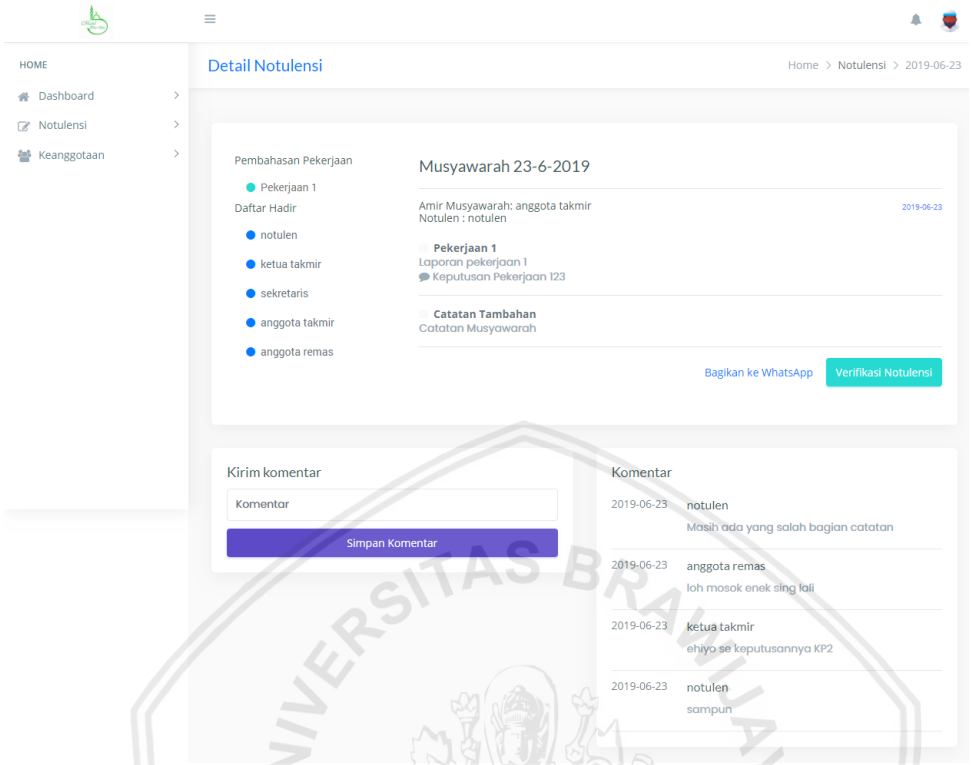
Implementasi desain antarmuka terdapat tampilan yang mewakili penggambaran bentuk sistem. Tampilan desain antarmuka dikembangkan berdasarkan perancangan antarmuka sebelumnya. Terdapat pencarian notulensi dalam Gambar 4.46, verifikasi notulensi Gambar 4.48, dan verifikasi pekerjaan Gambar 4.49.



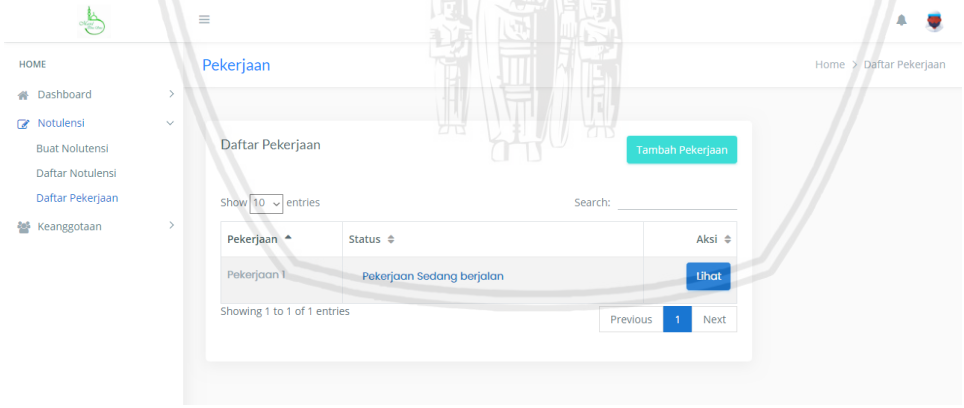
Gambar 4.46 Pencarian Notulensi



Gambar 4.47 Struktur Artefak Sistem



Gambar 4.48 Verifikasi Notulensi

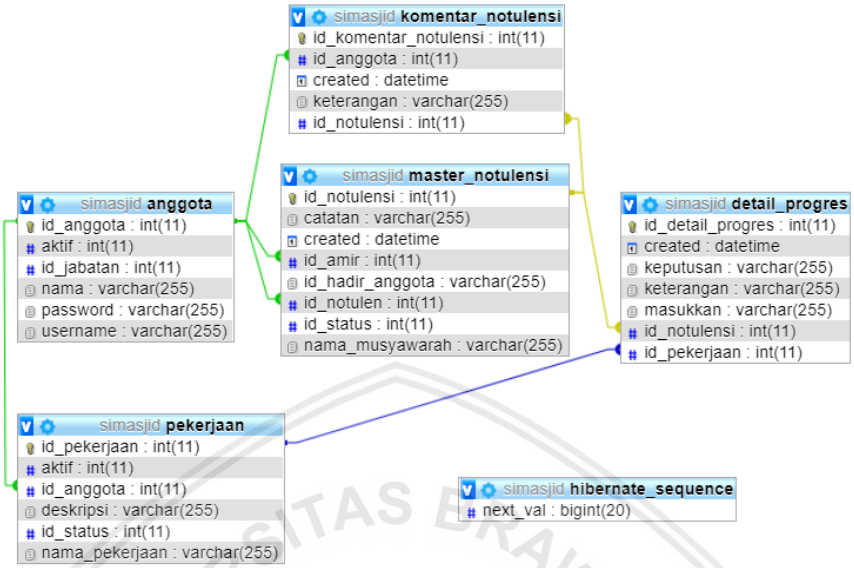


Gambar 4.49 Pekerjaan Sedang Berjalan

(b) Implementasi Data Definition Language

Dalam Gambar 4.50 merupakan hasil implementasi Data Definition Language (DDL) berdasarkan perubahan perancangan pada iterasi ini. Terdapat penambahan

id_status bernilai integer pada tabel anggota dan tabel master_notulensi pada DDL tersebut.



Gambar 4.50 Implementasi *Data Definition Language*

(c) Implementasi Fungsi

Berikut ini penambahan fungsi pada controller pekerjaan dan controller notulensi pada sistem. Mapping '/notulensi/cari' merupakan fungsi untuk mencari notulensi berdasarkan atribut tertentu yang disediakan pada form. Kemudian, mapping '/pekerjaan/setstatus/{id}/{status}' merupakan proses untuk mengganti status pekerjaan pada sistem. Pekerjaan akan diset menjadi terverifikasi, berjalan, dan selesai. Mapping '/notulensi/setstatus/{id}/{status}' merupkana proses untuk mengganti status notulensi, status yang dapat terjadi pada notulensi yaitu pending dan verifikasi.

```
Penambahan pada baris kode pada Controller Notulensi
 /*Iterasi luar 2*/
 @RequestMapping(value
   = "/notulensi/setStatus/{idNotulensi}/{idStatus}",
    method = RequestMethod.GET)
 public String setStatusNotulensi(@PathVariable Integer idNotulensi,
    @PathVariable Integer idStatus){
      ModelNotulensi mn = serviceNotulensi.getOne(idNotulensi);
      mn.setIdStatus(idStatus);
```

```
        serviceNotulensi.save(mn);
        return "redirect:/notulensi/detail/"+idNotulensi;
    }

    @RequestMapping(value = "/notulensi/cari",
        method = RequestMethod.GET)
    public String formNotulensi(Model model){
        model.addAttribute("pekerjaans",getPekerjaanAktif());
        return "notulensi/form_cari_notulensi";
    }

Penambahan pada baris kode pada Controller Pekerjaan
/*Iterasi luar 2*/
    @RequestMapping(value = "/pekerjaan/setstatus/{idPekerjaan}
       /{idStatus}", method = RequestMethod.GET)
    public String setStatusPekerjaan(@PathVariable Integer
      idPekerjaan, @PathVariable Integer idStatus) {
        ModelPekerjaan mp = servicePekerjaan.getOne(idPekerjaan);
        mp.setIdStatus(idStatus.toString());
        servicePekerjaan.save(mp);
        return "redirect:/pekerjaan/detail/" + idPekerjaan;
    }
```

4.5.6.4 Integrasi Sistem

Proses integrasi pada iterasi luar dua ini mengalami penambahan fitur seperti verifikasi dan pencarian. Baris kode terdapat penambahan pada controller dan model yang ada. Controller memiliki penambahan mapping halaman yang dijadikan sebagai fitur. Sedangkan pada model terdapat penambahan atribut yang digunakan sebagai penanda status.

4.5.7 Evaluasi Sistem

4.5.7.1 Pengujian Validasi

Pengujian validasi dilakukan untuk memastikan bahwa sistem dapat berjalan sesuai dengan persyaratan dan ekspetasi pengguna. Pengujian ini akan dilakukan berdasarkan scenario yang didapatkan dari beberapa alur *use case*. Kemudian kasus uji tersebut akan ditentukan berdasarkan scenario uji yang ditentukan. Pengujian ini

dilakukan untuk pada usecase mencari notulensi, menverifikasi notulensi, dan menverifikasi pekerjaan.

(a) Pengujian Validasi Mencari Notulensi

Pengujian validasi mencari notulensi menjelaskan pengujian sistem yang dapat dilakukan pengguna saat melakukan pencarian notulensi musyawarah. Skenario ini akan diuji kasuskan seperti pada Tabel 4.46.

Tabel 4.46 Matriks Skenario Mencari Notulensi

Kode Pengujian	Alur Awal	Alur Alternatif
V -MCN-01	Basic Flow	-
V- MCN-02	Basic Flow	A1. Tidak terdapat notulensi

(i) Pengujian Validasi Fungsi Mencari Notulensi

Tabel 4.47 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-21. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MCN-01. Pengujian ini menunjukan hasil valid pada fungsi mencari notulensi.

Tabel 4.47 Rencana Pengujian dan Kasus Uji Fungsi Mencari Notulensi

Kode Pengujian	V-MCN-01
Kode Persyaratan	FTF-AMM-21
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat mencari notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai anggota takmir dan remas 2. Penguji memilih fitur cari notulensi 3 Penguji memasukan data tanggal musyawarah, pekerjaan yang dicari, kata kunci yang terdapat pada musyawarah 5. Penguji mencari notulensi

Hasil yang diharapkan	Sistem dapat mencari notulensi yang memiliki atribut yang dimasukan
Hasil pengujian	Sistem berhasil menampilkan notulensi sesuai atribut yang dimasukan
Status pengujian	Valid
Masukan dan saran	

(ii) Pengujian Validasi Fungsi Tidak Terdapat Notulensi

Tabel 4.48 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-21 namun pada saat data tidak ditemukan. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MCN-02. Pengujian ini menunjukan hasil valid pada alternatif fungsi mencari notulensi.

Tabel 4.48 Rencana Pengujian dan Kasus Uji Fungsi Tidak terdapat Notulensi

Kode Pengujian	V-MCN-02
Kode Persyaratan	FTF-AMM-21
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat mencari notulensi musyawarah berdasarkan pada sistem namun tetap menampilkan pesan saat tidak terdapat data yang tersedia.
Prosedur Uji	1. Penguji teridentifikasi sebagai anggota takmir dan remas 2. Penguji memilih fitur cari notulensi 3 Penguji memasukan data tanggal musyawarah, pekerjaan yang dicari, kata kunci yang terdapat pada musyawarah 5. Penguji mencari notulensi
Hasil yang diharapkan	Sistem dapat menampilkan pesan tidak terdapat notulensi yang memiliki atribut yang dimasukan

Hasil pengujian	Sistem berhasil menampilkan pesan tidak terdapat notulensi yang memiliki atribut yang dimasukan
Status pengujian	Valid
Masukan dan saran	

(b) Pengujian Validasi Menverifikasi Status Pekerjaan

Pengujian validasi menverifikasi status pekerjaan menjelaskan pengujian sistem yang dapat dilakukan pengguna saat melakukan verifikasi status pekerjaan. Skenario ini akan diuji kasuskan seperti pada Tabel 4.49.

Tabel 4.49 Matriks Skenario Menverifikasi Status Pekerjaan

Kode Pengujian	Alur Awal	Alur Alternatif
V-MVP-01	Basic Flow	-

(i) Pengujian Validasi Fungsi Menverifikasi Status Pekerjaan

Tabel 4.50 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-22. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MVP-01. Pengujian ini menunjukan hasil valid pada fungsi menverifikasi status pekerjaan.

Tabel 4.50 Rencana Pengujian dan Kasus Uji Fungsi Menverifikasi Status Pekerjaan

Kode Pengujian	V-MVP-01
Kode Persyaratan	FTF-AMM-22
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat menverifikasi status pekerjaan pada sistem
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua atau sekretaris masjid 2. Penguji membuka detail pekerjaan 3. Penguji memilih menverifikasi pekerjaan
Hasil yang diharapkan	Sistem dapat mengubah status pekerjaan

Hasil pengujian	Sistem berhasil mengubah status pekerjaan
Status pengujian	Valid
Masukan dan saran	

(c) Pengujian Validasi Menverifikasi Notulensi

Pengujian validasi menverifikasi status pekerjaan menjelaskan pengujian sistem yang dapat dilakukan pengguna saat melakukan verifikasi notulensi. Skenario ini akan diuji kasuskan seperti pada Tabel 4.51.

Tabel 4.51 Matriks Skenario Menverifikasi Notulensi

Kode Pengujian	Alur Awal	Alur Alternatif
V-MVN-01	Basic Flow	-

(i) Pengujian Validasi Fungsi Menverifikasi Notulensi

Tabel 4.52 berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-24. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MSP-01. Pengujian ini menunjukan hasil valid pada fungsi menverifikasi notulensi.

Tabel 4.52 Rencana Pengujian dan Kasus Uji Fungsi Menverifikasi Notulensi

Kode Pengujian	V-MSP-01
Kode Persyaratan	FTF-AMM-24
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat menverifikasi notulensi pada sistem
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua atau sekretaris masjid 2. Penguji membuka detail notulensi 3. Penguji memilih menverifikasi notulensi
Hasil yang diharapkan	Sistem dapat menverifikasi notulensi
Hasil pengujian	Sistem berhasil menverfikasi notulensi

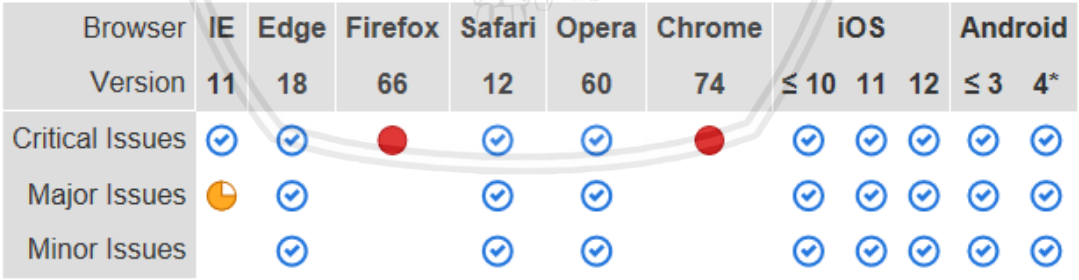
Status pengujian	Valid
Masukan dan saran	

4.5.7.2 Hasil Pengujian Validasi

Pengujian validasi diuji oleh Takmir dan Remas untuk menunjukan kebutuhan sistem sudah terimplementasi dan diterima oleh Takmir dan Remas. Pengujian validasi pada mencari notulensi, menverifikasi status pekerjaan, dan menverifikasi status notulensi menunjukan hasil yang valid. Terdapat masukan dan tanggapan dari penguji untuk menambahkan fitur tertentu untuk dikembangkan lebih lanjut. Pada fitur menverifikasi pekerjaan terdapat penambahan fitur untuk mengirim gambar sesuai dengan pekerjaan yang diberikan. Kemudian terdapat penambahan fitur untuk mengkategorikan pekerjaan untuk memberikan akses kepada anggota agar dapat bekerja sama pada satu pekerjaan. Pada pengujian validasi mencari notulensi dan menverifikasi notulensi belum terdapat masukan tetapi sudah divalidasi oleh penguji. Pada hasil pengujian tersebut dapat dinyatakan proses pengujian validasi proses iterasi luar ke-2 dinyatakan valid dan kebutuhan fungsional telah terpenuhi.

4.5.7.3 Pengujian Kompatibilitas Peramban

Pada subbab pengujian kompatibilitas peramban menjelaskan pengujian kompatibilitas sistem terhadap peramban tertentu untuk mengetahui permasalahan pada sistem di peramban yang ada. Pengujian ini menggunakan aplikasi SortSite untuk menguji kompatibilitas sistem. Dalam Gambar 4.51 menjelaskan skor kompatibilitas sistem pada aplikasi SortSite.



Gambar 4.51 Hasil Pengujian Kompatibilitas

Gambar 4.50 menunjukan hasil pengujian kompatibilitas terjadi *critical issue* pada peramban *Firefox* versi 66 dan *Chrome* versi 74. *Critical issue* ini terjadi karena sistem belum menggunakan *Secure Socket Layer* (SSL) sehingga peramban memberikan peringatan belum terdapat SSL. Kemudian juga terdapat *major issue* pada *Internet*

Explorer (IE) versi 11. Major issue ini terjadi karena IE belum belum mendukung *css* yang terdapat pada *framework layout boostrap*.

BAB 5 KESIMPULAN

5.1 Kesimpulan

Berdasarkan penelitian yang dilakukan pada Takmir Masjid Ibnu Sina, didapatkan kesimpulan penelitian sebagai berikut:

1. Hasil analisis persyaratan terdapat 4 hasil analisis proses bisnis saat ini, analisis masalah berdasarkan proses bisnis saat ini, identifikasi tipe pemangku kepentingan, dan identifikasi pengguna. Kemudian, hasil analisis persyaratan pada fitur, persyaratan fungsional dan nonfungsional dari sistem terdapat 9 fitur, 25 persyaratan fungsional, serta 1 persyaratan nonfungsional. Hasil analisis tersebut juga memuat informasi mengenai tujuan penggunaan sistem dan urutan aktivitas yang dilakukan untuk mencapai tujuan tersebut. Informasi yang terdokumentasikan terdapat 9 *use case.* Terdapat 6 *use c*ase pada iterasi dalam pertama dan 3 *use case* pada iterasi kedua. Setiap use case akan dijelaskan urutan aktivitas utama serta aktivitas alternatif yang didokumentasikan pada spesifikasi *use case*.
2. Hasil perancangan sistem yang dilakukan berdasarkan hasil analisis persyaratan terdiri dari rancangan arsitektur sistem untuk mengetahui struktur komponen dan hubungan antar komponen. Perancangan tersebut dipetakan dalam bentuk pola *Model-View-Controller* yang entitasnya digambarkan dalam bentuk *class* diagram. Pada hasil perancangan sistem terdapat 21 notasi pada struktur artefak sistem yang dimodelkan diantaranya merupakan 5 *model,* 5 *service,* 5 *controller, 3 config,* dan 3 *rest.* Lalu, hasil perancangan sistem juga menghasilkan pemodelan interaksi object berupa *sequence diagram*, *physical data mode*l, beberapa algoritme sistem, dan beberapa rancangan antarmuka pengguna. Hasil implementasi berdasarkan perancangan sistem berupa sistem informasi musyawarah berbasis situs web menggunakan framework spring boot. Spring boot menggunakan bahasa java sebagai dasar dalam mengembangan website sistem informasi musyawarah. spring boot juga dapat menggunakan thymeleaf sebagai library front-end. Terdapat javascript, css, dan library lainnya yang digunakan untuk membangun sistem sistem informasi musyawarah. hasil implementasi tersebut menghasilkan artefak sistem yang didokumentasikan pada struktur artefak sistem, beberapa implementasi algoritme, dan implementasi antarmuka pengguna.
3. Hasil pengujian validasi pada beberapa fungsi dinyatakan valid. Kemudian, hasil pengujian kompatibilitas menunjukan sistem berjalan pada critical issues karena masih belum menggunakan *keystore* pada *localhost*. Serta terdapat *major layout problem* pada *css* pada peramban *Internet Explore* Namun, sistem dapat berjalan dengan baik pada peramban lainnya.

5.2 Saran

Saran yang dapat diberikan sebagai bahan pertimbangan untuk melaksanakan pengembangan lebih lanjut pada Sistem Informasi Musyawarah Masjid Ibnu Sina ini sebagai berikut:

1. Penambahan fitur untuk manajemen notulensi dapat mendukung keuangan kas, pemasukan, dan pengeluaran masjid dapat diterapkan. Penambahan fitur manajemen aset juga diperlukan untuk memastikan alat yang terdapat pada masjid dapat berjalan untuk menghindari kerusakan dan kehilangan.
2. Perlu dilakukan evaluasi antarmuka pada peramban berbasis *mobile*, karena penggunaan *mobile* akan lebih cenderung digunakan dari *website.*

DAFTAR REFERENSI

Abdullah, F.H., Majid, N.H.A. and Othman, R., 2016. Defining Issue of Thermal Comfort Control through Urban Mosque Façade Design. *Procedia - Social and Behavioral Sciences*, 234, pp.416–423.

Allen, J.A., Lehmann-Willenbrock, N. and Landowski, N., 2014. Linking pre-meeting communication to meeting effectiveness. *Journal of Managerial Psychology*, 29(8), pp.1064–1081.

Booch, G., Maksimchuk, R. a., Engle, M.W., Young, B.J., Conallen, J. and Houston, K. a., 2007. *Object-Oriented Analysis and Design with Applications*. 3rd ed. *The Addison-Wesley Object Technology Series*, Boston.

Demir, A. and Bulut, I., 2018. A New Model for Respected Meetings. *Procedia Computer Science*, [online] 126, pp.1637–1655. Available at: <https://doi.org/10.1016/j.procs.2018.08.138>.

Faraday, J., 2018. *AKADEMIK SEKOLAH DASAR (STUDI PADA SDN WATES KABUPATEN KEDIRI) SKRIPSI memperoleh gelar Sarjana Komputer Disusun oleh : Jawara Wahyu Al Faraday*. Universitas Brawijaya.

IBM, 2007. *Writing Good Use Cases*.

Kute, S.S. and Thorat, S., 2014. A Review on Various Software Development Life Cycle (SDLC) Models. *International Journal of Research in Computer and Communication Technology*, 3(7), pp.776–781.

Larman, C., 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd ed. [online] Addison Wesley Professional. Available at: <http://www.utdallas.edu/~chung/SP/applying-uml-and-patterns.pdf>.

Leach, D.J., Rogelberg, S.G., Warr, P.B. and Burnfield, J.L., 2009. Perceived meeting effectiveness: The role of design characteristics. *Journal of Business and Psychology*, 24(1), pp.65–76.

Nurwansyah, E., 2018. *BERBASIS ANDROID DENGAN MENGGUNAKAN METODE (Studi Kasus : Fakultas Ilmu Komputer Universitas Brawijaya Malang)*. Universitas Brawijaya.

O'Brien, J.A. and Marakas, G.M., 2010. *Introduction to Information Systems. 15th Edition*. 15th ed. New York: Paul Ducham.

Object Management Group, 2011. Business Process Model and Notation. [online] 95(January), p.538. Available at: <http://link.springer.com/10.1007/978-3-642-25160-3>.

Pressman, R.S., 2010. *Software engineer Pressman, R. S. (n.d.). Software engineering (7nd ed.). New York: McGraw-Hill Book Company.ring*. 7nd ed. New York: Raghothaman Srinivasan Director.

Sanusi, Z.M., Johari, R.J., Said, J. and Iskandar, T., 2015. The Effects of Internal Control System, Financial Management and Accountability of NPOs: The Perspective of Mosques in Malaysia. *Procedia Economics and Finance*, [online] 28(April), pp.156–162. Available at: <http://dx.doi.org/10.1016/S2212-5671(15)01095-3>.

Shalahuddin, M. and S., R.A., 2010. *Java di WEB*. II ed. Bandung: Informatika Bandung.

Spring Boot, 2019. *Building an Application with Spring Boot*. [online] Available at: <https://spring.io/guides/gs/spring-boot/> [Accessed 21 Jun. 2019].

Wesley, P.A., 2015. Unified Modeling Language User Guide , The (2nd Edition) (Addison-Wesley Object Technology Series) Unified Modeling Language User Guide , The Unified Modeling Language User Guide , The Many of the designations used by manufacturers and sellers to dist. *ResearchGate*, 2nd(August), p.392.

repository.ub.ac.id

LAMPIRAN A HASIL PENGUJIAN VALIDASI

LAMPIRAN 1. Rencana Pengujian Skenario Mengelola Anggota

Rencana Pengujian Skenario

Pengujian Validasi Mengelola Anggota

Berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-06. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MA-01. Pengujian ini menunjukan hasil valid pada fungsi mengelola data anggota.

Kode Pengujian	V-MA-01
Kode Persyaratan	FTF-AMM-06
Penguji	Ahmad Buswori
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan menyimpan data anggota pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua takmir atau sekertaris 2. Penguji memilih fungsi menambah anggota 3. Penguji memasukan data yang akan disimpan pada sistem 4. Penguji menyimpan data
Hasil yang diharapkan	Sistem dapat menyimpan data anggota dan data anggota terdapat pada daftar anggota
Hasil pengujian	Sistem berhasil menyimpan data yang diinputkan dan menampilkan data pada sistem
Status pengujian	Valid / Tidak Valid
Masukkan dan saran	i/. Biodata ditambah dgn: nama ortu, telp rmh, hp, email, target selama jd. remas MIS hapal .... juz Bersedia & berjanji utk. taat & patuh thd aturan, & hasil Musyawarah & takmir demi lancarnya operasional MIS

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

LAMPIRAN 2. Rencana Pengujian Skenario Mengelola Notulensi

Rencana Pengujian Skenario

Pengujian Validasi Mengelola Notulensi

Berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-15. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MN-01. Pengujian ini menunjukan hasil valid pada fungsi mengelola notulensi.

Kode Pengujian	V-MN-01
Kode Persyaratan	FTF-AMM-15
Penguji	Aji Santoso - Administrasi
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat melakukan menambah data notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai notulensi 2. Penguji memilih fungsi menambah notulensi 3. Penguji memasukan data nama musyawarah, data amir musyawarah, data kehadiran anggota musyawarah yang akan disimpan pada sistem 4. Penguji menambah pekerjaan yang akan di progress 5. Penguji memasukkan data laporan, masukkan, keputusan, dan catatan tambahan musyawarah 6. Penguji menyimpan data
Hasil yang diharapkan	Sistem dapat menyimpan data yang dimasukkan
Hasil pengujian	Sistem berhasil menyimpan data yang dimasukkan dan menampilkan data pada sistem
Status pengujian	Valid
Masukkan dan saran Anggota Takmir	- 

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

LAMPIRAN 3. Rencana Pengujian Skenario Mengubah isi Notulensi

Rencana Pengujian Skenario

Pengujian Validasi Mengubah isi Notulensi

Berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-20. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MN-01. Pengujian ini menunjukan hasil valid pada fungsi mengapus notulensi.

Kode Pengujian	V-MN-01
Kode Persyaratan	FTF-AMM-20
Penguji	Aji Santoso - Administrasi
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat mengubah isi notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua atau sekertaris masjid 2. Penguji membuka detail notulensi 3. Penguji memilih fitur edit notulensi 4. Penguji melakukan perubahan data laporan, masukkan, keputusan, dan catatan tambahan musyawarah 5. Penguji menyimpan data
Hasil yang diharapkan	Sistem dapat mengubah data notulensi yang sudah tersimpan
Hasil pengujian	Sistem berhasil mengubah data notulensi yang sudah tersimpan
Status pengujian	Valid
Masukkan dan saran Anggota Takmir	

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

LAMPIRAN 4. Rencana Pengujian Skenario Mencari Notulensi

Rencana Pengujian Skenario

Pengujian Validasi Mencari Notulensi

Berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-21. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MCN-01. Pengujian ini menunjukan hasil valid pada fungsi mencari notulensi.

Kode Pengujian	V-MCN-01
Kode Persyaratan	FTF-AMM-21
Penguji	Muhammad Buswari
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat mencari notulensi musyawarah pada sistem.
Prosedur Uji	1. Penguji teridentifikasi sebagai anggota takmir dan remas 2. Penguji memilih fitur cari notulensi 3 Penguji memasukan data tanggal musyawarah, pekerjaan yang dicari, keyword yang terdapat pada musyawarah 5. Penguji mencari notulensi
Hasil yang diharapkan	Sistem dapat mencari notulensi yang memiliki atribut yang dimasukkan
Hasil pengujian	Sistem berhasil menampilkan notulensi sesuai atribut yang dimasukkan
Status pengujian	Valid / ~~Tidak Valid~~
Masukkan dan saran Anggota Takmir	untuk keslem notulensi bisa ditambahkan nama notulensi selain key word nya

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

LAMPIRAN 5. Rencana Pengujian Skenario Menverifikasi Pekerjaan

Rencana Pengujian Skenario

Pengujian Validasi Menverifikasi Pekerjaan

Berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-22. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MVP-01. Pengujian ini menunjukan hasil valid pada fungsi mencari notulensi.

Kode Pengujian	V-MVP-01
Kode Persyaratan	FTF-AMM-22
Penguji	Muhammad Buwani
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat menverifikasi status pekerjaan pada sistem
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua atau sekertaris masjid 2. Penguji membuka detail pekerjaan 3. Penguji memilih menverifikasi pekerjaan
Hasil yang diharapkan	Sistem dapat mengubah status pekerjaan
Hasil pengujian	Sistem berhasil mengubah status pekerjaan
Status pengujian	Valid / ~~Tidak Valid~~
Masukkan dan saran Anggota Takmir	Baik.

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

LAMPIRAN 6. Rencana Pengujian Skenario Menverifikasi Notulensi

Rencana Pengujian Skenario

Pengujian Validasi Menverifikasi Notulensi

Berisi rencana pengujian dengan kasus uji berdasarkan kode persyaratan FTF-AMM-24. Kasus uji berikut merupakan kasus uji dengan kode pengujian V-MSP-01. Pengujian ini menunjukan hasil valid pada fungsi mencari notulensi.

Kode Pengujian	V-MSP-01
Kode Persyaratan	FTF-AMM-24
Penguji	Muhammad Busuan
Tujuan Pengujian	Pengujian ini dilakukan untuk memastikan sistem dapat berjalan dengan baik, pada saat menverifikasi notulensi pada sistem
Prosedur Uji	1. Penguji teridentifikasi sebagai ketua atau sekertaris masjid 2. Penguji membuka detail notulensi 3. Penguji memilih menverifikasi notulensi
Hasil yang diharapkan	Sistem dapat menverifikasi notulensi
Hasil pengujian	Sistem berhasil menverfikasi notulensi
Status pengujian	Valid / ~~Tidak Valid~~
Masukkan dan saran Anggota Takmir	Sudah cukup baik dan sesuai dg kebutuhan

UNIVERSITAS BRAWIJAYA