IMPLEMENTASI SISTEM INFORMASI IZIN LOKASI (SILOKA) PADA DINAS CIPTA KARYA DAN TATA RUANG KABUPATEN MALANG DENGAN PEMROGRAMAN BERORIENTASI OBJEK

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Syeh Mukhamad Iqbal Abu Dafi
NIM: 125150407111006

PROGRAM STUDI SISTEM INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

PENGESAHAN

IMPLEMENTASI SISTEM INFORMASI IZIN LOKASI (SILOKA) PADA DINAS CIPTA KARYA DAN TATA RUANG KABUPATEN MALANG DENGAN PEMROGRAMAN BERORIENTASI OBJEK

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Syeh Mukhamad Iqbal Abu Dafi
125150407111006

Skripsi ini telah diuji dan dinyatakan lulus pada
22 Desember 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Ismiarta Aknuranda, S.T , M.Sc, Ph.D	Satrio Agung W., S.Kom, M.Kom
NIK: 2010006 740719 1 001	NIP: 19860521 201212 1 001

Mengetahui
Ketua Jurusan Sistem Informasi

Herman Tolle, Dr. Eng., S.T, M.T
NIP: 19710727 199603 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 22 Desember 2016

Syeh Mukhamad Iqbal Abu Dafi

NIM: 125150407111006

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan nikmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini yang berjudul "Implementasi Sistem Informasi Izin Lokasi (SILOKA) pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang Kabupaten Malang dengan Pemrograman Berorientasi Objek". Penulis ingin mengucapkan banyak terimakasih kepada pihak-pihak yang telah mendukung dalam menyelesaikan skripsi ini, diantaranya:

1. Bapak Ismiarta Aknuranda, S.T, M.Sc, Ph.D, yang sudah membimbing, memberikan waktu, saran, dan kesabarannya dalam pengerjaan hingga penyelesaian skripsi.
2. Bapak Satrio Agung Wicaksono, S.Kom, M.Kom yang sudah membimbing, memberikan waktu, saran, dan kesabarannya dalam pengerjaan penyelesaian skripsi.
3. Bapak Herman Tolle, Dr.Eng, S.T, M.T selaku ketua jurusan Sistem Informasi Universitas Brawijaya Malang.
4. Bapak Suprapto, S.T, M.T selaku ketua prodi Sistem Informasi Universitas Brawijaya Malang.
5. Bapak Yussi Tyroni, S.Kom, M.Kom selaku dosen pembimbing akademik
6. Dosen dan karyawan Fakultas Ilmu Komputer Universitas Brawijaya Malang
7. Bapak Baro'il, Ibu Kholifah, Kakak Uzma Daniya Anis Suhaili, Adik Farid Bani Adam Al-Farokhie Al-Muhaimin, dan Adik Salsabila Sayid Jazulli Abdullah yang selalu memberikan dukungan dan doa yang tiada habisnya setiap saat
8. Bella Pertiwi yang selalu menemani dan sangat banyak membantu.
9. Dimas Habib Rahmatullah, Dionysius Briananda Y atas bantuan dan kerjasamanya.
10. Keluarga besar mahasiswa Sistem Informasi angkatan 2012 Fakultas Ilmu Komputer Universitas Brawijaya Malang
11. Semua pihak yang tidak bisa penulis sebutkan satu persatu.

Penulis menyadari menyadari skripsi ini masih jauh dari sempurna dan masih banyak kekurangan, untuk itu penulis mengharapkan kritik dan saran yang membangun dari Bapak/Ibu/Saudara/i pembaca. Penulis juga berharap semoga skripsi ini bermanfaat khususnya bagi mahasiswa Sistem Informasi Fakultas Ilmu Komputer Universitas Brawijaya Malang. Demikian yang dapat penulis sampaikan, dan penulis mengucapkan terimakasih.

Malang, 22 Desember 2016

Syeh Mukhamad Iqbal Abu Dafi
mukhamad.iqbal.davi@gmail.com

ABSTRAK

Dinas Cipta Karya dan Tata Ruang merupakan salah satu instansi pemerintah di Kabupaten Malang yang mempunyai fungsi salah satunya adalah penyusunan kebijakan dan standarisasi teknis bangunan gedung termasuk pengelolaan gedung dan rumah asset daerah, sehingga terdapat proses izin lokasi ketika masyarakat akan melakukan pembangunan bangunan. Pada Dinas Cipta Karya dan Tata Ruang permohonan izin lokasi masih dilakukan secara manual. Warga harus datang terlebih dahulu ke kantor Dinas Cipta Karya dan Tata Ruang untuk mendapatkan formulir, kemudian menyerahkan formulir dan berkas, dilanjutkan dengan pemeriksaan berkas. Pemeriksaan dilakukan terutama pada lokasi perizinan, apakah wilayah atau lokasi termasuk diizinkan tanpa syarat, diizinkan dengan syarat, dapat dibangun secara terbatas atau tidak diperbolehkan. Pemeriksaan lokasi perizinan berdasarkan peraturan zonasi. Peraturan zonasi, meliputi ketentuan kegiatan penggunaan lahan, ketentuan tata bangunan, ketentuan prasarana dan sarana minimum, ketentuan pelaksanaan, ketentuan perubahan peraturan zonasi, serta ketentuan khusus. Ketentuan-ketentuan peraturan zonasi tersebut masih terdapat pada dokumen buku cetak, dan membutuhkan waktu yang lama serta teliti untuk mencari ketentuan zonasi, sehingga mempunyai resiko kesalahan lebih besar. Berdasarkan permasalahan tersebut, maka dibutuhkan suatu sistem informasi untuk proses izin lokasi. Untuk mengimplementasikan sebuah sistem informasi, dibutuhkan sebuah analisis kebutuhan yang menentukan kebutuhan apa yang dibutuhkan oleh pihak terkait. Selanjutnya, dilakukan perancangan untuk merancang bagaimana sistem akan dibangun. Pada analisis dan perancangan sudah dilakukan pada penelitian sebelumnya. Pada penelitian analisis dan perancangan belum disertai dengan adanya implementasi sistem. Tujuan penelitian ini mencakup perancangan untuk kebutuhan implementasi, implementasi SILOKA, dan pengujian SILOKA dengan menggunakan jenis pengujian *BlackBox*. Perancangan dilakukan untuk menyesuaikan dengan *framework* yang dipakai. Implementasi menggunakan pemrograman berorientasi objek, menggunakan konsep MVC dan menggunakan *framework* CodeIgniter. Hasil dari penelitian ini merupakan produk jadi SILOKA. Pengujian yang dilakukan pada penelitian ini menggunakan pengujian *blackbox* pada semua *use case* dan pengujian *whitebox* pada tiga *use case*. Hasil pengujian *blackbox* pada SILOKA adalah semua berjalan sesuai dengan spesifikasi *use case* yang telah didefinisikan pada penelitian sebelumnya. Sedangkan hasil pengujian *whitebox* merupakan perhitungan kompleksitas dan jalur independen, dimana setiap jalur independen telah dilalui minimal satu kali.

Kata kunci: Kontruksi, Implementasi, Sistem Informasi, Berorientasi Objek, CodeIgniter, pengujian *BlackBox,* pengujian *WhiteBox*

repository.ub.ac.id

ABSTRACT

Dinas Cipta Karya dan Tata Ruang is one of the government agencies in Malang which has the function of one of them is the development of policies and technical standardization buildings including asset building management area, so there is a location permit process when the public will do the construction of the building. In Dinas Cipta Karya dan Tata Ruang location permit is still done manually. Residents must come first to the office to get a registration form, then submit the form and file, and then the file will be checked. Examination of the file permormed mainly on the location permit, whether the region or location including unconditionally approved, permitted by the terms, can be build in a limited or not allowed. Examining location performed by the zoning regulation permit. Zoning regulations include land use activities, the procedures of buildings, provision of infrastructure and facilities for minimum, implementing provisions, the provision of the zoning regulation changes, as well as special provisions. The provisions of the zoning regulations are still contained in coduments printed books, and takes a long time to looking for zoning provisions, so have a greater risk of error. Based on the problems, Dinas Cipta Karya dan Tata Ruang need a information system to process the location permit. To implement an information system, it takes a needs analysis to determine the requirement. Furthermore, design phase how system will be build. In the analysis and design has been done in previous studies. In previous studies have not been accompanied by the implementation of the system. The purpose of this research include planning for the needs of implementation, implementation of SILOKA and testing SILOKA using blackbox testing types. The design needs to be done to adjust the framework. Implementation using object-oriented programming, using concept of MVC and using framework CodeIgniter. The result of this research are SILOKA product. Test performed in this research using blackbox testing on all use case and whitebox testing on three use case. The result of blackbox testing in SILOKA is system runs according use case spesification that defined in previous research. Whereas the result of whitebox testing is cyclomatic complexity and independent path, which each independent path has been passed minimal once.

Keywords: *Construction, Implementation, Information System, Object Oriented, CodeIgniter, Blackbox Testing, Whitebox Testing*

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Bab ini membahas tentang latar belakang yang melatar belakangi penulis mengangkat penelitian ini, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan.

1.1 Latar belakang

Dewasa ini, teknologi berkembang sangat pesat. Teknologi *internet* sudah merajalela terutama di Indonesia. Menurut Kemkominfo, jumlah pengguna *internet* di Indonesia hingga saat ini mencapai 82 juta orang (Ajo, 2016). Dengan berkembangnya *internet*, orang-orang dimudahkan dalam memperoleh informasi dengan cepat, sebagaimana *internet* merupakan sebuah media untuk mengakses informasi tersebut. Dengan adanya *internet* juga orang-orang dapat mencari sebuah informasi dengan cepat darimanapun dan kapanpun asalkan orang-orang tersebut mempunyai sebuah perangkat yang tersambung dengan *internet*. Perangkat yang dimaksud dapat berupa sebuah komputer personal, *smartphone* atau sebuah telepon seluler yang mempunyai fitur *internet*. Kecepatan dalam memperoleh informasi merupakan kebutuhan semua individu. Tidak hanya individu saja, sebuah organisasi juga membutuhkan sebuah aliran kecepatan informasi yang cepat.

Dinas Cipta Karya dan Tata Ruang merupakan salah satu Dinas Kabupaten Malang yang berlokasi di Kepanjen. Dinas Cipta Karya dan Tata Ruang mempunyai tugas pokok, salah satunya adalah melaksanakan urusan pemerintah daerah bidang Cipta Karya dan Tata Ruang berdasarkan asas otonomi. Untuk menyelenggarakan tugas tersebut, Dinas Cipta Karya dan Tata Ruang mempunyai fungsi salah satunya adalah penyusunan kebijakan dan standarisasi teknis bangunan gedung, termasuk pengelolaan gedung dan rumah asset daerah (DCKTR, 2013). Untuk itu terdapat proses permohonan izin lokasi ketika warga hendak melakukan pembangunan gedung maupun bangunan apakah layak dibangun di lokasi yang dimohon atau tidak. Untuk mengajukan permohonan izin lokasi, masih dilakukan dengan cara manual. Manual yang dimaksud, pemohon harus datang terlebih dahulu ke Kantor Dinas Cipta Karya dan Tata Ruang membawa berkas, lalu meminta formulir, mengisi formulir kemudian menyerahkan formulir dan berkas kepada Dinas. Kemudian pada berkas yang sudah dikumpulkan, dilakukan pemeriksaan berkas. Pemeriksaan dilakukan terutama pada lokasi perizinan, apakah wilayah termasuk diizinkan tanpa syarat, diizinkan dengan syarat, dapat dibangun secara terbatas atau tidak diperbolehkan. Untuk melakukan pemeriksaan tersebut menggunakan Buku Pedoman Peraturan Zonasi.

Peraturan zonasi, meliputi ketentuan kegiatan dan penggunaan lahan, ketentuan tata bangunan, ketentuan prasarana dan sarana minimum, ketentuan pelaksanaan, ketentuan perubahan peraturan zonasi, serta ketentuan khusus. Klasifikasi zona di kawasan Perkotaan Kepanjen didasarkan pada peraturan pemerintah PU nomor 20 tahun 2011 tentang Penyusunan RDTRK (Rencana Detail

Tata Ruang Kota) dan Peraturan Zonasi yang disesuaikan dengan kegiatan yang telah berkembang di wilayah perencanaan. Namun, Peraturan Zonasi pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang masih dalam bentuk dokumen buku cetak. Untuk mencari ketentuan-ketentuan peraturan zonasi harus membuka buku dokumen buku cetak peraturan zonasi dan membutuhkan waktu yang tidak sedikit serta ketelitian yang tinggi dimana mempunyai resiko kesalahan yang lebih besar. Dengan adanya hal permohonan izin lokasi dan proses pencarian data peraturan zonasi yang masih manual serta dengan seiring perkembangan terknologi, dibutuhkanlah sebuah Sistem Informasi Izin Lokasi (SILOKA) yang juga mencakup Peraturan Zonasi.

Untuk mengimplementasikan sebuah sistem informasi, dibutuhkan sebuah analisis kebutuhan. Analisis kebutuhan akan menentukan kebutuhan apa yang dibutuhkan oleh pihak terkait yang akan menggunakan sistem. Selanjutnya dilakukan sebuah perancangan untuk merancang bagaimana sistem akan dibangun. Pada penelitian sebelumnya sudah dilakukan sebuah analisis persyaratan dan perancangan Sistem Informasi Izin Lokasi (SILOKA) yang dilakukan oleh Bella Pertiwi selaku mahasiswi Sistem Informasi Universitas Brawijaya dengan judul penelitian Analisis dan Perancangan Sistem Informasi Izin Lokasi (SILOKA) pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang dengan Pendekatan Berorientasi Objek (Pertiwi, 2016). Analisis yang dilakukan, meliputi analisis proses bisnis yang dinotasikan dalam bentuk BPMN, analisis persyaratan, perancangan sistem dalam bentuk UML, dan evaluasi.

Berdasarkan analisis dan perancangan yang sudah dilakukan serta tanpa adanya implementasi, maka penulis akan melakukan implementasi SILOKA berdasarkan analisis dan perancangan yang sudah dilakukan. Implementasi menggunakan pemrograman berorientasi objek, karena pada analisis dan perancangan sebelumnya menggunakan pendekatan berorientasi objek. Pada pemrograman beorientasi objek di penelitian ini menggunakan konsep MVC (*Model, View, Controller*).

1.2 Rumusan masalah

Pada rumusan masalah terdapat dua rumusan masalah, yaitu rumusan masalah umum dan rumusan masalah khusus. Rumusan masalah umumnya adalah dapatkah SILOKA diimplementasikan berdasarkan perancangan yang sudah ada.

Sedangkan rumusan masalah khusus:

1. Bagaimanakah hasil perancangan detail dan implementasi SILOKA berdasarkan perancangan sebelumnya?
2. Bagaimanakah hasil pengujian SILOKA dengan metode *Blackbox* dan *Whitebox*?

1.3 Tujuan

Pada tujuan penelitian ini terdapat tujuan umum dan tujuan khusus. Tujuan umum adalah untuk mengimplementasikan SILOKA berdasarkan perancangan sebelumnya. Sedangkan tujuan khususnya adalah sebagai berikut:

1. Menghasilkan rancangan detail dan implementasi rancangan dari SILOKA pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang pada penelitian sebelumnya
2. Melakukan pengujian terhadap SILOKA yang sudah diimplementasikan dengan menggunakan pengujian *Blackbox* dan pengujian *Whitebox*.

1.4 Manfaat

Diharapkan dengan adanya implementasi SILOKA pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang sesuai dengan analisis dan perancangan yang sudah ada dapat memberikan manfaat sebagai berikut:

1. Dapat mengotomasisasi proses pencarian peraturan zonasi pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang
2. Dapat mempercepat pencarian data ketentuan Izin Lokasi pada subzona tertentu
3. Meningkatkan pelayanan kepada masyarakat dengan menyediakan informasi ketentuan Izin Lokasi yang dapat diakses oleh masyarakat dimanapun dan kapanpun karena Sistem Informasi Izin Lokasi yang berbasis web.
4. Mempermudah pegawai instansi yang terlibat dalam layanan pengajuan izin lokasi, karena dengan adanya Sistem Informasi Izin Lokasi proses pengajuan izin lokasi telah terkomputerisasi.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini mengikuti dari penelitian perancangan sebelumnya yang telah dilakukan oleh Bella Pertiwi. Namun terdapat tambahan batasan masalah pada penelitian ini adalah implementasi dilakukan dengan menggunakan *framework* CodeIgniter, dimana codeIgniter menganut sistem MVC (Model, View, Controller).

1.6 Sistematika Pembahasan

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang yang melatar belakangi penulis mengangkat penelitian ini, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini terdapat kajian pustaka yang membahas tentang penelitian sebelumnya, yaitu penelitian tentang perancangan dan penelitian yang berkaitan

dengan implementasi konsep MVC serta dasar teori yang memuat tentang teori-teori untuk mendukukng penelitian.

BAB III METODOLOGI

Bab ini membahas mengenai metodologi atau pendekatan yang dipakai untuk mengimplementasikan Sistem Informasi Izin Lokasi pada Dinas Cipta Karya dan Tata Ruang, mulai dari pendekatan yang digunakan sampai dengan diagram alur dari awal pengerjaan sampai akhir pengerjaan.

BAB IV TINJAUAN DAN ANALISIS LANJUT

Bab ini membahas tentang tinjauan dari penelitian sebelumnya yang sudah dilakukan dan hasil analisis lanjut mengenai perancangan SILOKA.

BAB V IMPLEMENTASI

Bab ini membahas mengenai implementasi dari SILOKA yang mencakup lingkungan implementasi dari perangkat lunak dan perangkat keras, kode program SILOKA dan hasil implementasi berupa *screenshot* program SILOKA ketika dijalankan.

BAB V PENGUJIAN

Bab ini membahas mengenai pengujian yang dilakukan pada SILOKA dengan menggunakan jenis pengujian *Blackbox* dan *Whitebox*, serta pengambilan hasil pengujian *Backbox* dan *Whitebox* dari SILOKA.

BAB VI PENUTUP

Bab ini membahas mngenai pengambilan kesimpulan yang diberikan serta saran dari SILOKA.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini terdapat kajian pustaka yang membahas tentang penelitian sebelumnya, yaitu penelitian tentang analisis dan perancangan yang merupakan penelitian sebelumnya serta penelitian yang berkaitan dengan implementasi konsep MVC serta dasar teori yang memuat tentang teori-teori untuk mendukukng penelitian.

2.1 Kajian Pustaka

2.1.1 Penelitian Sebelumnya

Pada perancangan yang sudah ada ini, dilakukan oleh Bella Pertiwi selaku mahasiswa Sistem Informasi, Fakultas Ilmu Komputer dengan judul Analisis dan Perancangan Sistem Informasi Izin Lokasi (SILOKA) pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang dengan Pendekaatan Berorientasi Objek (Pertiwi, 2016). Penelitian yang sudah dilakukan meliputi analisis proses bisnis, analisis persyaratan, perancangan sistem dan evaluasi.

Analisis proses bisnis meliputi alur proses bisnis pengajuan izin lokasi yang masih dilakukan secara manual atau belum menggunakan sistem dan alur proses bisnis pengajuan izin lokasi yang akan diusulkan dengan menggunakan sistem. Analisis proses bisnis yang dilakukan oleh peneliti tersebut dimodelkan dengan menggunakan *Business Process Modelling Notation* atau disebut dengan BPMN.

Analisis persyaratan yang dilakukan peneliti sebelumnya menggunakan pendekatan *Object Oriented Analysis Design* (OOAD) dimana pada analisis persyaratan ini dibagi menjadi tiga komponen, yaitu :

1. Identifikasi pernyataan masalah dan *stakeholder*, dimana peneliti mengidentifikasi permasalahan yang harus diselesaikan dengan adanya sistem baru.
2. Analisis fitur dan persyaratan deskriptif, dimana peneliti menganalisis fitur untuk mengetahui kesimpulan kemampuan dari Sistem Informasi Izin Lokasi yang akan dibangun secara umum.
3. Analisis persyaratan naratif untuk mengetahui nilai yang diberikan oleh sistem terhadap pengguna sistem.

Perancangan sistem yang dilakukan peneliti sebelumnya menggunakan pendekatan berorientasi objek. Untuk memodelkan diagram perancangannya dimodelkan dalam UML dimana terdapat empat komponen sebagai berikut :

1. Rancangan Arsitektur, dimana peneliti sebelumnya melakukan perancangan arsitektur sistem berdasarkan hasil analisis yang telah didapatkan ketika melakukan kebutuhan sistem.
2. Rancangan kelas untuk memberikan gambaran bagaimana sistem dan relasi yang terdapat di dalamnya. Perancangan kelas dimodelkan dengan

diagram kelas dan terdapat penjelasan mengenai tipe dan deskripsi kelas pada diagram kelas.

3. Pemodelan Data pada penelitian sebelumnya menggunakan *physical data model*.
4. Rancangan Antarmuka Pengguna pada penelitian sebelumnya menggunakan *sketch User Interface*.

Evaluasi yang dilakukan peneliti sebelumnya bertujuan untuk mengetahui kualitas dari perancangan yang sudah dilakukan oleh peneliti sebelumnya. Pengujian yang dilakukan berupa tinjauan dari sisi pengguna terkait dengan proses bisnis apakah sesuai atau tidak, persyaratan dan hasil rancangan antarmuka pengguna sistem. Peneliti sebelumnya juga menggunakan matriks kerunutan perancangan sistem dengan spesifikasi persyaratan yang telah dianalisis.

2.1.2 Penelitian Tentang Implementasi MVC

Terdapat penelitian tentang implementasi arsitektur MVC yang dilakukan oleh Ni L.P Pravina Utpatadevi, A. A. K. Oka Sudana dan A. A. Kt. Agung Cahyawan tentang Implementasi Arsitektur MVC pada Sistem Informasi Manajemen Akademik dengan *Platform* Android (Utpatadevi dkk, 2012). Arsitektur MVC pada penelitian tersebut diimplementasikan pada Sistem Informasi Manajemen Akademik berbasis Android. Sistem Informasi Manajemen Akademik adalah sumber yang terkait dengan masalah akademik dan sebagian besar digunakan untuk sumber informasi dalam suatu perguruan tinggi. Sistem Informasi Manajemen Akademik yang menggunakan teknologi menghasilkan Sistem Informasi Manajemen Akademik yang dapat diakses di dalam atau di luar kampus, dan bahkan dunia melalui media komputer atau *gadget* yang terhubung dengan *internet*. Aplikasi Sistem Informasi Manajemen Akademik berbasis *mobile application* untuk menciptakan lingkungan yang efisien antara mahasiswa dan dosen. Arsitektur MVC memisahkan mayor komputer dalam pengembangan aplikasi seperti manipulasi data, tampilan antarmuka pengguna dan *controller* untuk memastikan aplikasi dapat diperbaiki dengan mudah.

Pada pengembangan sistem informasi manajemen akademik tersebut diimplementasikan pada *platform* Android, setiap layer arsitektur MVC akan diimplementasikan secara berbeda, seperti database MySQL berfungsi sebagai *model*, berdasarkan *platform* Android akan terdapat *view* dan JSON *web service* sebagai *controller.*

2.2 Dasar Teori

Kajian Pustaka merupakan bahan-bahan atau teori-teori yang secara khusus berkaitan dengan penelitian yang sedang dikaji.

2.2.1 Sistem

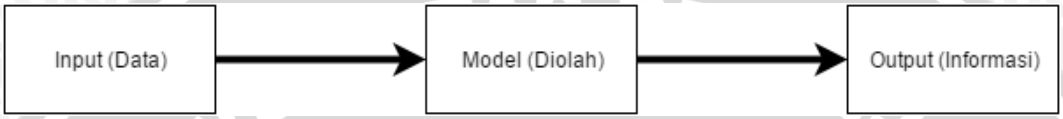
Menurut Jogiyanto (2005) mengemukakan bahwa sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu, sistem ini menggambarkan suatu kejadian-kejadian dan kesatuan yang nyata

adalah suatu objek nyata, seperti, tempat, benda dan orang-orang yang betul-betul ada dan terjadi.

Sistem juga merupakan kumpulan dari komponen yang saling berhubungan satu dengan yang lainnya membentuk satu kesatuan untuk mencapai tujuan tertentu. Sebagai contoh sistem komputer yang didefinisikan sebagai kumpulan dari perangkat keras dan perangkat lunak.

2.2.2 Sistem informasi

Informasi adalah data yang dioleh menjadi bentuk yang berguna bagi para pemakainya. Tujuan dari sistem informasi adalah menghasilkan sebuah informasi (Jogiyanto, 2009). Komponen sistem informasi dimulai dari siklus pengolahan data.



Gambar 2.1 Siklus Pengolahan Data

Sumber: Jogiyanto (2009)

Dari gambar 2.1 terlihat bahwa untuk melakukan siklus pengolahan data, dibutuhkan tiga komponen, yaitu *input* atau masukan, *model* dan *output* (keluaran) dimana data terletak pada masukan kemudian dioleh pada *model* sehingga menjadi sebuah informasi pada keluaran.

2.2.3 Pemrograman berorientasi objek

Pemrograman berorientasi objek merupakan suatu metode yang memungkinkan pengembang perangkat lunak untuk membangun sebuah sistem perangkat lunak yang dapat diandalkan, mudah digunakan, mudah dipelihara, terdokumentasi dengan baik dan *reusable* yang dapat memenuhi kebutuhan pengguna dengan sebuah komunitas objek yang bekerjasama dengan satu sama lain (Pillay, 2007). Teknik pemecahan masalah yang digunakan dalam pemrograman berorientasi objek lebih mirip dengan cara manusia dalam memecahkan masalah sehari-hari. Dalam pemrograman berorientasi objek, kita membuat sebuah objek perangkat lunak seperti objek dunia nyata yang juga memiliki atribut dan perilaku.

Sistem yang berorientasi objek lebih mudah untuk melakukan perubahan dari sistem yang dikembangkan menggunakan pendekatan fungsional. Objek termasuk atribut dan operasi untuk mengolah suatu data. Pada pemrograman berorientasi objek ketika mengubah suatu objek tidak mempengaruhi objek sistem yang lain.

Berikut merupakan konsep dasar dari pemrograman berorientasi objek:

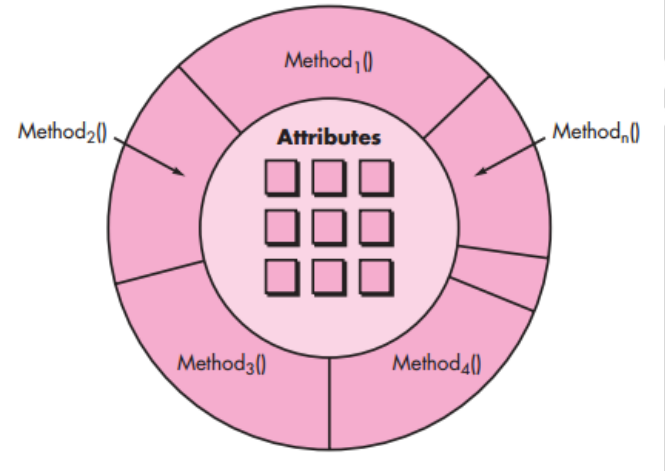
1. Objek dan kelas

Objek perangkat lunak dimodelkan setelah objek dunia nyata bahwa mereka juga memiliki atribut dan perilaku. Sebuah objek perangkat lunak mengelola atribut dalam satu atau lebih variabel. Variabel adalah item data yang

disebut *identifier*. Sebuah objek perangkat lunak mengimplementasikan perilaku dengan *method*. Sebuah *method* adalah fungsi yang berhubungan dengan suatu objek. Objek juga dikenal sebagai *instance* (Pillay, 2007).

Kelas merupakan *blueprint* dari objek, sebuah kelas digunakan untuk membuat sebuah objek. Kelas menyatakan variabel *instance* yang berisi objek. Kelas juga menyatakan dan mengimplementasikan *method* yang diperlukan untuk operasi objek. Jadi definisi dari kelas adalah *blueprint* yang mendefinisikan variabel dan *method* untuk semua objek dari jenis tertentu (Pillay, 2007).

Sebuah kelas merupakan konsep *object oriented* yang mengenkapsulasi data dan abstraksi prosedural yang diperlukan untuk menggambarkan atribut dan perilaku beberapa entitas dunia nyata. Ilustrasi kelas dapat digambarkan pada gambar



Gambar 2.2 Gambaran kelas

Sumber: Pressman (2010)

2. *Method*

Method pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. *Method* merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek. Dalam sebuah kelas, jumlah *method* dapat lebih dari satu (A.S dan Shalahudin, 2013).

3. Atribut

Atribut merupakan variabel yang dimiliki oleh sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas. Atribut dimiliki secara individual oleh sebuah objek, seperti nama, berat, jenis dan sebagainya (A. S dan Shalahudin, 2013).

4. Enkapsulasi

Enkapsulasi merupakan salah satu konsep utama pada perancangan berorientasi objek. Informasi dan *behaviour* yang saling berkaitan harus terdapat pada kelas yang sama, sehingga sistem dapat mencapai prinsip berorientasi objek.

Pada konsep enkapsulasi, data dan proses yang dapat memanipulasi data berada pada satu *package* sebagai unit yang kohesif (Pressman, 2010).

Enkapsulasi didefinisikan sebagai lokasi fisik dari fitur ke dalam abstraksi *blackbox*, sehingga dapat menyembunyikan implementasi dan berbagai keputusan perancangan terkait di belakang publik *interface* (*Dictionary Object Technology*, Firesmith, Eykholt, 1995). Enkapsulasi memungkinkan pengguna untuk menggunakan sistem tanpa mengetahui bagaimana sistem bekerja dengan mengakses *interface* sistem. Enkapsulasi memudahkan pengembang untuk melakukan perawatan sistem dengan biaya yang lebih murah karena pengguna sistem tidak terpengaruh secara langsung dengan adanya perubahan pada implementasi dengan adanya penyembunyian informasi. Pada konsep enkapsulasi, informasi dari sistem berada pada *packaged* yang sama dan dapat digunakan sebagai satu spesifikasi atau komponen dari program (Pressman, 2007).

5. Pewarisan (*Inheritance*)

Pewarisan merupakan salah satu karakteristik pembeda antara sistem konvensional dan sistem yang berorientasi pada objek. *Subclass* dapat mewarisi seluruh atribut dan operasi yang berkaitan dari *superclass*. Seluruh struktur data dan algoritma yang dirancang dan diimplementasikan pada *superclass* juga tersedia dan dapat digunakan oleh *subclass*. Pewarisan mendukung konsep penggunaan data kembali (*reuse*).

Pada masing-masing level dalam hierarki kelas, penambahan atribut dan operasi baru akan diwariskan dari kelas dengan level lebih tinggi pada hierarki. Terdapat beberapa opsi ketika pengembang menambahkan kelas baru, yaitu:

a. Kelas dapat dirancang dan dibangun melaluiu *scratch*, sehingga tidak perlu menggunakan konsep pewarisan
b. Hierarki kelas dapat digunakan jika kelas yang lebih tinggi pada hierarki memiliki paling banyak atribut dan operasi yang dibutuhkan sebagai persyaratan.
c. Susunan hierarki kelas dapat disusun kembali apabila dibutuhkan atribut dan operasi lain yang diwariskan oleh kelas baru.
d. Karakteristik dari kelas yang sudah ada dapat di*override*, dan kelas baru dibangun menggunakan atribut dan operasi dengan versi yang berbeda dari kelas yang sudah ada.

2.2.4 Use Case

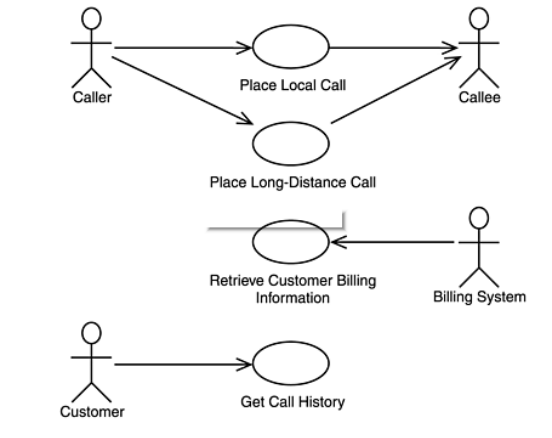
Ide dasar dibalik pemodelan *use case* adalah untuk mendapatkan inti dari apa yang harus dilakukan dan berfokus pada siapa yang akan menggunakan sistem.

Model *use case* terdiri dari komponen berikut ini (Bittner, 2004):

1. Aktor mewakili orang atau hal-hal yang berinteraksi dengan sistem. Aktor memiliki dan dan deskripsi singkat, dan mereka yang dihubungkan dengan *use case*

2. *Use case* merepresentasikan nilai yang dapat ditampilkan sistem pada aktor. *Use case* bukan merupakan fungsi atau fitur. *Use case* memiliki nama, deskripsi singkat dan deskripsi rinci yang menjelaskan bagaimana aktor menggunakan sistem untuk melakukan sesuatu dan apa yang dilakukan sistem untuk memenuhi kebutuhan. Masing-masing *use case* menggambarkan mengenai tujuan yang dingin dicapai oleh aktor pada sistem.

Aktor dan *use case* dapat digambarkan dengan diagram *use case*. Aktor digambarkan oleh orang, *use case* digambarkan dengan elips, sedangkan panah mewakili hubungan aktor dan *use case* (Bittner, 2002). Diagram *use case* berfungsi sebagai gambaran atau ringkasan dari perilaku sistem.



Gambar 2.3 Contoh diagram *use case* telepon

Sumber: Bittner (2002)

Hubungan antar *use case* (IBM, 2004a):

1. *Include*

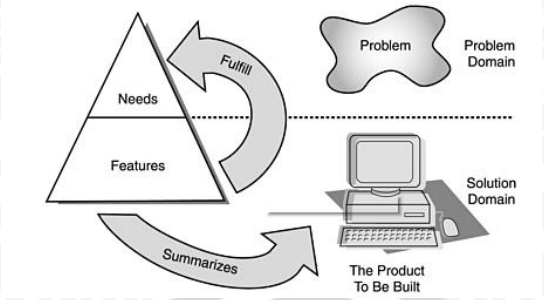
   Include adalah hubungan antar *use case* yang mengindikasikan bahwa *use case* membutuhkan *behavior* dari *use case* lainnya (*use case* yang di*include*). *Use case* yang memiliki hubungan *include*, harus menggunakan *included use case*

2. *Extend*

   Extend adalah hubungan antar *use case* yang mengindikasikan bahwa satu *use case* dapat di*extend* dengan *use case* yang lainnya. Hubungan *extend* adalah opsional untuk menggunakan *extended use case*

2.2.5 Fitur

Fitur merupakan kualitas dari sistem yang diperlukan untuk memberikan sebuah manfaat pada pengguna dan yang membantu untuk memenuhi kebutuhan dari pemangku kepentingan dan pengguna (Bittner, 2002). Gambar 2.2 merupakan gambaran hubungan antara kebutuhan, fitur, dan sistem yang akan dibangun.



Gambar 2.4 Gambar hubungan antar kebutuhan, fitur dan sistem

Sumber: Bittner (2002)

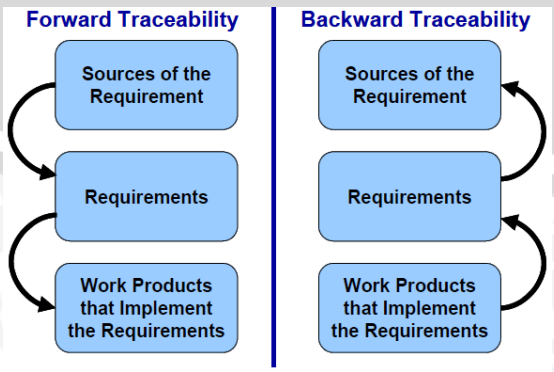
Fitur dibagi menjadi dua, yaitu fungsional dan non fungsional. Contoh fitur dari sistem yang dapat melacak permasalahan, yaitu kemampuan untuk menyediakan laporan. Fitur lain yang terdapat pada sistem berupa kualitas data yang digunakan untuk membuat laporan pada sistem tersebut. Fitur digunakan untuk menyimpulkan kemampuan dan kualitas dari produk yang akan dibuat, sehingga harus dapat diakses oleh seluruh anggota pada tim proyek dan seluruh pemangku kepentingan.

Setiap fitur akan diwujudkan secara rinci dalam model *use case* dan *supplementary specification*. Kombinasi dari fitur dan *use case* menyediakan mekanisme yang sangat kuat untuk mengelola ruang lingkup dari sistem, dan memastikan bahwa persyaratan spesifikasi sudah memenuhi seluruh kebutuhan pemangku kepentingan dan pengguna.

2.2.6 Matriks Kerunutan

Matriks kerunutan atau *traceability matrix* merupakan salah satu hal penting dalam manajemen persyaratan yang digunakan untuk memastikan sebuah produk dibangun dengan tepat di setiap fase dari *life cycle* pengembangan perangkat lunak untuk menelusuri progres pengembangan (Westfall, 2006).

Matriks kerunutan yang baik memungkinkan untuk melacak dari dua arah, yang berarti bahwa traceability dapat ditelusuri baik dari depan (*forward traceability*) maupun belakang (*Backward Traceability*), seperti pada gambar 2.3 menunjukkan *traceability* dua arah



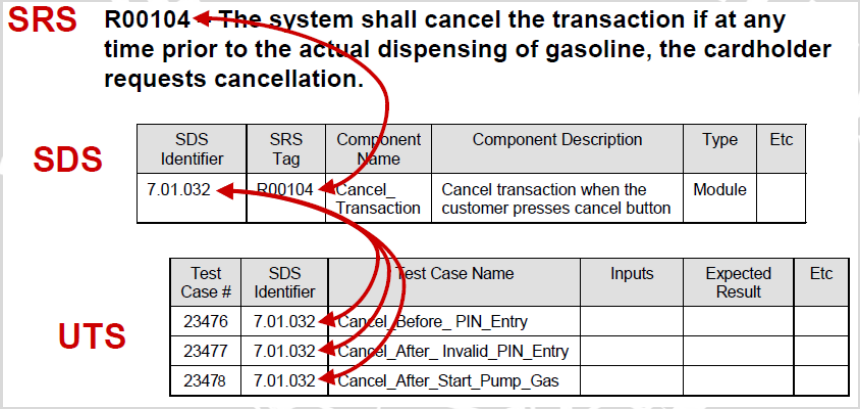
Gambar 2.5 Traceability Matriks Dua Arah

Sumber: Westfall (2006)

Forward Traceability berfungsi sebagai berikut:

1. Menelusuri sumber persyaratan ke persyaratan produk sehingga digunakan untuk memastikan kelengkapan spesifikasi persyaratan produk
2. Menelusuri persyaratan masing-masing produk yang unik ke dalam desain yang mengimplementasikan persyaratan tersebut, kode yang mengimplementasikan desain tersebut, dan tes yang memvalidasi persyaratan tersebut. Tujuannya adalah untuk memastikan bahwa setiap persyaratan diimplementasikan dalam bentuk produk dan setiap kebutuhan benar-benar teruji.
3. Menelusuri setiap produk yang dikerjakan kembali ke persyaratan yang terkait.

Untuk contoh bagaimana matriks kerunutan dapat dilihat pada *Software Requirement Spesification* pada gambar 2.4



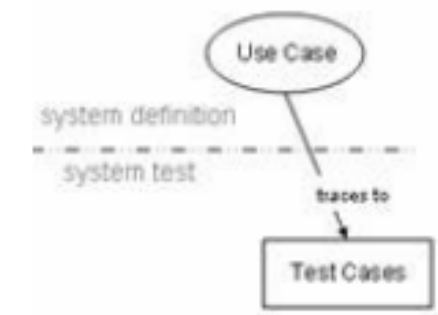
Gambar 2.6 Contoh Matriks Kerunutan

Sumber: Westfall (2006)

SDS (*Software Desain Spesification*) termasuk yang tag yang mengidentifikasi persyaratan yang dilakukan oleh setiap elemen desain yang diidentifikasi dan UTS (*Unit Test Spesification*) termasuk menelusuri tag elemen desain setiap *test case* yang diverifikasi.

2.2.6.1 Matriks Kerunutan Use Case dan Test Case

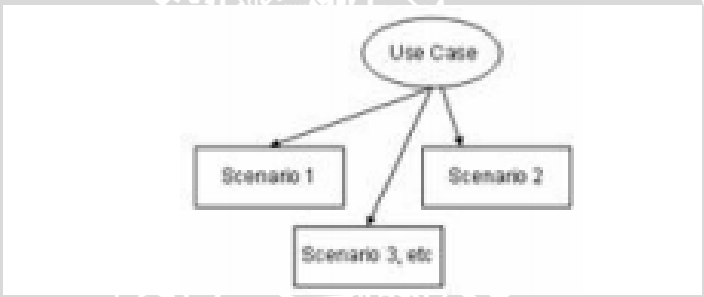
Menurut Heumann (2005) disitasi dalam Leffingwell (2002), salah satu pendekatan spesifik dalam pengujian persyaratan adalah untuk memastikan bahwa masing-masing *use case* sudah diuji oleh satu atau lebih *test case*. Gambar 2.5 merupakan bagian dari *traceability model* yang menggambarkan kerunutan antara *use case* dengan *test case*:



Gambar 2.7 *Traceability Model Use Case* dengan *Test Case*
Sumber: (Leffingwell, 2002)

Langkah-langkah untuk membuat matriks kerunutan antara *use case* dengan *test case* adalah sebagai berikut:

1. Identifikasi seluruh skenario yang telah dideskripsikan pada masing-masing *use case*. *Use case* akan memiliki berbagai kemungkinan skenario yang dapat diuji. Berdasarkan *traceability viewpoint*, masing-masing *use case* dapat dirunutkan dengan masing-masing skenario *use case* seperti pada Gambar 2.6:



Gambar 2.8 Kerunutan *Use Case* dengan Skenario *Use Case*
Sumber: (Leffingwell, 2002)

2. Masing-masing skenario *use case*, akan menghasilkan satu atau lebih spesifik *test case* pada matriks kerunutan. Penulisan pada mariks kerunutan adalah dengan menambahkan satu kolom *test case id* untuk merunutkan *use case* dengan *test case*.
3. Matriks kerunutan dari *use case* ke skenario dan skenario ke *test case* dapat mendeskripsikan seluruh hubungan antara elemen persyaratan dan pengujian. Gambar 2.7 merupakan contoh matriks kerunutan antara *use case*, skenario dan *test case* berdasarkan *traceability model:*

Use Case	Scenario Number ...	Test Case Id
Use Case #1	1	1.1
	2	2.1
	3	3.1
	4	4.1
	4	4.2
	4	4.3
	5	5.1
	6	6.1
	7	7.1
	7	7.2
	8	8.1
Use Case #2	1	1.1

Gambar 2.9 Gambar Matriks Kerunutan *Use Case*, Skenario dan *Test Case*
Sumber: (Leffingwell, 2002)

2.2.7 UML

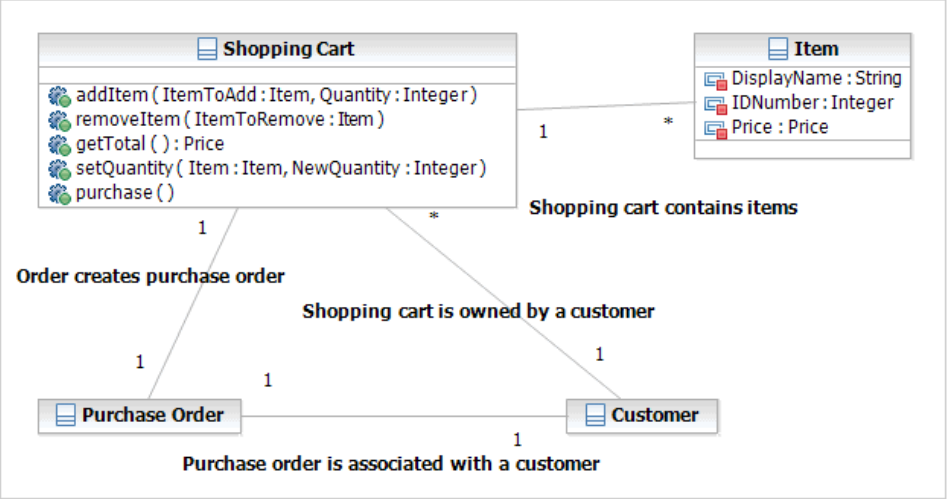
Menurut Nugroho (2009) UML (*Unified Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. UML juga merupakan kumpulan notasi grafis, yang membantu dalam menjelaskan dan merancangan sistem perangkat lunak, khususnya sistem perangkat lunak yang dibangun menggunakan gaya berorientasi objek (Fowler, 2003).

Pemodelan sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari. Berikut adalah jenis-jenis diagram:

1. **Diagram Kelas**

   Diagram kelas menggambarkan berbagai jenis objek dalam sistem dan berbagai macam hubungan statis yang ada. Diagram kelas juga menunjukkan sifat dan operasi dari kelas serta *constraint* yang berlaku pada objek yang terhubung (Fowler, 2003). Diagram kelas juga merupakan fundamental terhadap proses pemodelan objek dan model struktur statis dari suatu sistem (IBM Knowledge Center, 2004).

   Gambar 2.8 merupakan contoh dari diagram kelas.



Gambar 2.10 Class Diagram

Sumber: IBM (2004)

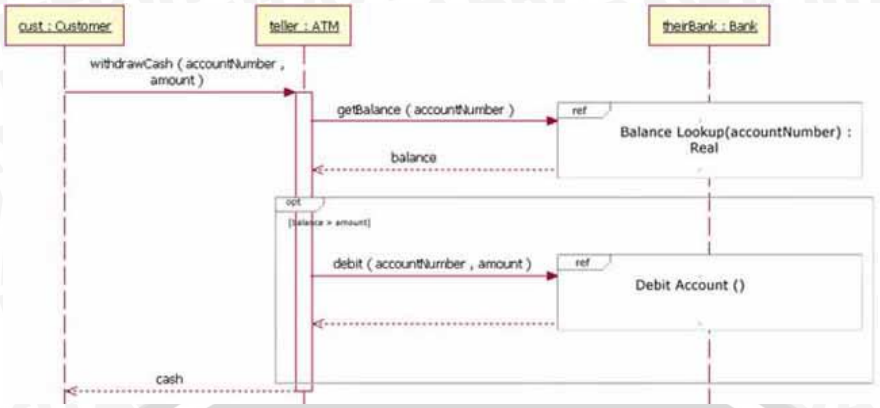
Pada diagram kelas dapat memiliki hubungan antar kelas, diantaranya sebagai berikut (IBMb, 2004):

a. Agregasi
   Agregasi adalah hubungan antar kelas yang menunjukkan kelas sebagai bagian dari kelas lain.
b. Asosiasi
   Asosiasi adalah hubungan struktural yang menunjukkan bahwa objek dari satu kelas terhubung dan dapat mengarah ke objek dari kelas lain.
c. Komposisi
   Komposisi adalah hubungan antar kelas yang menunjukkan kelas sebagai bagian utuh dan merupakan bentuk agregasi.
d. Generalisasi
   Generalisasi adalah hubungan antar kelas dimana satu model elemen atau *child* didasarkan pada elemen lain atau *parent*. Hubungan generalisasi digunakan dalam diagram kelas, komponen, *deployment*, dan *use case* untuk menunjukkan bahwa *child* menerima semua atribut, operasi dan hubungan yang didefinisikan *parent*.

2. Diagram *Sequence*

Diagram *sequence* merupakan diagram interaksi yang menunjukkan objek sebagai *lifeline* yang berjalan ke bawah, dengan interaksi dari waktu ke waktu tersebut direpresentasikan sebagai pesan yang digambarkan dengan garis dari sumber *lifeline* ke *lifeline* tujuan (Sparx, 2000). Diagram *sequence* menggambarkan bagaimana kelompok-kelompok objek berkolaborasi (Fowler, 2003).

Diagram *sequence* utamanya digunakan untuk menunjukkan interaksi antar objek dalam urutan interaksi yang terjadi. Gambar 2.9 merupakan contoh dari diagram *sequence*.



Gambar 2.11 *Sequence* Diagram
Sumber: IBM (2004)

2.2.8 Pemodelan Basis Data

Pemodelan basis data dalam UML dapat memberikan kemampuan untuk mengambil item lebih banyak pada diagram visual daripada dengan notasi ER (IBM, 2001). Pemodelan basis data menggunakan *physical data model* (PDM).

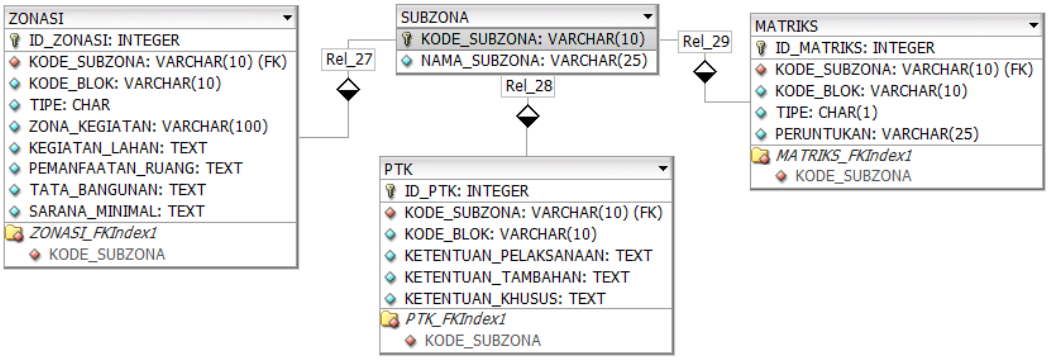
Berikut adalah elemen-elemen dari pemodelan data (IBM, 2001):

1. Tabel merupakan pengelompokan informasi pada basis data tentang subjek yang sama, terdiri dari kolom-kolom
2. Kolom merupakan sebuah komponen dari tabel yang memegang atribut tunggal dari tabel
3. *Primary Key* merupakan kandidat *key* yang dipilih untuk mengidentifikasi baris dalam sebuah table
4. *Foreign Key* merupakan kolom dalam tabel yang memetakan *primary key* ke tabel lain
5. *Identifying Relationship* merupakan hubungan antara dua tabel dimana *child* tabel harus berdampingan dengan *parent* tabel
6. *Non-Identifying Relationship* merupakan hubungan antara dua tabel dimana setiap tabel dapat berdiri secara independen

Langkah-langkah untuk membuat PDM adalah sebagai berikut (IBM, 2004):

1. Mengubah entitas menjadi tabel
2. Mengubah hubungan menjadi *foreign key*
3. Mengubah atribut menjadi kolom
4. Memodifikasi *physical data model* berdasarkan pada kebutuhan

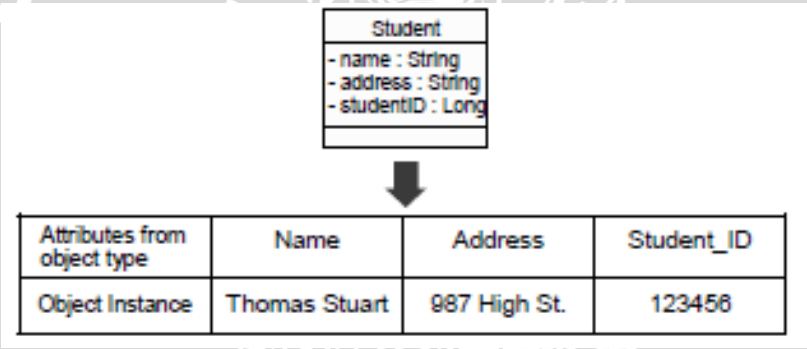
Gambar 2.10 merupakan contoh dari *physical data model*:



Gambar 2.12 Contoh *physical data model*

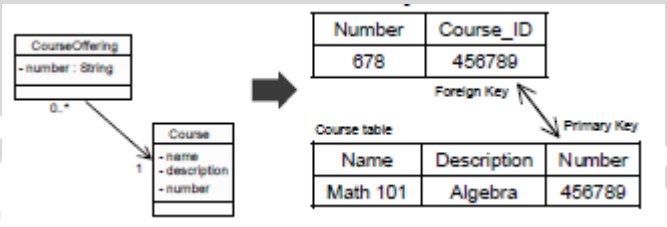
Tujuan dari relasional basis data adalah untuk melakukan normalisasi data, sedangkan tujuan dari pemodelan basis data adalah untuk mengenkapsulasi *behavior* sistem yang kompleks.

Kelas *persistent* merepresentasikan informasi yang harus disimpan pada sistem. Pada relational basis data, setiap baris pada tabel merupakan objek dan kolom pada tabel merupakan atribut dari kelas *persistent*. Gambar 2.11 adalah contoh pemetaan kelas *persistent* ke tabel basis data (IBM, 2004).



Gambar 2.13 Pemetaan kelas *persistent*
Sumber: IBM (2004)

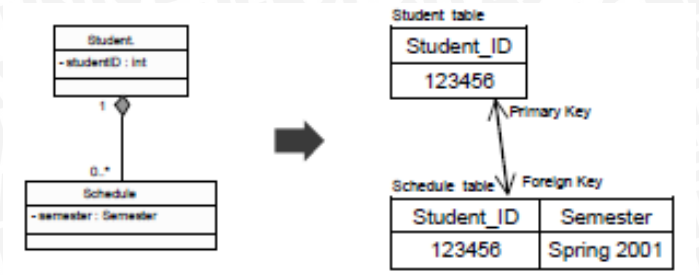
Asosiasi antara dua *persistent object* direalisasikan sebagai *foreign key* pada objek yang memiliki asosiasi. Gambar 2.12 merupakan pemetaan asosiasi antara dua *persistent* objek:



Gambar 2.14 Pemetaan kelas asosiasi
Sumber: IBM (2004)

Agregasi juga dimodelkan menggunakan hubungan *foreign key*. Pada hubungan agregasi, daftar kolom tabel akan ditambahkan dengan *primary key* dari

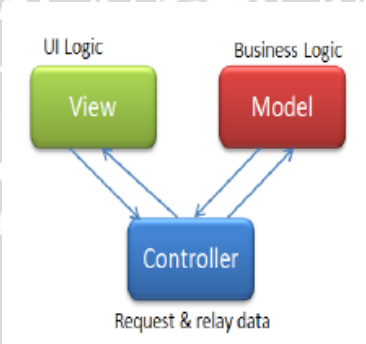
kelas yang berhubungan agregasi. Kolom tersebut akan bersifat sebagai *foreign key*. Gambar 2.13 adalah contoh pemetaan agregasi ke basis data:



Gambar 2.15 Pemetaan kelas agregasi
Sumber: IBM (2004)

2.2.9 MVC (*Model-View-Controller*)

MVC (*Model-View-Controller*) merupakan konsep arsitektur perangkat lunak yang terdiri dari pola arsitektur dalam rekayasa perangkat lunak yang mempunyai tiga komponen, yaitu *Model, View* dan *Controller* (Sarker dan Apu, 2014). Gambar 2.14 menunjukkan hubungan antar komponen MVC.



Gambar 2.16 Komponen MVC
Sumber: Sarker dan Apu (2014)

Dalam MVC, *view* dan *controller* memiliki *user interface*. Pada awalnya, pengguna mengirimkan *request* ke *controller* melalui *graphical user interface* (GUI). Kemudian *controller* mengakses *model* dan memberikan data sesuai *request* pengguna. Setelah itu, *model* mengembalikan data ke *controller*, dan *controller* mengirimkan data serta menampilkannya melalui *view*.

Berikut merupakan kelebihan MVC:

1. Penggunaan ulang komponen-komponen antarmuka pengguna
2. Kemampuan untuk mengembangkan aplikasi dengan antarmuka pengguna secara terpisah
3. Kemampuan untuk melakukan pewarisan (*inheritane*) dari beberapa bagian yang berbeda pada suatu hierarki kelas

2.2.10 Model

Model adalah bagian yang menangani hal yang berhubungan dengan pengolahan data ke dalam basis data, seperti halnya CRUD (Create, Read, Update, Delete). Semua hal yang terhubung dengan pengolahan basis data diletakkan pada *model* (malasngoding, 2016).

2.2.11 View

View adalah bagian yang menangani semua hal tentang tampilan *user interface* atau halaman yang diakses oleh pengguna. Tampilan pada *user interface* digabungkan semua ke dalam *view*. Dalam codeigniter, *view* juga dapat menjadi *fragmen* halaman, seperti *header* atau *footer*. (CodeIgniter, tanpa tahun).

2.2.12 Controller

Controller adalah kumpulan dari instruksi aksi yang menghubungkan *model* dan *view*, jadi pengguna tidak akan langsung berhubungan dengan *model* secara langsung (malasngoding, 2016).

2.2.13 CodeIgniter

CodeIgniter merupakan sebuah *framework* untuk pemrograman php yang bersiat *open source* menggunakan metode MVC (*Model*, *View*, *Controller*) (malasngoding, 2016). CodeIgniter dibangun dengan tujuan untuk memudahkan pengembang atau *programmer* dalam membangun atau mengembangkan sebuah aplikasi berbasis *web*.

CodeIgniter merupakan sebuah *toolkit* untuk orang-orang yang membangun aplikasi berbasis *web* dengan menggunakan PHP (CodeIgniter, 2016). CodeIgniter juga cocok untuk pengembang aplikasi berbasis *web* yang menginginkan *framework* yang sederhana, membutuhkan kompatibilitas yang luas dengan berbagai *web hosting,* menginginkan *framework* yang hampir tidak ada konfigurasi, menginginkan *framework* yang tidak menggunakan *command line*, menginginkan *framework* yang tidak mengharuskan mematuhi aturan penulisan *source code* (Utama, 2011). Tujuannya adalah untuk memungkinkan pengembang untuk mengembangkan proyek-proyek yang jauh lebih cepat daripada menulis kode dari awal, dengan mneyediakan *library* untuk tugas-tugas yang biasa diperlukan, serta antarmuka sederhana. CodeIgniter menggunakan pendekatan MVC (*Model-View—Controller*), yang memungkinkan pemisahan antar *model, view* dan *controller*nya.

Berikut ini merupakan keunggulan-keunggulan yang dimiliki oleh *framework* CodeIgniter (Utama, 2011):

1. Gratis
2. Ringan. Inti dari CI hanya membutuhkan sangat sedikit *library*. *Library* lainnya dapat digunakan dinamis berdasarkan kebutuhan.
3. Sampai saat ini CI masih diakui sebagai *framework* yang paling cepat

4. Menggunakan konsep MVC
5. Dukungan teknis yang lengkap di forum CI.

Meskipun CodeIgniter terdapat *template parser* sederhana yang dapat digunakan secara opsional, tetapi jika menggunakan CodeIgniter tidak harus menggunakannya. Pada tabel 2.1 merupakan contoh *template engine* dari CodeIgniter

Tabel 2.1 Contoh *Source Code* CodeIgniter

```
1  <ul>
2  <?php foreach ($addressbook as $name):?>
3          <li><?=$name?></li>
4  <?php endforeach; ?>
5  </ul>
```

Sumber: CodeIgniter (2016)

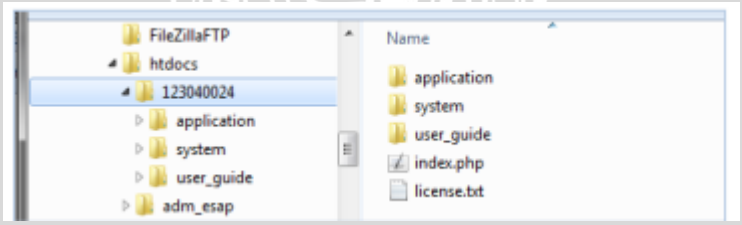
Pada tabel 2.1 sintak pada CodeIgniter dapat menggunakan sintaks dari PHP asli pada umumnya.

Mengapa menggunakan *framework* CodeIgniter? karena beberapa alasan sebagai berikut (Basuki, 2010):

1. CI merupakan *framework* dengan ukuran yang kecil
2. CI mempunyai performa yang handal
3. CI merupakan *framework* dengan konfigurasi minimal
4. CI merupakan *framework* dengan dokumentasi yang jelas

Berikut ini merupakan langkah-langkah bagaimana memasang *framework* CodeIgniter:

1. Unduh *framework* CI pada halaman resminya, yaitu www.codeigniter.com
2. Hasil unduhan berupa file zip, kemudian ekstrak pada direktori htdocs pada XAMPP seperti pada gambar 2.15.



Gambar 2.17 Direktori htdocs

3. Kemudian buka *web browser* dan ketikkan alamat http://localhost/123040024. Hasilnya akan seperti gambar 2.16.

Gambar 2.18 Halaman pembuka codeigniter

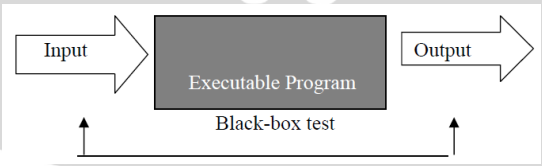
2.2.14 Pengujian

Pengujian adalah proses untuk menganalisis sebuah perangkat lunak untuk mendeteksi perbedaan antara kondisi yang ada dan yang dibutuhkan, serta untuk mengevaluasi fitur dari item perangkat lunak (Williams, 2006). Pada dasarnya, pengujian pada perangkat lunak terdapat dua jenis, yaitu pengujian *Black Box* dan pengujian *White Box*. Pengujian juga dimaksudkan untuk menunjukkan bahwa program melakukan apa yang seharusnya dilakukan dan untuk menemukan kecacatan program sebelum masuk ke tahap penggunaan (Sommerville, 2011).

Pada proses pengujian, terdapat dua tujuan, yaitu: (Sommerville, 2011)

1. Untuk menunjukkan kepada pelanggan bahwa perangkat lunak memenuhi persyaratan, harus ada pengujian untuk semua fitur dari sistem
2. Untuk menemukan kondisi dimana perilaku dari perangkat lunak tidak benar, tidak diinginkan, atau tidak sesuai dengan spesifikasinya. Hal ini merupakan konsekuensi dari cacat perangkat lunak.

Pendekatan yang digunakan dalam pengujian SILOKA adalah pengujian *Black Box*. Pengujian *Black Box* adalah pengujian yang tidak berfokus pada mekanisme internal sistem atau komponen, tetapi hanya berfokus pada keluaran yang dihasilkan sebagai respon terhadap masukan yang dipilih dan kondisi eksekusi (Williams, 2009). Pada gambar 2.17 merupakan ilustrasi pengujian *blackbox*.



Gambar 2.19 Pengujian *Black Box*
Sumber : Williams (2006)

Source code dari program tersebut dianggap sebagai sebuah kotak hitam yang besar dimana penguji tidak dapat melihat ke dalam kotak hitam tersebut. Yang diketahui dari penguji adalah apa yang menjadi masukan ke dalam kotak hitam, dan kotak hitam tersebut akan mengirimkan sebuah keluaran. Berdasarkan persyaratan yang diketahui oleh penguji, penguji tahu keluaran seperti apa yang

diharapkan dari kotak hitam tersebut, dan penguji memastikan apakah keluaran dari kotak hitam tersebut sesuai yang diharapkan atau tidak. Pengujian *Black Box* juga untuk memastikan bahwa fungsi yang ditetapkan dalam spesifikasi persyaratan bekerja.

Pengujian *black box* bertujuan untuk menemukan kesalahan dalam perilaku eksternal sistem yang dikategorikan sebagai berikut:

1. Kesalahan atau hilangnya fungsi
2. Kesalahan *interface*
3. Kesalahan data struktur atau akses database eksternal
4. Kesalahan behavior atau performa

Pengujian *white box* digunakan terutama untuk mendeteksi kesalahan logis dalam kode program (Nidhra dan Jagruthi, 2012). Pengujian *white box* digunakan untuk melakukan *debug* pada kode, menemukan kesalahan ketik atau *typo* secara acak, dan mengungkap asumsi pemrograman yang salah. Pengujian ini dilakukan pada kode program yang telah diimplementasikan.

Berikut ini merupakan beberapa teknik pengujian *white box* (Nidhra dan Jagruthi, 2012):

1. Pengujian *white box* statis:
   a. *Desk checking*
   b. *Code walkthrough*
   c. *Formal inspections*
2. Pengujian *white box* stuktural:
   a. *Control flow / coverage testing*
   b. *Basis path testing*
   c. *Loop testing*
   d. *Data flow testing*

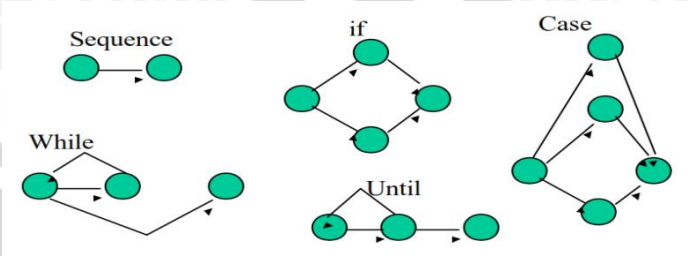
Basis path testing – flow graph notation

Control flow graph (CFG) merupakan grafik terarah yang terdiri dari dua tipe, yaitu *node* dan *control flow* (Nidhra dan Jagruthi, 2012). *Node* dinyatakan oleh lingkaran berlabel, yang mewakili satu atau lebih pernyataan, kondisi keputusan, prosedur program atau konvergensi dari dua atau lebih *node*. *Control flow* dinyatakan dengan panah atau garis yang mewakili aliran kontrol program. Dalam CFG, sebuah *node* dapat termasuk kondisi yang disebut predikat *node*, dan *edge* dari predikat *node* harus berkumpul di suatu *node* tertentu. *Area* ditetapkan oleh *edge* dan *node* yang disebut sebagai *region*.

Berikut ini merupakan aturan dalam *flow graph* (Nidhra dan Jagruthi, 2012):

1. Dalam *flow graph* simbol panah disebut sebagai *Edges* yang mewakili *control flow*
2. Lingkaran disebut sebagai *node*, yang merupakan satu atau lebih tindakan
3. *Area* yang dibatasi oleh *Edges* dan *Node* disebut *region*
4. Sebuah predikat *node* adalah *node* yang mengandung suatu kondisi

Pada gambar 2.18 merupakan notasi-notasi pada *flow graph*



Gambar 2.20 Notasi pada *flow graph*

Sumber: Nidhra dan Jagruthi (2012)

Basis path testing – cyclomatic complexity

Cyclomatic Complexity diperoleh dari jumlah *edges* dari *flow graph* program dikurangi jumlah *node* ditambah dua (Nidhra dan Jagruthi, 2012). *Cyclomatic complexity* murni tergantung pada *flow graph* dari program yang akan diuji.

Cyclomatic complexity dari struktur program dapat dikalkulasi dengan rumus sebagai berikut (Nidhra dan Jagruthi, 2012):

V (G) = e – n + 2

Cara alterntif untuk mengkalkulasi *cyclomatic complexity* program dari pengecekan *flow graph* adalah sebagai berikut (Nidhra dan Jagruthi, 2012):

V (G) = total dari daerah yang dibatasi oleh *edges* dan *node* (*region*)

V (G) = P + 1 dimana P adalah jumlah dari predikat *node*

Langkah-langkah untuk melakukan *cyclomatic complexity* (Nidhra dan Jagruthi, 2012):

1. Menggambar sebuah *flow graph*
2. Menentukan *cyclomatic complexity*
3. Menentukan *independent paths*
4. Persiapkan *test case*

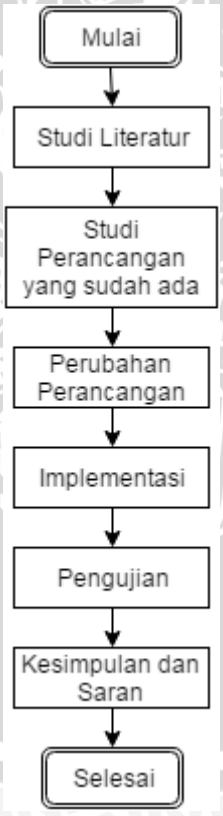
BAB 3 METODOLOGI

Bab ini membahas mengenai metodologi atau pendekatan yang dipakai untuk mengimplementasikan Sistem Informasi Izin Lokasi pada Dinas Cipta Karya dan Tata Ruang, mulai dari pendekatan yang digunakan sampai dengan diagram alur dari awal pengerjaan sampai akhir pengerjaan.

3.1 Pendekatan Penelitian

Pendekatan yang digunakan pada penelitian implementatif kontruksi ini adalah implementasi menggunakan pemrograman berorientasi objek, dimana dalam implementasi system informasi izin lokasi ini menganut sistem MVC, yaitu *Model, View* dan *Controller*. Alasan penggunaan pemrograman berorientasi objek oleh peneliti karena terdapat perancangan yang sudah dilakukan dan perancangan yang sudah dilakukan juga menggunakan pendekatan berorientasi objek. Untuk mewujudkan pemrograman berorientasi objek, penulis menggunakan sebuah *framework*, yaitu codeigniter.

3.2 Diagram Alir Pengerjaan



Gambar 3.1 Diagram Alir

3.2.1 Studi Literatur

Pada langkah pertama ini, Selain itu, penulis juga mempelajari tentang literatur-literatur yang berkaitan dengan penelitian ini. Studi literatur ini akan

bermanfaat bagi peneliti guna dijadikan sebuah referensi untuk melakukan pembangunan SILOKA pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang.

3.2.2 Studi Perancangan yang Sudah Ada

Pada langkah kedua, penulis mempelajari tentang penelitian sebelumnya, yaitu penelitian yang berjudul Analisis dan Perancangan Sistem Informasi Izin Lokasi (SILOKA) pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang dengan Pendekatan Berorientasi Objek yang dilakukan oleh Bella Pertiwi Mahasiswi Sistem Informasi Fakultas Ilmu Komputer (Pertiwi, 2016). Dari penelitian tersebut sudah dilakukan analisis persyaratan dan perancangan dari SILOKA serta evaluasi dari perancangan analisis dan perancangan tersebut, tetapi tanpa adanya implementasi.

3.2.3 Perubahan Perancangan

Setelah mempelajari tentang analisis dan perancangan yang telah dilakukan, maka langkah selanjutnya adalah penulis akan menyesuaikan rancangan sebelumnya dengan *framework* yang penulis gunakan, yaitu *framework* CodeIgniter. Penyesuaian dilakukan pada diagram kelas, terdapat penambahan kelas CI *Model* dan CI *Controller*.

3.2.4 Implementasi

Setelah penulis melakukan perubahan perancangan untuk kebutuhan implementasi, langkah selanjutnya adalah mengimplementasikan dari perancangan yang sudah dirubah ke dalam *code* program. Untuk mengimplementasikan SILOKA, penulis menggunakan *framework* CodeIgniter.

3.2.5 Pengujian

Setelah system sudah selesai diimplementasikan, maka harus dilakukan pengujian, yaitu dengan menggunakan jenis pengujian *black box*, dimana terdapat seorang pengguna langsung dari Dinas Cipta Karya dan Tata Ruang yang menggunakan system ini. Pengujian *black box* pada penelitian ini bertujuan untuk mengetahui apakah fungsi-fungsi yang ada sudah memenuhi kebutuhan yang sudah dispesifikasikan. Selain menggunakan pengujian *black box* juga digunakan pengujian *white box*. Pengujian *white box* bertujuan untuk mendeteksi kesalahan pada kode program, juga untuk mengetahui kompleksitas dan jalur independen yang terdapat pada kode program yang telah diimplementasikan.

3.2.6 Kesimpulan dan Saran

Setelah dilakukan pengujian pada sistem yang telah diimplementasikan, dan didapatkan kesimpulan dari implementasi SILOKA. Tidak hanya memberikan kesimpulan, tetapi juga bisa memberikan saran untuk Sistem Informasi Izin Lokasi, karena pada sistem yang sudah diimplementasikan mungkin masih belum sempurna, untuk itu penulis menerima segala saran yang membangun untuk SILOKA.

BAB 4 TINJAUAN DAN ANALISIS LANJUT

Bab ini membahas tentang tinjauan dari penelitian sebelumnya yang sudah dilakukan dan hasil analisis lanjut mengenai perancangan SILOKA.

4.1 Penjelasan Produk

4.1.1 Cakupan Produk

Sistem Informasi Izin Lokasi (SILOKA) adalah sebuah sistem informasi yang memungkinkan seseorang atau warga yang mewakili sebuah instansi atau perusahaan dapat mengajukan permohonan izin lokasi secara *online*. SILOKA juga memungkinkan pegawai Dinas Cipta Karya dan Tata Ruang Kabupaten Malang yang terlibat dalam permohonan izin lokasi untuk mengelola data izin lokasi secara *online* melalui sistem. SILOKA adalah sebuah sistem informasi berbasis *website*, jadi SILOKA dapat diakses dimanapun, pengguna hanya membutuhkan koneksi *internet*.

Dengan adanya SILOKA ini, calon pemohon izin atau warga dapat mengetahui informasi tentang bagaimana melakukan permohonan izin lokasi. SILOKA akan mempermudah melakukan izin lokasi karena warga yang akan melakukan permohonan izin tidak perlu datang ke kantor dinas, cukup hanya dengan mengakses SILOKA, kemudian melakukan pendaftaran dan melakukan pengajuan.

SILOKA juga dapat mempercepat dalam pencarian peraturan zonasi yang diperlukan oleh surveyor dinas yang akan melakukan survey. Pada kondisi sekarang peraturan zonasi pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang masih berupa modul buku cetak sebanyak enam modul. Jika pencarian peraturan zonasi dilakukan pada modul buku cetak tersebut membutuhkan waktu yang lama. SILOKA dapat mempercepat pencarian karena surveyor dinas cukup memasukkan kata kunci pencarian berupa kode blok, kode subzona dan peruntukan.

4.1.2 Fitur Produk

Fitur merupakan kemampuan yang dapat dilakukan oleh SILOKA secara umum. Fitur yang didefinisikan pada penelitian ini merujuk pada penelitian sebelumnya, yaitu perancangan SILOKA yang dilakukan oleh Bella Pertiwi. Pada tabel 4.1 merupakan fitur dari SILOKA:

Tabel 4.1 Fitur SILOKA

Kode Fitur	Nama Fitur	Deskripsi Fitur
FEAT1	Identifikasi Pengguna	Sistem dapat melakukan identifikasi pengguna
FEAT2	Pencarian Data Zonasi	Sistem dapat digunakan untuk melakukan pencarian data peraturan zonasi

Tabel 4.1 Fitur SILOKA (lanjutan)

Kode Fitur	Nama Fitur	Deskripsi Fitur
FEAT3	Mengelola Data Survey	Sistem dapat digunakan untuk melakukan pengelolaan data survey
FEAT4	Mengelola Data Zonasi	Sistem dapat digunakan untuk melakukan pengelolaan data zonasi
FEAT5	Menambah Jadwal	Sistem dapat digunakan untuk memasukan jadwal survey
FEAT6	Permohonan Izin Lokasi	Sistem dapat digunakan untuk melakukan proses izin lokasi
FEAT7	Mengelola Data Pengguna	Sistem dapat digunakan untuk melakukan pengelolaan data pengguna
FEAT8	Mengelola Data Berkas	Sistem dapat digunakan untuk mengelola data berkas
FEAT9	Mengelola Data Register	Sistem dapat digunakan untuk mengelola data register
FEAT10	Mengelola Disposisi	Sistem dapat digunakan untuk membuat lembar dan catatan disposisi
FEAT11	Verifikasi Berkas	Sistem dapat digunakan untuk membuat catatan kelengkapan berkas
FEAT12	Mengelola Data Pasca Rapat	Sistem dapat digunakan untuk mengelola data pasca rapat
FEAT13	Registrasi	Sistem dapat digunakan untuk mendaftarkan diri pada sistem
FEAT14	Ketersediaan Informasi	Sistem menyediakan informasi secara *real-time*
FEAT15	Waktu Akses	Sistem dapat diakses 24 jam sehari dan 7 hari seminggu

Sumber : Pertiwi (2016)

4.1.3 Persyaratan Fungsional

Dari fitur yang sudah didefinisikan di penelitian ini memiliki fungsi-fungsi yang terdapat di dalamnya. Persyaratan fungsional telah didefinisikan pada penelitian sebelumnya, yaitu perancangan SILOKA (Pertiwi, 2016). Pada tabel 4.2 merupakan persyaratan fungsional SILOKA.

Tabel 4.2 Persyaratan Fungsional SILOKA

Kode Fitur	Kode Dasar Persyaratan Fungsional	Kode Lengkap Persyaratan Fungsional	Deskripsi
FEAT1	SRS-F-SIL-P01	SRS-F-SIL-P01-1	Sistem dapat melakukan autorisasi apakah pengguna yang menggunakan sistem merupakan pengguna yang teridentifikasi dan terautorisasi untuk menggunakan fitur SILOKA
FEAT2	SRS-F-SIL-P02	SRS-F-SIL-P02-1	Sistem dapat digunakan untuk melakukan pencarian data peraturan zonasi dengan kata kunci pencarian berupa kode blok, kode subzona dan peruntukan bangunan
FEAT3	SRS-F-SIL-P03	SRS-F-SIL-P03-1	Sistem dapat digunakan untuk menampilkan data hasil survey
		SRS-F-SIL-P03-2	Sistem dapat digunakan untuk menambahkan data hasil survey pada sistem
		SRS-F-SIL-P03-3	Sistem dapat digunakan untuk melakukan perubahan data hasil survey yang sudah ditambahkan pada sistem
FEAT4	SRS-F-SIL-P04	SRS-F-SIL-P04-1	Sistem dapat digunakan untuk menampilkan data zonasi, matriks dan ketentuan tambahan.
		SRS-F-SIL-P04-2	Sistem dapat digunakan untuk menambahkan data zonasi, matriks dan ketentuan tambahan
		SRS-F-SIL-P04-3	Sistem dapat digunakan untuk melakukan penghapusan data zonasi, matriks dan ketentuan tambahan yang sudah ditambahkan pada sistem.
		SRS-F-SIL-P04-4	Sistem dapat digunakan untuk melakukan perubahan pada data zonasi, matriks dan ketentuan tambahan yang sudah ditambahkan pada sistem.
FEAT5	SRS-F-SIL-P05	SRS-F-SIL-P05-1	Sistem dapat digunakan untuk menampilkan data jadwal survey dan rapat izin lokasi.

Tabel 4.2 Persyaratan Fungsional SILOKA (lanjutan)

Kode Fitur	Kode Dasar Persyaratan Fungsional	Kode Lengkap Persyaratan Fungsional	Deskripsi
		SRS-F-SIL-P05-2	Sistem dapat digunakan untuk menambahkan jadwal survey dan rapat pada sistem
		SRS-F-SIL-P05-3	Sistem dapat digunakan untuk melakukan perubahan pada jadwal survey dan rapat pada sistem yang sudah ditambahkan pada sistem
		SRS-F-SIL-P05-2	Sistem dapat digunakan untuk menambahkan jadwal survey dan rapat pada sistem
FEAT6	SRS-F-SIL-P06	SRS-F-SIL-P06-1	Sistem dapat digunakan untuk melihat data kekurangan berkas
		SRS-F-SIL-P06-2	Sistem dapat digunakan untuk melihat progres permohonan izin lokasi yang diajukan
		SRS-F-SIL-P06-3	Sistem dapat menampilkan hasil pasca rapat izin lokasi
		SRS-F-SIL-P06-4	Sistem dapat digunakan untuk mendapatkan formulir permohonan izin lokasi dengan mengunduh formulir yang terdapat pada sistem
FEAT7	SRS-F-SIL-P07	SRS-F-SIL-P07-1	Sistem dapat digunakan untuk menampilkan data pengguna yang terdapat pada sistem
		SRS-F-SIL-P07-2	Sistem dapat digunakan untuk melakukan perubahan pada data pribadi dan *password* pemohon izin pada sistem
		SRS-F-SIL-P07-3	Sistem dapat digunakan untuk menambahkan pengguna pada sistem
		SRS-F-SIL-P07-4	Sistem dapat digunakan untuk melakukan penghapusan data pengguna yang sudah ditambahkan pada sistem
		SRS-F-SIL-P07-5	Sistem dapat digunakan untuk melakukan perubahan pada data pengguna yang sudah ditambahkan pada sistem

Tabel 4.2 Persyaratan Fungsional SILOKA (lanjutan)

Kode Fitur	Kode Dasar Persyaratan Fungsional	Kode Lengkap Persyaratan Fungsional	Deskripsi
FEAT8	SRS-F-SIL-P08	SRS-F-SIL-P08-1	Sistem dapat digunakan untuk menampilkan data berkas pemohon pada sistem
		SRS-F-SIL-P08-2	Sistem dapat digunakan untuk menambahkan data berkas izin lokasi pada sistem
		SRS-F-SIL-P08-3	Sistem dapat digunakan untuk melakukan perubahan pada data berkas yang sudah ditambahkan pada sistem
		SRS-F-SIL-P08-4	Sistem dapat digunakan untuk melakukan penghapusan data berkas izin lokasi yang sudah ditambahkan pada sistem
FEAT9	SRS-F-SIL-P09	SRS-F-SIL-P09-1	Sistem dapat digunakan untuk menampilkan data register
		SRS-F-SIL-P09-2	Sistem dapat digunakan untuk menambahkan data register pada sistem
		SRS-F-SIL-P09-3	Sistem dapat digunakan untuk melakukan perubahan data register pada sistem yang sudah ditambahkan pada sistem
		SRS-F-SIL-P09-4	Sistem dapat digunakan untuk menghapus data register yang terdapat pada sistem
FEAT10	SRS-F-SIL-P10	SRS-F-SIL-P10-1	Sistem dapat digunakan untuk menampilkan data disposisi
		SRS-F-SIL-P10-2	Sistem dapat digunakan untuk menambahkan data disposisi pada sistem
		SRS-F-SIL-P10-3	Sistem dapat digunakan untuk melakukan pencetakan data disposisi yang terdapat pada sistem
		SRS-F-SIL-P10-4	Sistem dapat digunakan untuk menerima notifikasi dan melihat data catatan disposisi

Tabel 4.2 Persyaratan Fungsional SILOKA (lanjutan)

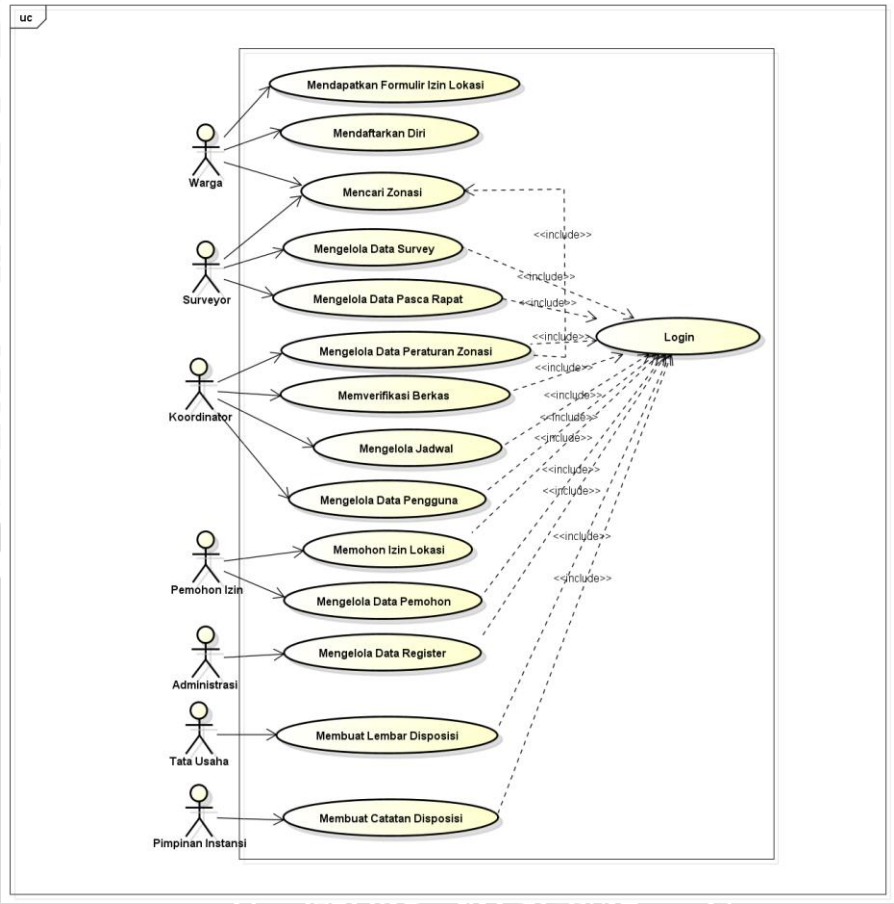
Kode Fitur	Kode Dasar Persyaratan Fungsional	Kode Lengkap Persyaratan Fungsional	Deskripsi
		SRS-F-SIL-P10-5	Sistem dapat digunakan untuk menambahkan catatan diposisi pada data disposisi yang terdapat pada sistem
FEAT11	SRS-F-SIL-P11	SRS-F-SIL-P11-1	Sistem dapat digunakan untuk melihat berkas permohonan izin lokasi pemohon
		SRS-F-SIL-P11-2	Sistem dapat digunakan untuk menambahkan data kelengkapan berkas pada sistem
FEAT12	SRS-F-SIL-P12	SRS-F-SIL-P12-1	Sistem dapat digunakan untuk menampilkan data pasca rapat
		SRS-F-SIL-P12-2	Sistem dapat digunakan untuk menambahkan data pasca rapat pada sistem
		SRS-F-SIL-P12-3	Sistem dapat digunakan untuk melakukan perubahan data pasca rapat pada sistem yang sudah ditambahkan pada sistem
FEAT13	SRS-F-SIL-P13	SRS-F-SIL-P13-1	Sistem dapat digunakan untuk pendaftaran pada sistem sehingga terdaftar pada sistem dan dapat melakukan pengajuan izin lokasi

Sumber: Pertiwi (2016)

Pada penelitian sebelumnya, lebih difokuskan pada persyaratan fungsional saja, sehingga persyaratan non fungsional tidak dibahas lebih lanjut (Pertiwi, 2016).

4.1.4 Pemodelan *use case*

Pada penelitian sebelumnya terdapat pemodelan *use case*, pemodelan *use case* bertujuan untuk menggambarkan mengenai tujuan yang ingin dicapai aktor pada sistem. Gambar 4.1 menunjukkan *use case* diagram SILOKA.



Gambar 4.1 *Use case* diagram SILOKA

Sumber : Pertiwi (2016)

SILOKA merupakan sistem informasi berbasis *website* yang ditujukan untuk masyarakat Kabupaten Malang yang hendak mengajukan permohonan izin lokasi. Dalam *use case* diagram SILOKA terdapat tujuh aktor, tabel 4.3 menunjukkan deskripsi dari setiap aktor dari *use case* diagram SILOKA

Tabel 4.3 Deskripsi Aktor *Use Case* Diagram SILOKA

Nama Aktor	Deskripsi Aktor
Warga	Orang yang hendak mencari informasi mengenai zonasi, tata cara permohonan izin lokasi dan mendaftar untuk mengajukan permohonan izin lokasi pada instansi
Surveyor	Orang yang bertanggungjawab untuk mengelola data hasil survey yang telah dilakukan
Koordinator	Orang yang bertanggungjawab untuk mengelola data pengguna, data berkas pengajuan, memilih surveyor dan menambahkan jadwal survey serta verifikasi berkas pengajuan
Pemohon Izin	Orang yang hendak mengajukan permohonan izin lokasi pada instansi
Administrasi	Orang yang bertanggungjawab untuk mengelola data register

Tabel 4.3 Deskripsi Aktor *Use Case* Diagram SILOKA (lanjutan)

Nama Aktor	Deskripsi Aktor
Tata Usaha	Orang yang bertanggungjawab untuk membuat lembar disposisi untuk setiap pengajuan permohonan izin lokasi
Pimpinan Instansi	Orang yang bertanggungjawab untuk menambahkan catatan pada lembar disposisi yang telah dibuat oleh tata usaha

Sumber : Pertiwi (2016)

4.1.5 Spesifikasi *Use Case*

Pada tabel 4.4 merupakan penjelasan mengenai spesifikasi *use case* mendaftarkan diri:

Tabel 4.4 Spesifikasi *Use Case* Mendaftarkan Diri

1. Brief Description	*Use case* ini menjelaskan mengenai bagaimana warga melakukan pendaftaran pada SILOKA sehingga terdaftar pada sistem dan dapat melakukan pengajuan permohonan izin lokasi
2. Basic Flow of Events	1. *Use case* dimulai ketika warga membuka sistem 2. Sistem menampilkan pilihan fungsi yang tersedia untuk warga **{Memilih Mendaftar}** 3. Warga memilih untuk melakukan pendaftaran 4. Sistem meminta warga untuk mengisi detail data pengguna yang dibutuhkan untuk melakukan pendaftaran **{Mengisi Detail Data}** 5. Warga mengisi detail data pengguna pada sistem **{Melakukan Pendaftaran}** 6. Warga melakukan pendaftaran dengan menyimpan detail data pengguna yang sudah diisi melalui sistem. 7. Sistem menampilkan pesan warga berhasil mendaftar **{*Use Case* Selesai}** 8. *Use Case* melakukan pendaftaran selesai
3. Alternative Flows	3.1 Terdapat data yang belum diisi Pada **{Mengisi Detail Data}** di *basic flow*, apabila warga tidak mengisi pada seluruh data yang diminta maka sistem akan menampilkan pesan untuk mengisi seluruh data dan melanjutkan *basic flow* pada **{Mengisi Detail Data}.** 3.2 Data yang didaftarkan sudah terdapat pada sistem Pada **{Melakukan Pendaftaran}** di *basic flow*, apabila data warga yang didaftarkan sudah terdapat pada sistem, maka sistem akan menampilkan pesan bahwa pengguna yang didaftarkan sudah terdapat pada sistem. 3.3 Tidak jadi melakukan pendaftaran Kapanpun pada alur *basic flow*, warga dapat tidak jadi melakukan pendaftaran dan **{*Use Case* Selesai}**
4. Subflow	Tidak terdapat *subflow* pada *use case* ini

Tabel 4.4 Spesifikasi *Use Case* Mendaftarkan Diri (lanjutan)

5. Key Scenarios	1. Warga berhasil melakukan pendaftaran : *Basic flow* 2. Terdapat data yang belum diisi : *Basic flow*, Terdapat data yang belum diisi, *Basic flow*. 3. Data yang didaftarkan sudah terdapat pada sistem : *Basic flow*, Data yang didaftarkan sudah terdapat pada sistem 4. Tidak jadi melakukan pendaftaran : *Basic flow*, Tidak jadi melakukan pendaftaran
6. Pre-Condition	Warga mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet
7. Post-Condition	Warga berhasil melakukan pendaftaran dan memiliki akun untuk masuk ke dalam sistem informasi izin lokasi
8. Extension Points	Tidak terdapat *extension points* pada *use case* ini
9. Special Requirements	Tidak terdapat *speial requirements* pada *use case* ini
***10. Glossary* Detail Data Pengguna**	Detail data pengguna merupakan detail pribadi yang membedakan pengguna dalam sistem. Informasi detail data pengguna terdiri dari nama pemohon, jenis identitas , nomor identitas, alamat, nomor telepon, email dan *password*.

Sumber: Pertiwi (2016)

Pada tabel 4.5 merupakan penjelasan mengenai spesifikasi *use case* mendapatkan formulir izin lokasi:

Tabel 4.5 Spesifikasi *Use Case* Mendapatkan Formulir Izin Lokasi

1. Brief Descrip ***2. tion***	*Use case* ini menjelaskan mengenai bagaimana warga mendapatkan formulir permohonan izin lokasi melalui SILOKA.
3. Basic Flow of Events	1. *Use case* dimulai ketika warga membuka sistem 2. Sistem menampilkan pilihan fungsi yang tersedia untuk warga **{Memilih Mengunduh}** 3. Warga memilih untuk mendapatkan formulir dengan mengunduh formulir dari sistem **{*Use Case* Selesai}** 4. *Use Case* mendapatkan formulir selesai
4. Alternative Flows	3.1 Tidak jadi mendapat formulir Kapanpun pada alur *basic flow*, warga dapat tidak jadi mendapat formulir dan **{*Use Case* Selesai}**
5. Subflow	Tidak terdapat *subflow* pada *use case* ini
6. Key Scenarios	*1.* Warga berhasil mendapat formulir : *Basic flow* *2.* Tidak jadi mendapat formulir : *Basic flow*, Tidak jadi mendapat formulir

Tabel 4.5 Spesifikasi *Use Case* Mendapatkan Formulir Izin Lokasi (lanjutan)

7. Pre-Condition	Warga mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan *internet*
8. Post-Condition	Warga berhasil mendapatkan formulir permohonan izin lokasi
9. Extension Points	Tidak terdapat *extension points* pada *use case* ini
10. Special Requirements	Tidak terdapat *speial requirements* pada *use case* ini

Sumber: Pertiwi (2016)

Pada tabel 4.6 merupakan penjelasan mengenai spesifikasi *use case* mencari zonasi:

Tabel 4.6 Spesifikasi *Use Case* Mencari Zonasi

1. Brief Description	*Use case* ini menjelaskan mengenai bagaimana warga,surveyor atau koordinator dapat melakukan pencarian data peraturan zonasi pada SILOKA.
2. Basic Flow of Events	1. *Use case* dimulai ketika aktor warga , koordinator atau surveyor membuka sistem 2. Sistem menampilkan pilihan fungsi yang tersedia untuk surveyor, koordinator atau warga 3. Surveyor, koordinator atau warga memilih untuk melakukan pencarian 4. Sistem meminta warga, koordinator atau surveyor untuk memasukan kata kunci pencarian **{Memasukan kata kunci pencarian}** 5. Warga, surveyor atau koordinator memasukan kata kunci pencarian **{Memilih cari}** 6. Warga, surveyor atau koordinator memilih untuk melakukan pencarian 7. Sistem menampilkan hasil pencarian **{*Use case* selesai}** 8. *Use case* mencari zonasi selesai
3. Alternative Flows	3.1 Tidak memasukan seluruh kata kunci Pada **{Memasukan kata kunci}** di *basic flow*, jika pemohon izin, koordinator atau surveyor tidak memasukan seluruh kata kunci yang diminta untuk pencarian maka sistem akan menampilkan pesan untuk memasukan seluruh kata kunci pencarian. 3.2 Data hasil pencarian tidak ditemukan Pada **{Memilih cari}** di *basic flow*, jika data hasil pencarian tidak ditemukan maka sistem akan menampilkan pesan data tidak ditemukan dan sistem kembali menampilkan form pencarian.

Tabel 4.6 Spesifikasi *Use Case* Mencari Zonasi (lanjutan)

3. Alternative Flows	3.3 Data hasil pencarian matriks lebih dari satu Pada **{Memilih cari}** di *basic flow*, jika data hasil pencarian lebih dari satu, maka 1. Sistem menampilkan opsi data matriks yang sama dengan kata kunci yang dimasukan 2. Warga, koordinator atau surveyor memilih melihat data ketentuan berdasarkan opsi data yang ditampilkan 3. Jika hasil pencarIan data ketentuan zonasi lebih dari satu maka sub alur **data ketentuan zonasi kembar** akan dijalankan 4. Sistem menampilkan hasil data ketentuan. 3.4 Data hasil pencarian ketentuan zonasi lebih dari satu Jika hasil pencarIan data ketentuan zonasi lebih dari satu maka sub alur **data ketentuan zonasi kembar** akan dijalankan 3.5 Tidak jadi melakukan pencarian Kapanpun pada alur *basic flow*, pemohon izin, koordinator atau surveyor dapat tidak jadi melakukan pencarian dan **(*Use case* selesai}**
4. Subflow	**4.1 Data Ketentuan Zonasi Kembar** 1. Sistem menampilkan opsi data peruntukan zonasi yang sama dengan kata kunci yang dimasukan 2. Warga, koordinator atau surveyor memilih melihat data ketentuan berdasarkan opsi data yang ditampilkan 3. Sistem menampilkan data ketentuan zonasi
5. Key Scenario	1. Berhasil melakukan pencarian : *Basic flow* 2. Hasil pencarian tidak ditemukan *: Basic flow*, Data hasil pencarian tidak ditemukan 3. Terdapat kata kunci pencarian yang belum diisi : *Basic flow* , Tidak memasukkan seluruh kata kunci 4. Hasil pencarian matriks lebih dari satu : *Basic flow*, Data hasil pencarian matriks lebih dari satu 5. Hasil pencarian zonasi lebih dari satu : *Basic flow*, Data hasil pencarian ketentuan zonasi lebih dari satu 6. Tidak jadi melakukan pencarian : *Basic flow*, Tidak jadi melakukan pencarian
6. Pre-Condition	Aktor mengakses SILOKA dengan perangkat yang memiliki koneksi internet, koneksi basis data pada sistem sudah tersedia serta aktor mengetahui blok dan subzona lokasi yang dicari.
7. Post-Condition	Sistem menampilkan hasil pencarian berupa ketentuan peruntukan bangunan.
8. Extension Points	Tidak ada *extension point* pada *use case* ini
9. Special Requirements	Tidak ada *special requirement* pada *use case* ini

Sumber: Pertiwi (2016)

Pada tabel 4.7 merupakan penjelasan mengenai spesifikasi *use case login*:

Tabel 4.7 Spesifikasi *Use Case Login*

1. *Brief Description*	*Use case* ini digunakan untuk mengautentifikasi bahwa pengguna yang menggunakan sistem adalah pengguna yang teridentifikasi dan terautorisasi untuk menggunakan fitur yang terdapat pada sistem.
2. *Basic Flow of Events*	1. *Use case* dimulai ketika aktor koordinator, surveyor, tata usaha, pemohon izin, administrasi atau pimpinan instansi membuka sistem 2. Sistem menampilkan pilihan fungsi yang tersedia untuk aktor **{Memilih *login*}** 3. Aktor memilih melakukan *login* 4. Sistem meminta aktor untuk memasukan nomor identitas dan kata kunci **{Memasukan nomor identitas dan kata kunci}** 5. Aktor memasukan nomor identitas dan kata kunci untuk dapat masuk ke dalam sistem **{Autentikasi identitas pengguna}** 6. Sistem melakukan proses autentikasi dengan pengecekan nomor identitas dan *password* pada basis data sistem. **{*Use case* selesai}** 7. *Use case* selesai
3. *Alternative Flows*	3.1 Nama atau kata kunci belum diisi Pada **{Memasukan nomor identitas dan kata kunci}** di *basic flow*, apabila terdapat kata kunci atau nomor identitas yang belum diisi maka sistem akan menampilkan pesan untuk mengisi nomor identitas atau kata kunci 3.2 Pengguna tidak teridentifikasi Pada **{Autentikasi identitas pengguna}** di *basic flow*, apabila nomor identitas dan kata kunci yang dimasukan tidak valid maka sistem akan menampilkan pesan autentifikasi gagal. Aktor dapat melanjutkan pada **{Memasukan nomor identitas dan kata kunci}** atau tidak jadi melakukan *login* dan **{*Use case* selesai}** 3.3 Tidak jadi melakukan proses autentifikasi Kapanpun pada alur *basic flow*, aktor dapat tidak jadi melakukan proses *login* pada sistem dan **{*Use Case* Selesai}**
4. *Subflow*	Tidak ada *subflow* pada *use case* ini
5. *Key Scenarios*	1. Proses autentikasi berhasil : *Basic flow* 2. Terdapat data yang kosong : *Basic flow*, Nama atau kata kunci belum diisi 3. Autentifikasi pengguna tidak valid : *Basic flow*, Pengguna tidak teridentifikasi 4. Tidak jadi melakukan *login* : *Basic flow*, Tidak jadi melakukan proses autentifikasi
6. *Pre-Condition*	Aktor mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet

Tabel 4.7 Spesifikasi *Use Case Login* (lanjutan)

7. Post-Condition	Aktor telah berhasil diautentifikasi sebagai pengguna terdaftar pada sistem dan terautorisasi untuk menggunakan fitur yang terdapat pada sistem.
8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini

Sumber: Pertiwi (2016)

Pada tabel 4.8 merupakan penjelasan mengenai spesifikasi *use case* mengelola data peraturan zonasi:

Tabel 4.8 Spesifikasi *Use Case* Mengelola Data Peraturan Zonasi

1. Brief Description	*Use case* ini menjelaskan mengenai bagaiman koordinator dapat mengelola data peraturan zonasi yang meliputi penambahan dan pembaharuan data zonasi, ptk dan matriks pada SILOKA.
2. Basic Flow of Events	1. *Use case* dimulai ketika koordinator membuka sistem. 2. Include *use case* ***Login*** **{Mengakses Modul}** 3. Koordinator masuk kedalam sistem kemudian mengakses modul untuk mengelola data peraturan zonasi 4. Sistem menampilkan pilihan fungsi yang tersedia untuk mengelola data peraturan zonasi. Fungsi tersebut meliputi pengelolaan data ketentuan zonasi, ketentuan P,T,K dan data matriks **{Memilih fungsi dan operasi pengolahan data}** 5. Jika koordinator memilih operasi membuat data matriks, maka sub alur **Menambah Data Matriks** akan dijalankan. 6. Jika koordinator memilih operasi membuat data zonasi, maka sub alur **Menambah Data Zonasi** akan dijalankan 7. Jika koordinator memilih operasi membuat data PTK, maka sub alur **Menambah Data PTK** akan dijalankan 8. Jika koordinator memilih operasi memperbaharui data zonasi, maka sub alur **Memperbarui Data Zonasi** akan dijalankan 9. Jika koordinator memilih operasi memperbaharui data ketentuan P,T,K maka sub alur **Memperbarui Data Ketentuan P,T,K** akan dijalankan 10. Jika koordinator memilih operasi membuat data matriks, maka sub alur **Memperbarui Data Matriks** akan dijalankan. 11. Jika koordinator memilih operasi menghapus data ketentuan P,T,K , maka sub alur **Menghapus Data P,T,K** akan dijalankan 12. Jika koordinator memilih operasi menghapus data zonasi maka sub alur **Menghapus Data Zonasi** akan dijalankan 13. Jika koordinator memilih operasi menghapus data matriks maka sub alur **Menghapus Data Matriks** akan dijalankan **{Use case selesai}** 14. *Use case* mengelola data zonasi selesai

Tabel 4.8 Spesifikasi *Use Case* Mengelola Data Peraturan Zonasi (lanjutan)

3. Alternative Flow	3.1 Terdapat data yang sama Pada **{Menambah data zonasi}** pada *subflow* dan **{Mengisi Data Zonasi}** pada *extension point subflow*, apabila data yang dimasukan oleh koordinator sudah terdapat pada sistem maka sistem akan menampilkan pesan bahwa data yang dimasukan sudah tersedia. 3.2 Tidak jadi melakukan pengelolaan data Kapanpun pada alur *basic flow*, koordinator dapat tidak jadi melakukan pengelolaan data dan **{*Use Case* Selesai}**
4. Subflows	**4.1 Menambah Data Matriks** a. Koordinator memilih fungsi untuk mengelola data matriks dan operasi untuk menambahkan data matriks b. Sistem meminta koordinator untuk mengisi data matriks yang diperlukan untuk penambahan data matriks **{Mengisi data matriks}** c. Koordinator mengisi data matriks dan menyimpan data matriks. d. Sistem menampilkan pesan berhasil menambah data matriks **4.2 Menambah Data Zonasi** a. Koordinator memilih fungsi untuk mengelola data ketentuan zonasi dan operasi untuk menambahkan data zonasi b. Sistem meminta koordinator untuk mengisi data zonasi yang diperlukan untuk penambahan data zonasi **{Mengisi data zonasi}** c. Koordinator mengisi data zonasi dan menyimpan data zonasi. d. Sistem menampilkan pesan berhasil menambah data zonasi **4.3 Menambah Data PTK** a. Sistem menampilkan seluruh data PTK dan koordinator memilih fungsi untuk mengelola data ketentuan P,T,K dan operasi untuk menambahkan data PTK b. Sistem meminta surveyor untuk mengisi data PTK yang diperlukan untuk penambahan data PTK **{Mengisi data zonasi}** c. Koordinator mengisi data PTK dan menyimpan data PTK. d. Sistem menampilkan pesan berhasil menambah data PTK **4.4 Memperbarui Data Zonasi** a. Koordinator memilih fungsi untuk mengelola data ketentuan zonasi dan operasi untuk memperbarui data zonasi b. Include use case **Mencari Zonasi** c. Koordinator memperbarui data ketentuan pada sistem dan menyimpan data tersebut. d. Sistem menampilkan pesan data berhasil diperbarui

Tabel 4.8 Spesifikasi *Use Case* Mengelola Data Peraturan Zonasi (lanjutan)

4. Subflows	**4.5 Menghapus Data Zonasi** **a.** Koordinator memilih fungsi untuk mengelola data ketentuan zonasi dan operasi untuk menghapus data zonasi **b.** Include use case **Mencari Zonasi** **c.** Koordinator memilih data yang akan dihapus dan sistem menampilkan pesan konfirmasi **d.** Koordinator melakukan konfirmasi penghapusan data **e.** Sistem menampilkan pesan data berhasil dihapus **4.6 Memperbarui Data Matriks** a. Koordinator memilih fungsi untuk mengelola data matriks dan operasi untuk memperbarui data matriks. b. Sistem meminta koordinator untuk memasukan kode blok dan kode subzona c. Koordinator memasukan kode blok dan kode subzona dan melakukan pencarian data matriks. d. Sistem menampilkan data matriks dan koordinator memilih data yang akan diperbarui e. Sistem menampilkan list data yang akan diperbarui dan meminta koordinator untuk memperbarui data matriks f. Koordinator memperbarui data matriks pada sistem dan meyimpan data tersebut. g. Sistem menampilkan pesan data berhasil diperbarui **4.7 Menghapus Data P,T,K** a. Koordinator memilih fungsi untuk mengelola data ketentuan P,T,K dan operasi untuk menghapus data ketentuan P,T,K b. Sistem menampilkan data berupa kode blok dan kode subzona yang terdapat pada sistem lalu koordinator memilih operasi untuk menghapus data c. Sistem menampilkan pesan konfirmasi penghapusan data dan koordinator melakukan konfirmasi penghapusan data d. Sistem menghapus data P,T,K dan menampilkan list data P,T,K **4.8 Memperbarui Data Ketentuan P,T,K** a. Koordinator memilih fungsi untuk mengelola data ketentuan P,T,K dan operasi untuk memperbarui data ketentuan P,T,K b. Sistem menampilkan data berupa kode blok dan kode subzona yang terdapat pada sistem lalu koordinator memilih operasi edit c. Sistem menampilkan daftar data yang akan diperbarui dan meminta koordinator untuk memperbarui data ketentuan P,T,K d. Koordinator memperbarui data ketentuan pada sistem dan meyimpan data tersebut.

Tabel 4.8 Spesifikasi *Use Case* Mengelola Data Peraturan Zonasi (lanjutan)

4. Subflows	e. Sistem menampilkan pesan data berhasil diperbarui **4.9 Menghapus Data Matriks** a. Koordinator memilih fungsi untuk mengelola data matriks dan operasi untuk menghapus data zonasi b. Sistem meminta koordinator untuk memasukan kode blok dan kode subzona c. Koordinator memasukan kode blok dan kode subzona dan melakukan pencarian data matriks. d. Sistem menampilkan data matriks dan koordinator memilih data yang akan dihapus e. Sistem menampilkan pesan konfirmasi penghapusan data f. Koordinator melakukan konfirmasi dan sistem menampilkan pesan data matriks berhasil dihapus
5. Key Scenarios	1. Proses mengelola data peraturan zonasi berhasil *: Basic flow* 2. Terdapat data kembar : *Basic flow*, Terdapat data yang sama 3. Tidak jadi mengelola data peraturan zonasi : *Basic flow*, Tidak jadi melakukan pengelolaan data
6. Pre-Condition	Koordinator mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan koneksi database sistem sudah tersedia
7. Post-Condition	Koordinator berhasil melakukan pengelolaan data yang meliputi penambahan atau pembaharuan data peraturan zonasi.
8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini

Sumber: Pertiwi (2016)

Pada tabel 4.9 merupakan penjelasan mengenai spesifikasi *use case* mengelola data pengguna:

Tabel 4.9 Spesifikasi *Use Case* Mengelola Data Pengguna

1. Brief Description	*Use case* ini menjelaskan mengenai bagaimana koordinator dapat mengelola data pengguna sistem yang meliputi pembuatan, pembaruan dan penghapusan data pengguna pada SILOKA.
2. Basic Flow of Events	1. *Use case* dimulai ketika koordinator membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 3. Koordinator masuk kedalam sistem dan memilih modul mengelola data pengguna

Tabel 4.9 Spesifikasi *Use Case* Mengelola Data Pengguna (lanjutan)

2. *Basic Flows of Events*	4. Sistem menampilkan data pengguna yang terdapat pada sistem dan fungsi yang tersedia untuk mengelola data pengguna. Fungsi tersebut meliputi membuat pengguna baru, memperbarui data pengguna dan menghapus pengguna dari sistem. **{Memilih operasi pengolahan data}** 5. Jika koordinator memilih operasi untuk membuat data pengguna, maka sub alur **Menambah Pengguna** akan dijalankan. 6. Jika koordinator memilih operasi untuk memperbarui data pengguna, maka sub alur **Memperbarui Data Pengguna** akan dijalankan 7. Jika koordinator memilih operasi untuk memperbarui *password* pengguna, maka sub alur **Memperbarui *Password*** akan dijalankan 8. Jika koordinator memilih operasi untuk menghapus data pengguna, maka sub alur **Menghapus Pengguna** akan dijalankan **{*Use case* selesai}** 9. *Use case* mengelola data pengguna selesai
3. *Alternative Flows*	3.1 Terdapat data yang belum diisi Pada **{Memilih fungsi pengolahan data}** di *basic flow* dan **{Mengisi data pengguna}** pada *subflow*, apabila koordinator tidak mengisi seluruh data maka sistem akan menampilkan pesan bahwa terdapat data yang belum diisi. 3.2 Terdapat data yang sama Pada **{Memilih fungsi pengolahan data}** di basic flow dan **{Mengisi data pengguna}** pada *subflow*, apabila nomor identitas yang dimasukan oleh koordinator sudah terdapat pada sistem maka sistem akan menampilkan pesan terdapat pengguna yang sama. Pengguna dapat melanjutkan pada **{Memilih fungsi pengolahan data}** 3.3 *Password* tidak sesuai Pada **{Memilih operasi pengolahan data}** di basic flow , apabila *password* lama yang dimasukan oleh koordinator tidak sesuai dengan yang terdaftar pada sistem, maka sistem akan menampilkan pesan bahwa *password* salah. Koordinator dapat memasukan *password* kembali atau tidak jadi memperbarui data *password* dan **{*Use Case* Selesai}** 3.4 Tidak jadi melakukan pengelolaan data Kapanpun pada alur *basic flow*, koordinator dapat tidak jadi melakukan pengelolaan data dan **{*Use Case* Selesai}**
4. *Subflow*	**4.1 Menambah Pengguna** a. Koordinator memilih operasi untuk menambahkan pengguna b. Sistem meminta agar koordinator mengisi data yang dibutuhkan untuk menambahkan pengguna sistem **{Mengisi data pengguna}**

Tabel 4.9 Spesifikasi *Use Case* Mengelola Data Pengguna (lanjutan)

4. Subflows	c. Koordinator mengisi data yang dibutuhkan dan menyimpan data untuk menambahkan pengguna pada sistem d. Sistem menampilkan pesan berhasil menambah pengguna baru **4.2 Memperbarui Data Pengguna** a. Koordinator memilih operasi untuk memperbarui data pengguna pada pengguna terdaftar dalam sistem b. Sistem menampilkan detail data pengguna yang akan diperbarui c. Koordinator memperbarui data pengguna pada sistem dan menyimpan hasil pembaruan d. Sistem menampilkan pesan data pengguna berhasil diperbarui **4.3 Memperbarui *Password*** a. Koordinator memilih operasi untuk memperbarui data pengguna pada sistem b. Sistem menampilkan detail data pengguna yang akan diperbarui dan opsi memperbarui *password* c. Koordinator memilih opsi memperbarui *password* d. Sistem meminta koordinator untuk memasukan *password* lama dan *password* baru e. Koordinator mengisi data yang dibutuhkan dan memperbarui data *password* pengguna pada sistem dengan menyimpan hasil pembaruan f. Sistem menampilkan pesan data pengguna berhasil diperbarui **4.4 Menghapus Pengguna** a. Koordinator memilih data yang akan dihapus dan sistem menampilkan pesan konfirmasi untuk menghapus data b. Koordinator melakukan konfirmasi penghapusan dan sistem menghapus data pengguna tersebut. c. Sistem menampilkan pesan berhasil menghapus data pengguna
5. Key Scenarios	*1.* Koordinator berhasil melakukan pengelolaan data : *Basic flow* 2. Terdapat data yang belum diisi : *Basic flow*, Terdapat data yang belum diisi 3. Nama pengguna sama : *Basic flow*, Terdapat data yang sama 4. Koordinator tidak jadi melakukan pengelolaan data : *Basic flow,* Tidak jadi melakukan pengelolaan data *Password* tidak sama : *Basic flow, password* tidak sesuai.
6. Pre-Condition	Koordinator mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan koneksi basis data sudah tersedia
7. Post-Condition	Koordinator berhasil melakukan pengelolaan data pengguna yang meliputi pembuatan, pembaruan atau penghapusan data akun pengguna pada sistem.

Tabel 4.9 Spesifikasi *Use Case* Mengelola Data Pengguna (lanjutan)

5. Extension Points	Tidak ada *special requirements* pada *use case* ini.
8. Special Requirements	Pembaruan data pengguna hanya dapat dilakukan pada pengguna selain pemohon izin.

Sumber: Pertiwi (2016)

Pada tabel 4.10 merupakan penjelasan mengenai spesifikasi *use case* mengelola jadwal:

Tabel 4.10 Spesifikasi *Use Case* Mengelola Jadwal

1. Brief Description	*Use case* ini menjelaskan mengenai bagaiman koordinator mengelola jadwal survey dan rapat yang meliputi penambahan serta pembaruan data jadwal survey dan rapat pada SILOKA
2. Basic Flow of Events	1. *Use case* dimulai ketika koordinator membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 3. Koodinator mengakses modul untuk mengelola jadwal survey dan rapat 4. Sistem menampilkan daftar berupa tanggal permohonan, nama pemohon, lokasi yang dimohon, fungsi yang tersedia untuk melihat detail data permohonan. 5. Sistem meminta koordinator melihat detail data permohonan pada pemohon izin tertentu. 6. Kooordinator melihat detail data permohonan 7. Sistem menampilkan detail data permohonan pemohon berupa nama pemohon, alamat pemohon, nomor telepon, nama perusahaan pemohon, lokasi, nama surveyor, jadwal pelaksanaan survey rapat dan fungsi yang tersedia untuk mengelola jadwal. Fungsi tersebut meliputi pembuatan dan pembaruan jadwal survey dan rapat **{Memilih operasi pengolahan data}** **8.** Jika koordinator memilih untuk membuat data jadwal survey dan rapat, maka sub alur **Menambah Jadwal** akan dijalankan **9.** Jika koordinator memilih untuk memperbarui data jadwal survey dan rapat, maka sub alur **Memperbarui Jadwal** akan dijalankan **{*Use case* selesai}** **10.** *Use case* selesai
3. Alternative Flows	3.1 Tidak jadi melakukan proses pengelolaan Kapanpun pada alur *basic flow*, koordinator dapat tidak jadi melakukan proses pengelolaan jadwal pada sistem dan **{*Use Case* Selesai}** 3.2 Jadwal yang sama Pada **{Memilih operasi pengolahan data}** di *basic flow*, apabila jadwal rapat yang dimasukan surveyor sama dengan jadwal yang sudah terdapat pada sistem, maka sistem akan menampilkan pesan notifikasi bahwa jadwal rapat sama.

Tabel 4.10 Spesifikasi *Use Case* Mengelola Jadwal (lanjutan)

3. Alternative Flows	3.3 Surveyor telah memiliki jadwal survey Pada **{Menambah Data Jadwal}** di *subflow* dan **{Mengisi data jadwal}** pada *extension point subflow*,apabila nama surveyor yang dimasukan telah memiliki jadwal survey yang sama dengan jadwal survey yang terdapat pada sistem, maka sistem akan menampilkan pesan notifikasi bahwa surveyor telah memiliki jadwal.
4. Subflow	**4.1 Menambah Jadwal** a. Surveyor memilih operasi untuk menambah data jadwal dan surveyor pada pemohon izin tertentu b. Sistem meminta agar koordinator mengisi data yang diperlukan untuk penambahan data jadwal **{Mengisi data jadwal}** c. Koordinator mengisi data yang diminta dan menyimpan data yang hendak ditambahkan d. Jadwal survey berhasil disimpan dan sistem menambahkan data progres e. Sistem menampilkan pesan berhasil menambah data jadwal **4.2 Memperbarui Jadwal** a. Surveyor memilih operasi untuk memperbarui data jadwal pada pemohon izin tertentu a. Sistem menampilkan detail data jadwal yang akan diperbarui b. Surveyor memperbarui data survey pada sistem dengan memasukan jadwal survey, rapat dan memasukan nama surveyor dan menyimpan data yang diperbarui c. Sistem menampilkan pesan berhasil memperbarui data survey
5. Key Scenarios	1. Proses pengelolaan jadwal berhasil : *Basic flow* 2. Terdapat jadwal rapat yang sama : *Basic flow*, Jadwal yang sama 3. Surveyor sudah memiliki jadwal : *Basic flow*, Surveyor telah memiliki jadwal survey 4. Tidak jadi melakukan pengelolaan : *Basic flow*, Tidak jadi melakukan proses pengelolaan
6. Pre-Condition	Koordinator mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, koneksi database sudah tersedia dan sudah mendapat laporan jadwal survey.
7. Post-Condition	Koordinator berhasil menambahkan data jadwal survey pada sistem
8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini

Sumber: Pertiwi (2016)

Pada tabel 4.11 merupakan penjelasan mengenai spesifikasi *use case* memverifikasi berkas

Tabel 4.11 Spesifikasi *Use Case* Memverifikasi Berkas

1. Brief Description	*Use case* ini menggambarkan bagaimana koordinator membuat daftar kelengkapan berkas apabila terdapat kekurangan atau kesalahan pada berkas yang diunggah pemohon izin pada SILOKA
2. Basic Flow of Events	1. *Use case* dimulai ketika koordinator membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 3. Koordinator mengakses modul untuk memverifikasi berkas 4. Sistem menampilkan list data pemohon yang belum melakukan survey dan operasi untuk menambah verifikasi berkas 5. Jika koordinator memilih untuk melihat berkas pemohon sebelum melakukan verifikasi, maka sub alur **Melihat Berkas** akan dijalankan 6. Koordinator memilih operasi menambah verifikasi berkas dan sistem meminta koordinator untuk mengisi daftar pengecekan dan catatan **{Membuat Verifikasi}** 7. Koordinator mengisi cheklist dan membuat catatan 8. Verifikasi berkasi berhasil disimpan dan sistem menambahkan data progres**.** 9. Sistem menampilkan pesan sukses membuat verifikasi berkas **{*Use case* selesai}** 10. *Use case* selesai
3. Alternative Flows	3.1 Tidak jadi melakukan proses verifikasi berkas Kapanpun pada alur *basic flow*, koordinator dapat tidak jadi melakukan proses membuat verifikasi berkas pada sistem dan **{*Use Case* Selesai}**
4. Subflow	**4.1 Melihat Berkas** a. Koordinator memilih operasi untuk melihat data berkas pada pemohon izin tertentuSistem menampilkan data pemohon dan list data berkas pemohon izin yang dipilih oleh koordinator b. Koordinator memilih berkas data tertentu c. Sistem menampilkan isi berkas pemohon izin sesuai dengan opsi berkas yang dipilih oleh koordinator
5. Key Scenarios	1. Koordinator berhasil membuat verifikasi berkas : *Basic flow* 2. Tidak jadi melakukan verifikasi : *Basic flow*, Tidak jadi melakukan proses verifikasi berkas
6. Pre-Condition	Koordinator mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan telah mendapat catatan disposisi serta melihat berkas pemohon izin.
7. Post-Condition	Koordinator berhasil memasukan verifikasi berkas pada sistem dan pemohon izin dapat melihat kekurangan berkas

Tabel 4.11 Spesifikasi *Use Case* Memverifikasi Berkas (lanjutan)

8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini

Sumber: Pertiwi (2016)

Pada tabel 4.12 merupakan penjelasan mengenai spesifikasi *use case* mengelola data pemohon:

Tabel 4.12 Spesifikasi *Use Case* Mengelola Data Pemohon

1. Brief Description	*Use case* ini menjelakan mengenai bagaimana pemohon izin dapat memperbarui biodata pribadi dan data *password*
2. Basic Flow of Events	1. *Use case* dimulai ketika pemohon izin membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 2. Pemohon izin masuk ke dalam sistem dan mengakses modul untuk mengelola data pemohon 3. Sistem menampilkan data berupa informasi pribadi dan fungsi yang tersedia untuk mengelola data pemohon. Fungsi tersebut meliputi memperbarui data pemohon dan memperbarui *password.* **{Memilih operasi pengolahan data}** 4. Jika pemohon izin memilih operasi untuk memperbarui data pemohon, maka sub alur **Memperbarui Data Pemohon** akan dijalankan. 5. Jika pemohon izin memilih operasi untuk memperbarui *password*, maka sub alur **Memperbarui *Password*** akan dijalankan. **{*Use Case* Selesai)** 6. *Use Case* melakukan pendaftaran selesai
3. Alternative Flows	1.1 Tidak jadi melakukan pengelolaan data Kapanpun pada alur *basic flow*, pemohon izin dapat tidak jadi melakukan pengelolaan data dan **{*Use Case* Selesai}** 1.2 Password tidak sesuai Pada **{Memperbarui *password*}** di *subflow* dan **{Mengisi *password*}** pada *extension point subflow*, apabila *password* lama yang dimasukan oleh pemohon izin tidak sesuai dengan yang terdaftar pada sistem, maka sistem akan menampilkan pesan bahwa *password* salah. Pemohon izin dapat memasukan *password* kembali atau tidak jadi memperbarui data dan **{*Use Case* Selesai}**
4. Subflow	**4.2 Memperbarui Data Pengguna** a. Pemohon izin memilih untuk melakukan pembaruan data pengguna

Tabel 4.12 Spesifikasi *Use Case* Mengelola Data Pemohon (lanjutan)

4. Subflows	b. Sistem menampilkan data dan meminta pemohon izin untuk mengisi data pemohon yang akan diperbarui. **{Mengisi Data Pembaruan}** c. Pemohon izin mengisi data pemohon yang hendak diperbarui. **{Melakukan Pembaruan}** d. Pemohon izin melakukan pembaruan data dengan menyimpan data yang sudah diperbarui pada sistem e. Sistem menampilkan pesan pembaruan data berhasil **4.2 Memperbarui *Password*** a. Pemohon izin memilih untuk melakukan pembaruan *password* b. Sistem meminta pemohon izin untuk mengisi *password* lama dan *password* baru **{Mengisi *Password*}** c. Pemohon izin mengisi *password* lama dan *password* yang hendak diperbarui. **{Melakukan Pembaruan}** d. Pemohon izin melakukan pembaruan data dengan menyimpan data yang sudah diperbarui pada sistem e. Sistem menampilkan pesan pembaruan data berhasil
5. Key Scenarios	1. Pemohon izin berhasil melakukan pengelolaan data : *Basic flow* 2. Tidak jadi mengelola data : *Basic flow*, Tidak jadi melakukan pengelolaan data. 3. *Password* Salah : *Basic flow*, *Password* tidak sesuai
6. Pre-Condition	Pemohon izin mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet.
7. Post-Condition	Pemohon izin berhasil melakukan pembaruan data terkait dengan identitas pemohon.
8. Extension Point	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini

Sumber: Pertiwi (2016)

Tabel 4.13 merupakan penjelasan mengenai spesifikasi *use case* memohon izin lokasi:

Tabel 4.13 Spesifikasi *Use Case* Memohon Izin Lokasi

1. Brief Description	*Use case* ini menjelaskan mengenai bagaimana pemohon izin dapat melakukan izin lokasi yang mencakup proses mengajukan permohonan, dan pengelolaan data berkas pada SILOKA.
2. Basic Flow of Events	1. *Use case* dimulai ketika pemohon izin membuka sistem 2. *Incluse use case* ***Login***

Tabel 4.13 Spesifikasi *Use Case* Memohon Izin Lokasi (lanjutan)

2. Basic Flow of Events	**{Mengakses Modul}** 3. Sistem menampilkan data permohonan izin lokasi pemohon dan fungsi untuk mengajukan permohonan serta melakukan proses izin lokasi *online* 4. Jika pemohon izin memilih melakukan pengajuan permohonan baru, maka sub alur **Mengajukan Permohonan** akan dijalankan. 5. Jika pemohon izin memilih melihat data pasca rapat pada permohonan tertentu , maka sub alur **Melihat Data Pasca Rapat** akan dijalankan 6. **{Memilih proses izin lokasi}** 7. Pemohon izin memilih untuk melakukan proses izin lokasi 8. Sistem meminta pemohon izin untuk melihat detail data permohonan 9. Pemohon izin memilih data permohonan izin lokasi dan sistem menampilkan data progress perizinan, data kelengkapan berkas, fungsi untuk mengelola data berkas dan melihat detail kelengkapan berkas 10. Jika pemohon izin memilih melihat detail kelengkapan berkas maka sub alur **Melihat Kelengkapan Berkas** akan dijalankan 11. Jika pemohon izin memilih operasi untuk membuat data berkas, maka sub alur **Menambah Data Berkas** akan dijalankan. 12. Jika pemohon izin memilih operasi untuk memperbarui data berkas maka sub alur **Memperbarui Data Berkas** akan dijalankan 13. **{*Use case* selesai}** 14. *Use case* selesai memohon izin lokasi selesai
3. Alternative Flows	1. Tidak terdapat proses perizinan Pada **{Memilih Proses Izin Lokasi}** di *basic flow*, apabila pemohon izin belum menyerahkan berkas maka sistem akan menampilkan pesan bahwa proses izin lokasi belum dilaksanakan. 2. Tidak jadi melakukan proses izin lokasi Kapanpun pada alur *basic flow*, pemohon izin dapat tidak jadi melakukan proses izin lokasi dan **{*Use Case* Selesai}** 3. Format berkas tidak sesuai Pada **{Membuat Data Berkas}** di *subflow* dan **{Memilih Berkas}** pada *extension point subflow*, apabila berkas yang diunggah oleh pemohon izin tidak sesuai dengan format maka sistem akan menampilkan pesan tidak berhasil mengunggah berkas.
4. Subflows	**4.3 Membuat Data Berkas** a. Pemohon izin memilih operasi untuk menambahkan data berkas pada sistem b. Sistem menampilkan daftar berkas yang harus diserahkan dan fungsi untuk mengunggah berkas

Tabel 4.13 Spesifikasi *Use Case* Memohon Izin Lokasi (lanjutan)

4. Subflows	**{Memilih berkas}** c. Pemohon izin memlih berkas yang akan diunggah dan mengunggah berkas kedalam sistem d. Sistem menyimpan data berkas dan menampilkan pesan berhasil mengunggah data berkas **4.2 Memperbarui Data Berkas** a. Pemohon izin memilih operasi untuk memperbarui data berkas pada sistem b. Sistem menampilkan daftar data berkas yang sudah diunggah c. Pemohon izin memilih berkas yang akan diperbarui d. Sistem meminta pemohon izin untuk mengunggah berkas baru **{Mengunggah Berkas}** e. Pemohon izin mengunggah berkas baru f. Sistem memperbarui data berkas dan menampilkan pesan berhasil memperbarui data berkas **4.3 Melihat Kelengkapan Berkas** a. Pemohon izin memilih operasi untuk melihat detail kekurangan berkas b. Sistem menampilkan data detail kelengkapan berkas dan pilihan untuk kembali melihat data proses izin lokasi c. Pemohon izin memilih untuk kembali dan melihat data proses izin lokasi **4.4 Melihat Data Pasca Rapat** a. Pemohon izin memilih operasi untuk melihat data hasil pasca rapat pada lokasi permohonan tertentu b. Sistem menampilkan hasil data pasca rapat berdasarkan data yang dipilih oleh pemohon izin **4.5 Mengajukan Permohonan** a. Pemohon izin memilih operasi untuk mengajukan permohonan izin lokasi online b. Sistem meminta pemohon izin untuk mengisi data yang dibutuhkan untuk mengajukan permohonan c. Pemohon izin mengisi data pada sistem dan melakukan pengajuan dengan menyimpan data permohonan d. Sistem menampilkan pesan berhasil mengajukan permohonan izin lokasi
5. Key Scenarios	1. Pemohon izin berhasil melakukan proses izin lokasi : *Basic flow* 2. Tidak terdapat proses perizinan : *Basic flow*, Tidak terdapat proses perizinan 3. Pemohon izin tidak jadi melakukan izin lokasi : *Basic flow*, Tidak jadi melakukan proses izin lokasi 3. Format berkas tidak sesuai : *Basic flow*, Format berkas tidak sesuai.

Tabel 4.13 Spesifikasi *Use Case* Memohon Izin Lokasi (lanjutan)

6. Pre-Condition	Pemohon izin mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan koneksi database sudah tersedia
7. Post-Condition	Pemohon izin berhasil melakukan proses permohonan izin lokasi yang meliputi mengajukan permohonan, melihat kekurangan berkas, melihat progress dan melakukan pengelolaan data berkas.
8. Extension Poin	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini
***Glossary* Memohon Izin Lokasi**	1. Data permohonan izin lokasi mencakup nama pemohon, alamat pemohon, nama perusahaan dan lokasi dimohon 2. Data Kelengkapan Berkas mencakup jenis berkas dan catatan kekurangan berkas 3. Data Berkas merupakan rincian formulir dan berkas yang harus diunggah kedalam sistem untuk melakukan proses izin lokasi. Data berkas terdiri dari formulir permohonan izin, surat pernyataan, Surat kuasa, Identitas diri, NPWP, Proposal rencana Kegiatan dan Pestetujuan Penanaman modal

Sumber: Pertiwi (2016)

Tabel 4.14 merupakan penjelasan mengenai spesifikasi *use case* mengelola data register

Tabel 4.14 Spesifikasi *Use Case* Mengelola Data Register

1. Brief Description	*Use case* ini menjelaskan mengenai bagaimana administrasi mengelola data register yang meliputi penambahan dan pembaruan data register pada SILOKA
2. Basic Flow of Events	1. *Use case* dimulai ketika administrasi membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 3. Administrasi masuk ke dalam sistem dan mengakses modul untuk mengelola data register 4. Sistem menampilkan list data berupa data register dan alamat dan fungsi yang tersedia untuk mengelola data register. **{Memilih Fungsi Pengolahan Data}** 5. Jika administrasi memilih operasi untuk membuat data register, maka sub alur **Menambah Data Register** akan dijalankan. 6. Jika administrasi memilih operasi untuk memperbarui data register, maka sub alur **Memperbarui Data Register** akan dijalankan 7. Jika administrasi memilih operasi untuk menghapus data register, maka sub alur **Menghapus Data Register** akan dijalankan **{*Use case* selesai}** 8. *Use case* mengelola data register selesai

Tabel 4.14 Spesifikasi *Use Case* Mengelola Data Register (lanjutan)

3. Alternative Flows	3.1 Terdapat data yang belum diisi Pada **{Memilih fungsi pengolahan data}** di *basic flow* dan **{Mengisi data regiter}** pada *subflow*, apabila administrasi tidak mengisi seluruh data yang dibutuhkan untuk membuat data register maka sistem akan menampilkan pesan bahwa terdapat data yang belum diisi. 3.2 Terdapat data yang sama Pada **{Memilih fungsi pengolahan data}** di *basic flow* dan **{Membuat data regiter}** pada *subflow*, apabila lokasi pemohon pada data register yang ditambahkan sudah terdapat pada sistem, maka sistem akan menampilkan pesan data sudah terdapat pada sistem. 3.3 Tidak jadi melakukan pengelolaan data Kapanpun pada alur *basic flow,* administrasi dapat tidak jadi melakukan pengelolaan data dan **{Use Case Selesai}**
4. Subflow	**4.1 Membuat data register** a. Administrasi memilih operasi untuk menambahkan data register b. Sistem meminta administrasi untuk mengisi data yang dibutuhkan untuk penambahan data register **{Mengisi data register}** c. Administrasi mengisi data yang dibutuhkan dan menyimpan data register d. Data register berhasil di simpan dan sistem menambahkan data progres e. Sistem menampilkan pesan berhasil membuat data register **3.2 Memperbarui data register** a. Sistem meminta administrasi untuk melihat detail data register pemohon izin b. Administrasi melihat detail data register dan memilih operasi untuk memperbarui data tertentu c. Sistem meminta administrasi untuk mengisi data pembaruan data register d. Administrasi memperbarui data register pada sistem dan menyimpan hasil pembaruan pada sistem **a.** Sistem menampilkan pesan berhasil memperbarui data register. **3.3 Menghapus data register** a. Administrasi memilih data yang akan dihapus dan sistem menampilkan pesan konfirmasi untuk menghapus data b. Administrasi melakukan konfirmasi penghapusan dan sistem menghapus data register tersebut c. Sistem menampilkan pesan berhasil menghapus data register.

Tabel 4.14 Spesifikasi *Use Case* Mengelola Data Register (lanjutan)

5. Key Scenarios	1. Administrasi berhasil melakukan pengelolaan data *: Basic flow* 2. Terdapat data yang belum diisi : *Basic flow*, Terdapat data yang belum diisi 3. Data registrasi kembar *: Basic flow*, Terdapat data yang sama 4. Administrasi tidak jadi melakukan pengelolaan data : *Basic flow*, Tidak jadi melakukan pengelolaan data
6. Pre-Condition	Administrasi mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan koneksi database sudah tersedia
7. Post-Condition	Administrasi berhasil melakukan pengelolaan data yang meliputi penambahan data register atau pembaruan data register.
8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini
***Glossary* Data Register**	Data register merupakan data yang dibutuhkan untuk membuat catatan register terkait dengan permohonan izin lokasi. Data register terdiri dari No. Registrasi, Tanggal Masuk, Nama Pemohon, Alamat Pemohon, Jenis Peruntukan, dan No.SK jadi

Sumber: Pertiwi (2016)

Pada tabel 4.15 merupakan penjelasan mengenai spesifikasi *use case* membuat lembar disposisi

Tabel 4.15 Spesifikasi *Use Case* Membuat Lembar Disposisi

1. Brief Description	*Use case* ini menjelaskan mengenai bagaimana tata usaha membuat lembar disposisi pada SILOKA
2. Basic Flow of Events	1. *Use case* dimulai ketika tata usaha membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 3. Tata usaha masuk ke dalam sistem dan dan mengakses modul lembar disposisi 4. Sistem menampilkan daftar data permohonan izin yang terdapat pada sistem dan pilihan fungsi untuk membuat lembar disposisi. Fungsi tersebut meliputi pembuatan lembar disposisi dan mencetak lembar disposisi **{Memilih operasi}** 5. Jika tata usaha memilih operasi untuk membuat lembar disposisi, maka sub alur **Membuat Lembar Disposisi** akan dijalankan. 6. Jika tata usaha memilih operasi untuk mencetak lembar disposisi, maka sub alur **Mencetak Lembar Disposisi** akan dijalankan. **{Use Case Selesai}** 7. Use Case melakukan membuat lembar disposisi pendaftaran selesai
3. Alternative Flows	3.1 Terdapat data yang belum diisi

Tabel 4.15 Spesifikasi *Use Case* Membuat Lembar Disposisi (lanjutan)

3. Alternative Flows	Pada **{Membuat Lembar Disposisi}** di *subflow* dan **{Mengisi Data Disposisi}** pada *extension point subflow*, apabila tata usaha tidak mengisi pada seluruh data yang dibutukan maka sistem akan menampilkan pesan untuk mengisi seluruh data dan melanjutkan basic flow pada **{Mengisi data Disposisi}.** 3.2 Tidak jadi membuat lembar disposisi 3.3 Kapanpun pada alur *basic flow*, tata usaha dapat tidak jadi membuat lembar disposisi dan **{*Use Case* Selesai}**
4. Subflow	**4.1 Membuat Lembar Disposisi** a. Tata usaha memilih untuk membuat lembar disposisi b. Sistem meminta tata usaha untuk mengisi data yang dibutuhkan untuk pembuatan lembar disposisi **{Mengisi Data Disposisi}** c. Tata usaha mengisi data disposisi melalui sistem. **{Melakukan Pembuatan}** d. Tata usaha melakukan pembuatan lembar disposisi dengan menyimpan data disposisi yang sudah diisi pada sistem e. Lembar disposisi berhasil disimpan dan sistem menambahkan data progres f. Sistem menampilkan pesan berhasil membuat lembar disposisi dan mengirimkan notifikasi kepada pimpinan instansi **4.2 Mencetak Lembar Disposisi** a. Tata usaha memilih untuk mencetak lembar disposisi b. Sistem meminta tata usaha untuk memilih lembar disposisi yang akan dicetak c. Tata usaha memilih lembar disposisi yang akan dicetak. d. Sistem mencetak lembar disposisi sesuai dengan pilihan tata usaha
5. Key Scenarios	**1.** Tata usaha berhasil melakukan pembuatan disposisi : *Basic flow* **2.** Terdapat formulir yang belum diisi *: Basic flow*, Terdapat data yang belum diisi, *Basic flow*. **3.** Tidak jadi melakukan pembuatan disposisi : *Basic flow*, Tidak jadi membuat lembar disposisi
6. Pre-Condition	Tata usaha mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan koneksi database sudah tersedia
7. Post-Condition	Tata usaha berhasil membuat catatan disposisi dan catatan ditambahkan pada sistem.
8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini
***10. Glossary* Data Disposisi**	Data disposisi meliputi data yang dibutuhkan untuk membuat lembar dispoisisi. Data disposisi meliputi nomor agenda, tujuan lembar disposisi dan tindakan

Sumber: Pertiwi (2016)

Pada tabel 4.16 merupakan penjelasan mengenai spesifikasi *use case* membuat catatan disposisi:

Tabel 4.16 Spesifikasi *Use Case* Membuat Catatan Disposisi

1. *Brief Description*	*Use case* ini menjelaskan mengenai bagaimana pimpinan instansi dapat membuat catatan disposisi apabila terdapat kekurangan atau kesalahan pada berkas pada SILOKA
2. *Basic Flow of Events*	1. *Use case* dimulai ketika pimpinan instansi membuka sistem 2. *Include use case* ***Login*** **{Mengakses Notifikasi}** 3. Pimpinan instansi masuk kedalam sistem dan melihat notifikasi 4. Sistem menampilkan list lembar disposisi, data berkas pemohon dan fungsi melihat lembar disposisi serta berkas 5. Jika pimpinan instansi memilih operasi untuk melihat berkas maka sub alur **Melihat berkas** akan dijalankan 6. Pimpinan instansi melihat detail lembar disposisi 7. Sistem menampilkan fungsi untuk menambahkan catatan disposisi **{Memilih membuat catatan disposisi}** 8. Pimpinan instansi memilih menambahkan catatan disposisi dan sistem meminta pimpinan instansi untuk mengisi catatan yang dibutuhkan dalam pembuatan catatan disposisi. **{Mengisi form catatan}** 9. Pimpinan instansi mengisi data yang dibutuhkan untuk membuat catatan disposisi **{Menyimpan data catatan}** 10. Pimpinan instansi menyimpan catatan disposisi pada sistem 11. Catatan disposisi berhasil disimpan dan sistem menambahkan data progres 12. Sistem menampilkan pesan berhasil membuat catatan disposisi **{*Use case* selesai}** 13. *Use case* membuat catatan disposisi selesai
3. *Alternative Flows*	3.1 Terdapat catatan yang belum diisi Pada **{Menyimpan data catatan}** di *basic flow*, apabila pimpinan instansi belum mengisi seluruh data yang dibutuhkan pada formulir penambahan catatan maka sistem akan menampilkan pesan data belum terisi dan melanjutkan *basic flow* pada **{Mengisi form catatan}**. 3.2 Tidak jadi menambahkan catatan Kapanpun pada alur *basic flow*, pimpinan instansi atau koordinator dapat tidak jadi melakukan penambahan catatan dan ***{Use Case* Selesai}**
4. *Subflow*	4.1 Melihat Berkas a. Sistem meminta pimpinan instansi untuk memilih melihat berkas permohonan tertentu b. Pimpinan instansi memilih melihat berkas pemohon tertentu

Tabel 4.16 Spesifikasi *Use Case* Membuat Catatan Disposisi (lanjutan)

4.	**c.** Sistem menampilkan data berkas dan pimpinan memilih kembali untuk membuat catatan
5. Key Scenarios	1. Pimpinan instansi berhasil melakukan pengelolaan data : *Basic flow* 2. Terdapat data yang belum diisi *: Basic flow*, Terdapat catatan yang belum diisi 3. Tidak jadi melakukan penambahan catatan : *Basic flow,* Tidak jadi menambahkan catatan
6. Pre-Condition	Pimpinan instansi mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan koneksi database sudah tersedia
7. Post-Condition	Pimpinan instansi berhasil membuat catatan disposisi melalui sistem.
8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Pembuatan catatan disposisi dilakukan secara berurutan dari kepala dinas, kepala bidang dan kepala seksi

Sumber: Pertiwi (2016)

Pada tabel 4.17 merupakan penjelasan mengenai spesifikasi *use case* mengelola data survey:

Tabel 4.17 Spesifikasi *Use Case* Mengelola Data Survey

1. Brief Description	*Use case* ini menjelaskan mengenai bagaimana surveyor dapat mengelola data survey yang meliputi pembuatan, dan pembaruan data drive dan peta berdasarkan hasil survey lokasi pada SILOKA
2. Basic Flow of Events	1. *Use case* dimulai ketika surveyor membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 3. Surveyor masuk kedalam sistem kemudian mengakses modul untuk mengelola data survey. 4. Sistem menampilkan data berupa nama pemohon, jadwal survey, status dan meminta surveyor untuk melihat detail data survey 5. Surveyor melihat detail data survey dan sistem menampilkan pilihan fungsi yang tersedia untuk mengelola data survey. Fungsi tersebut meliputi membuat data survey dan memperbarui data survey. 6. **{Memilih operasi pengolahan data}** 7. Jika surveyor memilih operasi untuk membuat data survey, maka sub alur **Membuat Data Survey** akan dijalankan. 8. Jika surveyor memilih operasi untuk memperbaharui data survey, maka sub alur **Memperbarui Data Survey** akan dijalankan **{*Use case* selesai}**

Tabel 4.17 Spesifikasi *Use Case* Mengelola Data Survey (lanjutan)

2. Basic Flow of Events	**9.** *Use case* mengelola data survey selesai
3. Alternative Flows	3.1 Terdapat data yang belum diisi Pada **{Membuat Data Survey}** di *subflow* dan **{Mengisi Data Survey}** pada *extension point subflow*, apabila surveyor tidak mengisi seluruh data yang diminta maka sistem akan menampilkan pesan bahwa terdapat data yang belum diisi. 3.2 Tidak jadi melakukan pengelolaan data Kapanpun pada alur *basic flow*, surveyor dapat tidak jadi melakukan pengelolaan data dan **{*Use Case* Selesai}**
4. Subflow	**4.1 Membuat Data Survey** a. Sistem meminta surveyor untuk melihat data detail permohonan b. Surveyor memilih melihat detail permohonan dan memilih operasi untuk menambahkan data survey pada pemohon izin tertentu c. Sistem meminta agar mengunggah data yang diperlukan untuk penambahan data survey **{Mengisi data survey}** d. Surveyor mengunggah data hasil survey e. Sistem menampilkan pesan berhasil membuat data survey **4.2 Memperbarui Data Survey** a. Sistem meminta surveyor untuk melihat data detail permohonan b. Surveyor memilih melihat detail permohonan dan Sistem menampilkan detail data survey yang akan diperbarui c. Surveyor memilih operasi untuk memperbarui data survey pada pemohon izin tertentu d. Surveyor memperbarui data survey pada sistem dan memyimpan hasil pembaruan pada sistem e. Sistem menampilkan pesan berhasil memperbarui data survey
5. Key Scenarios	1. Surveyor berhasil melakukan pengelolaan data *: Basic flow* 2. Terdapat data yang belum diisi : *Basic flow*, Terdapat data yang belum diisi 3. Surveyor tidak jadi melakukan pengelolaan data : *Basic flow*, Tidak jadi melakukan pengelolaan data
6. Pre-Condition	Surveyor mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, dan koneksi database sudah tersedia
7. Post-Condition	Surveyor berhasil melakukan pengelolaan data yang meliputi penambahan data survey atau pembaruan data survey.
8. Extension Points	Tidak ada *extension points* pada *use case* ini
9. Special Requirements	Tidak ada *special requirements* pada *use case* ini

Tabel 4.17 Spesifikasi *Use CaseI* Mengelola Data Survey (lanjutan)

***Glossary* Data Survey**	Data survey merupakan informasi berupa drive dan peta terkait dengan hasil survey. Data Survey terdiri dari peta pertimbangan teknis pertanahan, foto peninjauan lokasi, peta izin lokasi dan laporan penelitian lapangan

Sumber: Pertiwi (2016)

Pada tabel 4.18 merupakan penjelasan mengenai spesifikasi menambah data pasca rapat:

Tabel 4.18 Spesifikasi *Use Case* Mengelola Data Pasca Rapat

1. ***Brief Description***	*Use case* ini menjelaskan mengenai bagaimana koordinator menggunakan SILOKA untuk mengelola data pasca rapat yang meliputi penambahan dan pembaruan data pada pemohon yang sudah melakukan rapat pada sistem.
2. ***Basic Flow of Events***	1. *Use case* dimulai ketika surveyor membuka sistem 2. *Include use case* ***Login*** **{Mengakses Modul}** 3. Surveyor masuk ke dalam sistem dan mengakses modul untuk mengelola data pasca rapat 4. Sistem menampilkan data berupa nama pemohon, dan lokasi permohonan. Sistem meminta surveyor untuk melihat detail data pada pemohon izin tertentu. 5. Surveyor melihat detail data pada pemohon izin tertentu 6. Sistem menampilkan pilihan fungsi yang tersedia untuk mengelola data pasca rapat, yaitu menambah data dan memperbarui data pasca rapat. **{Memilih operasi perngolahan data}** 7. Jika surveyor memilih operasi untuk menambah data pasca rapat, maka sub alur **Menambah Data Pasca Rapat** akan dijalankan 8. Jika surveyor memilih operasi untuk memperbarui data pasca rapat, maka sub alur **Memperbarui Data Pasca Rapat** akan dijalankan **{*Use case* selesai}** 9. *Use case* selesai
3. ***Alternative Flows***	3.1 Tidak jadi melakukan pengelolaan data Kapanpun pada alur *basic flow*, pemohon izin dapat tidak jadi melakukan pengelolaan data pasca rapat dan **{*Use Case* Selesai}** 3.2 Format berkas tidak sesuai Pada **{Mengunggah berkas}** di *basic flow* , apabila berkas yang diunggah oleh surveyor tidak sesuai dengan format maka sistem akan menampilkan pesan tidak berhasil mengunggah berkas.
4. ***Subflow***	**4.1 Menambah Data Pasca Rapat** a. Surveyor memilih operasi untuk menambahkan data pasca rapat pada pemohon izin tertentu

Tabel 4.19 Spesifikasi *Use Case* Mengelola Data Pasca Rapat (lanjutan)

4. ***Subflows***	b. Sistem meminta surveyor untuk mengunggah berkas laporan pasca rapat **{Mengunggah Berkas}** c. Surveyor memilih berkas yang akan diunggah dan mengunggah berkas ke dalam sistem. d. Sistem menyimpan penambahan data dan menampilkan pesan berhasil mengunggah berkas pasca rapat **4.2 Memperbarui Data Pasca Rapat** a. Surveyor memilih operasi untuk memperbarui data pasca rapat pada pemohon izin tertentu b. Sistem meminta surveyor untuk memperbarui berkas laporan pasca rapat dengan mengunggah berkas baru. c. Surveyor memperbarui data pasca rapat pada sistem dan menyimpan hasil pembaruan pada sistem d. Sistem menampilkan pesan berhasil memperbarui data data pasca rapat
5. ***Key Scenarios***	1. Surveyor berhasil mengelola data berkas : *Basic flow* 2. Fortmat berkas yang diunggah tidak sesuai : *Basic flow*, Format berkas tidak sesuai
6. ***Pre-Condition***	Surveyor mengakses SILOKA menggunakan perangkat yang memiliki koneksi dengan internet, koneksi database sudah tersedia dan sudah melaksanakan rapat.
7. ***Post-Condition***	Surveyor berhasil menambahkan berkas pasca rapat pada sistem
8. ***Extension Points***	Tidak ada *extension points* pada *use case* ini
9. ***Special Requirements***	Tidak ada *special requirements* pada *use case* ini

Sumber: Pertiwi (2016)

4.2 Perubahan Perancangan ke Implementasi

Pada penelitian sebelumnya, telah dilakukan perancangan SILOKA dengan pendekatan berorientasi objek. Perancangan yang dilakukan mulai dari perancangan kelas analisis, kelas desain, kelas diagram dan diagram *sequence*. Pada diagram kelas implementasi dibutuhkan penyesuaian dengan *framework* CodeIgniter.

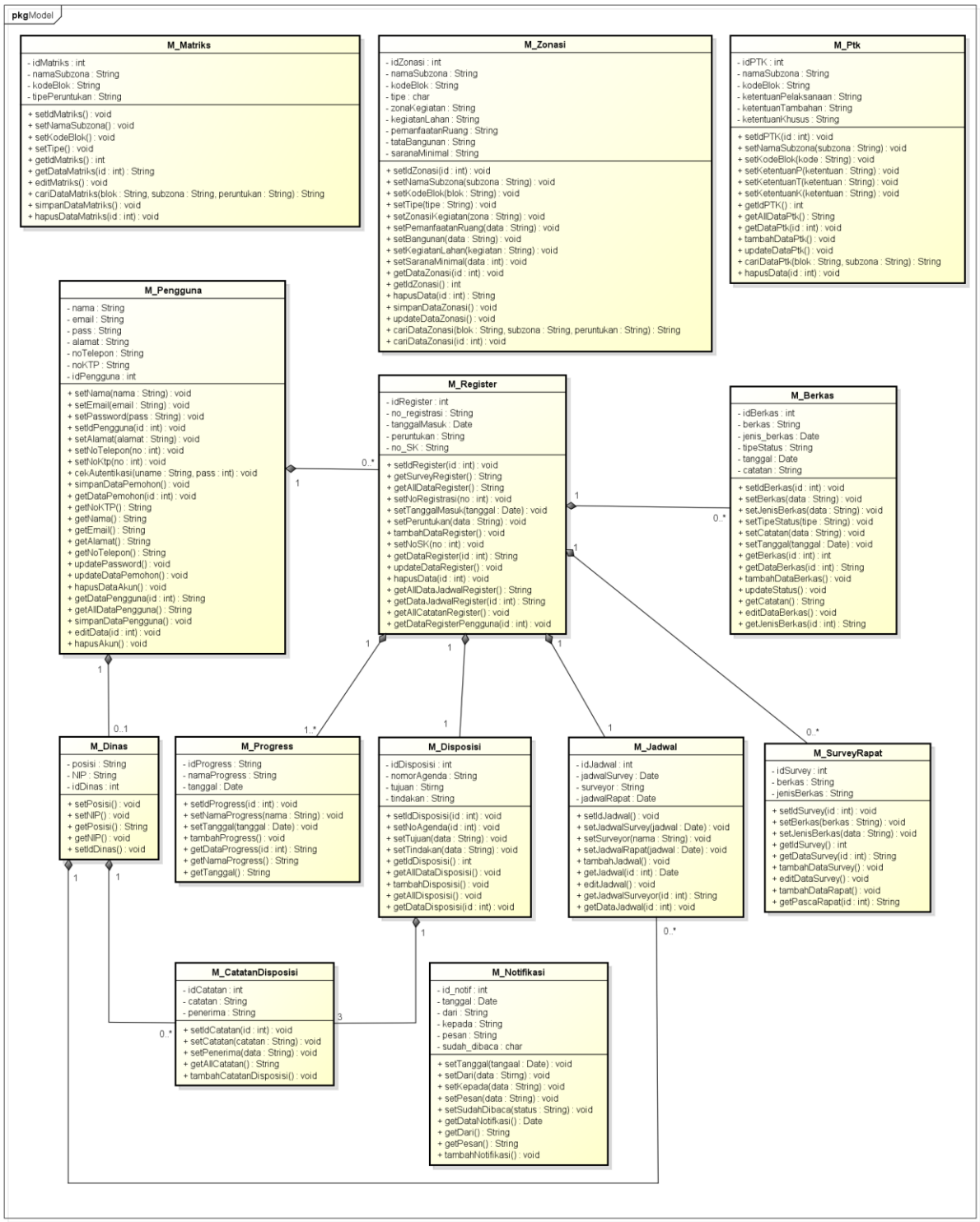
4.2.1 Diagram Kelas

Berikut ini merupakan diagram kelas perancangan dari penelitian sebelumnya dan diagram kelas implementasi yang sudah menyesuaikan dengan *framework* codeigniter serta penjelasan kelas.

4.2.1.1 Diagram Kelas Perancangan

Pada penelitian sebelumnya telah dirancang diagram kelas dengan konsep MVC (*Model*, *View, Controller*) secara umum. Pada diagram kelas dibagi menjadi

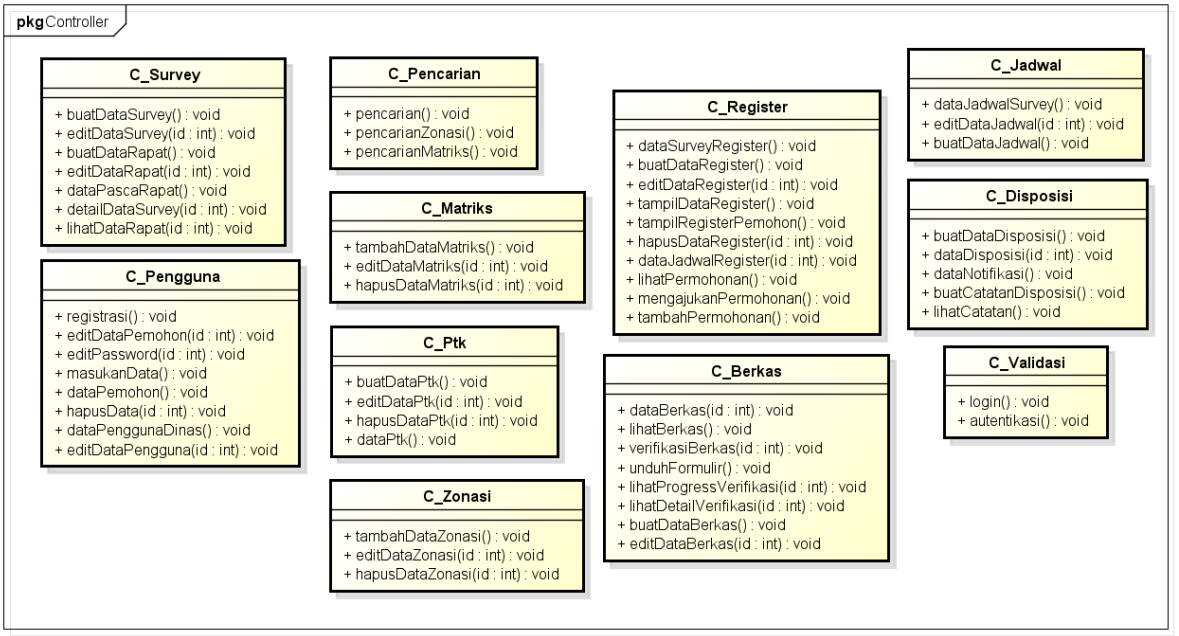
tiga bagian, yaitu *model, controller* serta gabungan *model* dan *controller*. *Model* merupakan bagian yang menangani hal yang berhubungan dengan basis data. Gambar 4.2 menunjukkan *model* dari diagram kelas SILOKA.



Gambar 4.2 *Model* dari Diagram Kelas Perancangan

Sumber: Pertiwi (2016)

Controller merupakan kumpulan instruksi yang menghubungkan antara *model* dan *view*. Gambar 4.3 menunjukkan *controller* dari diagram kelas perancangan.

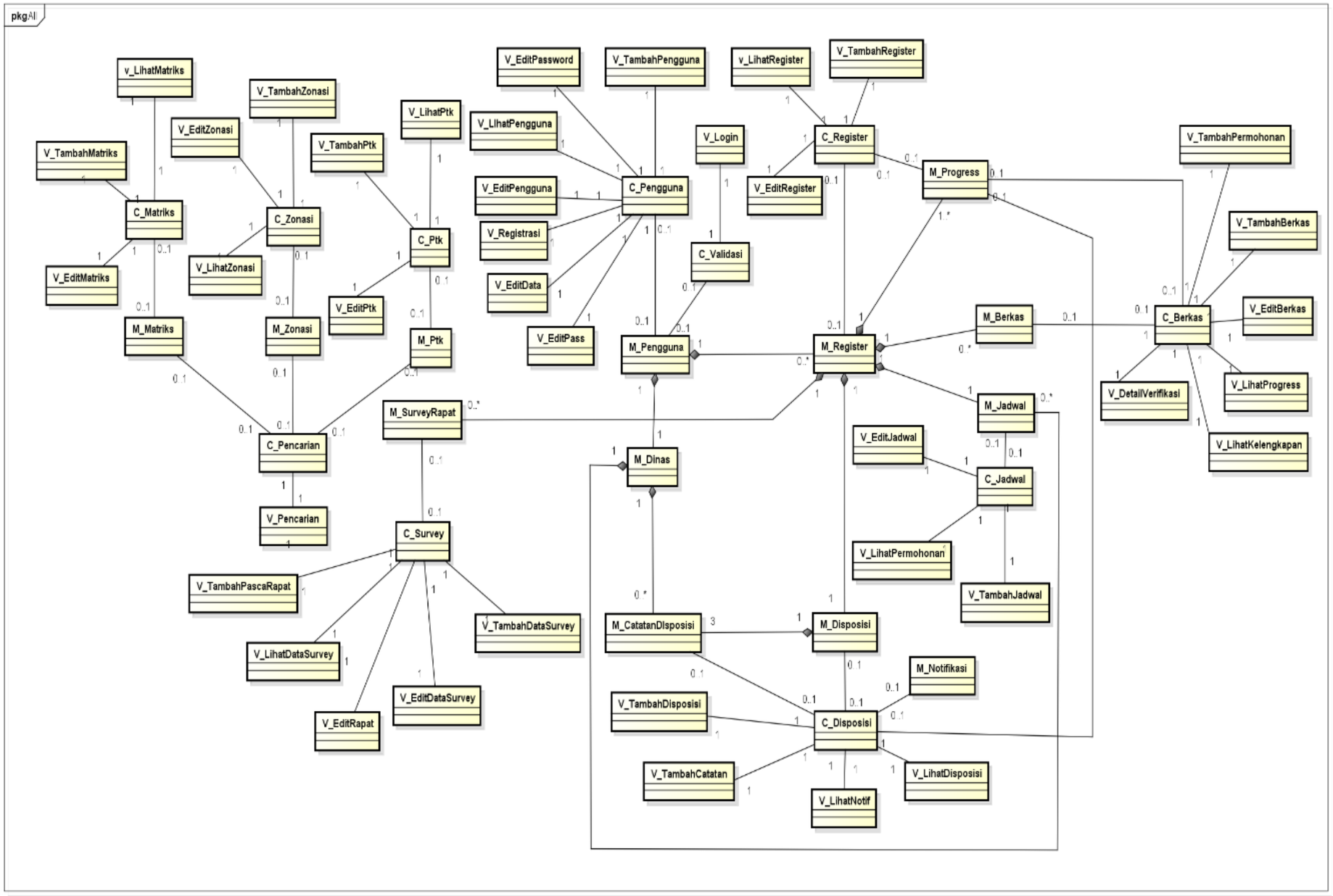


Gambar 4.3 *Model* dari diagram kelas SILOKA

Sumber : Pertiwi (2016)

Dari *controller* dan *model* pada gambar 4.2 dan 4.3 digabungkan menjadi diagram kelas perancangan. Gabungan diagram kelas perancangan keseluruhan dapat dilihat pada gambar 4.4.

repository.ub.ac.id



Gambar 4.4 Diagram kelas perancangan keseluruhan
Sumber : Pertiwi (2016)

Penjelasan Diagram Kelas Perancangan

Dari diagram kelas perancangan terdapat penjelasan di setiap kelas. Berikut merupakan penjelasan dari setiap kelas dari diagram kelas perancangan (Pertiwi, 2016)

Tabel 4.19 menjelaskan mengenai deskripsi kelas C_Zonasi

Tabel 4.20 Deskripsi kelas C_Zonasi

Nama Kelas	C_Zonasi
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan data zonasi dan data P, T, K dalam melakukan pengelolaan data zonasi

Sumber : Pertiwi, (2016)

Tabel 4.20 menjelaskan mengenai deskripsi kelas C_Matriks

Tabel 4.21 Deskripsi kelas C_Matriks

Nama Kelas	C_Matriks
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan data matriks dalam melakukan pengelolaan data matriks yang digunakan pada pencarian zonasi

Sumber : Pertiwi, (2016)

Tabel 4.21 menjelaskan mengenai deskripsi kelas C_Pengguna

Tabel 4.22 Deskripsi kelas C_Pengguna

Nama Kelas	C_Pengguna
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan akun pengguna terkait dengan data pengguna pada sistem

Sumber : Pertiwi, (2016)

Tabel 4.22 menjelaskan mengenai deskripsi kelas C_Pencarian

Tabel 4.23 Deskripsi kelas C_Pencarian

Nama Kelas	C_pencarian
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan pencarian data peraturan zonasi

Sumber : Pertiwi, (2016)

Tabel 4.23 menjelaskan mengenai deskripsi kelas C_Register

Tabel 4.24 Deskripsi kelas C_Register

Nama Kelas	C_Register

Tabel 4.23 Deskripsi kelas C_Register (Lanjutan)

Deskripsi	**Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan pengelolaan data register oleh dinas**

Sumber : Pertiwi, (2016)

Tabel 4.24 menjelaskan mengenai deskripsi kelas C_Survey

Tabel 4.25 Deskripsi kelas C_Survey

Nama Kelas	C_Survey
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan pengelolaan data hasil survey dan pasca rapat izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.25 menjelaskan mengenai deskripsi kelas C_Berkas

Tabel 4.26 Deskripsi kelas C_Berkas

Nama Kelas	C_Berkas
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan pengelolaan data berkas pemohon untuk melakukan proses izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.26 menjelaskan mengenai deskripsi kelas C_Jadwal

Tabel 4.27 Deskripsi kelas C_Jadwal

Nama Kelas	C_Jadwal
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan data jadwal pelaksanaan survey

Sumber : Pertiwi, (2016)

Tabel 4.27 menjelaskan mengenai deskripsi kelas C_Disposisi

Tabel 4.28 Deskripsi kelas C_Disposisi

Nama Kelas	C_Disposisi
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan data disposisi dan catatan disposisi izin lokasi.

Sumber : Pertiwi, (2016)

Tabel 4.28 menjelaskan mengenai deskripsi kelas C_Validasi

Tabel 4.29 Deskripsi kelas C_Validasi

Nama Kelas	C_Validasi
Deskripsi	Kelas yang menjembatani antara entity dan boundary yang berkaitan dengan validasi data untuk autentikasi pengguna

Sumber : Pertiwi, (2016)

Tabel 4.29 menjelaskan mengenai deskripsi kelas M_Matriks

Tabel 4.30 Deskripsi kelas M_Matriks

Nama Kelas	M_Matriks
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data matriks untuk proses pencarian zonasi pada izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.30 menjelaskan mengenai deskripsi kelas M_Notifikasi

Tabel 4.31 Deskripsi kelas M_Notifikasi

Nama Kelas	M_Notifikasi
Deskripsi	Kelas yang berkaitan dengan pembuatan notifikasi terkait dengan lembar dan catatan disposisi

Sumber : Pertiwi, (2016)

Tabel 4.31 menjelaskan mengenai deskripsi kelas M_Zonasi

Tabel 4.32 Deskripsi kelas M_Zonasi

Nama Kelas	M_Zonasi
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data zonasi untuk proses pencarian zonasi pada izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.32 menjelaskan mengenai deskripsi kelas M_Ptk

Tabel 4.33 Deskripsi kelas M_Ptk

Nama Kelas	M_Ptk
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data ptk untuk proses pencarian zonasi pada izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.33 menjelaskan mengenai deskripsi kelas M_Akun

Tabel 4.34 Deskripsi kelas M_Akun

Nama Kelas	M_Akun
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data pengguna terdaftar pada sistem

Sumber : Pertiwi, (2016)

Tabel 4.34 menjelaskan mengenai deskripsi kelas M_Register

Tabel 4.35 Deskripsi kelas M_Register

Nama Kelas	M_Register

Tabel 4.34 Deskripsi kelas M_Register (Lanjutan)

Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data register izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.35 menjelaskan mengenai deskripsi kelas M_SurveyRapat

Tabel 4.36 Deskripsi kelas M_SurveyRapat

Nama Kelas	M_SurveyRapat
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data hasil survey dan rapat perizinan

Sumber : Pertiwi, (2016)

Tabel 4.36 menjelaskan mengenai deskripsi kelas M_Berkas

Tabel 4.37 Deskripsi kelas M_Berkas

Nama Kelas	M_Berkas
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data berkas yang diperlukan untuk melakukan proses izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.37 menjelaskan mengenai deskripsi kelas M_Jadwal

Tabel 4.38 Deskripsi kelas M_Jadwal

Nama Kelas	M_Jadwal
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data jadwal pelaksanaan survey

Sumber : Pertiwi, (2016)

Tabel 4.38 menjelaskan mengenai deskripsi kelas M_Disposisi

Tabel 4.39 Deskripsi kelas M_Disposisi

Nama Kelas	M_Disposisi
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data disposisi permohonan izin lokasi

Sumber : Pertiwi, (2016)

Tabel 4.39 menjelaskan mengenai deskripsi kelas M_CatatanDisposisi

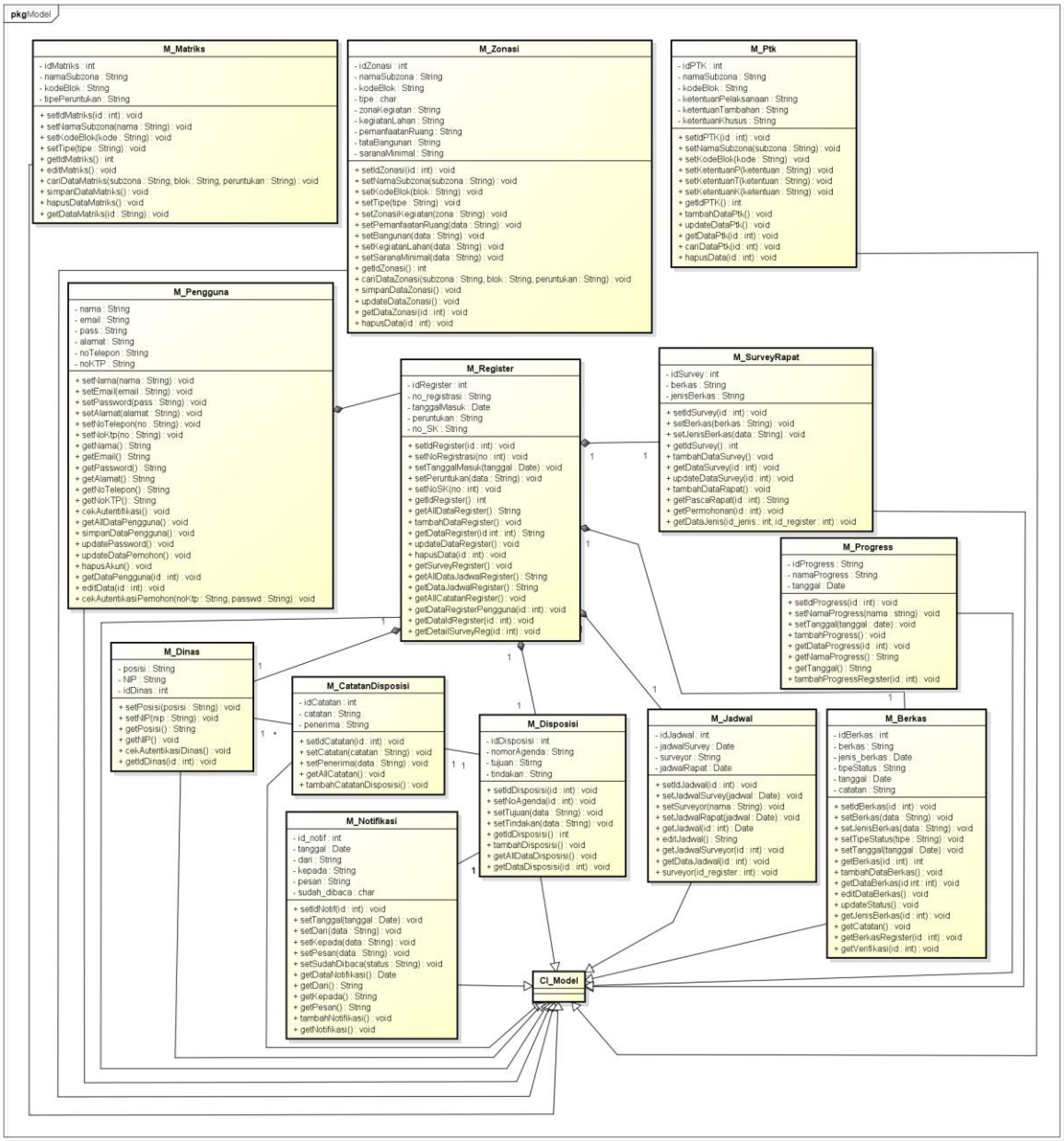
Tabel 4.40 Deskripsi kelas M_CatatanDisposisi

Nama Kelas	M_CatatanDisposisi
Deskripsi	Kelas yang berkaitan dengan penyimpanan dan pengelolaan data catatan disposisi permohonan izin lokasi

Sumber : Pertiwi, (2016)

4.2.1.2 Diagram Kelas Implementasi

Pada penelitian ini diagram kelas telah menyesuaikan dengan *framework* yang digunakan untuk implementasi SILOKA, yaitu CodeIgniter. Diagram kelas implementasi dipisahkan menjadi tiga bagian, yaitu *controller*, *model* serta gabungan dari *controller* dan *model.* Gambar 4.5 merupakan *model* diagram kelas implementasi dari SILOKA.

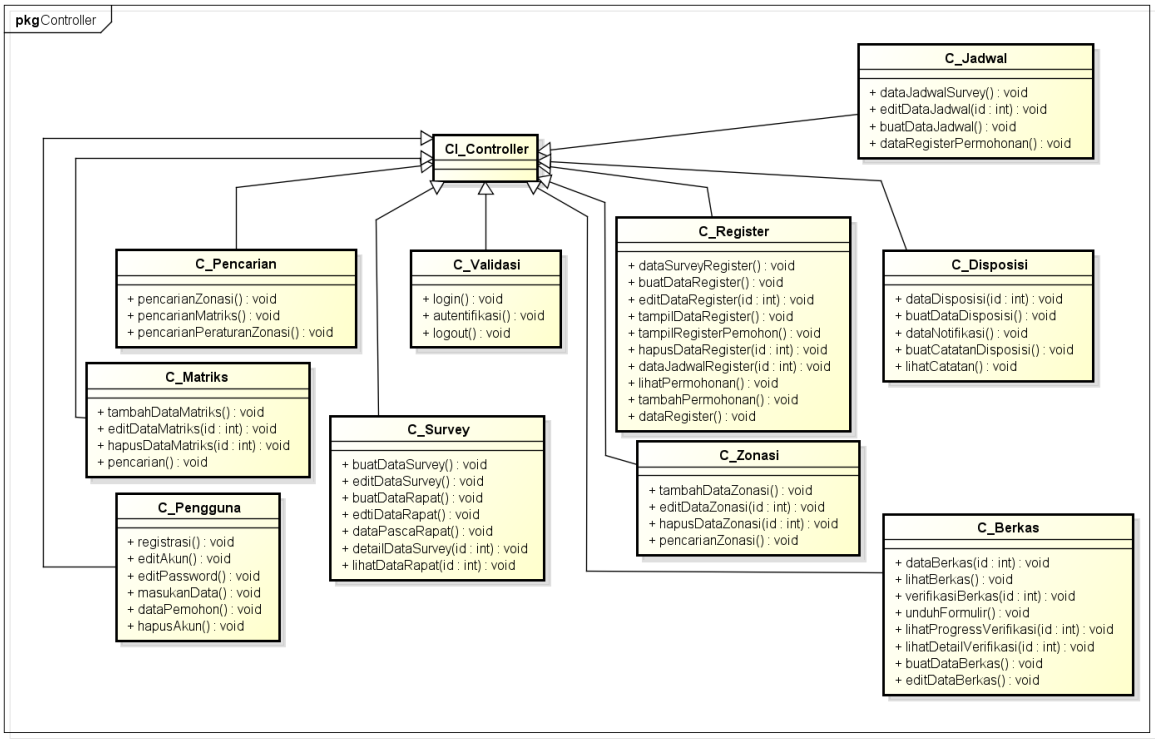


Gambar 4.5 *Model* diagram kelas implementasi

Pada *model* diagram kelas implementasi terdapat penambahan kelas CI_Model, dimana semua kelas *model* diturunkan dari CI_Model. Selain itu, juga terdapat penambahan *method*, yaitu pada kelas M_Disposisi ditambahkan

method getDisposisiRegister(id). *Method* getDisposisiRegister(id) digunakan untuk memperoleh data permohonan dari pemohon izin ketika tata usaha akan menambah lembar disposisi. Pada kelas M_Jadwal terdapat penambahan *method* surveyor($id_register). *Method* surveyor(id_register) digunakan untuk memperoleh data surveyor ketika koordinator melihat detail jadwal, karena data surveyor harus ditampilkan dengan perulangan.

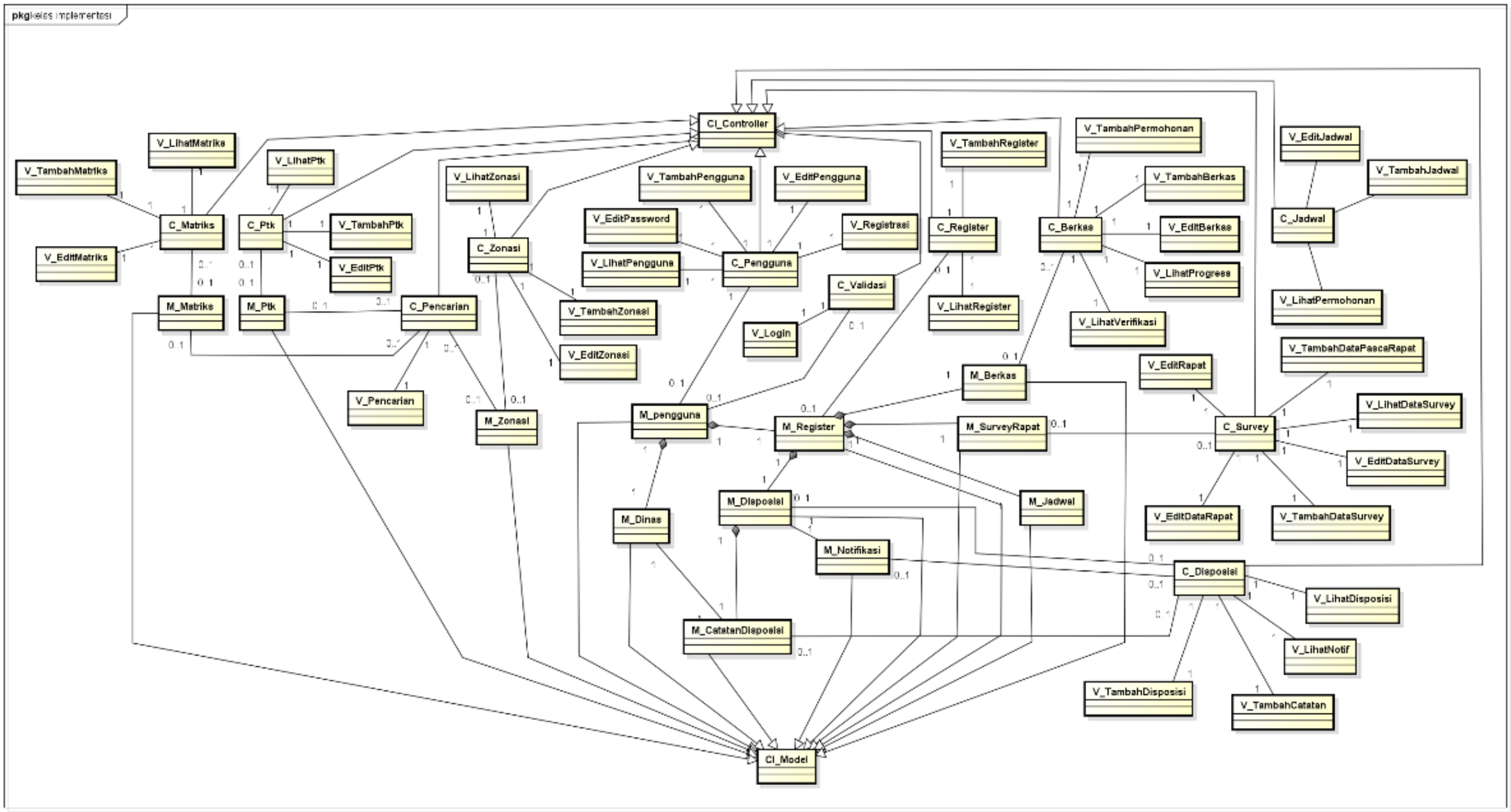
Pada *controller* diagram kelas implementasi juga terjadi perubahan penyesuaian *framework* dimana semua kelas *controller* juga diturunkan dari CI *Controller*. Pada *controller* diagram kelas implementasi tidak terdapat penambahan *method*. Pada gambar 4.6 menunjukkan *controller* dari diagram kelas implementasi



Gambar 4.6 *Controller* diagram kelas implementasi

Gambar 4.7 menunjukkan diagram kelas implementasi keseluruhan. Diagram kelas implementasi keseluruhan mencakup *model, controller* dan *view*.

repository.ub.ac.id



Gambar 4.7 Diagram Kelas Implementasi keseluruhan

Penjelasan dan Tipe Kelas Diagram Implementasi

Diagram kelas implementasi terdapat penyesuaian *framework* CodeIgniter dengan terdapat penambahan dua kelas, yaitu CI_Model dan CI_Controller. CI_Model merupakan kelas *model* dari CodeIgniter, sedangkan kelas CI_Controller merupakan kelas *controller* dari Codeigniter yang objek dari kelas ini adalah super kelas yang menyimpan seluruh *library* dari codeigniter

4.2.2 Matriks Kerunutan Diagram Kelas

Matriks kerunutan bertujuan untuk merunut antara diagram kelas perancangan ke diagram kelas implementasi. Pada tabel 4.40 merupakan matriks kerunutan antara diagram kelas perancangan dan diagram kelas implementasi

Tabel 4.41 Matriks Kerunutan Diagram Kelas

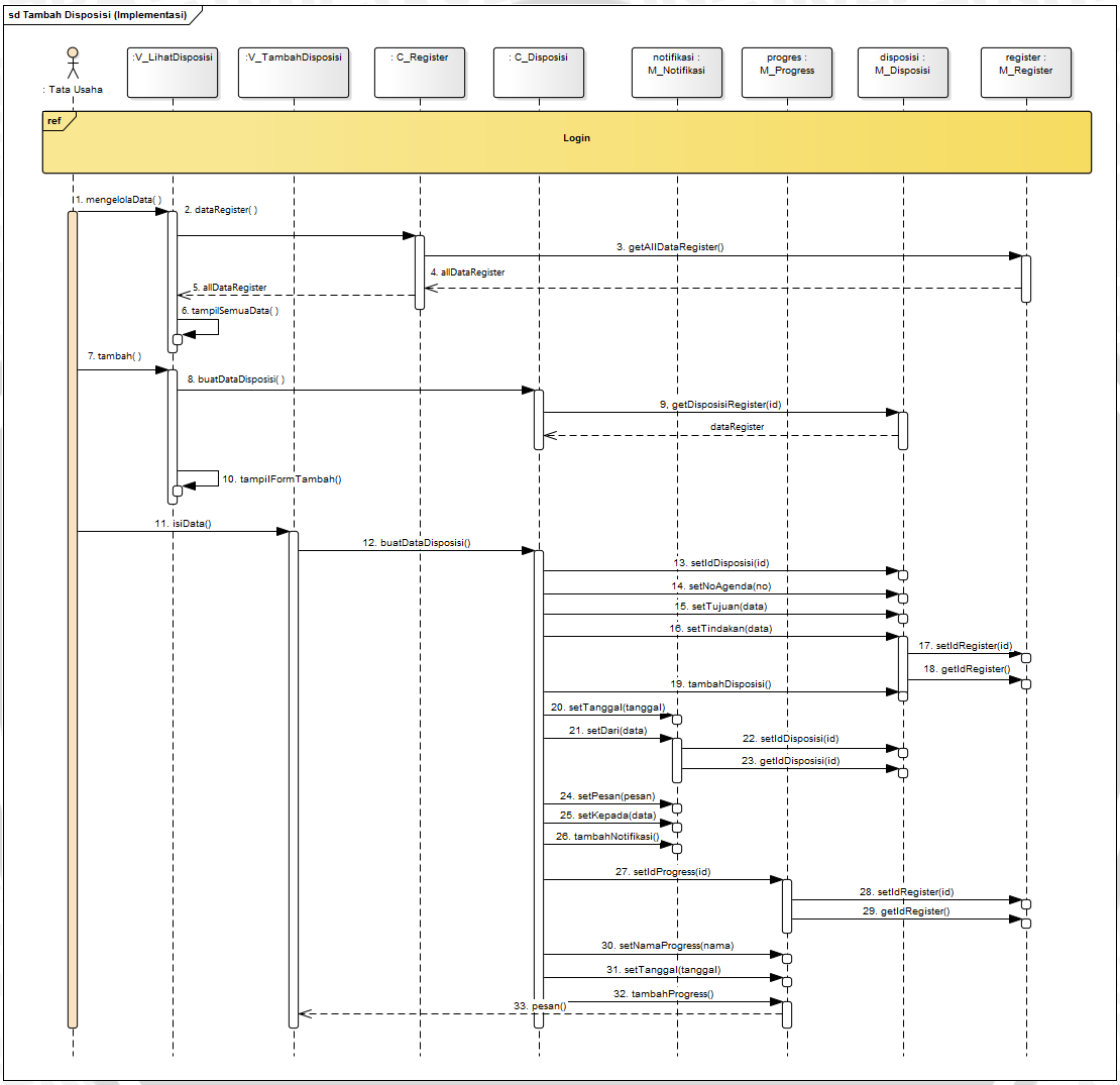
No	Kelas Perancangan	Kelas Implementasi
1	C_Zonasi	C_Zonasi
2	C_Matriks	C_Matriks
3	C_Pengguna	C_Pengguna
4	C_Pencarian	C_Pencarian
5	C_Register	C_Register
6	C_Survey	C_Survey
7	C_Berkas	C_Berkas
8	C_Jadwal	C_Jadwal
9	C_Disposisi	C_Disposisi
10	C_Validasi	C_Validasi
11	M_Matriks	M_Matriks
12	M_Notifikasi	M_Notifikasi
13	M_Zonasi	M_Zonasi
14	M_Ptk	M_Ptk
15	M_Akun	M_Akun
16	M_Dinas	M_Dinas
17	M_Register	M_Register
18	M_SuveyRapat	M_SuveyRapat
19	M_Berkas	M_Berkas
20	M_Jadwal	M_Jadwal
21	M_Disposisi	M_Disposisi
22	Tidak ada	CI_Model
23	Tidak ada	CI_Controller

4.2.3 Diagram *Sequence*

Diagram *sequence* merupakan diagram yang menjelaskan bagaimana suatu operasi dilakukan yang diatur berdasarkan waktu. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang berurut. Pada penelitian ini, terjadi perubahan pada diagram *sequence* dari penelitian sebelumnya, diantaranya adalah sebagai berikut:

1. Membuat Lembar Disposisi

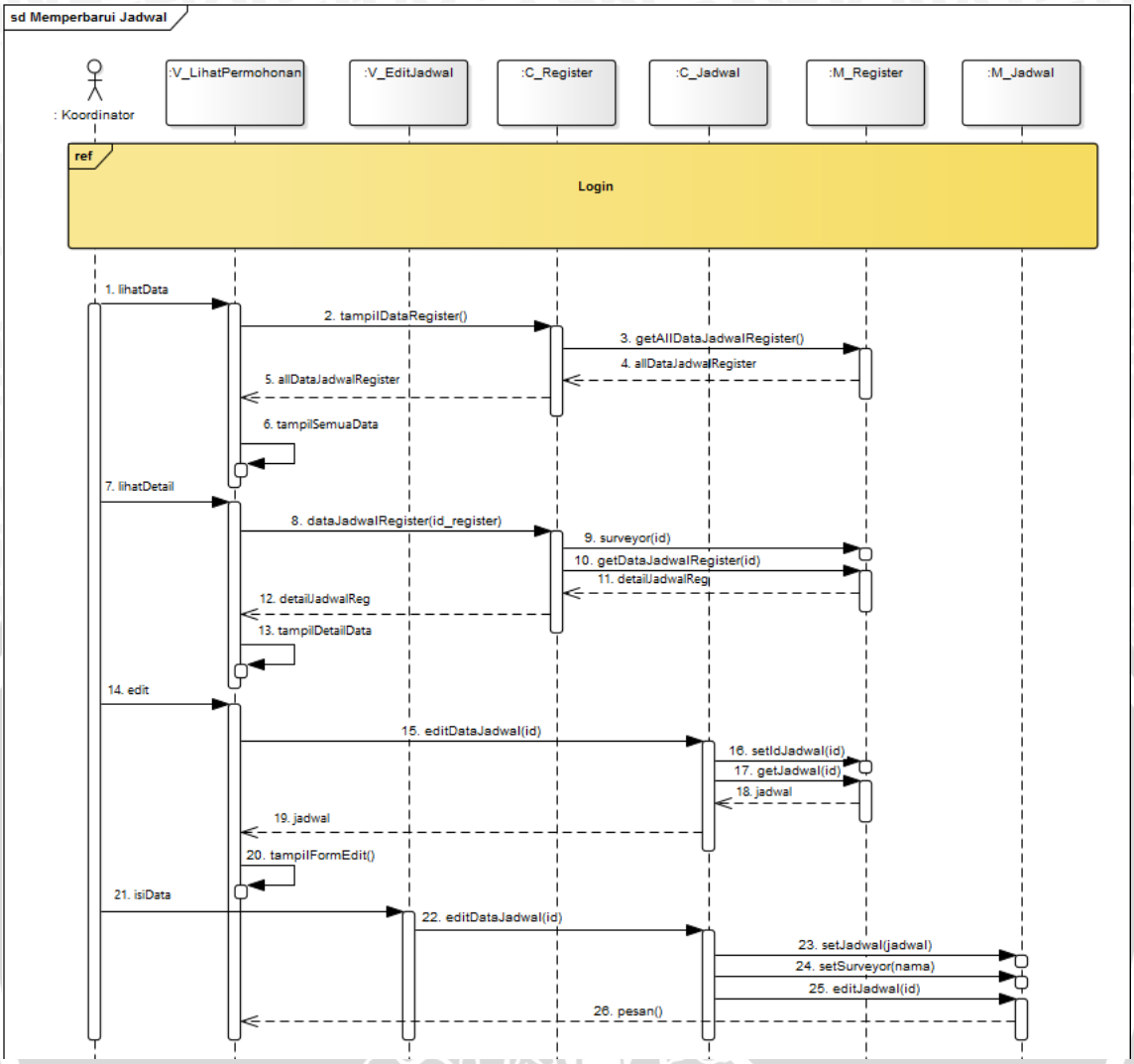
Pada diagram *sequence* membuat lembar disposisi terjadi penambahan *method* pada kelas M_Disposisi, yaitu getDisposisiRegister(id) dimana *method* tersebut digunakan untuk mengambil data register dan data pemohon izin ketika tata usaha akan menambahkan lembar disposisi. Pada gambar 4.8 merupakan perubahan diagram *sequence* membuat lembar disposisi.



Gambar 4.8 Diagram *sequence* membuat lembar disposisi

2. Edit Jadwal

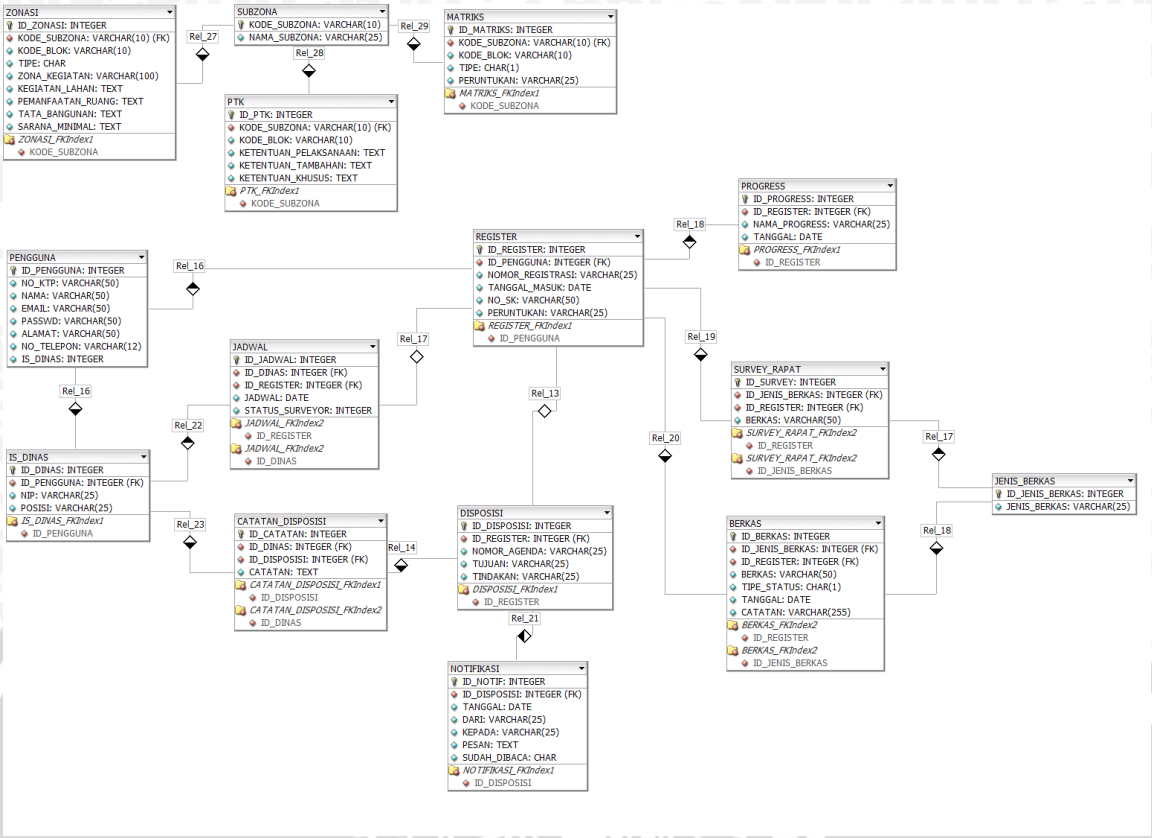
Pada diagram *sequence* memperbarui data jadwal terdapat penambahan *method* pada kelas M_Jadwal, yaitu surveyor(id_register) dimana *method* tersebut digunakan untuk mengambil data surveyor ketika koordinator melihat detail data survey yang sudah terdapat jadwal survey dan nama surveyor. Pada gambar 4.9 merupakan diagram *sequence* edit jadwal.



Gambar 4.9 Diagram *sequence* memperbarui data jadwal

4.2.4 Pemodelan Basis Data

Basis data bertujuan untuk menyimpan semua data yang diperlukan pada SILOKA. Basis data untuk implementasi SILOKA merujuk pada penelitian sebelumnya (Pertiwi, 2016). Untuk membuat basis data berasal dari diagram kelas dengan langkah-langkah yang pertama mengubah entitas menjadi tabel, yang kedua mengubah hubungan menjadi *foreign key*, yang ketiga mengubah atribut menjadi kolom, dan yang terakhir adalah memodifikasi *physical data model* berdasarkan pada kebutuhan atau normalisasi. Gambar 4.10 merupakan basis data SILOKA.



Gambar 4.10 Pemodelan Basis Data SILOKA
Sumber: Pertiwi (2016)

4.2.4.1 Tabel PTK

Nama tabel : PTK
Jumlah *field* : 6
Fungsi : Untuk menyimpan data ketentuan pelaksanaan,tambahan dan khusus

Tabel 4.41 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel PTK

Tabel 4.42 Tabel PTK

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_PTK	Integer	-	Id ptk
2.	KODE_SUBZONA	Varchar	10	Kode subzona
3.	KODE_BLOK	Varchar	10	Kode blok
4.	KETENTUAN_PELAKSANAAN	Text	-	Ketentuan Pelaksanaan pada Peraturan Zonasi

Tabel 4.41 Tabel PTK (lanjutan)

No.	Nama *Field*	Tipe	Lebar	Keterangan
5.	KETENTUAN_TAMBAHAN	Text	-	Ketentuan Tambahan pada Peraturan Zonasi
6.	KETENTUAN_KHUSUS	Text	-	Ketentuan Khusus pada Peraturan Zonasi

Sumber: Pertiwi (2016)

4.2.4.2 Tabel ZONASI

Nama tabel : ZONASI
Jumlah *field* : 9
Fungsi : Untuk menyimpan data ketentuan zonasi pada peraturan zonasi

Tabel 4.42 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel ZONASI

Tabel 4.43 Tabel ZONASI

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_ZONASI	Integer	-	Id zonasi
2.	KODE_SUBZONA	Varchar	10	Kode subzona
3.	KODE_BLOK	Varchar	10	Kode blok
4.	ZONA_KEGIATAN	Varchar	255	Zona kegiatan zonasi
5.	KEGIATAN_LAHAN	Text	-	Ketentuan kegiatan lahan
6.	PEMANFAATAN_RUANG	Text	-	Ketentuan pemanfaatan ruang
7.	TATA_BANGUNAN	Text	-	Ketentuan tata banguna
8.	SARANA_MINIMAL	Text	-	Ketentuan prasarana dan sarana minimal

Tabel 4.42 Tabel ZONASI (lanjutan)

No.	Nama *Field*	Tipe	Lebar	Keterangan
9.	TIPE	Char	-	Tipe peruntukan

Sumber: Pertiwi (2016)

4.2.4.3 Tabel MATRIKS

Nama tabel : MATRIKS
Jumlah *field* : 4
Fungsi : Untuk menyimpan data matriks peraturan zonasi

Tabel 4.43 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel MATRIKS

Tabel 4.44 Tabel MATRIKS

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_MATRIKS	Integer	-	Id matriks
2.	KODE_SUBZONA	Varchar	10	Kode subzona
3.	KODE_BLOK	Varchar	10	Kode blok
4.	PERUNTUKAN	Varchar	255	Peruntukan bangunan

Sumber: Pertiwi (2016)

4.2.4.4 Tabel SUBZONA

Nama tabel : SUBZONA
Jumlah *field* : 2
Fungsi : Untuk menyimpan data subzona

Tabel 4.44 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel SUBZONA

Tabel 4.45 Tabel SUBZONA

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	KODE_SUBZONA	Varchar	10	Id subzona
2.	NAMA_SUBZONA	Varchar	50	Nama wilayah subzona

Sumber: Pertiwi (2016)

4.2.4.5 Tabel PENGGUNA

Nama tabel : PENGGUNA
Jumlah *field* : 7

Fungsi : Untuk menyimpan data pengguna terdaftar pada sistem

Tabel 4.45 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel PENGGUNA

Tabel 4.46 Tabel PENGGUNA

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_PENGGUNA	Integer	-	Id Pengguna
2.	NO_KTP	Varchar	25	No KTP pengguna
3.	NAMA	Varchar	50	Nama pengguna
4.	ALAMAT	Varchar	50	Alamat pengguna
5.	NO_TELEPON	Varchar	12	Nomor telepon pengguna
6.	EMAIL	Varchar	50	Email pemohon
7.	PASSWD	Varchar	20	Password akun pengguna
8.	IS_DINAS	Integer	-	Status Dinas Akun

Sumber: Pertiwi (2016)

4.2.4.6 Tabel IS_DINAS

Nama tabel : IS_DINAS
Jumlah *field* : 4
Fungsi : Untuk menyimpan data pegawai dinas

Tabel 4.46 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel IS_DINAS

Tabel 4.47 Tabel IS_DINAS

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_DINAS	Integer	-	Id dinas
2.	NO_KTP	Varchar	50	No KTP pegawai dinas
3.	NIP	Varchar	25	NIP pegawai dinas
4.	POSISI	Varchar	25	Posisi pegawi terkait dengan proses izin lokasi

Sumber: Pertiwi (2016)

4.2.4.7 Tabel REGISTER

Nama tabel : REGISTER
Jumlah *field* : 6
Fungsi : Untuk menyimpan data register

Tabel 4.47 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel REGISTER

Tabel 4.48 Tabel REGISTER

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_REGISTER	Integer	-	Id register
2.	ID_PENGGUNA	Varchar	25	Id pengguna
3.	NOMOR_REGISTRASI	Varchar	25	Nomor registrasi
4.	TANGGAL_,MASUK	Date	-	Tanggal masuk register
5.	PERUNTUKAN	Varchar	25	Jenis peruntukan permohonan
6.	NO_SK	Varchar	50	No surat keputusan

Sumber: Pertiwi (2016)

4.2.4.8 Tabel DISPOSISI

Nama tabel : DISPOSISI
Jumlah *field* : 5
Fungsi : Untuk menyimpan data lembar disposisi

Tabel 4.48 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel DISPOSISI

Tabel 4.49 Tabel DISPOSISI

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_DISPOSISI	Integer	-	Id disposisi
2.	ID_REGISTER	Integer	-	Id register
3.	NOMOR_AGENDA	Varchar	50	Nomor agenda disposisi
4.	TUJUAN	Varchar	25	Tujuan disposisi
5.	TINDAKAN	Varchar	25	Tindakan disposisi

Sumber: Pertiwi (2016)

4.2.4.9 Tabel CATATAN_DISPOSISI

Nama tabel : CATATAN_DISPOSISI
Jumlah *field* : 4
Fungsi : Untuk menyimpan data catatan disposisi

Tabel 4.49 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel CATATAN_DISPOSISI

Tabel 4.50 Tabel CATATAN_DISPOSISI

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_CATATAN	Integer	-	Id catatan disposisi
2.	NIP	Varchar	25	Nip pegawai dinas
3.	ID_DISPOSISI	Integer	-	Id disposisi
4.	CATATAN	Text	-	Catatan disposisi

Sumber: Pertiwi (2016)

4.2.4.10 Tabel JADWAL

Nama tabel : JADWAL
Jumlah *field* : 5
Fungsi : Untuk menyimpan data jadwal survey

Tabel 4.50 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel JADWAL

Tabel 4.51 Tabel JADWAL

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_JADWAL	Integer	-	Id jadwal
2.	NIP	Varchar	25	Nip pegawai dinas
3.	ID_REGISTER	Integer	-	Id register
4.	JADWAL	Date	-	Jadwal survey
5.	STATUS_SURVEYOR	Integer	-	Status surveyor

Sumber: Pertiwi (2016)

4.2.4.11 Tabel SURVEY_RAPAT

Nama tabel : SURVEY_RAPAT
Jumlah *field* : 7
Fungsi : Untuk menyimpan data hasil survey dan pasca rapat

Tabel 4.51 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel SURVEY_RAPAT

Tabel 4.52 Tabel SURVEY_RAPAT

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_DATA	Integer	-	Id data survey
2.	ID_JENIS_BERKAS	Integer	-	Id jenis berkas
3.	ID_REGISTER	Integer	-	Id register
4.	BERKAS	Varchar	50	Berkas

Sumber: Pertiwi (2016)

4.2.4.12 Tabel JENIS_BERKAS

Nama tabel : JENIS_BERKAS
Jumlah *field* : 2
Fungsi : Untuk menyimpan data jenis berkas

Tabel 4.52 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel JENIS_BERKAS

Tabel 4.53 Tabel JENIS_BERKAS

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_JENIS_BERKAS	Integer	-	Id jenis berkas
2.	JENIS_BERKAS	Vaarchar	25	Jenis berkas pemohon

Sumber: Pertiwi (2016)

4.2.4.13 Tabel BERKAS

Nama tabel : BERKAS
Jumlah *field* : 7
Fungsi : Untuk menyimpan data berkas pemohon izin lokasi

Tabel 4.53 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel BERKAS

Tabel 4.54 Tabel BERKAS

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_BERKAS	Integer	-	Id berkas
2.	ID_REGISTER	Integer	-	Id register
3.	JENIS_BERKAS	Integer	-	Jenis berkas pemohon
4.	BERKAS	Varchar	50	Berkas pemohon

Tabel 4.53 Tabel BERKAS (lanjutan)

No.	Nama *Field*	Tipe	Lebar	Keterangan
5.	TIPE_STATUS	Varchar	25	Status verifikasi berkas
6.	TANGGAL	Timestamp	-	Tanggal verifikasi berkas
7.	CATATAN	Varchar	255	Catatan Koordinator

Sumber: Pertiwi (2016)

4.2.4.14 Tabel NOTIFIKASI

Nama tabel : NOTIFIKASI
Jumlah *field* : 7
Fungsi : Untuk menyimpan data pesan notifikasi disposisi dan catatan disposisi

Tabel 4.54 merupakan penjelasan nama, tipe, lebar, dan keterangan dari tabel NOTIFIKASI

Tabel 4.55 Tabel NOTIFIKASI

No.	Nama *Field*	Tipe	Lebar	Keterangan
1.	ID_NOTIF	Integer	-	Id notifikasi
2.	ID_DISPOSISI	Integer	-	Id disposisi
3.	TANGGAL	Timestamp	-	Tanggal notifikasi
4.	DARI	Varchar	50	Pengirim pesan notifikasi
5.	KEPADA	Varchar	50	Penerima pesan notifikasi
6.	PESAN	Text	-	Pesan notifikasi
7.	SUDAH_DIBACA	Char	1	Status pesan notifikasi

Sumber: Pertiwi (2016)

BAB 5 IMPLEMENTASI

Bab ini membahas mengenai implementasi dari SILOKA yang mencakup lingkungan implementasi dari perangkat lunak dan perangkat keras, kode program SILOKA dan hasil implementasi berupa *screenshot* program SILOKA ketika dijalankan.

5.1 Lingkungan Implementasi

Pada lingkungan implementasi menjelaskan tentang lingkungan spesifikasi perangkat lunak maupun perangkat keras yang digunakan untuk mengimplementasikan SILOKA.

5.1.1 Perangkat Keras

Berikut merupakan spesifikasi perangkat keras yang digunakan untuk pengembangan SILOKA adalah sebagai berikut.

1. Manufaktur : HP (*Hewlett-Packard*)
2. Model : Envy M6
3. *Processor* : Intel Core-i7 3632QM CPU @2.20GHz (8Cpu)
4. *Memory* : 8192 MB RAM
5. *Harddisk* : 750GB
6. VGA : AMD Radeon 7600M Series

5.1.2 Perangkat Lunak

Berikut ini merupakan perangkat lunak yang digunakan untuk pengembangan SILOKA adalah sebagai berikut.

1. Sistem Operasi Microsoft Windows 10 Pro *Single Language* merupakan seri terakhir sistem operasi dari Microsoft
2. XAMPP Versi 3.2.1 (Apache untuk *Web Server* dan MySQL untuk basis data)

   XAMPP adalah perangkat lunak yang berguna untuk pengembangan website berbasis PHP dan MySQL. XAMPP memiliki kelebihan untuk bisa berperan sebagai server web Apache untuk simulasi pengembangan website. Tool pengembangan web ini mendukung teknologi web populer, seperti PHP dan MySQL.
3. Sublime Text V.3
4. *Web Browser* Google Chrome

5.2 Implementasi Sistem

Pada sub bab Implementasi penulis terdapat *source code* dari program hasil implementasi dari SILOKA dan beserta *screenshot* hasil implementasi dari SILOKA.

5.2.1 Kode Program Implementasi

Source Code merupakan kode program implementasi dari SILOKA. Berikut ini merupakan *source code* dari setiap fitur SILOKA

1. Pencarian zonasi

Tabel 5.1 merupakan kode program *view* dari pencarian zonasi pada warga

Tabel 5.1 Kode program *view* pencarian zonasi

```
<html>
<head>
      <title></title>
      <link    href="<?php    echo    base_url();    ?>assets/css/pure.css"
rel="stylesheet" type="text/css" />
      <style scoped>

        .button-success,
        .button-error,
        .button-warning,
        .button-secondary {
            color: white;
            border-radius: 4px;
            text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
        }

        .button-success {
            background: rgb(28, 184, 65); /* this is a green */
        }

        .button-error {
            background: rgb(202, 60, 60); /* this is a maroon */
        }

        .button-warning {
            background: rgb(223, 117, 20); /* this is an orange */
        }

        .button-secondary {
            background: rgb(66, 184, 221); /* this is a light blue */
        }

    </style>
</head>
<body>
<div class="navluar">
      <div class="navdalam">
            <h3 align="center">PENCARIAN ZONASI</h3>
      </div>
      <form class="pure-form" action="<?php echo base_url();
?>index.php/C_Pencarian/pencarianZonasi" method="post" style="font-
size:12px;">
          <fieldset>
              <input type="text" placeholder="Kode Blok" name="blok"
required>
```

Tabel 5.1 Kode program *view* pencarian zonasi (lanjutan)

```
                <input type="text" placeholder="Kode Subzona"
name="subzona" required>
                    <input type="text" placeholder="Peruntukan"
name="peruntukan" required>
                    <input type="submit" name = "cariZonasi" class="pure-
button pure-button-primary" value="Cari">
            </fieldset>
        </form>
</body>
</html>
```

Pada tabel 5.2 merupakan kode program *controller* C_Pencarian untuk pencarian zonasi

Tabel 5.2 Kode program *controller* C_Pencarian untuk pencarian zonasi

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class C_Pencarian extends CI_Controller {


    public function __construct(){
            parent::__construct();
            $this->load->model('M_Matriks','M_Zonasi','M_Ptk',true);
    }

    public function pencarianZonasi(){
            if($this->input->post('cariZonasi')){
                    $subzona = $this->input->post('subzona');
                    $blok = $this->input->post('blok');
                    $peruntukan = $this->input->post('peruntukan');
                    $id = $this->input->post('id');

                    $dataMatriks =  $this->M_Matriks-
>getDataMatriks($subzona,$blok,$peruntukan);
                    $dataZonasi = $this->M_Zonasi-
>getDataZonasi($subzona,$blok,$peruntukan);
                    $dataPtk = $this->M_Ptk->getDataPtk($subzona,$blok);

                    $data['matriks'] = $dataMatriks->result();

                    $data['zonasi'] = $dataZonasi->result();
                    $data['ptk'] = $dataPtk->result();

                    if(($dataMatriks->num_rows() == 1) and ($dataZonasi-
>num_rows() == 1)){
                            if($this->session->userdata('jabatan') ==
'Koordinator'){
                                    $data['judul']="Halaman Pencarian";

        $data['main']="koordinator/ketentuanZonasi";
                                    $data['nav']="koordinator/nav";
                                    $this->load->view('admin',$data);
                            }elseif($this->session->userdata('jabatan') ==
'Surveyor'){
                                    $data['judul']="Halaman Pencarian";
                                    $data['main']="surveyor/ketentuanZonasi";
                                    $data['nav']="surveyor/nav";
                                    $this->load->view('admin',$data);
                            }else{
                                    $data['judul']="Halaman Pencarian";
```

Tabel 5.2 Kode program *controller* C_Pencarian untuk pencarian zonasi (lanjutan)

```
47                                      $data['main']="warga/wargaHasilPencarian";
48                                      $data['footer']="footer";
49                                      $data['laman']="warga/laman";
50                                      $this->load->view('home',$data);
51                              }
52                      }elseif($dataMatriks->num_rows()  >1 and ($peruntukan !=
53  ' ' or $peruntukan != NULL)){
54                              if($this->session->userdata('jabatan') ==
55  'Koordinator'){
56                                      $data['judul']="Halaman Pencarian";
57
58          $data['main']="koordinator/ketentuanZonasiKembar";
59                                      $data['nav']="koordinator/nav";
60                                      $this->load->view('admin',$data);
61                              }elseif($this->session->userdata('jabatan') ==
62  'Surveyor'){
63                                      $data['judul']="Halaman Pencarian";
64
65          $data['main']="surveyor/ketentuanZonasiKembar";
66                                      $data['nav']="surveyor/nav";
67                                      $this->load->view('admin',$data);
68                              }else{
69                                      $data['judul']="Halaman Pencarian";
70
71          $data['main']="warga/wargaHasilPencarianKembar";
72                                      $data['footer']="footer";
73                                      $data['laman']="warga/laman";
74                                      $this->load->view('home',$data);
75                              }
76                      }elseif(($dataMatriks->num_rows() == 1) and
77  ($dataZonasi->num_rows() > 1)){
78                              if($this->session->userdata('jabatan') ==
79  'Koordinator'){
80                                      $data['judul']="Halaman Pencarian";
81
82          $data['main']="koordinator/ketentuanZonasiKembar2";
83                                      $data['nav']="koordinator/nav";
84                                      $this->load->view('admin',$data);
85                              }elseif($this->session->userdata('jabatan') ==
86  'Surveyor'){
87                                      $data['judul']="Halaman Pencarian";
88
89          $data['main']="surveyor/ketentuanZonasiKembar2";
90                                      $data['nav']="surveyor/nav";
91                                      $this->load->view('admin',$data);
92                              }else{
93                                      $data['judul']="Halaman Pencarian";
94
95          $data['main']="warga/wargaHasilPencarianKembar2";
96                                      $data['footer']="footer";
97                                      $data['laman']="warga/laman";
98                                      $this->load->view('home',$data);
99                              }
100                     }else{
101                             $zonasi = $this->M_Zonasi->cariDataZonasi($id);
102                             $data['zona'] = $zonasi->result();
103                             if($zonasi->num_rows() == 1){
104                                     if($this->session->userdata('jabatan') ==
105 'Koordinator'){
106                                             $data['judul']="Halaman Pencarian";
107
108         $data['main']="koordinator/ketentuanZonasi2";
109                                             $data['nav']="koordinator/nav";
```

Tabel 5.2 Kode program *controller* C_Pencarian untuk pencarian zonasi (lanjutan)

```
110 $this->load->view('admin',$data);
111 }elseif($this->session->userdata
112 ('jabatan') == 'Surveyor'){
113                                         $data['judul']="Halaman Pencarian";
114
115       $data['main']="surveyor/ketentuanZonasi2";
116                                         $data['nav']="surveyor/nav";
117                                         $this->load->view('admin',$data);
118                                   }else{
119                                         $data['judul']="Halaman Pencarian";
120
121       $data['main']="warga/wargaHasilPencarian2";
122                                         $data['footer']="footer";
123                                         $data['laman']="warga/laman";
124                                         $this->load->view('home',$data);
125                                   }
126                             }else{
127                                   echo "<script>alert('Data Zonasi tidak
128 Ditemukan');"
129                                   .
130 "window.location='".base_url()."index.php/C_Pencarian/pencarian';</script>";
131
132                             }
133                       }
134             }else{
135                   redirect('C_Pengguna/registrasi','refresh');
136             }
137       }
138 }
139 ?>
```

Pada tabel 5.3 merupakan kode program *model* M_Matriks untuk pencarian zonasi.

Tabel 5.3 Kode program *model* M_Matriks untuk pencarian zonasi

```
1  <?php
2  class M_Matriks extends Ci_Model{
3
4        private $idMatriks;
5        private $namaSubzona;
6        private $kodeBlok;
7        private $peruntukan;
8        private $tipe;
9
10       function __construct()
11       {
12             parent::__construct();
13             $this->load->database();
14       }
15
16       public function setIdMatriks($id){
17             $this->idMatriks = $id;
18       }
19
20       public function setNamaSubzona($nama){
21             $this->namaSubzona = $nama;
22       }
23
24       public function setKodeBlok($kode){
25             $this->kodeBlok = $kode;
26       }
```

Tabel 5.3 Kode program *model* M_Matriks untuk pencarian zonasi (lanjutan)

```
27
28          public function setTipe($tipe){
29                  $this->tipe = $tipe;
30          }
31
32          public function setPeruntukan($peruntukan){
33                  $this->peruntukan = $peruntukan;
34          }
35
36          public function getDataMatriks($subzona, $blok, $peruntukan){
37                  return  $this->db->query("select  ID_MATRIKS,  KODE_SUBZONA,
38  KODE_BLOK, FLEKSIBEL_ZONING, PERUNTUKAN from matriks where KODE_SUBZONA =
39  '$subzona' AND KODE_BLOK = '$blok' AND PERUNTUKAN LIKE'%$peruntukan%'");
40          }
41  }
42  ?>
```

Pada tabel 5.4 merupakan kode program *model* M_Zonasi *method* getDataZonasi($subzona, $blok, $peruntukan) untuk pencarian zonasi.

Tabel 5.4 Kode program *model* M_Zonasi untuk pencarian zonasi

```
1   <?php
2   class M_Ptk extends Ci_Model{
3
4           private $idPtk;
5           private $namaSubzona;
6           private $kodeBlok;
7           private $ketentuanPelaksanaan;
8           private $ketentuanTambahan;
9           private $ketentuanKhusus;
10
11          function __construct()
12          {
13                  parent::__construct();
14                  $this->load->database();
15          }
16
17          public function setIdPtk($id){
18                  $this->idPtk = $id;
19          }
20
21          public function setNamaSubzona($subzona){
22                  $this->namaSubzona = $subzona;
23          }
24
25          public function setKodeBlok($kode){
26                  $this->kodeBlok = $kode;
27          }
28
29          public function setKetentuanP($ketentuan){
30                  $this->ketentuanPelaksanaan = $ketentuan;
31          }
32
33          public function setKetentuanT($ketentuan){
34                  $this->ketentuanTambahan = $ketentuan;
35          }
36
37          public function setKetentuanK($ketentuan){
38                  $this->ketentuanKhusus = $ketentuan;
39          }
40
```

Tabel 5.4 Kode program *model* M_Zonasi untuk pencarian zonasi (lanjutan)

```
41          public function cariDataPtk(){
42
43          }
44
45          public function getDataPtk($id){
46                  return    $this->db->query("select    ID_PTK,    KODE_SUBZONA,
47  KODE_BLOK, KETENTUAN_PELAKSANAAN, KETENTUAN_TAMBAHAN, KETENTUAN_KHUSUS from
48  ptk where ID_PTK = '$id'");
49          }
50  }
51  ?>
```

Pada tabel 5.5 merupakan kode program *model* M_Ptk *method* getDataPTK($subzona, $blok) untuk pencarian zonasi.

Tabel 5.5 Kode program *model* M_Ptk untuk pencarian zonasi

```
1   <?php
2   class M_Zonasi extends Ci_Model{
3
4           private $idZonasi;
5           private $namaSubzona;
6           private $kodeBlok;
7           private $tipe;
8           private $zonaKegiatan;
9           private $kegiatanLahan;
10          private $pemanfaatanRuang;
11          private $tataBangunan;
12          private $saranaMinimal;
13
14          function __construct()
15          {
16                  parent::__construct();
17                  $this->load->database();
18          }
19
20          public function setIdZonasi($id){
21                  $this->idZonasi = $id;
22          }
23
24          public function setNamaSubzona($subzona){
25                  $this->namaSubzona = $subzona;
26          }
27
28          public function setKodeBlok($blok){
29                  $this->kodeBlok = $blok;
30          }
31
32          public function setTipe($tipe){
33                  $this->tipe = $tipe;
34          }
35
36          public function setZonasiKegiatan($zona){
37                  $this->zonaKegiatan = $zona;
38          }
39
40          public function setPemanfaatanRuang($data){
41                  $this->pemanfaatanRuang = $data;
42          }
43
44          public function setBangunan($data){
45                  $this->tataBangunan = $data;
```

Tabel 5.5 Kode program *model* M_Ptk untuk pencarian zonasi (lanjutan)

```
46          }
47
48          public function setKegiatanLahan($kegiatan){
49                  $this->kegiatanLahan = $kegiatan;
50          }
51
52          public function setSaranaMinimal($data){
53                  $this->saranaMinimal = $data;
54          }
55
56          public function getDataZonasi($subzona, $blok, $peruntukan){
57                  return   $this->db->query("select   ID_ZONASI,   KODE_SUBZONA,
58  KODE_BLOK,   TIPE,   ZONA_KEGIATAN,   KEGIATAN_LAHAN,   PEMANFAATAN_RUANG,
59  TATA_BANGUNAN, SARANA_MINIMAL from zonasi where KODE_SUBZONA = '$subzona'
60  AND KODE_BLOK = '$blok' AND ZONA_KEGIATAN LIKE'%$peruntukan%'");
61          }
62  }
63  ?>
```

2. Mendaftarkan Diri

Pada tabel 5.6 merupakan kode program *view* untuk registrasi pada warga

Tabel 5.6 Kode program *view* mendaftarkan diri

```
1   <html>
2   <head>
3           <title></title>
4           <link href="<?php echo base_url(); ?>assets/css/pure.css"
5   rel="stylesheet" type="text/css" />
6           <style scoped>
7
8             .button-success,
9             .button-error,
10            .button-warning,
11            .button-secondary {
12                color: white;
13                border-radius: 4px;
14                text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2);
15            }
16
17            .button-success {
18                background: rgb(28, 184, 65); /* this is a green */
19            }
20
21            .button-error {
22                background: rgb(202, 60, 60); /* this is a maroon */
23            }
24
25            .button-warning {
26                background: rgb(223, 117, 20); /* this is an orange */
27            }
28
29            .button-secondary {
30                background: rgb(66, 184, 221); /* this is a light blue */
31            }
32
33      </style>
34  </head>
35  <body>
36  <div class="navluar">
37          <div class="navdalam">
```

Tabel 5.6 Kode program *view* mendaftarkan diri (lanjutan)

```
38              <h3 align="center">PENDAFTARAN</h3>
39          </div>
40          <form action="<?php echo base_url();
41  ?>index.php/C_Pengguna/registrasi" method="post" class="pure-form">
42              <table align="center" class="pure-table pure-table-
43  horizontal" style="font-size:12px" width="80%">
44                  <tr>
45                      <td>Nama Pemohon</td>
46                      <td>:</td>
47                      <td><input type="text" name="nama"
48  placeholder="Nama Pemohon" required /></td>
49                  </tr>
50                  <tr>
51                      <td>No. Identitias</td>
52                      <td>:</td>
53                      <td><input type="text" name="no_ktp"
54  placeholder="No. Identitas" required /></td>
55                  </tr>
56                  <tr>
57                      <td>Alamat</td>
58                      <td>:</td>
59                      <td><input type="text" name="alamat"
60  placeholder="Alamat" required /></td>
61                  </tr>
62                  <tr>
63                      <td>No. Handphone</td>
64                      <td>:</td>
65                      <td><input type="text" name="telepon"
66  placeholder="No. Handphone" required /></td>
67                  </tr>
68                  <tr>
69                      <td>E-mail</td>
70                      <td>:</td>
71                      <td><input type="email" name="email"
72  placeholder="E-mail" required /></td>
73                  </tr>
74                  <tr>
75                      <td>Password</td>
76                      <td>:</td>
77                      <td><input type="password" name="password"
78  placeholder="Password" required /></td>
79                  </tr>
80                  <tr>
81                      <td></td>
82                      <td></td>
83                      <td><input type="submit" name="daftar"
84  value="Daftar" class="button-secondary pure-button" /></td>
85                  </tr>
86              </table>
87          </td>
88  </div>
89  </body>
90  </html>
```

Pada tabel 5.7 merupakan *source code method* registrasi pada kelas C_Pengguna

Tabel 5.7 *Method* registrasi

```
1   public function registrasi()
2   {
3           if($this->session->userdata('isDinas') == "0"){
4                   $data['judul']="Halaman Admin Home";
```

Tabel 5.7 *Method* registrasi (lanjutan)

```
5               $data['main']="koordinator/tambahPengguna";
6               $data['nav']="koordinator/nav";
7               $this->load->view('admin',$data);
8       }else{
9               if($this->input->post('daftar')){
10                      $this->M_Pengguna->setNama($this->input->post('nama'));
11                      $this->M_Pengguna->setNo_ktp($this->input-
    >post('no_ktp'));
12                      $this->M_Pengguna->setAlamat($this->input-
    >post('alamat'));
13                      $this->M_Pengguna->setTelepon($this->input-
    >post('telepon'));
14                      $this->M_Pengguna->setEmail($this->input-
    >post('email'));
15                      $this->M_Pengguna->setPassword($this->input-
    >post('password'));
16
17                      $daftar = $this->M_Pengguna->simpanDataPemohon();
18                      if($daftar == "TRUE"){
19                              echo "<script>alert('Anda telah berhasil
20  mendaftar, silahkan Login');"
21                                                      .
22  "window.location='".base_url()."index.php/C_Validasi/login';</script>";
23                      }elseif($daftar == "FALSE"){
24                              echo "<script>alert('Nomor identitas yang Anda
25  masukkan sudah terdapat pada sistem');"
26                                                      .
27  "window.location='".base_url()."index.php/C_Pengguna/registrasi';</script>"
28  ;
29                      }
30              }else{
31                      $data['judul']="Halaman Pendaftaran";
32                      $data['main']="warga/wargaPendaftaran";
33                      $data['footer']="footer";
34                      $data['laman']="warga/laman";
35                      $this->load->view('home',$data);
36              }
37      }
38  }
```

Tabel 5.8 merupakan *source code model* M_Pengguna

Tabel 5.8 *Source Code* M_Pengguna

```
1   <?php
2   class M_Pengguna extends Ci_Model{
3
4           private $nama;
5           private $no_ktp;
6           private $alamat;
7           private $telepon;
8           private $email;
9           private $password;
10
11          public function setNama($nama){
12                  $this->nama = $nama;
13          }
14
15          public function setNo_ktp($no){
16                  $this->no_ktp = $no;
17          }
18
```

Tabel 5.8 *Source Code* M_Pengguna (lanjutan)

```
19          public function setAlamat($alamat){
20                  $this->alamat = $alamat;
21          }
22
23          public function setTelepon($telepon){
24                  $this->telepon = $telepon;
25          }
26
27          public function setEmail($email){
28                  $this->email = $email;
29          }
30
31          public function setPassword($pass){
32                  $this->password = $pass;
33          }
34
35          public function getNama(){
36                  return $this->nama;
37          }
38
39          public function getNo_ktp(){
40                  return $this->no_ktp;
41          }
42
43          public function getAlamat(){
44                  return $this->alamat;
45          }
46
47          public function getTelepon(){
48                  return $this->telepon;
49          }
50
51          public function getEmail(){
52                  return $this->email;
53          }
54
55          public function getPassword(){
56                  return $this->password;
57          }
58
59          public function simpanDataPemohon(){
60                  $boolean = NULL;
61                  $pengguna = $this->db->query("select no_ktp from
62  akun where no_ktp = '$this->no_ktp'")->row();
63                  if($pengguna->no_ktp == NULL){
64                          $dataPemohon = array(
65                          'no_ktp' => $this->no_ktp,
66                          'nama' => $this->nama,
67                          'email' => $this->email,
68                          'passwd' => $this->password,
69                          'alamat' => $this->alamat,
70                          'no_telp' => $this->telepon,
71                          'isDinas' => 1
72                           );
73                          $this->db->insert('akun', $dataPemohon);
74                          $boolean = "TRUE";
75                  }else{
76                          $boolean = "FALSE";
77                  }
78                  return $boolean;
79          }
80  }
```

3. Mengelola Data Pemohon

Tabel 5.9 merupakan *source code method* editData Pengguna

Tabel 5.9 *method* editDataPengguna

```
1   public function editDataPemohon(){
2   if($this->session->userdata('pengguna')){
3         if($this->session->userdata('isDinas') != 0){
4               if($this->input->post('editPemohon')){
5                     $this->M_Pengguna->setNama($this->input-
6   >post('nama'));
7                     $this->M_Pengguna->setAlamat($this->input-
8   >post('alamat'));
9                     $this->M_Pengguna->setTelepon($this->input-
10  >post('telepon'));
11
12                    $editPemohon = $this->M_Pengguna-
13  >updateDataPemohon();
14
15                    echo "<script>alert('Data Anda telah berhasil
16  diperbarui');"
17                                                .
18  "window.location='".base_url()."index.php/C_Pengguna/dataPemohon';
19  </script>";
20              }else{
21                    $getDataPemohon = $this->M_Pengguna-
22  >getDataPemohon($this->session->userdata('pengguna'))->row();
23                    $this->M_Pengguna->setNama($getDataPemohon-
24  >nama);
25                    $this->M_Pengguna->setNo_ktp($getDataPemohon-
26  >no_ktp);
27                    $this->M_Pengguna->setAlamat($getDataPemohon-
28  >alamat);
29                    $this->M_Pengguna->setTelepon($getDataPemohon-
30  >no_telp);
31                    $data['nama'] = $this->M_Pengguna->getNama();
32                    $data['no_ktp'] = $this->M_Pengguna-
33  >getNo_ktp();
34                    $data['alamat'] = $this->M_Pengguna-
35  >getAlamat();
36                    $data['telepon'] = $this->M_Pengguna-
37  >getTelepon();
38                    $data['judul']="Halaman Data Pemohon";
39                    $data['main']="pemohon/pemohonEditPemohon";
40                    $data['footer']="footer";
41                    $data['laman']="pemohon/lamanPemohonIzin";
42                    $this->load->view('home',$data);
43              }
44        }else{
45        redirect(base_url('index.php/C_Validasi/autentifikasi'));
46        }
47  }else{
48        redirect(base_url('index.php/C_Validasi/autentifikasi'));
49  }
50  }
```

Tabel 5.10 merupakan *source code class* M_Pengguna

Tabel 5.10 M_Pengguna

```
1   private $nama;
2   private $no_ktp;
3   private $alamat;
```

Tabel 5.10 M_Pengguna (lanjutan)

```
4   private $telepon;
5   private $email;
6   private $password;
7
8   public function setNama($nama){
9         $this->nama = $nama;
10  }
11
12  public function setNo_ktp($no){
13        $this->no_ktp = $no;
14  }
15
16  public function setAlamat($alamat){
17        $this->alamat = $alamat;
18  }
19
20  public function setTelepon($telepon){
21        $this->telepon = $telepon;
22  }
23
24  public function setEmail($email){
25        $this->email = $email;
26  }
27
28  public function setPassword($pass){
29        $this->password = $pass;
30  }
31
32  public function getNama(){
33        return $this->nama;
34  }
35
36  public function getNo_ktp(){
37        return $this->no_ktp;
38  }
39
40  public function getAlamat(){
41        return $this->alamat;
42  }
43
44  public function getTelepon(){
45        return $this->telepon;
46  }
47
48  public function getEmail(){
49        return $this->email;
50  }
51
52  public function getPassword(){
53        return $this->password;
54  }
55
56  public function updateDataPemohon(){
57        $dataUpdate = array('nama' => $this->nama,
58                            'alamat' => $this->alamat,
59                            'no_telp' => $this->telepon
60         );
61        $this->db->where('no_ktp',$this->session-
    >userdata('pengguna'));
62        $this->db->update('akun',$dataUpdate);
63        return;
64  }
```

5.2.2 Hasil Implementasi

1. Halaman Mendaftarkan Diri

Halaman mendaftarkan diri merupakan halaman yang berfungsi untuk melakukan pendaftaran diri ke dalam sistem sehingga pengunjung atau warga dapat masuk ke dalam sistem dan mengajukan permohonan izin lokasi. Gambar 5.1 menunjukkan antarmuka pengguna halaman mendaftarkan diri

Gambar 5.1 Halaman Mendaftarkan Diri

2. Memohon Izin Lokasi

a. Halaman Home Pemohon Izin

Halaman home pemohon izin merupakan halaman awal pemohon izin. Halaman ini berisi informasi tata cara untuk mengajukan permohonan izin lokasi, yaitu berupa alur dalam bentuk gambar dan penjelasan dari masing-masing alur. Gambar 5.2 menunjukkan antarmuka pengguna halaman *home* pemohon izin.

Gambar 5.2 Halaman Home Pemohon izin

b. Halaman Permohonan Izin Lokasi

Halaman permohonan izin lokasi merupakan halaman yang berisi tentang informasi daftar permohonan izin yang sudah dilakukan oleh pemohon izin. Gambar 5.3 menunjukkan antarmuka pengguna halaman permohonan izin lokasi.

Gambar 5.3 Halaman Permohonan Izin Lokasi

c. Halaman Pengajuan Permohonan Izin Lokasi

Halaman pengajuan izin lokasi merupakan halaman yang berfungsi untuk mengajukan permohonan izin lokasi. Pada halaman ini, pemohon izin diminta untuk mengisikan nama perusahaan yang diwakilkan dan lokasi yang dimohon. Gambar 5.4 menunjukkan antarmuka pengguna halaman pengajuan permohonan izin lokasi.

Gambar 5.4 Halaman Pengajuan Permohonan Izin Lokasi

d. Halaman Detail Permohonan

Halaman detail permohonan merupakan halaman yang menyediakan informasi mengenai progres izin lokasi dan notifikasi kekurangan berkas dari pemohon izin. Pada halaman pemohon izin dapat mengetahui bagaimana progres izin lokasi yang dimohon. Gambar 5.5 menunjukkan antarmuka pengguna halaman detail permohonan.

Gambar 5.5 Halaman Detail Permohonan

e. Halaman Mengelola Berkas Permohonan

Halaman mengelola berkas permohonan merupakan halaman yang digunakan pemohon izin untuk menambahkan berkas, mengedit berkas juga menghapus berkas. Pada halaman ini pemohon izin juga

dapat mengetahui apakah berkas yang diajukan sudah diverifikasi atau belum. Gambar 5.6 menunjukkan antarmuka pengguna halaman mengelola berkas permohonan

Gambar 5.6 Halaman Mengelola Berkas Permohonan

3. Halaman Pencarian Zonasi

a. Halaman Pencarian Zonasi (Memasukkan Kata Kunci)

Halaman pencarian zonasi merupakan halaman yang berfungsi untuk melakukan pencarian peraturan zonasi. Untuk melakukan pencarian zonasi, warga diminta untuk memasukkan kode blok, kode subzona serta jenis peruntukan. Gambar 5.7 menunjukkan antarmuka pengguna halaman pencarian zonasi

Gambar 5.7 Halaman Pencarian Zonasi

b. Halaman Hasil Pencarian Zonasi

Halaman hasil pencarian zonasi merupakan halaman yang menampilkan hasil dari pencarian zonasi berupa subzona, kode blok, peruntukan, fleksible zoning, peraturan zonasi, dan peraturan PTK. Gambar 5.8 menunjukkan antarmuka pengguna halaman hasil pencarian zonasi.

Gambar 5.8 Halaman Hasil Pencarian Zonasi

c. Halaman Hasil Pencarian Matriks Kembar

Halaman hasil pencarian matriks kembar merupakan halaman yang menampilkan jika hasil pencarian pada matriks terdapat data yang sama atau kembar, maka akan ditampilkan pada halaman ini. Gambar 5.9 menunjukkan antarmuka pengguna halaman hasil pencarian matriks kembar.

Gambar 5.9 Halaman Hasil Pencarian Matriks Kembar

d. Halaman Hasil Pencarian Zonasi Kembar

Halaman hasil pencarian zonasi kembar merupakan halaman yang menampilkan jika hasil pencarian pada zonasi terdapat data yang sama

atau kembar, maka akan ditampilkan pada halaman ini. Gambar 5.10 menunjukkan antarmuka pengguna halaman hasil pencarian zonasi kembar

Gambar 5.10 Halaman Hasil Pencarian Zonasi Kembar

BAB 6 PENGUJIAN

Bab ini membahas mengenai pengujian yang dilakukan pada Sistem Informasi IZIN LOKASI dengan menggunakan jenis pengujian *Black Box* dan *White Box*, serta pengambilan hasil pengujian *black box* dan *white box* dari SILOKA.

6.1 Rencana Pengujian

Pengujian menggunakan metode *blackbox* dan pengujian dilakukan berdasarkan skenario yang sudah didefinisikan pada spesifikasi *use case* di penelitian sebelumnya (Pertiwi, 2016). Tabel 6.1 menunjukkan rencana pengujian *blackbox*. Pengujian *blackbox* tidak berfokus pada internal sistem, tetapi berfokus kepada apa masukan yang dimasukkan dan apa keluaran yang dihasilkan apakah sesuai dengan hasil yang diharapkan.

Tabel 6.1 Rencana Pengujian

No	Kode Fitur	Nama Fitur	*Use Case*	Jenis Pengujian
1	FEAT1	Identifikasi pengguna	*Login*	*Blackbox*
2	FEAT2	Pencarian zonasi	Mencari zonasi	*Blackbox, Whitebox*
3	FEAT3	Mengelola data survey	Mengelola data survey	*Blackbox*
4	FEAT4	Mengelola data zonasi	Mengelola data peraturan zonasi	*Blackbox*
5	FEAT5	Menambah jadwal	Mengelola jadwal	*Blackbox*
6	FEAT6	Permohonan izin lokasi	Mendapatkan formulir izin lokasi Memohon izin lokasi	*Blackbox, Whitebox*
7	FEAT7	Mengelola data pengguna	Mengelola data pengguna Mengelola data pemohon	*Blackbox, Whitebox*
8	FEAT8	Mengelola data berkas	Memohon izin lokasi	*Blackbox*
9	FEAT9	Mengelola data register	Mengelola data register	*Blackbox*
10	FEAT10	Mengelola disposisi	Membuat lembar disposisi Membuat catatan disposisi	*Blackbox*
11	FEAT11	Verifikasi berkas	Memverifikasi berkas	*Blackbox*
12	FEAT12	Mengelola data pasca rapat	Mengelola data pasca rapat	*Blackbox*
13	FEAT13	Registrasi	Mendaftarkan diri	*Blackbox*

6.2 Pengujian *Blackbox*

Test case merupakan sekumpulan dari pengujian masukan, kondisi yang akan dieksekusi dan hasil yang diharapkan. Pada penelitian ini, *test case* dibuat berdasarkan skenario yang sudah didefinisikan pada spesifikasi *use case* di penelitian sebelumnya atau perancangan SILOKA (Pertiwi, 2016). Pada *test case* ini akan didefinisikan skenario yang akan dijalankan, kemudian mendefinisikan masukan, lalu keluaran yang diharapkan sesuai dengan *post-condition* dan *alternative*, dan yang terakhir adalah keluaran yang dihasilkan oleh sistem.

6.2.1 Mendapatkan Formulir Izin Lokasi

Pada tabel 6.2 merupakan *test case* dari mendapatkan formulir dimana terdapat dua skenario dan juga dua *test case*.

Tabel 6.2 Mendapatkan Formulir Izin Lokasi

Mendapatkan formulir Izin Lokasi		
Test case 1	*Use case* mendapatkan formulir izin lokasi Skenario 1	Warga berhasil mendapatkan formulir
	Masukan	Warga memilih tombol unduh formulir
	Keluaran yang diharapkan	Warga berhasil mendapatkan formulir permohonan izin lokasi
	Keluaran	Warga berhasil mendapatkan formulir permohonan izin lokasi
Test case 2	*Use case* mendapatkan formulir izin lokasi Skenario 2	Tidak jadi mendapat formulir
	Masukan	-
	Keluaran yang diharapkan	Warga tidak mendapatkan formulir
	Keluaran	Warga tidak mendapatkan formulir

Dari dua *test case* yang diuji berdasarkan skenario *use case* mendapatkan formulir pada tabel 6.2, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mendapatkan formulir.

6.2.2 Mendaftarkan Diri

Tabel 6.3 merupakan *test case* dari mendaftarkan diri, terdapat empat skenario dan empat *test case*.

Tabel 6.3 Mendaftarkan Diri

Mendaftarkan Diri		
Test case 3	*Use case* mendaftarkan diri Skenario 1	Warga berhasil melakukan pendaftaran : Basic Flow
	Masukan	Nama Pemohon : Syeh Mukhamad Iqbal Abu Dafi Nomor Identitas : 125150407111006 Alamat : Jln. Bendungan Jatiluhur no. 17 Nomor *Handphone* : 082245090093 Email : dafi@gmail.com Password : 12345
	Keluaran yang diharapkan	Warga berhasil melakukan pendaftaran dan memiliki akun untuk masuk ke dalam SILOKA
	Keluaran	Warga berhasil melakukan pendaftaran dan muncuk pesan "Anda Berhasil Mendaftar, Silahkan Login"
Test case 4	*Use case* mendaftarkan diri Skenario 2	Terdapat data yang belum diisi
	Masukan	Nama Pemohon : NULL Nomor identitas : 125150407111007 Alamat : Jln. Bendungan Nawangan no. 20 *Email* : siapa@gmail.com *Password* : 12345
	Keluaran yang diharapkan	Sistem akan menampilkan pesan untuk mengisi seluruh data
	Keluaran	Pesan peringatan untuk mengisi seluruh data
Test case 5	*Use case* mendaftarkan diri Skenario 3	Data yang didaftarkan (Nomor Identitas) sudah terdapat pada sistem
	Masukan	Nama Pemohon : Syeh Dafi Nomor Identitas : 125150407111006 Alamat : Jln. Bendungan Sengguruh no. 4 Nomor *Handphone* : 085755559399 *Email* : Syeh@gmail.com *Password* : 12345
	Keluaran yang diharapkan	Sistem akan menampilkan pesan bahwa pengguna yang didaftarkan sudah terdapat pada sistem

Tabel 6.3 Mendaftarkan diri (lanjutan)

Mendaftarkan Diri		
Test case 5	Keluaran	Sistem menampilkan pesan "Nomor identitas yang Anda masukkan sudah terdaftar"
Test case 6	*Use case* mendaftarkan diri Skenario 4	Tidak jadi melakukan pendaftaran
	Masukan	-
	Keluaran yang diharapkan	Warga tidak jadi melakukan pendaftaran
	Keluaran	Warga tidak jadi melakukan pendaftaran dan kembali ke halaman sebelumnya atau ke halaman yang dituju.

Dari empat *test case* yang diuji berdasarkan skenario *use case* mendaftarkan diri pada tabel 6.3, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mendaftarkan diri.

6.2.3 Mencari Zonasi

Tabel 6.4 merupakan *test case* dari mencari zonasi, terdapat tujuh skenario dan tujuh *test case*

Tabel 6.4 Mencari Zonasi

Mencari Zonasi		
Test case 7	*Use case* mencari zonasi Skenario 1	Berhasil melakukan pencarian
	Masukan	Kode Blok : A1 Kode Subzona : PS Peruntukan : Rumah Tunggal
	Keluaran yang diharapkan	Sistem menampilkan hasil pencarian berupa ketentuan peruntukan bangunan
	Keluaran	Sistem menampilkan hasil pencarian zonasi berupa ketentuan peruntukan bangunan
Test case 8	*Use case* mencari zonasi Skenario 2	Hasil Pencarian tidak ditemukan
	Masukan	Kode Blok : A9 Kode Subzona : PS1 Peruntukan : Rumah

Tabel 6.4 Mencari Zonasi (lanjutan)

Mencari Zonasi		
Test case 8	Keluaran yang dihasilkan	Sistem akan menampilkan pesan data tidak ditemukan dan sistem kembali menampilkan *form* pencarian
	Keluaran	Muncul pesan "Data Zonasi Tidak Ditemukan" dan kembali ke halaman *form* pencarian
Test case 9	*Use case* mencari zonasi Skenario 3	Terdapat kata kunci pencarian yang belum disi
	Masukan	Kode Blok : NULL Kode Subzona : PS Peruntukan : Rumah Tunggal
	Keluaran yang diharapkan	Sistem akan menampilkan pesan untuk memasukkan seluruh kata kunci pencarian
	Keluaran	Pesan peringatan untuk mengisi seluruh kata kunci pencarian
Test case 10	*Use case* mencari zonasi Skenario 4	Hasil pencarian matriks lebih dari satu
	Masukan	Kode Blok : A1 Kode Subzona : PS Peruntukan : Rumah
	Keluaran yang diharapkan	Sistem menampilkan opsi data matriks yang sama dengan kata kunci yang dimasukkan
	Keluaran	Sistem emnampilkan opsi data matriks yang sama dengan kata kunci yang dimasukkan
Test case 11	*Use case* mencari zonasi Skenario 5	Hasil Pencarian zonasi lebih dari satu
	Masukan	Kode Blok : A1 Kode Subzona : PS Peruntukan : Rumah Kapel
	Keluaran yang diharapkan	Sistem akan menampilkan opsi data peruntukan oznasi yang sama dengan kata kunci yang dimasukkan
	Keluaran	Sistem menampilkan opsi data zonasi yang sama dengan kata kunci yang dimasukkan

Tabel 6.4 Mencari Zonasi (lanjutan)

Mencari Zonasi		
Test case 12	*Use case* mencari zonasi Skenario 7	Tidak jadi melakukan pencarian
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan pencarian
	Keluaran	Tidak jadi melakukan pencarian dan kembali ke halaman sebelumnya atau halaman yang dituju

Dari enam *test case* yang diuji berdasarkan skenario *use case* mencari zoansi pada tabel 6.4, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mencari zonasi.

6.2.4 Mengelola Data Survey

Tabel 6.5 merupakan *test case* dari mengelola data survey

Tabel 6.5 Data Survey

Mengelola Data Survey		
Test case 13	*Use case* mengelola data survey Skenario 1	Surveyor berhasil melakukan pengelolaan data
	Masukan	Peta teknis : paper.pdf
	Keluaran yang diharapkan	Surveyor berhasil melakukan pengelolaan data penambahan data survey
	Keluaran	Surveyor berhasil melakukan pengelolaan data penambahan data survey
Test case 14	*Use case* mengelola data survey Skenario 2	Terdapat data yang belum diisi
	Masukan	Peta teknis : NULL
	Keluaran yang dihasilkan	Sistem akan menampilkan pesan bahwa terdapat data yang belum diisi
	Keluaran	Sistem menampilkan pesan bahwa terdapat data yang belum diisi
Test case 15	*Use case* mengelola data survey Skenario 3	Surveyor tidak jadi melakukan pengelolaan data

Tabel 6.5 Data Survey (lanjutan)

Mengelola Data Survey		
Test case 15	Masukan	-
	Keluaran yang diharapkan	Surveyor tidak jadi melakukan pengelolaan data
	Keluaran	Surveyor tidak jadi melakukan pengelolaan data dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju

Dari tiga *test case* yang diuji berdasarkan skenario *use case* mengelola data survey pada tabel 6.5, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mengelola data survey.

6.2.5 Mengelola Data Pasca Rapat

Tabel 6.6 merupakan *test case* dari mengelola data pasca rapat

Tabel 6.6 Mengelola Data Pasca Rapat

Mengelola Data Pasca Rapat		
Test case 16	*Use case* mengelola data pasca rapat Skenario 1	Berhasil mengelola data pasca rapat (menambah data pasca rapat pada permohonan tertentu)
	Masukan	Data pasca rapat : usecase.pdf
	Keluaran yang diharapkan	Surveyor berhasil menambahkan berkas pasca rapat pada sistem.
	Keluaran	Sistem menampilkan pesan "Data berhasil ditambahkan" dan menampilkan halaman data pasca rapat pemohon
Test case 17	*Use case* mengelola data pasca rapat Skenario 2	Format berkas yang diunggah tidak sesuai
	Masukan	Data pasca rapat : usecase.txt
	Keluaran yang diharapkan	Sistem menampilkan pesan tidak berhasil mengunggah berkas.
	Keluaran	Sistem menampilkan pesan "Tipe file yang anda unggah tidak sesuai. Harap unggah file lain"

Dari dua *test case* yang diuji berdasarkan skenario *use case* mengelola data pasca rapat pada tabel 6.6, didapatkan hasil keluaran sesuai dengan hasil keluaran

yang diharapkan yang sidah didefinisikan pada spesifikasi *use case* mengelola data pasca rapat.

6.2.6 Mengelola Data Peraturan Zonasi

Tabel 6.7 merupakan *test case* dari mengelola data peraturan zonasi

Tabel 6.7 Mengelola Data Peraturan Zonasi

Mengelola Data Peraturan Zonasi		
Test case 18	*Use case* mengelola data peraturan zonasi Skenario 1	Berhasil mengelola data peraturan zonasi (Tambah Zonasi)
	Masukan	Blok : A1 Subzona : PS Peruntukan : Rumah Tunggal Fleksibel Zoning : I Penggunaan Lahan : Lahan harus digunakan dengan semestinya Pemanfaatan Ruang : Ruang kosong harus dimanfaatkan Tata Bangunan : Tata bangunan harus sesuai dengan kriteria minimal Sarana dan Prasarana Minimal : Sarana mobilitas harus terjangkau
	Keluaran yang diharapkan	Berhasil melakukan pengelolaan data, yaitu penambahan data peraturan zonasi.
	Keluaran	Sistem menampilkan pesan “Berhasil menambahkan data peraturan zonasi”
Test case 19	*Use case* mengelola data peraturan zonasi Skenario 2	Terdapat data atau form yang kosong
	Masukan	Blok : A1 Subzona : PS Peruntukan : - Fleksibel Zoning : I Penggunaan Lahan : Lahan harus digunakan dengan semestinya

Tabel 6.7 Mengelola Data Peraturan Zonasi (lanjutan)

Mengelola Data Peraturan Zonasi		
Test case 19	Masukan	Pemanfaatan Ruang : Ruang kosong harus dimanfaatkan Tata Bangunan : Tata bangunan harus sesuai dengan kriteria minimal Sarana dan Prasarana Minimal : Sarana mobilitas harus terjangkau
	Keluaran yang diharapkan	
	Keluaran	Sistem menampilkan peringatan untuk mengisi seluruh data atau form
Test case 20	*Use case* mengelola data peraturan zonasi Skenario 3	Tidak jadi mengelola data peraturan zonasi
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan pengelolaan data
	Keluaran	Tidak jadi melakukan pengelolaan data dan kembali ke halaman sebelumnya atau ke halaman yang dituju

Dari tiga *test case* yang diuji berdasarkan skenario *use case* mengelola data peraturan zonasi pada tabel 6.7, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mengelola data peraturan zonasi.

6.2.7 Memverifikasi Berkas

Tabel 6.8 merupakan *test case* dari memverifikasi berkas

Tabel 6.8 Memverifikasi Berkas

Memverifikasi Berkas		
Test case 21	*Use case* memverifikasi berkas Skenario 1	Koordinator berhasil membuat verifikasi berkas
	Masukan	*Checkbox* Catatan verifikasi : Berkas yang Anda masukkan sudah benar

Tabel 6.8 Memverifikasi Berkas (lanjutan)

Memverifikasi Berkas		
Test case 21	Keluaran yang diharapkan	Berhasil memasukkan verifikasi berkas pada sistem
	Keluaran	Sistem menampilkan pesan "Berkas berhasil diverifikasi"
Test case 22	*Use case* memverifikasi berkas Skenario 2	Tidak jadi melakukan verifikasi
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan proses membuat verifikasi berkas pada sistem
	Keluaran	Tidak jadi melakukan proses membuat verifikasi berkas dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju

Dari dua *test case* yang diuji berdasarkan skenario *use case* memverifikasi berkas pada tabel 6.8, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* memverifikasi berkas.

6.2.8 Mengelola Jadwal

Tabel 6.9 merupakan *test case* dari mengelola jadwal

Tabel 6.9 Mengelola Jadwal

Mengelola Jadwal		
Test case 23	*Use case* mengelola jadwal Skenario 1	Proses pengelolaan jadwal berhasil
	Masukan	Jadwal Survey : 2016/08/09 Surveyor 1 : Bella Pertiwi Surveyor 2 : Indra Purnama Putra
	Keluaran yang diharapkan	Koordinator berhasil menambahkan data jadwal survey pada sistem
	Keluaran	Koordinator berhasil menambahkan data jadwal survey pada sistem
Test case 24	*Use case* mengelola jadwal Skenario 2	Terdapat jadwal yang sama

Tabel 6.9 Mengelola Jadwal (lanjutan)

Mengelola Jadwal		
Test case 24	Masukan	Jadwal rapat : 2016/08/09
	Keluaran yang diharapkan	Sistem akan menampilkan pesan bahwa jadwal rapat sama
	Keluaran	Sistem menampilkan pesan bahwa jadwal rapat sama
Test case 25	*Use case* mengelola jadwal Skenario 3	Surveyor sudah memiliki jadwal
	Masukan	Jadwal Survey : 2016/08/09 Surveyor 1 : Bella Pertiwi Surveyor 2 : Dito Priyanto
	Keluaran yang diharapkan	Sistem akan menampilkan pesan bahwa surveyor telah memiliki jadwal
	Keluaran	Sistem menampilkan pesan surveyor telah memiliki jadwal
Test case 26	*Use case* mengelola jadwal Skenario 4	Tidak jadi melakukan pengelolaan jadwal
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan pengelolaan jadwal
	Keluaran	Tidak jadi melakukan pengelolaan jadwal dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju

Dari empat *test case* yang diuji berdasarkan skenario *use case* mengelola data jadwal pada tabel 6.9, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mengelola data jadwal.

6.2.9 Mengelola Data Pengguna

Tabel 6.10 merupakan *test case* dari mengelola data pengguna

Tabel 6.10 Mengelola Data Pengguna

Mengelola Data Pengguna		
Test case 27	*Use case* mengelola	Koordinator berhasil melakukan pengelolaan data

Tabel 6.10 Mengelola Data Pengguna (lanjutan)

Mengelola Data Pengguna		
Test case 27	data pengguna Skenario 1	
	Masukan	ID Dinas : 5 Nomor KTP : 125150407111010 Nama : Supardi *Email* : supardi@gmail.com *Password* : 12345 Alamat : Jln. Bendungan Sengguruh no. 17 No. Telp : 085755559399 NIP : 125150407111010 Jabatan : Surveyor
	Keluaran yang diharapkan	Berhasil melakukan pengelolaan data pengguna pada sistem.
	Keluaran	Pesan "Berhasil Melakukan Pengelolaan data"
Test case 28	*Use case* mengelola data pengguna Skenario 2	Terdapat data yang belum diisi
	Masukan	ID Dinas : 5 Nomor KTP : 125150407111010 Nama : Supardi *Email* : NULL *Password* : 12345 Alamat : Jln. Bendungan Sengguruh no. 17 No. Telp : 085755559399 NIP : 125150407111010 Jabatan : Surveyor
	Keluaran yang diharapkan	Sistem akan menampilkan pesan bahwa terdapat data yang belum diisi.
	Keluaran	Muncul pesan peringatan untuk mengisi seluruh data
Test case 29	*Use case* mengelola data pengguna Skenario 3	Nomor Identitas sama

Tabel 6.10 Mengelola Data Pengguna (lanjutan)

Mengelola Data Pengguna		
Test case 29	Masukan	ID Dinas : 5 Nomor KTP : 125150407111006 (Sudah terdaftar) Nama : Supardi *Email* : supardi@gmail.com *Password* : 12345 Alamat : Jln. Bendungan Sengguruh no. 17 No. Telp : 085755559399 NIP : 125150407111010 Jabatan : Surveyor
	Keluaran yang diharapkan	Sistem akan menampilkan pesan terdapat pengguna yang sama
	Keluaran	Pesan "Terdapat data yang sama"
Test case 30	*Use case* mengelola data pengguna Skenario 4	*Password* lama yang dimasukkan tidak sama
	Masukan	*Password* lama : skjdngkjngrg (tidak sesuai) *Password* baru : 12345
	Keluaran yang diharapkan	Sistem menampilkan pesan bahwa *password* salah
	Keluaran	Pesan "*Password* lama salah"
Test case 31	*Use case* mengelola data pengguna Skenario 5	Tidak jadi melakukan pengelolaan data
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan pengelolaan data
	Keluaran	Tidak jadi melakukan pengelolaan data dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju

Dari lima *test case* yang diuji berdasarkan skenario *use case* mengelola data pengguna pada tabel 6.10, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mengelola data pengguna.

6.2.10 Memohon Izin Lokasi

Tabel 6.11 merupakan *test case* dari memohon izin lokasi

Tabel 6.11 Memohon Izin Lokasi

Memohon Izin Lokasi		
Test case 32	*Use case* memohon izin lokasi Skenario 1	Pemohon izin berhasil melakukan proses izin lokasi
	Masukan	Tanggal masuk : 09/08/2016 Peruntukan : Rumah Tunggal Nam Perusahaan : PT. Jasa Tirta Lokasi Dimohon : Jln. Surabaya
	Keluaran yang dihasilkan	Pemohon izin berhasil memproses permohonan izin lokasi mengajukan permohonan
	Keluaran	Pemohon izin berhasil memproses permohonan izin lokasi dan sistem menampilkan pesan "Permohonan berhasil diajukan"
Test case 33	*Use case* memohon izin lokasi Skenario 2	Tidak terdapat proses perizinan
	Masukan	-
	Keluaran yang diharapkan	Sistem akan menampilkan pesan bahwa proses izin lokasi belum dilaksanakan
	Keluaran	Sistem menampilkan pesan proses perizinan lokasi belum dilaksanakan
Test case 34	*Use case* memohon izin lokasi Skenario 3	Pemohon izin tidak jadi melakukan izin lokasi
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan proses izin lokasi
	Keluaran	Tidak jadi melakukan proses izin lokasi dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju
Test case 35	*Use case* memohon izin lokasi Skenario 4	Format berkas tidak sesuai

Tabel 6.11 Memohon Izin Lokasi (lanjutan)

Memohon Izin Lokasi		
Test case 35	Masukan	Berkas : file.txt
	Keluaran yang diharapkan	Sistem akan menampilkan pesan tidak berhasil mengunggah berkas
	Keluaran	Sistem menampilkan pesan "Tidak berhasil mengunggah berkas"

Dari empat *test case* yang diuji berdasarkan skenario *use case* memohon izin lokasi pada tabel 6.11, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* memohon izin lokasi.

6.2.11 Mengelola Data Pemohon

Tabel 6.12 merupakan *test case* dari mengelola data pemohon

Tabel 6.12 Mengelola Data Pemohon

Mengelola Data Pemohon		
Test case 36	*Use case* mengelola data pemohon Skenario 1	Berhasil mengelola data pemohon (Memperbarui nomor telepon pengguna)
	Masukan	No Telepon : 081111111111111
	Keluaran yang diharapkan	Pemohon izin berhasil melakukan pembaruan data terkait dengan identitas pemohon
	Keluaran	Pemohon izin berhasil melakukan pembaruan data identitas pemohon dan sistem menampilkan pesan "Data anda berhasil diperbarui"
Test case 37	*Use case* mengelola data pemohon Skenario 2	Tidak jadi melakukan pengelolaan data
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan pengelolaan data
	Keluaran	Tidak jadi melakukan pengelolaan data dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju
Test case 38	*Use case* mengelola data pemohon Skenario 3	*Password* salah

Tabel 6.12 Mengelola Data Pemohon (lanjutan)

Mengelola Data Pemohon		
Test case 38	Masukan	*Password* lama : 123456 Konfirmasi *Password* : 123oke
	Keluaran yang diharapkan	Sistem akan menampilkan pesan bahwa *password* salah
	Keluaran	Sistem menampilkan pesan "*Password* yang anda masukan tidak sesuai, mohon coba lagi" dan sistem menampilkan halaman memperbarui data *password* pemohon

Dari tiga *test case* yang diuji berdasarkan skenario *use case* mengelola data pemohon pada tabel 6.12, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mengelola data pemohon.

6.2.12 Mengelola Data Register

Tabel 6.13 merupakan *test case* dari mengelola data register

Tabel 6.13 Mengelola Data Register

Mengelola Data Register		
Test case 39	*Use case* mengelola data register Skenario 1	Administrasi berhasil melakukan pengelolaan data
	Masukan	Nomor KTP : 125150400111016 Nomor Registrasi : 0005/2016 Tanggal Masuk 26/04/2016 Jenis Peruntukan : Rumah Tunggal Nama Perusahaan : Joyo Group Lokasi Dimohon : Jln. MT Haryono no. 103
	Keluaran yang diharapkan	Administrasi berhasil melakukan pengelolaan data penambahan data register
	Keluaran	Administrasi berhasil melakukan penambahan data register dan sistem mengeluarkan pesan "Data Register berhasil ditambahkan"
Test case 40	*Use case* mengelola data register Skenario 2	Terdapat data yang belum diisi

Tabel 6.13 MengelolaData Register (lanjutan)

Mengelola Data Register		
Test case 40	Masukan	Nomor KTP : 125150400111016 Nomor Registrasi : NULL Tanggal Masuk : 26/04/2016 Jenis Peruntukan : Rumah Tunggal Nama Perusahaan : Joyo Group Lokasi Dimohon : Jln. MT Haryono no. 103
	Keluaran	Pesan peringatan untuk mengisi seluruh data
Test case 41	*Use case* mengelola data register Skenario 3	Data Registrasi kembar
	Masukan	Nomor KTP : 125150407111006 (Sudah terdaftar) Nomor Registrasi : 1003/2016 Tanggal masuk : 27/05/2016 Jenis Peruntukan : Rumah Tunggal Nama Perusahaan : Joyo Group Lokasi Dimohon : Jln. Bendungan Sengguruh no. 103
	Keluaran yang diharapkan	Sistem akan menampilkan pesan data sudah terdapat pada sistem
	Keluaran	Sistem menampilkan pesan "Data sudah terdapat pada sistem"
Test case 42	*Use case* mengelola data register Skenario 4	Tidak jadi melakukan pengelolaan data
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi melakukan pengelolaan data
	Keluaran	Tidak jadi melakukan pengelolaan data dan kembali ke halaman sebelumnya atau halaman yang akan dituju

Dari empat *test case* yang diuji berdasarkan skenario *use case* mengelola data register pada tabel 6.13, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* mengelola data register.

6.2.13 Membuat Lembar Disposisi

Tabel 6.14 merupakan *test case* dari membuat lembar disposisi

Tabel 6.14 Membuat Lembar Disposisi

Membuat Lembar Disposisi		
Test case 43	*Use case* membuat lembar disposisi Skenario 1	Berhasil membuat lembar disposisi
	Masukan	No Agenda : SRS1234-NF Ditujukan : Dinas Cipta Karya dan Tata Ruang Tindakan : Segera
	Keluaran yang diharapkan	Tata usaha berhasil membuat lembar disposisi dan lembar disposisi ditambahkan ke sistem
	Keluaran	Tata usaha berhasil membuat lembar disposisi dan sistem menampilkan pesan "Data berhasil ditambahkan", kemudian sistem menampilkan halaman utama lembar disposisi
Test case 44	*Use case* membuat lembar disposisi Skenario 2	Terdapat formulir yang belum diisi
	Masukan	No Agenda : SRS1234-NF Ditujukan : - Tindakan : -
	Keluaran yang diharapkan	Sistem akan menampilkan pesan untuk mengisi seluruh data
	Keluaran	Sistem menampilkan pesan untuk mengisi seluruh data
Test case 45	*Use case* membuat lembar disposisi Skenario 3	Tidak jadi melakukan pembuatan data disposisi
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi membuat lembar disposisi
	Keluaran	Tidak jadi membuat lembar disposisi dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju

Dari tiga *test case* yang diuji berdasarkan skenario *use case* membuat lembar disposisi pada tabel 6.14, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* membuat lembar disposisi.

6.2.14 Membuat Catatan Disposisi

Tabel 6.15 merupakan *test case* dari membuat catatan disposisi

Tabel 6.15 Membuat Catatan Disposisi

Membuat Catatan Disposisi		
Test case 46	*Use case* membuat catatan disposisi Skenario 1	Berhasil melakukan pembuatan catatan disposisi
	Masukan	Isi Catatan : Berkas permohonan belum lengkap
	Keluaran yang diharapkan	Pimpinan instansi berhasil membuat catatan disposisi melalui sistem
	Keluaran	Pimpinan instansi berhasil membuat catatan disposisi dan sistem menampilkan pesan "Data catatan disposisi berhasil ditambahkan"
Test case 47	*Use case* membuat catatan disposisi Skenario 2	Terdapat data yang belum diisi
	Masukan	Isi Catatan : -
	Keluaran yang diharapkan	Sistem akan menampilkan pesan data belum terisi
	Keluaran	Sistem menampilkan pesan pesan data belum terisi
Test case 48	*Use case* membuat catatan disposisi Skenario 3	Tidak jadi melakukan penambahan catatan
	Masukan	-
	Keluaran yang diharapkan	Tidak jadi menambahkan catatan
	Keluaran	Tidak jadi menambahkan catatan dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju

Dari tiga *test case* yang diuji berdasarkan skenario *use case* membuat catatan disposisi membuat catatan disposisi pada tabel 6.15, didapatkan hasil

keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case* membuat catatan disposisi.

6.2.15 *Login*

Tabel 6.16 merupakan *test case* dari *login*

Tabel 6.16 *Login*

Login		
Test case 49	*Use case login* Skenario 1	Berhasil melakukan login pada sistem
	Masukan	No. KTP : 125150407111006 *Password* : 12345
	Keluaran yang diharapkan	Aktor telah berhasil diautentifikasi sebagai pengguna terdaftar pada sistem dan terautorisasi untuk menggunakan fitur yang terdapat pada sistem
	Keluaran	Aktor telah berhasil diautentifikasi sebagai pengguna terdaftar pada sistem dan terautorisasi untuk menggunakan fitur yang terdapat pada sistem, kemudian menampilkan halaman *home* sesuai dengan aktor yang terautorisasi
Test case 50	*Use case login* Skenario 2	Terdapat data yang kosong
	Masukan	No. KTP : 125150407111006 *Password* : -
	Keluaran yang diharapkan	Sistem akan menampilkan pesan untuk mengisi nomor identitas atau kata kunci
	Keluaran	Sistem menampilkan pesan untuk mengisi nomor identitas atau kata kunci
Test case 51	*Use case login* Skenario 3	Autentikasi pengguna tidak valid
	Masukan	No. KTP : 125150407111006 *Password* : qwerty
	Keluaran yang diharapkan	Sistem akan menampilkan pesan autentifikasi gagal
	Keluaran	Sistem menampilkan pesan “Data yang anda masukan salah”
Test case 52	*Use case login* Skenario 4	Tidak jadi melakukan *login*
	Masukan	-

Tabel 6.16 *Login* (lanjutan)

Login		
Test case 52	Keluaran yang diharapkan	Tidak jadi melakukan *login*
	Keluaran	Tidak jadi melakukan *login* dan kembali ke halaman sebelumnya atau ke halaman yang akan dituju.

Dari empat *test case* yang diuji berdasarkan skenario *use case login* pada tabel 6.16, didapatkan hasil keluaran sesuai dengan hasil keluaran yang diharapkan yang sudah didefinisikan pada spesifikasi *use case login*.

6.3 Pengujian *White Box*

Pengujian *white box* dilakukan dengan tujuan untuk mendeteksi kesalahan pada kode program, juga untuk mengetahui kompleksitas yang terdapat pada kode program yang telah diimplementasikan. Untuk langkah-langkah melakukan pengujian *white box* adalah sebagai berikut:

1. Membuat *flow graph*
2. Menentukan *cyclomatic complexity*
3. Menentukan jalur independen

Berikut merupakan pengujian *white box* pada fitur SILOKA:

6.3.1 Pencarian Zonasi

Pada tabel 6.17 merupakan kode program pencarian zonasi

Tabel 6.17 Tabel kode program pencarian zonasi

```
1   public function pencarianZonasi(){
2           if($this->input->post('cariZonasi')){
3                   $subzona = $this->input->post('subzona');
4                   $blok = $this->input->post('blok');
5                   $peruntukan = $this->input->post('peruntukan');
6                   $id = $this->input->post('id') ;
7
8                   $dataMatriks = $this->M_Matriks-
9   >getDataMatriks($subzona,$blok,$peruntukan);
10                  $dataZonasi = $this->M_Zonasi-
11  >getDataZonasi($subzona,$blok,$peruntukan);
12                  $dataPtk = $this->M_Ptk->getDataPtk($subzona,$blok);
13
14                  $data['matriks'] = $dataMatriks->result();
15                  $data['zonasi'] = $dataZonasi->result ();
16                  $data['ptk'] = $dataPtk->result();
17
18                  if(($dataMatriks->num_rows() == 1) and ($dataZonasi-
19  >num_rows() == 1)){
20                          if($this->session->userdata('jabatan') ==
21  'Koordinator'){
22                                  $data['judul']="Halaman Pencarian";
23                                  $data['main']="koordinator/ketentuanZonasi
24                                  ";
25                                  $data['nav']="koordinator/nav";
26                                  $this->load->view('admin',$data);
27                          }elseif($this->session->userdata('jabatan') ==
```

Tabel 6.17 Tabel kode program pencarian zonasi (lanjutan)

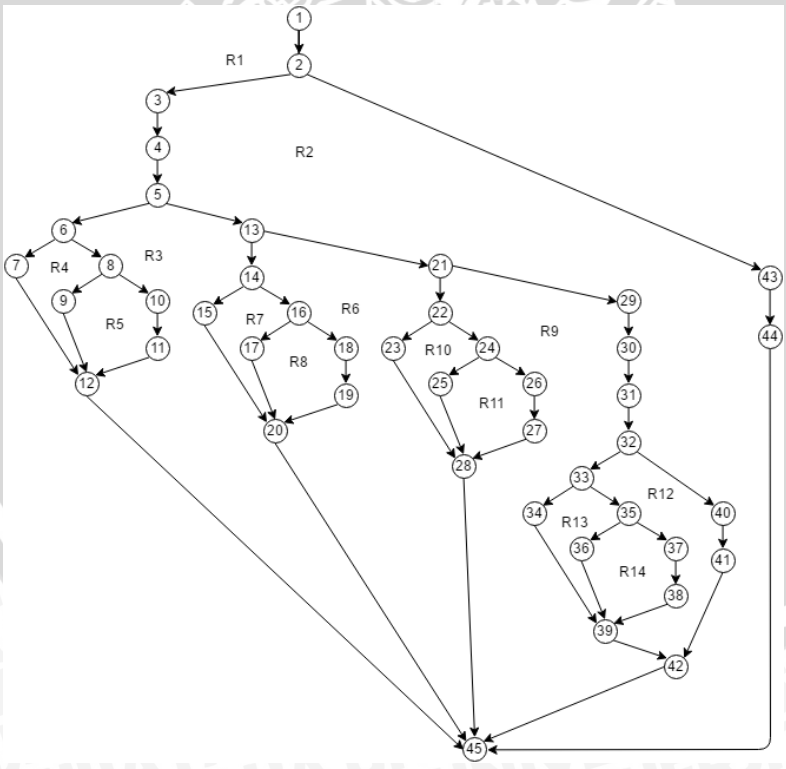
```
'Surveyor'){
                                $data['judul']="Halaman Pencarian";
                                $data['main']="surveyor/ketentuanZonasi";
                                $data['nav']="surveyor/nav";
                                $this->load->view('admin',$data);
                        }else{
                                $data['judul']="Halaman Pencarian";
                                $data['main']="warga/wargaHasilPencarian";
                                        $data['footer']="footer";
                                $data['laman']="warga/laman";
                                $this->load->view('home',$data);
                        }
                }elseif($dataMatriks->num_rows()  >1 and ($peruntukan !=
' ' or $peruntukan != NULL)){
                        if($this->session->userdata('jabatan') ==
'Koordinator'){
                                $data['judul']="Halaman Pencarian";
                                $data['main']="koordinator/ketentuanZonasi
                                Kembar";
                                $data['nav']="koordinator/nav";
                                $this->load->view('admin',$data);
                        }elseif($this->session->userdata('jabatan') ==
'Surveyor'){
                                $data['judul']="Halaman Pencarian";
                                $data['main']="surveyor/ketentuanZonasiKem
                                bar";
                                $data['nav']="surveyor/nav";
                                $this->load->view('admin',$data);
                        }else{
                                $data['judul']="Halaman Pencarian";
                                $data['main']="warga/wargaHasilPencarianKe
                                mbar";
                                $data['footer']="footer";
                                $data['laman']="warga/laman";
                                $this->load->view('home', $data);
                        }
                }elseif(($dataMatriks->num_rows() == 1) and ($dataZonasi-
>num_rows() > 1)){
                        if($this->session->userdata('jabatan') ==
'Koordinator'){
                                $data['judul']="Halaman Pencarian";
                                $data['main']="koordinator/ketentuanZonasi
                                Kembar2";
                                $data['nav']="koordinator/nav";
                                $this->load->view('admin',$data) ;
                        }elseif($this->session->userdata('jabatan') ==
'Surveyor'){
                                $data['judul']="Halaman Pencarian";
                                $data['main']="surveyor/ketentuanZonasiKem
                                bar2";
                                $data['nav']="surveyor/nav";
                                $this->load->view('admin',$data);
                        }else{
                                $data['judul']="Halaman Pencarian";
                                $data['main']="warga/wargaHasilPencarianKe
                                mbar2";
                                $data['footer']="footer";
                                $data['laman']="warga/laman";
                                $this->load->view('home',$data);
                        }
                }else{
                        $zonasi = $this->M_Zonasi->cariDataZonasi($id);
                        $data['zona'] = $zonasi->result();
                        if($zonasi->num_rows () == 1){
                                if($this->session->userdata('jabatan') ==
'Koordinator'){
                                        $data['judul']="Halaman
Pencarian";
                                        $data['main']="koordinator/ketentua
                                        nZonasi2";
```

9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

Tabel 6.17 Tabel kode program pencarian zonasi (lanjutan)

```
98                                      $data['nav']="koordinator/nav";
99                                      $this->load->view('admin',$data);
100                                 }elseif($this->session-
101 >userdata('jabatan') == 'Surveyor'){
102                                     $data['judul']="Halaman
103 Pencarian";
104                                     $data['main']="surveyor/ketentuanZo
105                                     nasi2";
106                                     $data['nav']="surveyor/nav";
107                                     $this->load->view('admin',$data);
108                                 }else{
109                                     $data['judul'] ="Halaman
110 Pencarian";
111                             $data['main']="warga/wargaHasilPencarian2";
112                                     $data['footer']="footer";
113                                     $data['laman']="warga/laman";
114                                     $this->load->view('home',$data);
115                                 }
116                             }else{
117                                 echo "<script>alert('Data Zonasi tidak
118 Ditemukan');".
119 "window.location='".base_url()."index.php/C_Pencarian/pencarian';<
120 /script>";                      }
121                     }
122             }else{
123                     redirect('C_Pengguna/registrasi','refresh');
124             }
125 }
```

Pada gambar 6.1 merupakan *flow graph* dari pencarian zonasi.



Gambar 6.1 *Flow graph* pancarian zonasi

Berikut merupakan perhitungan *cyclomatic complexity* dari pencarian zonasi:

V(G) = 14

V(G) = E – N + 2

= 57 – 45 + 2

= 14

V(G) = PN + 1

= 13 + 1 = 14

Berikut merupakan jalur independen dari pencarian zonasi:

1) 1-2-44-45-45
2) 1-2-3-4-5-6-7-12-43-45
3) 1-2-3-4-5-6-8-9-12-43-45
4) 1-2-3-4-5-6-8-10-11-12-43-45
5) 1-2-3-4-5-13-14-15-20-43-45
6) 1-2-3-4-5-13-14-16-17-20-43-45
7) 1-2-3-4-5-13-14-16-18-19-20-43-45
8) 1-2-3-4-5-13-21-22-23-28-43-45
9) 1-2-3-4-5-13-21-22-24-25-28-43-45
10) 1-2-3-4-5-13-21-22-24-26-27-28-43-45
11) 1-2-3-4-5-13-21-29-30-31-32-33-34-39-42-43-45
12) 1-2-3-4-5-13-21-29-30-31-32-33-35-36-39-42-43-45
13) 1-2-3-4-5-13-21-29-30-31-32-33-35-37-38-39-42-43-45
14) 1-2-3-4-5-13-21-29-30-31-32-40-41-42-43-45

Berdasarkan jumlah jalur independen, maka diperoleh *test case* sesuai dengan tabel 6.18.

Tabel 6.18 *Test case* pencarian zonasi

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	Data pencarian zonasi dan *session* kosong	Menuju ke halaman registrasi	Menuju ke halaman registrasi	Valid
2	Submit: cariZonasi ID Zonasi: *NULL* *Session*: Koordinator Kode blok: A1 Kode subzona: PS Peruntukan: Rumah tunggal	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan,	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan,	Valid

Tabel 6.19 *Test case* pencarian zonasi (Lanjutan)

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
		pemanfaatan ruang, tata bangunan serta sarana dan prasana pada Koordinator	pemanfaatan ruang, tata bangunan serta sarana dan prasana pada Koordinator	
3	Submit: cariZonasi ID Zonasi: *NULL* *Session*: Surveyor Kode blok: A1 Kode subzona: PS Peruntukan: Rumah tunggal	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan serta sarana dan prasana pada surveyor	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan serta sarana dan prasana pada surveyor	Valid
4	Submit: cariZonasi ID Zonasi: *NULL* *Session*: *NULL* Kode blok: A1 Kode subzona: PS Peruntukan: Rumah tunggal	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan serta sarana dan prasana pada warga	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan serta sarana dan prasana pada warga	Valid
5	Submit: cariZonasi ID Zonasi: *NULL* *Session*: Koordinator Kode blok: A1 Kode subzona: PS Peruntukan: Rumah kapel	Menampilkan halaman hasil pencarian data matriks kembar berupa daftar	Menampilkan halaman hasil pencarian data matriks kembar berupa daftar	Valid

Tabel 6.20 *Test case* pencarian zonasi (Lanjutan)

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
		matriks yang mempunyai peruntukan sama di blok yang sama pada koordinator	matriks yang mempunyai peruntukan sama di blok yang sama pada koordinator	
6	Submit: cariZonasi ID Zonasi: *NULL* *Session*: Surveyor Kode blok: A1 Kode subzona: PS Peruntukan: Rumah kapel	Menampilkan halaman hasil pencarian data matriks kembar berupa daftar matriks yang mempunyai peruntukan sama di blok yang sama pada surveyor	Menampilkan halaman hasil pencarian data matriks kembar berupa daftar matriks yang mempunyai peruntukan sama di blok yang sama pada surveyor	Valid
7	Submit: cariZonasi ID Zonasi: *NULL* *Session*: *NULL* Kode blok: A1 Kode subzona: PS Peruntukan: Rumah kapel	Menampilkan halaman hasil pencarian data matriks kembar berupa daftar matriks yang mempunyai peruntukan sama di blok yang sama pada warga	Menampilkan halaman hasil pencarian data matriks kembar berupa daftar matriks yang mempunyai peruntukan sama di blok yang sama pada warga	Valid
8	Submit: cariZonasi ID Zonasi: *NULL* *Session*: Koordinator Kode blok: A1 Kode subzona: PS Peruntukan: Rumah	Menampilkan halaman hasil pencarian data zonasi kembar berupa daftar zonasi yang mempunyai peruntukan sama pada koordinator	Menampilkan halaman hasil pencarian data zonasi kembar berupa daftar zonasi yang mempunyai peruntukan sama pada koordinator	Valid
9	Submit: cariZonasi ID Zonasi: *NULL* *Session*: Surveyor Kode blok: A1 Kode subzona: PS	Menampilkan halaman hasil pencarian data zonasi kembar	Menampilkan halaman hasil pencarian data zonasi kembar	Valid

Tabel 6.21 *Test case* pencarian zonasi (Lanjutan)

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
	Peruntukan: Rumah	berupa daftar zonasi yang mempunyai peruntukan sama pada surveyor	berupa daftar zonasi yang mempunyai peruntukan sama pada surveyor	
10	Submit: cariZonasi ID Zonasi: *NULL* *Session*: *NULL* Kode blok: A1 Kode subzona: PS Peruntukan: Rumah	Menampilkan halaman hasil pencarian data zonasi kembar berupa daftar zonasi yang mempunyai peruntukan sama pada warga	Menampilkan halaman hasil pencarian data zonasi kembar berupa daftar zonasi yang mempunyai peruntukan sama pada warga	Valid
11	Submit: cariZonasi ID Zonasi: 1 *Session*: Koordinator Kode blok: *NULL* Kode subzona: *NULL* Peruntukan: *NULL*	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan, serta sarana dan prasarana minimal pada koordinator	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan, serta sarana dan prasarana minimal pada koordinator	Valid
12	Submit: pencarianZonasi ID Zonasi: 1 *Session*: Surveyor Kode blok: *NULL* Kode subzona: *NULL* Peruntukan: *NULL*	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan, serta sarana dan prasarana minimal pada surveyor	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan, serta sarana dan prasarana minimal pada surveyor	Valid

Tabel 6.22 *Test case* pencarian zonasi (Lanjutan)

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
13	Submit: pencarianZonasi ID Zonasi: 1 *Session*: *NULL* Kode blok: *NULL* Kode subzona: *NULL* Peruntukan: *NULL*	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan, serta sarana dan prasarana minimal pada warga	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan, serta sarana dan prasarana minimal pada warga	Valid
14	Submit: pencarianZonasi ID Zonasi: *NULL* *Session*: *NULL* Kode blok: *NULL* Kode subzona: *NULL* Peruntukan: *NULL*	Menampilkan pesan "Data zonasi tidak ditemukan"	Menampilkan perisan "Data zonasi tidak ditemukan"	Valid

6.3.2 Memohon izin lokasi

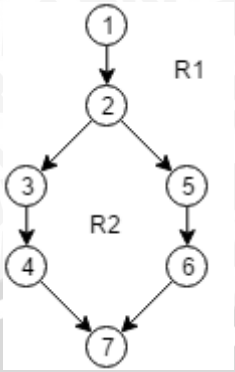
a. Melihat progress perizinan dan verifikasi berkas

Pada tabel 6.19 merupakan kode program *method* lihatDetailVerifikasi kelas C_Berkas dari melihat progress verifikasi berkas.

Tabel 6.23 Kode program lihatDetailVerifikasi

```
1  public function lihatDetailVerifikasi($id_register){
2          if($this->session->userdata('pengguna')){
3                  $data['dataVerifikasi'] = $this->M_Berkas-
4  >getVerifikasi($id_register)->result();
5                  $data['id_register'] = $id_register;
6                  $data['judul'] = "Halaman Detail Notifikasi";
7                  $data['main']="pemohon/pemohonDetailVerifikasi";
8                  $data['footer']="footer";
9                  $data['laman']="pemohon/lamanPemohonIzin";
10                 $this->load->view('home',$data);
11         }else{
12                 redirect(base_url('index.php/C_Validasi/autentifikasi'));
13         }
14 }
```

Pada gambar 6.2 merupakan *flow graph* dari melihat progress dan verifikasi berkas



Gambar 6.2 *Flow graph* melihat progress dan verifikasi berkas

Berikut merupakan perhitungan *cyclomatic complexity* dari melihat progress dan verifikasi berkas:

V(G) = 2

V(G) = E – N + 2

= 7 – 7 + 2

= 2

V(G) = PN + 1

= 1 + 1

= 2

Jalur independen:

1) 1-2-3-4-7
2) 1-2-5-6-7

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai dengan tabel 6.22

Tabel 6.24 *Test case* melihat progress dan verifikasi berkas

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	*Session*: 125150407111006 Id Register: 1	Menampilkan notifikasi kekurangan berkas berupa nama berkas dan catatan kekurangan berkas pemohon izin	Menampilkan notifikasi kekurangan berkas berupa nama berkas dan catatan kekurangan berkas pemohon izin	Valid
2	*Session*: *NULL* Id Register: 1	Menampilkan halaman *login*	Menampilkan halaman *login*	Valid

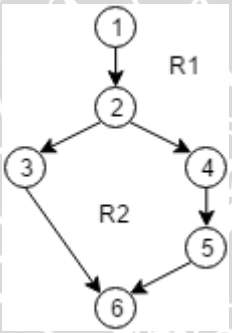
b. Melihat data pasca rapat

Pada tabel 6.23 merupakan kode program *method* lihatPascaRapat kelas C_Survey dari melihat data pasca rapat.

Tabel 6.25 Kode Program *method* lihatPascaRapat

```
1   public function lihatPascaRapat($id_register){
2           $dataPascaRapat = $this->M_SurveyRapat-
3   >getPascaRapat($id_register)->row();
4           if(empty($dataPascaRapat->BERKAS)){
5                   echo "<script>alert('Data Pasca Rapat belum tersedia') ;"
6                           .
7   "window.location='".base_url()."index.php/C_Register/lihatPermohonan';</
8   script>";
9           }else{
10                  force_download("assets/pascaRapat/".$dataPascaRapat-
11  >BERKAS."", NULL);
12          }
13  }
```

Pada gambar 6.3 merupakan *flow graph* dari melihat data pasca rapat



Gambar 6.3 *Flow graph* melihat data pasca rapat

Berikut merupakan perhitungan *cyclomatic complexity* dari melihat data pasca rapat:

V(G) = 2

V(G) = E – N + 2

= 6 – 6 + 2

= 2

V(G) = PN + 1

= 1 + 1

= 2

Berikut merupakan jalur independen dari melihat data pasca rapat:

1) 1-2-3-6
2) 1-2-4-5-6

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai dengan tabel 6.24.

Tabel 6.26 *Test case* melihat data pasca rapat

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	ID Register: *NULL*	Sistem menampilkan pesan "Data pasca rapat belum tersedia"	Sistem menampilkan pesan "Data pasca rapat belum tersedia"	Valid
2	ID Register: 1	*Download* data pasca rapat	*Download* data pasca rapat	Valid

c. Membuat data berkas

Pada tabel 6.25 merupakan kode program *method* buatDataBerkas kelas C_Berkas dari membuat data berkas.

Tabel 6.27 Kode program *method* buatDataBerkas

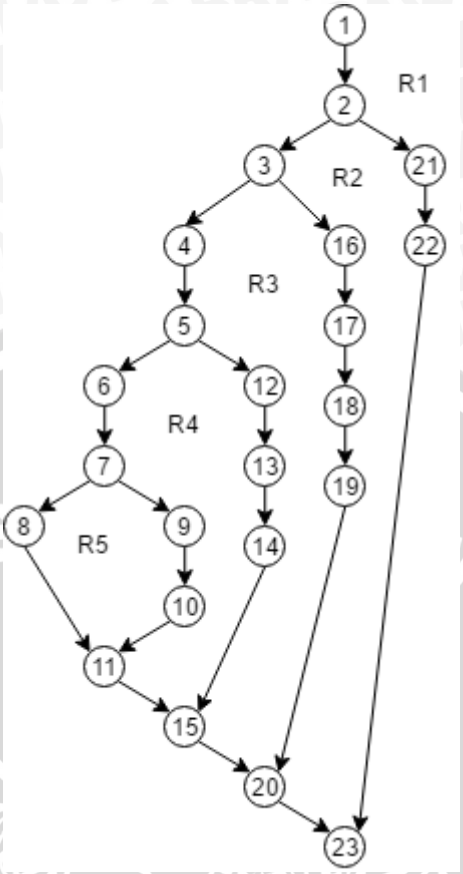
```
1   public function buatDataBerkas($id_register){
2           if($this->session->userdata('pengguna') ){
3                   if($this->input->post('upload')){
4                           $config['upload_path'] = './assets/berkas/';
5                           $config['allowed_types'] = 'pdf|doc|docx';
6                           $config['max_size']     = '30000000000000';
7                           $config['file_name'] = $this->input-
8   >post('Userfile');
9
10                          $this->load->library('upload', $config);
11                          $this->upload->initialize($config);
12
13                          if ( ! $this->upload->do_upload()){
14                                  $createdDate = date('Y-m-d H:i:s');
15                                  $file_data = $this->upload->data();
16                                  $file_name = $file_data['file_name'];
17                                  $this->M_Berkas->setBerkas($file_name);
18                                  $nama = $this->M_Berkas->get_berkas();
19                                  $tipe = explode(".", $nama);
20                                  $tipe_data = $tipe[1];
21
22                                  if($tipe_data != "pdf" and $tipe_data !=
23  "doc" and $tipe_data != "docx" and $tipe_data != NULL){
24                                          echo "<script>alert('Format file
25  harus pdf atau word, silahkan unggah lagi');"
26                                                  .
27  "window.location='".base_url()."index.php/C_Berkas/buatDataBerkas/".$id_
28  register."';</script>";
29                                  }else{
30                                          echo "<script>alert('Besar data
31  melebihi kapasitas');"
32                                                  .
33  "window.location='".base_url()."index.php/C_Berkas/buatDataBerkas/".$id_
34  register."';</script>";
35                                  }
36                          }else{
37                                  $id_register = $this->input-
38  >post('id_register');
39                                  $createdDate = date('Y-m-d H:i:s');
40
41
```

Tabel 6.25 Kode program *method* buatDataBerkas (lanjutan)

```
42
43                              $file_data = $this->upload->data();
44                              $file_name = $file_data['file_name'];
45                              $createdDate = date('Y-m-d H:i:s');
46
47                              $this->M_Register-
48  >setIdRegister($id_register);
49                              $this->M_Berkas->setBerkas($file_name);
50                              $this->M_Berkas->setJenis_berkas($this-
51  >input->post('jenis_berkas'));
52                              $this->M_Berkas->setTipe_status("B");
53                              $this->M_Berkas->setTanggal($createdDate);
54
55                              $tambahBerkas = $this->M_Berkas-
56  >tambahDataBerkas();
57                              echo "<script>alert('Berhasil menambah
58  Data Berkas');"
59                                      .
60  "window.location='".base_url()."index.php/C_Berkas/buatDataBerkas/".$id_
61  register."';</script>";
62                          }
63                  }else{
64                          $data['permohonan'] = $this->M_Berkas-
65  >getBerkasRegister($id_register)->result();
66                          $data['berkas'] = $this->M_Berkas-
67  >getDataBerkas($id_register)->result();
68                          $data['jenis1'] = $this->M_Berkas-
69  >getDataJenis(1,$id_register)->row();
70                          $data['jenis2'] = $this->M_Berkas-
71  >getDataJenis(2,$id_register)->row();
72                          $data['jenis3'] = $this->M_Berkas-
73  >getDataJenis(3,$id_register)->row();
74                          $data['jenis4'] = $this->M_Berkas-
75  >getDataJenis(4,$id_register)->row();
76
77                          $data['status1'] = $this->M_Berkas-
78  >getDataJenis(1,$id_register)->row();
79                          $data['status2'] = $this->M_Berkas-
80  >getDataJenis(2,$id_register)->row();
81                          $data['status3'] = $this->M_Berkas-
82  >getDataJenis(3,$id_register)->row();
83                          $data['status4'] = $this->M_Berkas-
84  >getDataJenis(4,$id_register)->row();
85                          $data['judul'] = "Halaman Mengelola Data Berkas";
86                          $data['main']="pemohon/pemohonKelolaBerkas";
87                          $data['footer']="footer";
88                          $data['laman']="pemohon/lamanPemohonIzin";
89                          $this->load->view('home',$data);
90                  }
91          }else{
92                  redirect(base_url('index.php/C_Validasi/autentifikasi'));
93          }
94  }
```

13 14 15 16 17 18 19 20 21 22 23

Pada gambar 6.4 merupakan *flow graph* dari membuat data berkas



Gambar 6.4 *Flow graph* membuat data berkas

Berikut merupakan perhitungan *cyclomatic complexity* dari membuat data berkas:

V(G) = 5

V(G) = E – N +2

= 26 – 23 + 2

= 5

V(G) = PN + 1

= 4 + 1

= 5

Berikut merupakan jalur independen dari membuat data berkas:

1) 1-2-21-22-23
2) 1-2-3-16-17-18-19-20-23
3) 1-2-3-4-5-12-13-14-15-20-23
4) 1-2-3-4-5-6-7-9-10-11-15-20-23
5) 1-2-3-4-5-6-7-8-11-15-20-23

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai dengan tabel 6.26.

Tabel 6.28 *Test case* membuat data berkas

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	*Session*: *NULL* Submit: *NULL* *User file*: file_benar.pdf	Sistem menampilkan halaman *login*	Sistem menampilkan halaman *login*	Valid
2	*Session*: 125150407111006 Submit: *NULL* *User file*: *NULL*	Sistem menampilkan halaman mengelola berkas	Sistem menampilkan halaman mengelola berkas	Valid
3	*Session*: 125150407111006 Submit: upload *Use file*: file_benar.pdf	Sistem menampilkan pesan "Berhasil menambah data berkas"	Sistem menampilkan pesan "Berhasil menambah data berkas"	Valid
4	*Session*: 125150407111006 Submit: upload *Use file*: file_salah.sql	Sistem menampilkan pesan "Format *file* harus pdf atau *word*, silahkan unggah lagi"	Sistem menampilkan pesan "Format *file* harus pdf atau *word*, silahkan unggah lagi"	Valid
5	*Session*: 125150407111006 Submit: upload *User file*: file_besar.docx	Sistem menampilkan pesan "Besar data melebihi kapasitas"	Sistem menampilkan pesan "Besar data melebihi kapasitas"	Valid

d. Memperbarui data berkas

Pada tabel 6.27 merupakan kode program *method* editDataBerkas kelas C_Berkas dari memperbarui berkas.

Tabel 6.29 Kode program *method* editDataBerkas

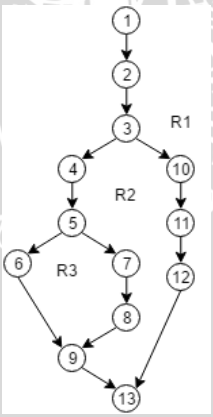
```
1   public function editDataBerkas($id){
2           $config['upload_path'] = './assets/berkas/';
3           $config['allowed_types'] = 'pdf|doc|docx';
4           $config['max_size']     = '20000000000000';
5           $config['file_name'] = $this->input->post('Userfile');
6           $this->load->library('upload', $config);
7           $this->upload->initialize($config);
8
9           if ( ! $this->upload->do_upload()){
10                  $createdDate = date('Y-m-d H:i:s');
11                  $file_data = $this->upload->data();
12                  $file_name = $file_data['file_name'];
13                  $this->M_Berkas->setBerkas($file_name);
```

Tabel 6.27 Kode program *method* editDataBerkas (Lanjutan)

```
14                  $nama = $this->M_Berkas->get_berkas();
15                  $tipe = explode(".", $nama);
16                  $tipe_data = $tipe[1];
17
18                  if($tipe_data != "pdf" and $tipe_data != "doc" and
19  $tipe_data != "docx" and $tipe_data != NULL){
20                          echo "<script>alert('Format file harus pdf atau
21  word, silahkan unggah lagi');"
22                                  .
23  "window.location='".base_url()."index.php/C_Berkas/buatDataBerkas/".$id.
24  "';</script>";
25                  }else{
26                          echo "<script>alert('Besar data melebihi
27  kapasitas');"
28                                  .
29  "window.location='".base_url()."index.php/C_Berkas/buatDataBerkas/".$id.
30  "';</script>";
31                  }
32          }else{
33                  $id_register = $this->input->post('id_register');
34                  $createdDate = date('Y-m-d H:i:s') ;
35                  $file_data = $this->upload->data();
36                  $file_name = $file_data['file_name'];
37                  $createdDate = date('Y-m-d H:i:s');
38
39                  $this->M_Berkas->setJenis_berkas($this->input-
40  >post('jenis_berkas'));
41                  $this->M_Berkas->setBerkas($file_name);
42
43                  $editBerkas = $this->M_Berkas-
44  >editDataBerkas($id_register);
45                  echo "<script>alert('Berhasil memperbarui Data Berkas');"
46                                  .
47  "window.location='".base_url()."index.php/C_Berkas/buatDataBerkas/".$id.
48  "';</script>";
49          }
50  }
```

5
6
7
8
9
10
11
12
13

Pada gambar 6.5 merupakan *flow graph* memperbarui data berkas



Gambar 6.5 *Flow graph* memperbarui data berkas

Berikut merupakan perhitungan *cyclomatic complexity* dari memperbarui data berkas:

V(G) = 3

V(G) = E – N + 2

= 14 – 13 + 2

= 3

V(G) = PN +1

= 2 + 1

= 3

Berikut merupakan jalur independen dari memperbarui data berkas:

1) 1-2-3-10-11-12-13
2) 1-2-3-4-5-6-9-13
3) 1-2-3-4-5-7-8-9-13

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai dengan tabel 6.28.

Tabel 6.30 *Test case* memperbarui data berkas

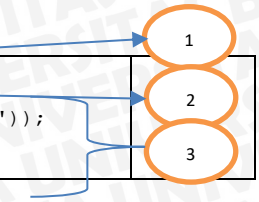
Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	ID Register: 1 ID Jenis Berkas: 1 *User file*: Berkas_benar.pdf	Sistem menampilkan pesan "Berhasil memperbarui data berkas"	Sistem menampilkan pesan "Berhasil memperbarui data berkas"	Valid
2	ID Register: 1 ID Jenis Berkas: 1 *User file*: Berkas_salah.sql	Sistem menampilkan pesan "Format *file* harus pdf atau *word*, silahkan unggah lagi"	Sistem menampilkan pesan "Format *file* harus pdf atau *word*, silahkan unggah lagi"	Valid
3	ID Register: 1 ID Jenis Berkas: 1 *User file*: Berkas_Besar.pdf	Sistem menampilkan pesan "Besar data melebihi kapasitas"	Sistem menampilkan pesan "Besar data melebihi kapasitas"	Valid

e. Mengajukan permohonan

Pada tabel 6.29 merupakan kode program *method* tambahPermohonan kelas C_Register dari mengajukan permohonan.

Tabel 6.31 Kode program *method* tambahPermohonan

```
1  public function tambahPermohonan(){
2          if($this->input->post('ajukanIzin')){
3                  $pisah = explode('/',$this->input->post('tanggal'));
4                  $urutan = array($pisah[2],$pisah[1],$pisah[0]);
5                  $tanggal_masuk = implode('-',$urutan);
```

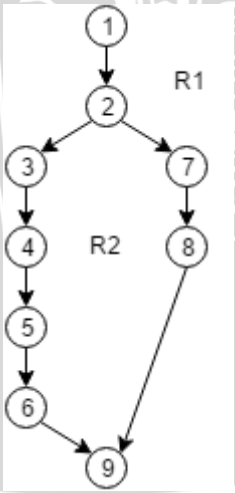


Tabel 6.29 Kode program *method* tambahPermohonan (lanjutan)

```
6                   $this->M_Register->setTanggal_masuk($tanggal_masuk);
7                   $this->M_Register->setPeruntukan($this->input-
9   >post('autocomplete'));
10                  $this->M_Register->setNama_perusahaan($this->input-
11  >post('nama_perusahaan'));
12                  $this->M_Register->setAlamat_dimohon($this->input-
13  >post('alamat_dimohon'));
14
15                  $insert = $this->M_Register->tambahDataRegister();
16                  $idRegister= $this->M_Register->getDataIdRegister($this-
17  >session->userdata('id_pengguna'))->row();
18                  $idData = $idRegister->id_register;
19
20                  $this->M_Progress->setNamaProgress('Pembuatan Data
21  Register');
22                  $this->M_Progress->setTanggal(date('Y-m-d'));
23                  $progress = $this->M_Progress-
24  >tambahProgressRegister($idData);
25
26                  echo "<script>alert('Permohonan telah berhasil
27  diajukan');"
28                          .
29  "window.location='".base_url()."index.php/C_Register/lihatPermohonan';</
30  script>";
31          }else{
32                  $data['dataPeruntukan'] = $this->M_Register-
33  >getDataPeruntukan()->result();
34                  $data['judul']="Halaman Permohonan Izin Lokasi Online";
35                  $data['main']="pemohon/pemohonIzinLokasi";
36                  $data['footer']="footer";
37                  $data['laman']="pemohon/lamanPemohonIzin";
38                  $this->load->view('home',$data);
39          }
40  }
```

4
5
6
7
8
9

Pada gambar 6.6 merupakan *flow graph* mengajukan permohonan



Gambar 6.6 *Flow graph* mengajukan permohonan

Berikut merupakan perhitungan *cyclomatic complexity* dari mengajukan permohonan:

V(G) = 2

V(G) = E – N + 2

= 9 – 9 + 2

= 2

V(G) = PN + 1

= 1 + 1

= 2

Berikut merupakan jalur independen dari mengajukan permohonan:

1) 1-2-3-4-5-6-9
2) 1-2-7-8-9

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai dengan tabel 6.30.

Tabel 6.32 *Test case* mengajukan permohonan

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	Submit: AjukanIzin Tanggal Masuk: 2016-12-28 Peruntukan: Rumah Kos Nama Perusahaan: PT Elex Media Alamat: Jln. Bend. Jatiluhur No. 17	Sistem menampilkan pesan "Permohonan telah berhasil diajukan"	Sistem menampilkan pesan "Permohonan telah berhasil diajukan"	Valid
2	Submit: *NULL* Tanggal masuk: *NULL* Peruntukan: *NULL* Nama Perusahaan: *NULL* Alamat: *NULL*	Sistem menampilkan halaman *form* pengajuan permohonan izin lokasi	Sistem menampilkan halaman *form* pengajuan permohonan izin lokasi	Valid

6.3.3 Mengelola data pengguna

a. Menambah pengguna

Pada tabel 6.31 merupakan kode program *method* masukanData kelas C_Pengguna dari menambah pengguna

Tabel 6.33 Kode program *method* masukanData

```
1  public function masukanData(){
2          $this->M_Pengguna->setNo_ktp($this->input->post('no_ktp'));
3          $this->M_Pengguna->setNama($this->input->post('nama'));
4          $this->M_Pengguna->setEmail($this->input->post('email'));
5          $this->M_Pengguna->setPassword($this->input->post('password'));
6          $this->M_Pengguna->setAlamat($this->input->post('alamat'));
7          $this->M_Pengguna->setTelepon($this->input->post('telepon'));
8
```

1

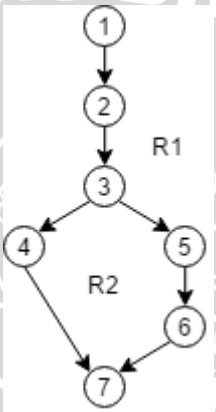
2

Tabel 6.31 Kode program *method* masukanData (Lanjutan)

```
9           $tambahPengguna = $this->M_Pengguna->simpanDataPengguna();
10
11          if($tambahPengguna == "TRUE"){
12                  echo "<script>alert('Data Pengguna Berhasil
13  ditambahkan');"
14                          .
15  "window.location='".base_url()."index.php/C_Pengguna/dataPenggunaDinas';
16  </script>";
17          }else{
18                  echo "<script>alert('Terdapat data yang sama');"
19                          .
20  "window.location='".base_url()."index.php/C_Pengguna/registrasi';</scrip
21  t>";
22          }
23  }
```

3
4
5
6
7

Pada gambar 6.7 merupakan *flow graph* dari menambah pengguna



Gambar 6.7 *Flow graph* menambah pengguna

Berikut merupakan perhitungan *cyclomatic complexity* dari menambah pengguna:

V(G) = 2

V(G) = E – N + 2

= 7 – 7 + 2

= 2

V(G) = PN + 1

= 1 + 1 = 2

Berikut merupakan jalur independen dari menambah pengguna:

1) 1-2-3-4-7
2) 1-2-3-5-6-7

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai pada tabel 6.32

Tabel 6.34 *Test case* menambah pengguna

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	Semua *form* data pengguna terisi dan $tambahPengguna = TRUE	Sistem menampilkan pesan "Data pengguna berhasil ditambahkan"	Sistem menampilkan pesan "Data pengguna berhasil ditambahkan"	Valid
2	Semua *form* data pengguna terisi dan $tambahPengguna = FALSE	Sistem menampilkan pesan "Terdapat data yang sama"	Sistem menampilkan pesan "Terdapat data yang sama"	Valid

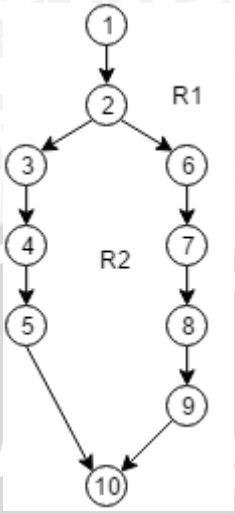
b. Memperbarui pengguna

Pada tabel 6.33 merupakan kode program *method* editDataPengguna kelas C_Pengguna dari memperbarui pengguna.

Tabel 6.35 Kode program *method* editDataPengguna

```
1   public function editDataPengguna($id){
2           if($this->input->post("editPengguna")){
3                   $this->M_Pengguna->setNama($this->input->post('nama'));
4                   $this->M_Pengguna->setEmail($this->input->post('email'));
5                   $this->M_Pengguna->setAlamat($this->input-
6   >post('alamat'));
7                   $this->M_Pengguna->setTelepon($this->input-
8   >post('telepon'));
9
10                  $editData = $this->M_Pengguna->editData($id);
11                  echo "<script>alert('Data pengguna berhasil
12  diperbarui');"
13                  .
14  "window.location='".base_url()."index.php/C_Pengguna/dataPenggunaDinas';
15  </script>";
16          }else{
17                  $dataPengguna = $this->M_Pengguna->getDataPengguna($id)-
18  >row();
19                  $this->M_Pengguna->setNo_ktp($dataPengguna->no_ktp);
20                  $this->M_Pengguna->setNama($dataPengguna->nama);
21                  $this->M_Pengguna->setEmail($dataPengguna->email);
22                  $this->M_Pengguna->setAlamat($dataPengguna->alamat);
23                  $this->M_Pengguna->setTelepon($dataPengguna->no_telp);
24                  $this->M_Dinas->setNIP($dataPengguna->nip);
25                  $this->M_Dinas->setPosisi($dataPengguna->posisi);
26
27                  $data['no_ktp'] = $this->M_Pengguna->getNo_ktp();
28                  $data['nama'] = $this->M_Pengguna->getNama();
29                  $data['email'] = $this->M_Pengguna->getEmail();
30                  $data['alamat'] = $this->M_Pengguna->getAlamat();
31                  $data['telepon'] = $this->M_Pengguna->getTelepon();
32                  $data['nip'] = $this->M_Dinas->getNip();
33                  $data['posisi'] = $this->M_Dinas->getPosisi();
34                  $data['judul']="Halaman Edit Data Pengguna";
25                  $data['main']="koordinator/editPengguna";
26                  $data['nav']="koordinator/nav";
27                  $this->load->view('admin',$data) ;
28          }
29  }
```

Pada gambar 6.8 merupakan *flow graph* dari memperbarui pengguna



Gambar 6.8 Memperbarui pengguna

Berikut merupakan perhitungan *cyclomatic complexity* dari memperbarui pengguna:

V(G) = 2

V(G) = E – N + 2

= 10 – 10 + 2

= 2

V(G) = PN + 1

= 1 + 1 = 2

Berikut merupakan jalur independen dari memperbarui pengguna:

1) 1-2-3-4-5-10
2) 1-2-6-7-8-9-10

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai tabel 6.34

Tabel 6.36 *Test case* memperbarui pengguna

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	Submit: editPengguna Semua data terisi	Sistem menampilkan pesan "Data pengguna berhasil diperbarui"	Sistem menampilkan pesan "Data pengguna berhasil diperbarui"	Valid
2	Submit: *NULL* Semua data tidak terisi	Sistem menampilkan halaman *form* edit data pengguna	Sistem menampilkan halaman *form* edit data pengguna	Valid

c. Memperbarui *password*

Pada tabel 6.35 merupakan kode program *method* editPassword kelas C_Pengguna dari memperbarui *password*.

Tabel 6.37 Kode program *method* editPassword

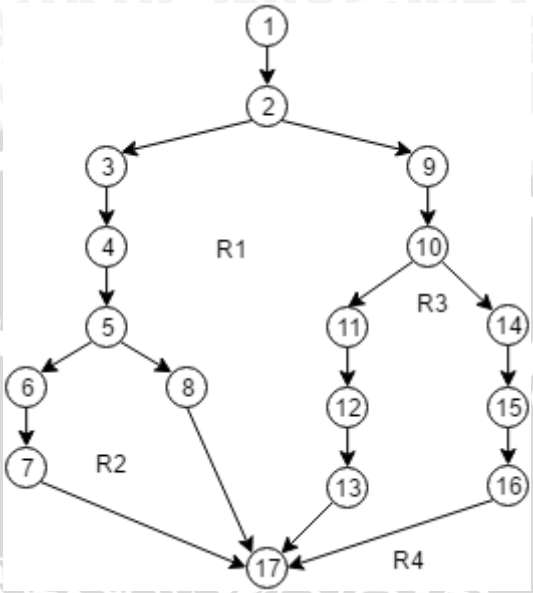
```
1   public function editPassword($id){
2           if($this->input->post('editPassword')){
3                   $this->M_Pengguna->setNo_ktp($id);
4                   $this->M_Pengguna->setPassword($this->input-
5   >post('passwordLama'));
6                   $autentikasi = $this->M_Pengguna->cekAutentikasi()-
7   >row();
8                   if(!empty($autentikasi)){
9                           $updatePassword = $this->M_Pengguna-
1   >updatePassword($id, $this->input->post('passwordBaru'));
11                          echo "<script>alert('Password Anda telah berhasil
12  diperbarui');"
13                                  .
14  "window.location='".base_url()."index.php/C_Pengguna/editPassword/".$id.
15  "';</script>";
16                  }else{
17                          echo "<script>alert('Password yang Anda masukkan
18  tidak sesuai, mohon coba lagi');"
19                                  .
20  "window.location='".base_url()."index.php/C_Pengguna/editPassword/".$id.
21  "';</script>";
22                  }
23          }else{
24                  if($this->session->userdata('isDinas') == 1){
25                          $getDataPemohon = $this->M_Pengguna-
26  >getDataPemohon($this->session->userdata('pengguna'))->row();
27                          $this->M_Pengguna->setNama($getDataPemohon-
28  >nama);
29                          $this->M_Pengguna->setNo_ktp($getDataPemohon-
30  >no_ktp);
31                          $this->M_Pengguna->setAlamat($getDataPemohon-
32  >alamat);
33                          $this->M_Pengguna->setTelepon($getDataPemohon-
34  >no_telp);
35                          $data['nama'] = $this->M_Pengguna->getNama();
36                          $data['no_ktp'] = $this->M_Pengguna->getNo_ktp();
37                          $data['alamat'] = $this->M_Pengguna->getAlamat();
38                          $data['telepon'] = $this->M_Pengguna-
39  >getTelepon();
40                          $data['judul']="Halaman Edit Password Pemohon";
41                          $data['main']="pemohon/pemohonEditPassword";
42                          $data['footer']="footer";
43                          $data['laman']="pemohon/lamanPemohonIzin";
44                          $this->load->view('home',$data);
45                  }else{
46                          $getDataPengguna = $this->M_Pengguna-
47  >getDataPengguna($id)->row();
48                          $this->M_Pengguna->setNo_ktp($getDataPengguna-
49  >no_ktp);
50                          $this->M_Pengguna->setNama($getDataPengguna-
51  >nama);
52                          $this->M_Pengguna->setAlamat($getDataPengguna-
53  >alamat);
54                          $this->M_Pengguna->setEmail($getDataPengguna-
55  >email);
56
57                          $data['no_ktp'] = $this->M_Pengguna->getNo_ktp();
58                          $data['nama'] = $this->M_Pengguna->getNama();
59                          $data['alamat'] = $this->M_Pengguna->getAlamat();
60                          $data['email'] = $this->M_Pengguna->getEmail();
61                          $data['judul']="Halaman Edit Password Pengguna";
62                          $data['main']="koordinator/editPassword";
63                          $data['nav']="koordinator/nav";
64                          $this->load->view('admin',$data);
65                  }
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Tabel 6.35 Kode program *method* editPassword (lanjutan)

66 67	} }	

Pada gambar 6.9 merupakan *flow graph* dari memperbarui *password*



Gambar 6.9 *Flow graph* memperbarui *password*

Berikut merupakan perhitungan *cyclomatic complexity* dari memperbarui *password*:

V(G) = 4

V(G) = E – N + 2

= 19 – 17 + 2

= 4

V(G) = PN + 1

= 3 + 1 = 4

Berikut merupakan jalur independen dari memperbarui *password*:

1) 1-2-3-4-5-6-7-17
2) 1-2-3-4-5-8-17
3) 1-2-9-10-11-12-13-17
4) 1-2-9-10-14-15-16-17

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai dengan tabel 6.36

Tabel 6.38 *Test case* memperbarui *password*

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	Submit: editPassword *Session*: *NULL* No Ktp: 125150407111006 *Password* Lama: 12345 *Password* Baru: 123456 (Password lama sesuai)	Sistem menampilkan pesan "*Password* Anda telah berhasil diperbarui"	Sistem menampilkan pesan "*Password* Anda telah berhasil diperbarui"	Valid
2	Submit: editPassword *Session*: *NULL* No Ktp: 125150407111006 *Password* Lama: qwerty *Passwiord* Baru: 12345 (Password lama tidak sesuai)	Sistem menampilkan pesan "*Password* yang Anda masukkan tidak sesuai, mohon coba lagi"	Sistem menampilkan pesan "*Password* yang Anda masukkan tidak sesuai, mohon coba lagi"	Valid
3	Submit: *NULL* *Session*: 1 No Ktp: *NULL* *Password* Lama: *NULL* *Password* Baru: *NULL*	Sistem menampilkan halaman *form* edit *password* pemohon izin	Sistem menampilkan halaman *form* edit *password* pemohon izin	Valid
4	Submit: *NULL* *Session*: 0 No Ktp: *NULL* *Password* Lama: *NULL* *Password* Baru: *NULL*	Sistem menampilkan halaman *form* edit *password* koordinator	Sistem menampilkan halaman *form* edit *password* koordinator	Valid

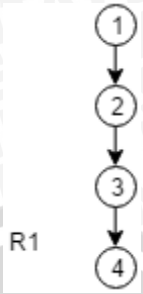
d. Hapus pengguna

Pada tabel 6.39 merupakan kode program *method* hapusData kelas C_Pengguna dari hapus pengguna.

Tabel 6.39 hapusData

```
1  public function hapusData($id){
2          $hapusData = $this->M_Pengguna->hapusPengguna($id);
3          echo "<script>alert('Akun berhasil dihapus');"
4          .
5  "window.location='".base_url()."index.php/C_Pengguna/dataPenggunaDinas';
6  </script>";
7  }
```

Pada gambar 6.10 merupakan *flow graph* dari hapus pengguna



Gambar 6.10 *Flow graph* hapus pengguna

Berikut merupakan perhitungan *cyclomatic complexity* dari hapus pengguna:

V(G) = 1

V(G) = E - N + 2

= 3 – 4 + 2 = 1

V(G) = PN + 1

= 0 + 1 = 1

Berikut merupakan jalur independen dari hapus pengguna:

1) 1-2-3-4

Berdasarkan jumlah jalur independen, diperoleh *test case* sesuai dengan tabel 6.38

Tabel 6.40 *Test case* hapus pengguna

Jalur	Data masukan	Hasil yang diharapkan	Hasil yang diperoleh	Status
1	ID Pengguna: 1	Sistem menampilkan pesan “Akun berhasil dihapus”	Sistem menampilkan pesan “Akun berhasil dihapus”	Valid

6.4 Matriks Kerunutan Pengujian

Matriks kerunutuan sangat penting dalam aktivitas pengembangan perangkat lunak, karena dapat digunakan untuk melacak kesesuaian dari fitur hingga *test case*. Pelacakan dimlau dari kode fitur yang didefinisikan, kemudian pada kode persyaratan fungsional, *use case* hingga *test case*. Pada tabel 6.18 merupakan matriks *traceability* dari pengujian *black box*.

Tabel 6.41 Matriks Kerunutan Pengujian

Kode Fitur	Kode Dasar Fungsional	Kode Lengkap Fungsional	*Use Case*	Nomor Skenario	*Test case*	Validasi
FEAT 1	SRS-F-SIL-P01	SRS-F-SIL-P01-1	*Login*	1	*Test case* 49	Valid
				2	*Test case* 50	Valid
				3	*Test case* 51	Valid
				4	*Test case* 52	Valid
FEAT 2	SRS-F-SIL-P02	SRS-F-SIL-P02-1	Mencari Zonasi	1	*Test case* 7	Valid
				2	*Test case* 8	Valid
				3	*Test case* 9	Valid
				4	*Test case* 10	Valid
				5	*Test case* 11	Valid
				6	*Test case* 12	Valid
FEAT 3	SRS-F-SIL-P03	SRS-F-SIL-P03-1	Mengelola Data Survey	1	*Test case* 13	Valid
		SRS-F-SIL-P03-2		2	*Test case* 14	Valid
		SRS-F-SIL-P03-3		3	*Test case* 15	Valid
FEAT 4	SRS-F-SIL-P04	SRS-F-SIL-P04-1	Mengelola Data Peraturan Zonasi	1	*Test case* 18	Valid
		SRS-F-SIL-P04-2		2	*Test case* 19	Valid
		SRS-F-SIL-P04-3		3	*Test case* 20	Valid
		SRS-F-SIL-P04-4				
FEAT 5	SRS-F-SIL-P05	SRS-F-SIL-P05-1	Mengelola Jadwal	1	*Test case* 23	Valid
		SRS-F-SIL-P05-2		2	*Test case* 24	Valid
		SRS-F-SIL-P05-3		3	*Test case* 25	Valid
				4	*Test case* 26	Valid
FEAT 6	SRS-F-SIL-P06	SRS-F-SIL-P06-1	Memohon Izin Lokasi	1	*Test case* 32	Valid
		SRS-F-SIL-P06-2		2	*Test case* 33	Valid
		SRS-F-SIL-P06-3		3	*Test case* 34	Valid
		SRS-F-SIL-P06-4	Mendapatkan formulir	1	*Test case* 1	Valid
		SRS-F-SIL-P06-5		2	*Test case* 2	Valid
FEAT 7	SRS-F-SIL-P07	SRS-F-SIL-P07-1	Mengelola Data Pemohon dan Mengelola Data Pengguna	1	*Test case* 27	Valid
				2	*Test case* 28	Valid
		SRS-F-SIL-P07-2		3	*Test case* 29	Valid
				4	*Test case* 30	Valid
		SRS-F-SIL-P07-3		5	*Test case* 31	Valid
				1	*Test case* 36	Valid
		SRS-F-SIL-P07-4		2	*Test case* 37	Valid
		SRS-F-SIL-P07-5		3	*Test case* 38	Valid
FEAT 8	SRS-F-SIL-P08	SRS-F-SIL-P08-1	Memohon Izin Lokasi	4	*Test case* 35	Valid
		SRS-F-SIL-P08-2				
		SRS-F-SIL-P08-3				
		SRS-F-SIL-P08-4				

Tabel 6.40 Matriks Kerunutan Pengujian (lanjutan)

Kode Fitur	Kode Dasar Fungsional	Kode Lengkap Fungsional	*Use Case*	Nomor Skenario	*Test case*	Validasi
FEAT 9	SRS-F-SIL-P09	SRS-F-SIL-P09-1	Mengelola Data Register	1	*Test case* 39	Valid
		SRS-F-SIL-P09-2		2	*Test case* 40	Valid
		SRS-F-SIL-P09-3		3	*Test case* 41	Valid
		SRS-F-SIL-P09-4		4	*Test case* 42	Valid
FEAT 10	SRS-F-SIL-P10	SRS-F-SIL-P10-1	Membuat Lembar Disposisi dan Membuat Catatan Disposisi	1	*Test case* 43	Valid
		SRS-F-SIL-P10-2		2	*Test case* 44	Valid
		SRS-F-SIL-P10-3		3	*Test case* 45	Valid
		SRS-F-SIL-P10-4		1	*Test case* 46	Valid
		SRS-F-SIL-P10-5		2	*Test case* 47	Valid
				3	*Test case* 48	Valid
FEAT 11	SRS-F-SIL-P11	SRS-F-SIL-P11-1	Memverifikasi Berkas	1	*Test case* 21	Valid
		SRS-F-SIL-P11-2		2	*Test case* 22	Valid
FEAT 12	SRS-F-SIL-P12	SRS-F-SIL-P12-1	Mengelola Data Pasca Rapat	1	*Test case* 16	Valid
		SRS-F-SIL-P12-2		2	*Test case* 17	Valid
		SRS-F-SIL-P12-3				
FEAT 13	SRS-F-SIL-P13	SRS-F-SIL-P13-1	Mendaftarkan Diri	1	*Test case* 3	Valid
				2	*Test case* 4	Valid
				3	*Test case* 5	Valid
				4	*Test case* 6	Valid

6.5 Analisis Hasil Pengujian

Berdasarkan hasil pengujian *blackbox* dan whitebox yang telah dilakukan dapat diambil kesimpulan sebagai berikut :

1. Terdapat 52 *test case* yang masing-masing *test case* diambil berdasarkan skenario yang terdapat pada setiap spesifikasi *use case*.
2. Semua *test case* yang diuji dengan pengujian *black box* mendapatkan hasil yang sesuai dengan skenario yang terdapat pada masing-masing spesifikasi *use case*, sehingga mendapatkan hasil valid.
3. Hasil pengujian *whitebox* yang telah dilakukan pada pencarian zonasi, memohon izin lokasi dan mengelola data pengguna telah menghasilkan *flow graph* dengan hasil perhitungan *cyclomatic complexity,* jalur independen dan *test case* sebagai berikut:
   a. Pencarian zonasi
      Hasil perhitungan *cyclomatic complexity* dari pencarian zonasi adalah 14, 14 jalur independen dan 14 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.
   b. Memohon izin lokasi
      1) Melihat progress perizinan dan verifikasi berkas

Hasil perhitungan *cyclomatic complexity* dari melihat progress perizinan dan verifikasi berkas adalah 2, 2 jalur independen dan 2 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.

2) Melihat data pasca rapat
   Hasil perhitungan *cyclomatic complexity* dari melihat data pasca rapat adalah 2, 2 jalur independen dan 2 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.
3) Membuat data berkas
   Hasil perhitungan *cyclomatic complexity* dari membuat data berkas adalah 5, 5 jalur independen dan 5 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.
4) Memperbarui data berkas
   Hasil perhitungan *cyclomatic complexity* dari memperbarui data berkas adalah 3, 3 jalur independen dan 3 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.
5) Mengajukan permohonan
   Hasil perhitungan *cyclomactic complexity* dari mengajukan permohonan adalah 2, 2 jalur independen dan 2 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.

c. Mengelola data pengguna
   1) Menambah pengguna
      Hasil perhitungan *cyclomatic complexity* dari menambah pengguna adalah 2, 2 jalur independen dan 2 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.
   2) Memperbarui pengguna
      Hasil perhitungan *cyclomatic complexity* dari memperbarui pengguna adalah 2, 2 jalur independen dan 2 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.
   3) Memperbarui *password*
      Hasil perhitungan *cyclomatic complexity* dari memperbarui *password* adalah 4, 4 jalur independen dan 4 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.
   4) Hapus pengguna
      Hasil perhitungan *cyclomatic complexity* dari hapus pengguna adalah 1, 1 jalur independen dan 1 *test case*. Setiap *test case* yang diujikan mendapatkan hasil valid.

BAB 7 PENUTUP

Bab ini membahas mengenai pengambilan kesimpulan serta saran dari SILOKA.

7.1 Kesimpulan

1. SILOKA dapat diimplementasikan berdasarkan perancangan yang telah dilakukan. Implementasi dilakukan dengan meninjau kembali analisis kebutuhan pada penelitian sebelumnya. Kemudian penulis melakukan penyesuaian diagram kelas perancangan dengan *framework* yang digunakan, yaitu *framework* CodeIgniter hingga menjadi kelas-kelas implementasi. Kelas implementasi didapatkan dari kelas perancangan yang telah sudah penulis sesuaikan dengan *framework* CodeIgniter, sehingga kelas implementasi mendapat penambahan kelas CI_Model dan CI_Controller. Selanjutnya penulis melakukan implementasi dengan *framework* CodeIgniter berdasarkan perancangan yang telah dilakukan untuk . Setelah SILOKA selesai diimplementasikan, kemudian dilakukan pengujian dengan menggunakan jenis pengujian *blackbox* dan *whitebox*.
2. Hasil dari perancangan detail adalah kelas-kelas implementasi, dimana kelas-kelas implementasi sudah menyesuaikan dengan *framework* CodeIgniter. Penyesuaian yang dimaksud adalah semua kelas *controller* diturunkan dari kelas CI *Controller*, sedangkan semua kelas *model* diturunkan dari kelas CI *Model*. Jadi terdapat penambahan kelas berupa CI_Model dan CI_Controller yang merupakan kelas dari *framework*. Pada kelas implementasi terdapat penambahan *method* pada model kelas impelementasi, yaitu pada *model* M_Disposisi terdapat penambahan *method* getDisposisiRegister($id) dimana *method* tersebut berfungsi untuk mengambil data pemohon dan permohonan. Penambahan *method* getDisposisiRegister($id) juga mempengaruhi diagram *sequence* membuat lembar disposisi, karena terdapat penambahan alur dari *controller* C_Disposisi *method* buatDataDisposisi() ke *model* M_Disposisi *method* getDisposisiRegister(id) ketika Tata Usaha akan menambah lembar disposisi. Penambahan *method* yang kedua adalah surveyor($id_register), dimana *method* getDisposisiRegister($id) berfungsi untuk mengambil data surveyor. Penambahan *method* surveyor($id_register) juga mempengaruhi diagram *sequence* memperbarui jadwal, karena terdapat penambahan alur dari *controller* C_Register dataJadwalRegister() ke *model* M_Jadwal *method* surveyor($id_register), dimana *method* surveyor($id_register) berfungsi mengambil data surveyor yang harus ditampilkan menggunakan perulangan ketika koordinator akan melihat detail jadwal. Hasil implementasi berupa Program SILOKA dengan fitur yang dapat dijalankan, yaitu identifikasi pengguna, pencarian zonasi, mengelola data survey, mengelola data zonasi, menambah jadwal, permohonan izin lokasi, mengelola data pengguna, mengelola data berkas,

mengelola data register, mengelola disposisi, verifikasi berkas, mengelola data pasca rapat dan registrasi.

3. Hasil pengujian *blackbox* pada SILOKA dari 15 *use case* diagram adalah berjalan sesuai dengan spesifikasi *use case* yang telah dispesifikasikan pada penelitian sebelumnya. Dari 15 *use case* diagram yang telah diimplementasikan dilakukan validasi *testing* dan hasilnya adalah valid. Sedangkan hasil pengujian *whitebox* pada SILOKA dengan menguji fitur pencarian zonasi, memohon izin lokasi, mengelola data pengguna, telah menghasilkan *flow graph* dan perhitungan kompleksitas, jalur independen dan *test case* berdasarkah jumlah jalur indepnden bahwa setiap *test case* yang diujikan mendapatkan hasil valid.

7.2 Saran

1. SILOKA diharapkan dapat dilanjutkan ke penelitian selanjutnya, yaitu dengan penambahan fitur GIS (*Geographic Information System*) pada fitur pencarian zonasi yang dapat digunakan untuk keperluan mencari data kode blok, kode subzona, dan peruntukan berdasarkan wilayah peta.
2. Penelitian dapat dilanjutkan pada penelitian selanjutnya dimana penelitian selanjutnya melanjutkan SDLC selanjutnya, yaitu penelitian tentang pengujian dari SILOKA.

DAFTAR PUSTAKA

A.S, R dan M Shalahudin. 2013. Rekayasa Perangkat Lunak Terstuktur dan Berorientasi Objek. Bandung: Informatika.

Ajo. 2016. Kemkominfo : Pengguna *internet* di Indonesia capai 82 juta. [Online]. Tersedia di : <https://kominfo.go.id/content/detail/3980/kemkominfo-pengguna-internet-di-indonesia-capai-82-juta/0/berita_satker> [Diakses 9 April 2016].

Basuki, Awan Pribadi. 2010. Membangun *Web* Berbasis PHP dengan *Framework* Codeigniter. Yogyakarta: Lokomedia.

CodeIgniter. 2014. *Application Flowchart*. [Online]. Tersedia di : <http://www.codeigniter.com/user_guide/overview/appflow.html> [Diakses 26 Juni 2016]

Danuri, Muhammad., 2013. "*Object Oriented Programming*" Pembangunan Program Aplikasi Berbasis Windows. Volume 5 No. 1, [e-journal]. Tersedia melalui: Perpustakaan Amiko JTC <http://amikjtc.com/jurnal/index.php/jurnal/article/view/35> [Diakses 1 September 2015].

DCKTR (Dinas Cipta Karya dan Tata Ruang), 2013. Tugas Pokok dan Fungsi Dinas Cipat Karya dan Tata Ruang Kabupaten Malang. [Online]. Tersedia di : <http://ciptakarya.malangkab.go.id/konten-22.html> [Diakses 5 April 2016].

Fowler, Martin. 2003. UML *Distilled Third Editian : A Brief Guide to The Standard Object Modelling Language*. [Pdf]. Tersedia di : <http://www.agentgroup.unimore.it/~nicola/courses/IngegneriaDelSoftware/uml/books/UMLDistilled.pdf> [Diakses oada 3 Mei 2016].

George H, William S, Hopwood., 2005. *Sistem Informasi Akuntansi, Buku Satu.*. Jakarta: Salemba Empat.

Gordon, B, Davis., 2010. *Kerangka Dasar Sistem Informasi Manajemen Bagian 1*. Jakarta : PT. Pustaka Binamas Pressindo.

Herlawati dan Widodo., 2011. *Menggunakan UML*. Bandung: Informatika.

IBM. 2004. *An Introduction to Structure Diagrams in UML 2 : The Class Diagram*. [Online]. Tersedia di : <https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/> [Diakses 25 Juni 2016]

IBM. 2004a. *IBM Knowledge Center: Relationship types.* [Online]. Tersedia di : <http://www.ibm.com/support/knowledgecenter/SS8PJ7_9.1.1/com.ibm.xtools.modeler.doc/topics/rreltyp.html> [Diakses 25 Desember 2016].

IBM. 2004b. *IBM Knowledge Center: Relationship in class diagrams*. [Online}. Tersedia di:

<http://www.ibm.com/support/knowledgecenter/SS8PJ7_9.1.2/com.ibm.xtools.modeler.doc/topics/crelsme_clssd.html> [Diakses 30 Desember 2016]

Indrajit., 2007. *Analisis dan Perancangan Sistem Berorientasi Objek*. Bandung: Informatika.

Jogiyanto, HM., 2005. *Sistem Teknologi Informasi*. Yogyakarta: Andi.

Leffingwell, D dan Don Widrig. 2002. *The Rational Edge: The Role of Requirements Traceability in System Development*. [Pdf}. Tersedia di: <http://www.therationaledge.com/content/sep-02/m_requirementsTraceability_di.jsp> [Diakses 5 Agustus 2016]

Malasngoding. 2016. CodeIgniter *part* 1. [Online]. Tersedia di : <http://www.malasngoding.com/pengertian-dan-cara-menggunakan-codeigniter/> [Diakses pada 30 Mei 2016].

Nidhra, S dan Jagruthi, Dondeti. 2012. *Black Box and White Box Testing Techniques – A Literature Review*. Vol. 2, No. 2. [e-Journal]. Tersedia melalui : <http://airccse.org/journal/ijesa/papers/2212ijesa04.pdf> [Diakses 1 Oktober 2016].

Nugroho, Adi., 2009. *Rekayasa Perangkat Lunak Menggunakan UML dan Java*. Yogyakarta: Andi Offset.

Nur Hidayat, Arif., 2010. Rancang Bangun dan Desain Sistem Informasi Geografis Profil Daerah Kota Blitar Berbasis *Web*. S1. Fakultas Sains dan Teknologi. Universitas Islam Negeri Maulana Malik Ibrahim. Malang.

Pertiwi, Bella., 2016. Analisis dan Perancangan Sistem Informasi Izin Lokasi (SILOKA) pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang dengan Pendekatan Berorientasi Objek. S1. Fakultas Ilmu Komputer. Universitas Brawijaya. Malang.

Pillay, Anban. 2007. *Object Oriented Programming using Java*. [pdf]. Tersedia di : <http://math.hws.edu/eck/cs124/downloads/OOP2_from_Univ_KwaZulu-Natal.pdf> [Diakses 4 Juni 2016].

Pressman, Roger S. 2010. *Software Engingeering: A Practitioner's Approach 7th*. United States: Raghothaman Srinivasan.

Robert G Murdick, dkk., 2011. *Sistem Informasi untuk Manajemen Modern*. Jakarta: Erlangga.

Sarker, H dan K. Apu., 2014. "*MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application*". Volume 7 No. 6. [e-journal]. Tersedia melalui : < http://www.sersc.org/journals/IJHIT/vol7_no5_2014/29.pdf> [Diakses 16 Juni 2016].

Sommerville, 2011. *Software Engineering 9th*. U.S.A : Addison-Wesley.

Sparx System. 2000. UML 2 *Sequence Diagram*. [Online]. Tersedia melalui : <http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_sequencediagram.html> [Diakses pada 23 Mei 2016].

Sutedjo, Budi dan Dharma Oetomo., 2012. *Perancangan dan Pengembangan Sistem Informasi*. Yogyakarta: Andi.

Utama, Candra. 2011. *CodeIgniter Framework*. [Online]. Tersedia melalui : <https://www.academia.edu/5024946/CodeIgniter_Framework_IT507_-Rekayasa_Web?auto=download> [Diakses 20 Oktober 2016}.

Utpata, Ni. L. P, dkk., 2012. "*Implementation of MVC* (*Model-View-Controller*) *Architectural to Academic Management Information System with Android Platform Base*". Volume 57 No. 8. [e-journal]. Tersedia melalui : < http://research.ijcaonline.org/volume57/number8/pxc3883313.pdf> [Diakses 31 Mei 2016].

Westfall, Linda., 2006. "*Bidirectional Requirements Traceability*". Tersedia melalui : <http://www.compaid.com/caiinternet/ezine/westfall-bidirectional.pdf> [Diakses 1 Juni 2016]

Williams, Laurie., 2006. "*Testing Overview and Black-Box Testing Techniques*". [e-journal]. Tersedia melalui : < http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf> [Diakses 27 Maret 2016].