

**PENGEMBANGAN SISTEM PEMETAAN OTOMATIS *ENTITY*
RELATIONSHIP DIAGRAM KE DALAM DATABASE**

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:

Michelle Larassati Ayusmara L

NIM: 135150201111240



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

PENGEMBANGAN SISTEM PEMETAAN OTOMATIS ENTITY RELATIONSHIP
DIAGRAM KE DALAM DATABASE

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Michelle Larassati Ayusmara L
NIM: 135150201111240

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Januari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I



Achmad Arwan, S.Kom, M.Kom
NIP: 198408152008121004

Dosen Pembimbing II



Mahardeka Tri Ananta, S.Kom., M.T.,
M.Sc.
NIK: 2016078912041001

Mengetahui

Ketua Jurusan Teknik Informatika



Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

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

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

Malang, 3 Januari 2019



Michelle Larassati Ayusmara L

NIM: 135150201111240

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas berkat, dan anugerah-Nya penulis dapat menyelesaikan tugas akhir yang berjudul “Pengembangan Sistem Pemetaan Otomatis *Entity Relationship Diagram* ke dalam *Database*” sebagai salah satu syarat untuk menyelesaikan studi di Fakultas Ilmu Komputer Universitas Brawijaya.

Dalam penyusunan laporan skripsi ini, penulis mendapatkan bantuan dan dorongan dari berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Bapak Achmad Arwan selaku dosen pembimbing I dan Mahardeka Tri Ananta selaku dosen pembimbing II yang telah membimbing penulis dengan penuh kesabaran dan memberikan ilmu serta saran dalam penyusunan skripsi ini.
2. Dosen pembimbing akademik yang telah membantu mengarahkan pelaksanaan skripsi.
3. Orang tua penulis, (Alm) Bapak Willilam Wijnand Latukolan dan Ibu Suharlin yang telah memberikan dukungan, semangat, dan doa. Saudara penulis Pinandhito Nararyawirya Latukolan dan Sharon Jedidia Amaris yang memberikan dukungan semangat.
4. Para dosen Fakultas Ilmu Komputer yang membantu penulis dalam berbagi ilmu dan pengalaman sehingga penulis dapat menyelesaikan studi dan penulisan skripsi ini.
5. Teman-teman dan sahabat-sahabat penulis yang mendukung, memberi semangat, membagi ilmu, dan membantu penulis dalam proses penulisan skripsi.
6. Dan semua pihak yang telah membantu penulis dalam menyelesaikan studi dan penulisan skripsi ini.

Semoga Tuhan selalu melimpahkan berkat untuk semua pihak yang telah disebutkan di atas. Penulis menyadari bahwa penulisan skripsi ini memiliki banyak kekurangan, untuk itu penulis sangat terbuka dalam menerima kritik dan saran yang membangun untuk penulisan skripsi ini yang lebih baik. Semoga skripsi ini berguna dan bermanfaat untuk pembaca dan semua pihak

Malang, 3 Januari 2019

Penulis

michellarass@gmail.com

ABSTRAK

Salah satu tahap dalam perancangan basis data untuk pengembangan sebuah perangkat lunak adalah proses perancangan basis data yang dibuat berdasarkan hasil perancangan model konseptual dan relasional. Proses yang cukup sulit adalah pada saat pemetaan konsep basis data. Ketika daftar kebutuhan pengguna mengalami perubahan, akan berdampak pada perancangan Sistem Basis data yang sudah dibuat sebelumnya. Selain itu, struktur basis data yang kompleks, ketika mengalami perubahan akan memakan banyak waktu untuk dipetakan kembali secara manual. Berdasarkan permasalahan tersebut, maka dibuatlah suatu Sistem untuk memetakan model basis data berbentuk ERD agar langsung dapat digunakan pada database MySQL. Dengan tujuan mempermudah pemetaan, maka sistem ini menggunakan notasi Chen. Notasi ini adalah notasi yang paling sering digunakan dalam *data modeling tool* karena bentuknya yang mudah dipahami dan digambarkan. Namun notasi ini jarang digunakan oleh aplikasi ERD mapping. Sistem ini menggunakan tujuh langkah pemetaan model relasi entitas sehingga sesuai dengan kaidah Rekaya Perangkat Lunak. Sistem dibuat dengan berbasis web, menggunakan pemrograman berorientasi objek, dan menggunakan framework codeigniter. Sistem menghasilkan 16 buah kebutuhan yang telah diuji dengan menggunakan pengujian *white-box* dan *black-box* mendapatkan hasil valid untuk keseluruhannya dan juga 95% untuk pengujian *compatibility* sistem.

Kata kunci: Pemetaan ERD, Diagram Relasi Entitas, Notasi Chen

ABSTRACT

One phase in database design for developing a software is a database design, this phase created based on conceptual and relational models. The process that is quite difficult is when mapping databases. When user requirement changed, it will have an impact on the design of the Database System that has been made before. In addition, a complex database structure, when experiencing changes will take a lot of time to be manually mapped. Based on these problems, A system was created to map the database model in the form of ERD so that it can be directly used on MySQL databases. With the goal to ease the mapping process, this system uses Chen's notation. This notation is the most used notation in the data modeling tool because it is easily read and interpreted. But this notation is rarely used by the ERD mapping application. This system uses seven steps of Entity Relationship diagram mapping so that it complies with the rules of Software Engineering. The system is made by web-based, using object-oriented programming, and using the Codeigniter framework. The system develop 16 requirement that has been tested using white-box and black-box testing in order to get best result in Chen's notation ER diagram mapping and the final testing results as expected, and compatibility testing got 95% valid result.

Keywords: ERD Mapping, Entity Relationship Diagram, Chen's Notation

DAFTAR ISI

PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	v
ABSTRACT	vi
DAFTAR ISI.....	vii
DAFTAR GAMBAR.....	ix
DAFTAR TABEL.....	xi
DAFTAR LAMPIRAN	xiv
BAB 1 PENDAHULUAN.....	15
1.1 Latar belakang	15
1.1 Rumusan masalah	16
1.2 Tujuan.....	16
1.3 Manfaat	17
1.4 Batasan masalah.....	17
1.5 Sistematika pembahasan	17
BAB 2 LANDASAN KEPUSTAKAAN	19
2.1 Kajian pustaka	19
2.2 Pemetaan <i>Entity Relationship Diagram</i>	19
2.3 Rekayasa Perangkat Lunak	23
2.4 UML (Unified Modelling Language).....	24
2.5 Analisis berorientasi objek	28
2.6 Perancangan berorientasi objek	28
2.7 Model pengembangan perangkat lunak <i>waterfall</i>	29
2.8 Pemrograman berorientasi objek	30
2.9 <i>Framework CodeIgniter</i>	30
2.10 GoJS	31
2.11 ERDPlus.....	31
2.12 Pengujian	32
BAB 3 METODOLOGI	35



3.1 Studi literatur.....	35
3.2 Analisis kebutuhan	36
3.3 Perancangan sistem	36
3.4 Implementasi.....	36
3.5 Pengujian dan analisis	37
3.6 Pengambilan kesimpulan dan saran.....	37
BAB 4 Analisis kebutuhan.....	38
4.1 Gambaran Umum Sistem	38
4.2 Identifikasi Aktor	38
4.3 Daftar Kebutuhan Fungsional.....	38
4.4 Daftar Kebutuhan Non-Fungsional.....	42
4.5 <i>Use Case Diagram</i>	42
4.6 Usecase Scenario.....	43
BAB 5 PERANCANGAN DAN IMPLEMENTASI	54
5.1 Perancangan.....	54
5.2 Implementasi.....	75
BAB 6 PENGUJIAN	93
6.1 Pengujian Unit	93
6.2 Pengujian Validasi.....	100
6.3 Pengujian Non-fungsional (Pengujian Compatibility)	110
6.4 Analisis Pengujian	110
BAB 7 PENUTUP	111
7.1 Kesimpulan	111
7.2 Saran.....	111
DAFTAR PUSTAKA.....	112
Lampiran 1 EKSPERIMEN MANDIRI.....	114
Lampiran 2 HASIL WAWANCARA	117
Lampiran 3 HASIL PENGUJIAN COMPATIBILITY	119
Lampiran 4 HASIL UJI COBA FUNGSI PEMETAAN.....	123
Lampiran 5 HASIL UJI COBA FUNGSI PEMETAAN.....	126



DAFTAR GAMBAR

Gambar 1.1 Contoh <i>mapping</i> ERD ke dalam tabel	15
Gambar 2.1 Komponen pada <i>Entity Relationship Diagram</i> (ERD).....	20
Gambar 2.2 Contoh <i>Entity Relationship Diagram</i> (ERD) Notasi Chen	21
Gambar 2.3 Contoh <i>Entity Relationship Diagram</i> (ERD) Notasi <i>Information Engineering</i>	21
Gambar 2.4 Contoh <i>Entity Relationship Diagram</i> (ERD) Notasi <i>Crow's Foot</i>	21
Gambar 2.5 Contoh <i>Entity Relationship Diagram</i> (ERD) Notasi <i>Barker</i>	22
Gambar 2.6 Contoh <i>Entity Relationship Diagram</i> (ERD) Notasi IDEF1X	22
Gambar 2.7 Aturan Format pada JSON.....	22
Gambar 2.8 <i>Layer-layer</i> pada Perangkat Lunak.....	23
Gambar 2.9 Struktur Diagram UML	24
Gambar 2.10 Langkah-Langkah Metode <i>Waterfall</i>	29
Gambar 2.11 Interaksi Objek pada Pemrograman berorientasi Objek	30
Gambar 2.12 Contoh Pembuatan <i>Diagram Editor</i> pada GoJS.....	31
Gambar 2.13 Tampilan Aplikasi Web ERD Plus.....	32
Gambar 2.14 <i>Black Box Testing</i>	32
Gambar 2.15 <i>White Box Testing</i>	33
Gambar 2.16 Jenis-Jenis Kebutuhan Non-Fungsional.....	34
Gambar 2.17 Hasil <i>Survey</i>	34
Gambar 3.1 <i>Diagram</i> Alur Metodologi Penelitian	35
Gambar 4.1 Use Case Diagram.....	43
Gambar 5.1 <i>Sequence</i> diagram Pemetaan Upload ERD.....	55
Gambar 5.2 <i>Sequence</i> diagram Pemetaan ERD baru	56
Gambar 5.3 <i>Sequence</i> diagram buat ERD	57
Gambar 5.4 <i>Sequence</i> diagram Download SQL.....	57
Gambar 5.5 Class Diagram	59
Gambar 5.6 Conceptual Data Model	68
Gambar 5.7 Perancangan antarmuka login	71
Gambar 5.8 perancangan antar muka registrasi	71
Gambar 5.9 perancangan antar muka lupa <i>password</i>	72

Gambar 5.10 perancangan antar muka detail <i>project</i> kosong	72
Gambar 5.11 perancangan antar muka detail <i>project</i>	73
Gambar 5.12 perancangan antar muka daftar <i>project member</i>	74
Gambar 5.13 perancangan antar muka Daftar <i>project</i>	74
Gambar 5.14 Perancangan Daftar <i>Member</i>	75
Gambar 5.15 perancangan buat <i>project</i> baru	75
Gambar 5.16 Hasil implementasi Basis data pada phpmyadmin	77
Gambar 5.17 Tampilan antarmuka home	86
Gambar 5.18 Tampilan antarmuka login	87
Gambar 5.19 Tampilan antarmuka registrasi	87
Gambar 5.20 Tampilan antarmuka lupa <i>password</i>	88
Gambar 5.21 Tampilan antarmuka daftar <i>member</i>	88
Gambar 5.22 Tampilan antarmuka daftar <i>project</i> oleh administrator	89
Gambar 5.23 Tampilan antarmuka daftar <i>project</i> oleh <i>member</i>	90
Gambar 5.24 Tampilan antarmuka detail <i>project</i> berisi erd yang dibuat	91
Gambar 5.25 Tampilan antarmuka detail <i>project</i> berisi erd yang diupload	91
Gambar 5.26 Tampilan antarmuka detail <i>project</i> kosong	92
Gambar 5.27 Tampilan antarmuka detail <i>project</i> berisi erd	92
Gambar 6.1 Flowgraph operasi <i>mappingErdupload</i>	94
Gambar 6.2 Flowgraph operasi <i>mappingbaru</i>	96
Gambar 6.3 Flowgraph operasi <i>downloadSql</i>	98
Gambar 6.4 Flowgraph operasi <i>addERD</i>	99

DAFTAR TABEL

Tabel 2.1 Simbol pada <i>Use case Diagram</i>	25
Tabel 2.2 Simbol pada <i>Sequence Diagram</i>	26
Tabel 2.3 Simbol pada <i>Class Diagram</i>	27
Tabel 4.1. Tabel Identifikasi Aktor	38
Tabel 4.2. Tabel Kebutuhan Fungsional User.....	39
Tabel 4.3. Tabel Kebutuhan Fungsional <i>Member</i>	40
Tabel 4.4. Tabel Kebutuhan Fungsional <i>Administrator</i>	42
Tabel 4.5. Tabel Kebutuhan Non-fungsional.....	42
Tabel 4.6. Tabel Usecase <i>scenario</i> Pemetaan ERD <i>upload</i>	44
Tabel 4.7. Tabel Usecase <i>Scenario</i> Pemetaan ERD baru.....	45
Tabel 4.8 Tabel Usecase <i>Scenario</i> Buat ERD	45
Tabel 4.9 Tabel Usecase <i>scenario download</i> Sql.....	46
Tabel 4.10 Tabel Usecase <i>scenario Edit</i> ERD.....	46
Tabel 4.11 Tabel Usecase <i>scenario login</i>	47
Tabel 4.12 Tabel Usecase <i>scenario</i> registrasi.....	47
Tabel 4.13 Tabel Usecase <i>scenario</i> lupa <i>password</i>	48
Tabel 4.14 Tabel Usecase <i>scenario logout</i>	49
Tabel 4.15 Tabel Usecase <i>scenario</i> lihat daftar <i>member</i>	49
Tabel 4.16 Tabel Usecase <i>scenario</i> hapus <i>member</i>	50
Tabel 4.17 Tabel Usecase <i>scenario</i> lihat daftar <i>project</i>	50
Tabel 4.18 Tabel Usecase <i>scenario</i> lihat daftar <i>project member</i>	51
Tabel 4.19 Tabel Usecase <i>scenario</i> buat <i>project</i> baru	52
Tabel 4.20 Tabel Usecase <i>scenario</i> hapus <i>project</i>	52
Tabel 4.21 Tabel Usecase <i>scenario</i> lihat detail <i>project</i>	53
Tabel 5.1 Perancangan detail kelas controller <i>member</i>	60
Tabel 5.2 Perancangan detail kelas controller erd	60
Tabel 5.3 Perancangan detail kelas controller admin.....	61
Tabel 5.4 Perancangan detail kelas controller <i>user</i>	61
Tabel 5.5 Potongan algoritma operasi mapping ERD <i>upload</i>	62
Tabel 5.6 Potongan algoritma operasi transform regular entity.....	63

Tabel 5.7 Potongan algoritma operasi transform relation one to one.....	64
Tabel 5.8 Potongan algoritma operasi mapping ERD baru	66
Tabel 5.9 Potongan algoritma operasi buat erd	67
Tabel 5.10 Potongan algoritma operasi download erd	67
Tabel 5.11 Struktur tabel User	68
Tabel 5.12 Struktur tabel <i>Member</i>	69
Tabel 5.13 Struktur tabel <i>Administrator</i>	69
Tabel 5.14 Struktur tabel <i>Project</i>	69
Tabel 5.15 Struktur tabel Erd	70
Tabel 5.16 Tabel Spesifikasi perangkat keras	76
Tabel 5.17 Tabel Spesifikasi Perangkat Lunak.....	76
Tabel 5.18 implementasi kelas.....	77
Tabel 5.19 Potongan kode program mapping upload	79
Tabel 5.20 Potongan kode program method transform regular	80
Tabel 5.21 Potongan kode program method transform relation one to one	81
Tabel 5.22 Potongan kode program method transform multivalued attribute ...	83
Tabel 5.23 Potongan kode program Adderd.....	85
Tabel 5.24 Potongan kode program download sql	85
Tabel 6.1 Potongan algoritma method <i>mappingUpload</i>	93
Tabel 6.2 Test Case algoritma proses mapping upload	95
Tabel 6.3 Potongan source code dari method mappingBaru	95
Tabel 6.4 Test Case algoritma proses mappingbaru	97
Tabel 6.5 Potongan source code dari method downloadSql.....	97
Tabel 6.6 Test Case algoritma proses <i>download sql</i>	98
Tabel 6.7 Potongan source code dari method addERD	99
Tabel 6.8 Test Case Algoritma proses tambah ERD	100
Tabel 6.9 hasil pengujian pemetaan ERD <i>upload</i>	100
Tabel 6.10 hasil pengujian pemetaan ERD baru	101
Tabel 6.11 hasil pengujian buat ERD.....	102
Tabel 6.12 hasil pengujian edit ERD.....	102
Tabel 6.13 hasil pengujian <i>download sql</i>	103
Tabel 6.14 hasil pengujian login.....	103



Tabel 6.15 hasil pengujian registrasi.....	104
Tabel 6.16 hasil pengujian lupa <i>password</i>	105
Tabel 6.17 Hasil pengujian <i>logout</i>	105
Tabel 6.18 Hasil pengujian lihat daftar <i>member</i>	106
Tabel 6.19 Hasil pengujian hapus <i>member</i>	106
Tabel 6.20 Hasil pengujian lihat daftar <i>project</i>	107
Tabel 6.21 Hasil pengujian lihat daftar <i>project member</i>	107
Tabel 6.22 hasil pengujian buat <i>project member</i>	108
Tabel 6.23 hasil pengujian hapus <i>project</i>	108
Tabel 6.24 hasil pengujian lihat detail <i>project</i>	109
Tabel 6.25 Hasil pengujian Compatibility.....	110



DAFTAR LAMPIRAN

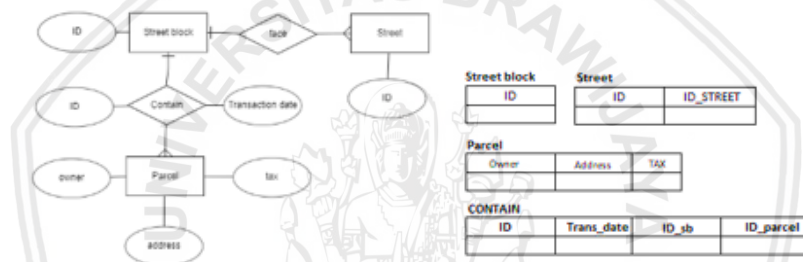
Lampiran 1 EKSPERIMEN MANDIRI.....	114
Lampiran 2 HASIL WAWANCARA	117
Lampiran 3 HASIL PENGUJIAN COMPATIBILITY	119
Lampiran 4 HASIL UJI COBA FUNGSI PEMETAAN.....	123
Lampiran 5 HASIL UJI COBA FUNGSI PEMETAAN.....	126



BAB 1 PENDAHULUAN

1.1 Latar belakang

Dalam membangun suatu perangkat lunak, terdapat beberapa tahap yang penting dan menjadi dasar keberhasilan pengembangan yaitu tahap perancangan. Dalam tahap perancangan ini, terdapat proses perancangan *database* yang terdiri dari tahap perancangan *database* secara konseptual, logikal, dan juga fisik. Perancangan *database* secara fisik dikembangkan dari perancangan *database* secara konseptual dan logikal, di mana akan menghasilkan *Entity Relationship Diagram* atau diagram yang menggambarkan apa saja entitas yang dibutuhkan oleh sistem dan juga hubungan antar entitasnya. Kesulitan yang sering terjadi pada tahap pengembangan *database* adalah pada saat pembangunan model konseptual dan pada tahap pemetaan, di mana jika ditemukan *ER Diagram* dengan struktur yang kompleks atau tidak sederhana, sehingga membutuhkan banyak waktu untuk pemetaannya.



Gambar 1.1 Contoh *mapping* ERD ke dalam tabel

Sumber : penulis.

Berdasarkan hasil eksperimen perhitungan waktu pemetaan ERD secara manual (Lampiran 1) didapatkan hasil semakin kompleks diagram yang ingin dikerjakan maka semakin banyak waktu yang dibutuhkan. Dari hasil penghitungan pada uji coba, waktu yang dibutuhkan dalam pengerjaan diagram dengan tiga buah entitas, dua buah relasi dan tujuh buah atribut seperti pada Gambar 1.1, dibutuhkan waktu tujuh menit enam detik. Dan untuk diagram dengan struktur yang lebih kompleks seperti pada lampiran, membutuhkan waktu lebih lama 13 menit, 58 detik dibandingkan diagram pada Gambar 1.1. Dari hasil eksperimen juga didapatkan bahwa apabila suatu diagram yang kompleks dikerjakan dengan waktu setengah dari waktu pengerjaan akan menghasilkan lebih banyak kesalahan dan meningkatkan probabilitas terjadinya kesalahan atau *human error*, yang berupa kesalahan pada saat pemetaan maupun kesalahan penulisan. Dari hasil observasi penulis terhadap aplikasi-aplikasi *tool* yang membantu perancangan database dengan memangkas waktu pengerjaan, hanya sedikit aplikasi yang mendukung penggunaan diagram ERD dengan notasi yang sederhana. Dari beberapa aplikasi yang ditemukan (Lampiran 5), tidak banyak yang dapat memetakan sebuah diagram ERD bernotasi sederhana menjadi tabel-tabel dalam sebuah database yang juga sudah memiliki relasi didalamnya.

Pada penelitian sebelumnya (Mohammed, Abdul, Muhammed, & Abdullah, 2015), hanya menjelaskan pendekatan seperti apa yang dibutuhkan untuk mentransformasi sebuah *ER Diagram* kedalam tabel. Melihat masalah tersebut, maka dibuatlah program untuk memetakan *ER Diagram* secara otomatis ke dalam *database*. Dalam membangun sistem automisasi ERD ini cukup sederhana dan memiliki daftar kebutuhan yang harus cukup jelas di proses awal, karena itu pengembangan sistem ini akan dibangun dengan menggunakan metode *waterfall*, yaitu dengan runtutan yang sistematis. Dari hasil observasi perangkat-perangkat lunak serupa, juga didapatkan bahwa sebagian besar menggunakan *ER Diagram* dengan notasi *crow's foot*. Perangkat lunak dengan penambahan fitur pemetaan *ER Diagram* bernotasi Chen sulit ditemukan. Padahal notasi Chen dianggap sebagai notasi yang paling sering digunakan dalam *data modeling tool* dan bentuknya yang mudah dibentuk dan dipahami (CS Odessa, 2017). Karena itu, perangkat lunak akan terfokus menggunakan notasi Chen, yang lebih mudah dimengerti dan jelas.

Dari permasalahan yang ditemui penulis, disimpulkan bahwa diperlukan sebuah alat atau *tool* yang bisa memotong waktu dan biaya pengerjaan pada saat pemetaan ERD dilakukan. Sehingga dalam proses pemetaan, akan memangkas tingkat kesalahan yang terjadi akibat *human error* dan menyingkat waktu. Berdasarkan uraian di atas maka penelitian ini mengambil judul *Pembangunan Sistem Pemetaan Otomatis Entity Diagram ke dalam Database*. Dengan program ini diharapkan dapat membantu mahasiswa maupun pengembang perangkat lunak, dalam suatu proyek untuk memudahkan perancang *database* dan juga membantu *programmer* dalam membangun *database* dengan lebih mudah, dengan waktu yang efisien, dan mengurangi tingkat kesalahan pada saat pemetaan yang disebabkan oleh *human error*.

1.1 Rumusan masalah

Berdasarkan analisis pembangunan Sistem Pemetaan Otomatis Entity Diagram ke dalam Database, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana analisis proses Sistem Pemetaan Otomatis Entity Diagram ke dalam Database?
2. Bagaimana perancangan Sistem Pemetaan Otomatis Entity Diagram ke dalam Database serupa SQL secara otomatis?
3. Bagaimana implementasi Sistem Pemetaan Otomatis Entity Diagram ke dalam Database serupa SQL secara otomatis?
4. Bagaimana hasil pengujian Sistem Pemetaan Otomatis Entity Diagram ke dalam Database?

1.2 Tujuan

Tujuan penelitian ini adalah:

1. Mengetahui proses pemetaan otomatis *Entity Relationship Diagram* ke dalam *database* sesuai kaidah Rekayasa Perangkat Lunak secara otomatis.

2. Mengetahui proses analisis dan perancangan pemetaan otomatis *Entity Relationship Diagram* ke dalam *database*.
3. Mengetahui proses implementasi pemetaan otomatis *Entity Relationship Diagram* ke dalam *database*.
4. Mengetahui hasil pengujian *black box* dan *white box* dari pemetaan otomatis *Entity Relationship Diagram* ke dalam *database*.

1.3 Manfaat

Manfaat dari penelitian ini, diharapkan dapat membantu memudahkan mahasiswa, maupun pengembang perangkat lunak dalam memetakan *Entity Relationship Diagram* dengan lebih efisien dibandingkan pemetaan secara manual.

1.4 Batasan masalah

Berdasarkan permasalahan yang disampaikan pada latar belakang, maka didapatkan batasan masalah dalam penelitian ini adalah sebagai berikut :

1. Pengembangan sistem dilakukan menggunakan Bahasa program PHP, *Javascript*, dan media manajemen *database* MySQL.
2. Menggunakan *Entity Relationship Diagram* yang dibuat dengan menggunakan Perangkat lunak *ERDPlus Online*.
3. *Entity Relationship Diagram* yang diinputkan sudah di ekspor ke dalam bentuk JSON langsung di ekspor dari *ERDPlus* dengan ekstensi “.erdplus”.
4. *Entity Relationship Diagram* dibuat dengan menggunakan notasi Chen.
5. Sistem dibangun dengan menggunakan metode pengembangan *waterfall*.
6. Tahapan pemetaan dilakukan hanya sebatas 1-6 tahapan pemetaan *Entity Relationship Diagram*.

1.5 Sistematika pembahasan

Dalam penyusunan skripsi ini terdapat penulisan yang terstruktur sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab pertama, akan berisi masalah umum terkait dengan topik penelitian yang diuraikan secara sistematis. Penulisan bab ini berisi latar belakang mengapa peneliti mengambil topik ini, rumusan masalah dari penelitian, tujuan melakukan penelitian ini, manfaat yang didapatkan dari hasil penelitian, batasan-batasan masalah apa saja yang ditentukan oleh peneliti dan terakhir sistematika penulisan penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab kedua, berisi kajian pustaka dan juga teori-teori apa saja yang digunakan pada penelitian. Teori-teori ini dipilih berdasarkan referensi yang berkaitan dan mendukung penelitian Pengembangan Sistem Pemetaan Otomatis Entity Diagram ke dalam Database.

BAB 3 METODOLOGI PENELITIAN

Kemudian, Pada bab ketiga akan menguraikan metode atau langkah-langkah apa saja yang digunakan dalam penulisan penelitian. Pada bab ini akan terdiri dari studi literatur, analisis kebutuhan, perancangan sistem perangkat lunak, implementasi sistem dari hasil perancangan, pengujian sistem dan terakhir adalah analisis hasil pengujian sistem yang akan dituangkan untuk penarikan kesimpulan.

BAB 4 ANALISIS KEBUTUHAN

Pada bab ini membahas tentang analisis kebutuhan apa saja yang dibutuhkan dalam membangun Sistem Pemetaan Otomatis Entity Diagram ke dalam Database yang akan ditampilkan dengan *use case* dan juga *use case scenario*.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Bab lima ini berisi tentang perancangan sistem yang akan diimplementasikan, yaitu Sistem Pemetaan Otomatis Entity Diagram ke dalam Database, berupa perancangan *database*, *sequence diagram*, dan *class diagram*. Dalam bab ini juga berisi, bagaimana Pengembangan Sistem Pemetaan Otomatis Entity Diagram ke dalam Database serta prosedur apa saja yang dilakukan pada proses implementasi sistem ini.

BAB 6 PENGUJIAN

Bab ini membahas tentang pengujian Pengembangan Sistem Pemetaan Otomatis Entity Diagram ke dalam Database dengan menggunakan beberapa metode. Pada bab ini akan didapatkan hasil-hasil pengujian sistem dimana digunakan untuk pengambilan kesimpulan dan saran.

BAB 7 PENUTUP

Pada Bab ketujuh, akan membahas hasil kesimpulan dari penelitian yang diteliti. Pada bab ini akan berisi apa saja yang telah didapatkan dari penelitian beserta saran yang didapatkan oleh peneliti, dimana diharapkan dengan saran-saran ini dapat bermanfaat untuk perkembangan dan perbaikan sistem.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab kajian pustaka berisi pembahasan tentang dasar teori maupun penelitian yang terkait dengan topik penelitian untuk mendukung penyusunan penelitian ini.

2.1 Kajian pustaka

Pada penelitian sebelumnya (Mohammed, Muhammed dan Abdullah 2015) dilakukan pendekatan praktik untuk mengubah *ER Diagram* ke dalam sebuah tabel. Dalam penelitian ini, *output* yang dihasilkan hanyalah sebuah tabel, dan hasil yang didapat untuk memberikan gambaran mengenai *ER Diagram*, dan pendekatan yang harus dilakukan untuk mentransform *ER Diagram* ke dalam tabel yang paling sesuai untuk diimplementasikan.

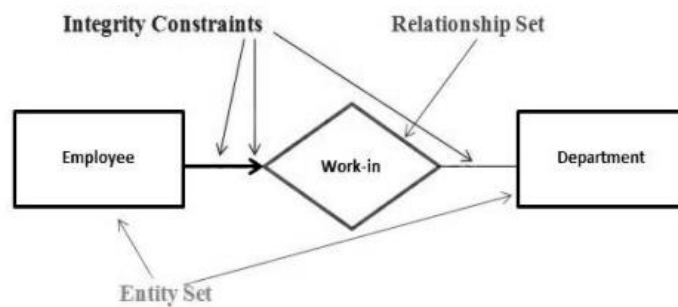
Dalam penelitian sebelumnya (Gaikwad, Kadri, Khandagle, & Tava, 2017), pemetaan dilakukan dengan menerapkan algoritme normalisasi untuk menghapus tingkat ambiguitas dalam *ER Diagram*. Dalam penelitian ini pun juga akan menghasilkan keluaran berupa *database* melalui tahap *diagram parser* dan normalisasi. Dalam penelitian ini, peneliti lebih terfokus dalam menganalisis manfaat dan bagaimana dampak dari algoritme normalisasi ERD pada sistem.

Sedangkan dalam penelitian oleh Santoso (Santoso 2004) berbentuk sama persis dengan dengan penelitian oleh Khaled (Khaled Al-Masree, 2015) yang *inputnya* bukan sebuah diagram namun sebuah *database* ataupun *script SQL* yang nantinya akan menghasilkan suatu *ER Diagram*.

2.2 Pemetaan Entity Relationship Diagram

2.2.1 Entity Relationship Diagram

Entity Relationship Diagram atau ERD adalah sebuah diagram struktural yang digunakan untuk merancang sebuah *database*. Sebuah ERD mendeskripsikan data yang akan disimpan dalam sebuah sistem maupun batasannya. Komponen utama yang terdapat di dalam sebuah ERD adalah *entity set*, *relationship set*, dan juga *constraints* (Mohammed et al., 2015). Seperti pada Gambar 2.1, sebuah *entity set* atau entitas menggambarkan obyek yang ada di dunia nyata yang dapat direpresentasikan melalui gambaran entitas itu sendiri. Sedangkan *relationship set* menggambarkan hubungan antar entitas. Dan terakhir, *integrity constraint* adalah batasan yang dari relasi yang terjadi antar entitas satu dengan yang lain.



Gambar 2.1 Komponen pada *Entity Relationship Diagram* (ERD)

Sumber: (Mohammed et al., 2015)

2.2.2 Langkah-langkah pemetaan Entity Relationship Diagram

Pemetaan ERD adalah suatu langkah di mana memetakan atau mengubah suatu diagram ERD ke dalam suatu bentuk baru dengan elemen yang sama. ERD *mapping* contohnya adalah mengubah ERD ke dalam suatu tabel. Langkah-langkah yang dilakukan dalam *mapping* adalah (Elmasri & Navathe, 2016):

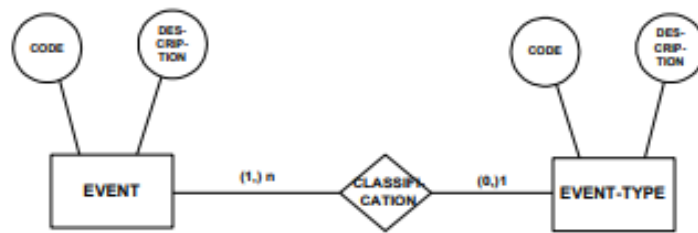
1. Memetakan entitas reguler atau entitas yang kuat,
2. Memetakan entitas lemah,
3. Memetakan entitas dengan jenis relasi 1:1 (satu dengan satu),
4. Memetakan entitas dengan jenis relasi 1:N (satu dengan banyak),
5. Memetakan entitas dengan jenis relasi N:N (banyak dengan banyak),
6. Memetakan atribut *multivalued*, atau atribut yang memiliki sekelompok nilai untuk setiap entitasnya,
7. Memetakan entitas dengan tipe relasi N-ary atau entitas dengan jumlah lebih dari 3 yang saling terhubung dalam relasi.

2.2.3 Jenis Entity Relationship Diagram berdasarkan notasi

Dalam merancang ERD, terdapat beberapa macam jenis berdasarkan notasi-notasi yang digunakan (Hay, 1999), antara lain sebagai berikut.

2.2.3.1 Notasi Chen

Notasi ini dibuat oleh Peter Chen pada tahun 1977, merupakan notasi asli dari ERD Gambar 2.2. Untuk menyatakan bahwa suatu entitas, maupun relasi itu bertipe lemah, disimbolkan dengan garis double atau garis ganda pada persegi panjang maupun lingkarannya. Scheuermann, Schiffner, dan Weber mengubah konsep asli dengan menambahkan generalisasi, agregasi, dan batasan partisipasi (Song, Evans, & Park, 1995)

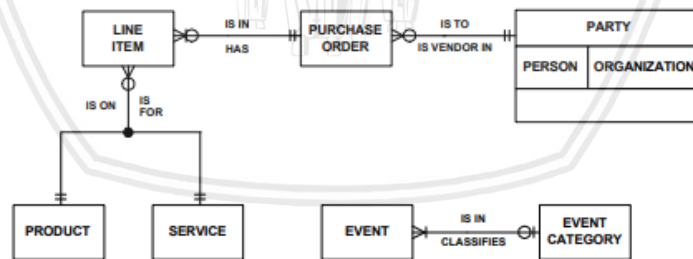


Gambar 2.2 Contoh *Entity Relationship Diagram (ERD)* Notasi Chen

Sumber: (Hay, 1999)

2.2.3.2 Notasi notasi yang lain

Notasi Crow's foot pertama kali dimunculkan pada artikel oleh Gordon Everest, dan kemudian digunakan juga pada notasi *Barker*, *SSADM*, dan juga *Information Engineering*. Pada notasi ini seperti yang tergambarkan pada Gambar 2.4, representasi entitas digambarkan dengan persegi, relasi dengan *diamond*, dan dihubungkan dengan garis yang ujungnya terdapat bentuk garis bercabang tiga yang menyerupai kaki burung. Notasi IDEF1X adalah teknik pemodelan data yang digunakan oleh banyak cabang di pemerintahan Amerika Serikat (Hay, 1999) seperti pada Gambar 2.6. yang ketiga adalah Notasi *Barker*, notasi ini awalnya dikembangkan oleh CACI, yang kemudian dipromosikan oleh Richard Barker (1990), dan kemudian diadopsi oleh perusahaan Oracle. Seperti pada Gambar 2.5. Dan kemudian ada notasi *Information engineering*, entitas ini direpresentasikan dalam bentuk persegi, yang berisi nama entity, dan atributnya tidak ditampilkan seperti pada Gambar 2.3. Untuk relasinya, menggunakan simbol garis *Crow's Foot* untuk relasi satu atau lebih, dan untuk relasi satu dan hanya satu menggunakan garis biasa



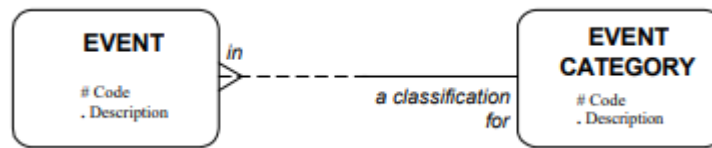
Gambar 2.3 Contoh *Entity Relationship Diagram (ERD)* Notasi *Information Engineering*

Sumber: (Hay, 1999)



Gambar 2.4 Contoh *Entity Relationship Diagram (ERD)* Notasi *Crow's Foot*

Sumber: (Song et al., 1995)



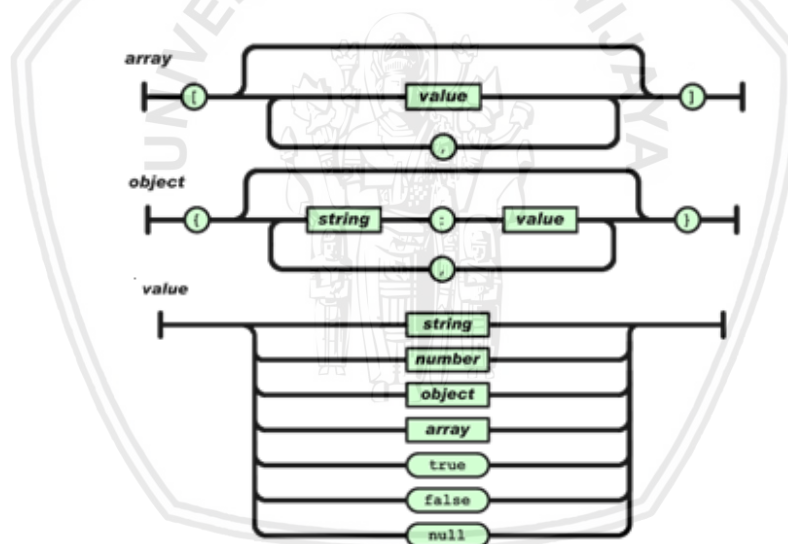
Gambar 2.5 Contoh *Entity Relationship Diagram* (ERD) Notasi Barker

Sumber: (Hay, 1999)



Gambar 2.6 Contoh *Entity Relationship Diagram* (ERD) Notasi IDEF1X

Sumber: (Hay, 1999)



Gambar 2.7 Aturan Format pada JSON

Sumber: (JSON.org, 2002)

2.2.4 JSON

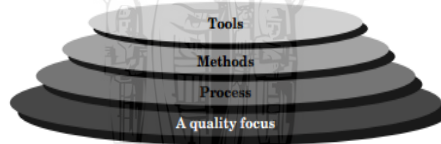
JSON atau *Javascript Object Notation* adalah *data interchange format* yang ringan (JSON.org, 2002). Format ini mudah dibaca dan ditulis oleh manusia dan juga mudah di uraikan dan diturunkan oleh mesin. Dalam JSON, terdapat sintaks dan simbol yang digunakan agar mesin dapat membaca, seperti '{ }' kurung kurawal sebagai tanda memulai dan mengakhiri objek, '[]' kurung siku sebagai tanda awal dan akhir dari sebuah *array*, ' : ' titik dua sebagai pemisah antara

sebuah kunci dan nilai dari kunci itu sendiri, dan ‘,’ koma sebagai pemisah tiap pasangan kunci dan nilai dengan yang berikutnya (Bassett, 2015). Gambar 2.7, merupakan contoh dari aturan *format* penulisan JSON. Sebuah *value*, dapat berupa *string*, dengan diberi ‘ ’’ petik dua pada penulisannya. Dapat juga berupa angka, objek, *array*, bahkan boolean dan juga null.

2.2.5 SQL

Structured Query Language atau yang dikenal dengan SQL adalah bahasa yang berdomain spesifik yang digunakan dalam pemrograman untuk mengelola data pada sistem manajemen *database* relasional (RDBMS). SQL pertama kali dikembangkan oleh Donald D. Chamberlin dan Raymond F. Boyce pada tahun 1970an di IBM. Pada awalnya diberi nama SEQUEL atau *Structured English Query Language*.

SQL terdiri dari beberapa elemen bahasa yaitu *Clauses* atau klausa yang berfungsi sebagai komponen penyusun dari kueri atau pernyataan, *Expression* atau ekspresi yang menghasilkan nilai atau tabel berisi kolom dan baris data, *Predicates* atau predikat yang menentukan kondisi dalam bentuk boolean, *Queries* atau kueri yang mengambil data berdasarkan suatu kriteria tertentu, *Statements* atau pernyataan yang memiliki efek tetap pada skema atau data yang mengontrol transaksi, alur program, koneksi, sesi, maupun diagnostik, dan terakhir adalah *Insignificant whitespace* atau spasi yang umumnya diabaikan dalam pernyataan di SQL, namun dibutuhkan untuk mempermudah pembacaan kode.



Gambar 2.8 *Layer-layer* pada Perangkat Lunak

Sumber: (Pressman, 2001)

2.3 Rekayasa Perangkat Lunak

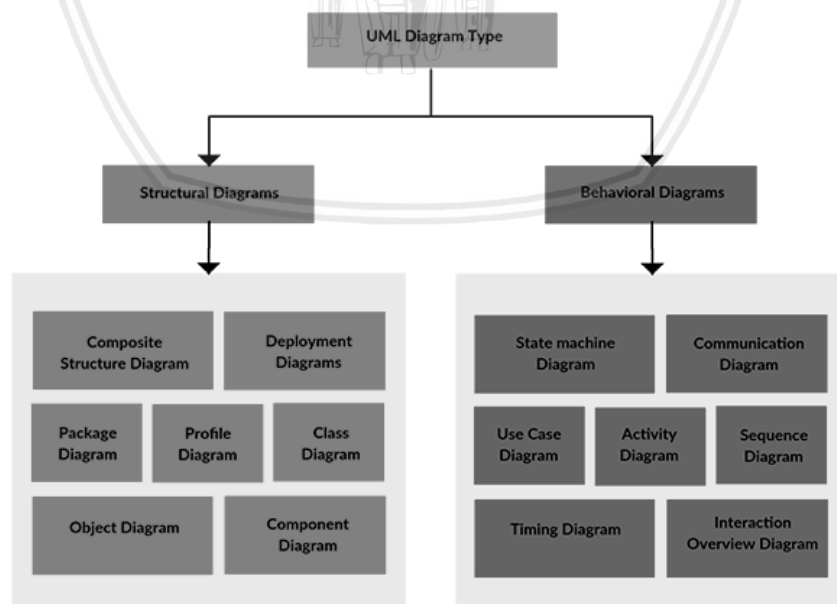
Perangkat lunak tidak hanya berupa aplikasi interface pada komputer saja melainkan juga semua dokumen-dokumen perancangan hingga pengujian yang saling berhubungan. Rekayasa perangkat lunak atau yang disingkat dengan RPL adalah pembuatan dan penggunaan prinsip-prinsip dalam dunia teknik untuk menghasilkan perangkat lunak yang ekonomis. Disini yang dimaksud dengan perangkat lunak ekonomis adalah perangkat lunak yang dapat diandalkan dan juga bekerja secara efisien pada mesin yang sesungguhnya (Pressman, 2001) dan merupakan sebuah prinsip perkerayaan yang berhubungan dengan semua aspek pembuatan perangkat lunak dari tahap awal spesifikasi sampai tahap perawatan sistem setelah digunakan (Sommerville, 2011). Rekayasa perangkat lunak dapat dibagi menjadi 3 *layer*. Seperti pada Gambar 2.8, terdapat lapisan fondasi paling

bawah yaitu *a quality focus*, kemudian proses, *methods* dan *tools*. Proses sebagai fondasi dari rekayasa perangkat lunak, *methods* menyediakan rangkaian teknis untuk membangun perangkat lunak, dan *tools* menyediakan pendukung untuk *process* dan *methods* secara otomatis maupun semi-otomatis (Pressman, 2001).

2.4 UML (Unified Modelling Language)

Dikembangkan pada 1990an oleh James Rumbaugh, Grady Booch, dan Ivar Jacobson, UML atau *Unified Modelling Language* adalah bahasa grafik standar yang digunakan untuk memodelkan perangkat lunak yang berbasis *Object Oriented* (Lethbridge & Leganiere, 2002). UML digunakan untuk memahami, merancang, melihat, mengkonfigurasi, memelihara, dan mengatur informasi pada sistem. Selain itu juga digunakan pada semua metode pengembangan, setiap tahapan, domain aplikasi, dan juga media (Rumbaugh, Jacobson, & Booch, 2004).

Pada tahun 1997, UML pun diadopsi menjadi standar oleh *Object Management Group*. Menggunakan UML dianggap menguntungkan karena sebagai notasi standar, siapapun yang menganalisis menjadi lebih mudah karena menginterpretasikan tanpa adanya kesalahpahaman. Selain itu, banyak *tools* yang dapat membantu membangun UML model dan dapat menyimulasikan, menganimasikan, maupun menggenerasikan kode dari sistem (Lethbridge & Leganiere, 2005). Diagram UML, terdiri dari dua macam yaitu *structural diagram* dan *behavior diagram*. Pada Gambar 2.9 dijelaskan bahwa *Composite Structure Diagram*, *Deployment Diagram*, *Package Diagram*, *Profil Diagram*, *Class diagram*, *Object Diagram*, dan *Component Diagram* adalah bagian dari diagram struktural pada UML. Sedangkan *State Machine Diagram*, *Communication Diagram*, *Use case diagram*, *Activity Diagram*, *Sequence Diagram*, *Timing Diagram*, *Interaction Overview Diagram* adalah bagian dari diagram perilaku pada UML.



Gambar 2.9 Struktur Diagram UML

Sumber: (Nishadha, 2012)


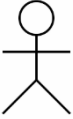


2.4.1 Use case diagram

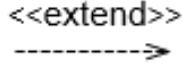
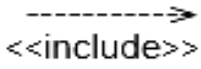
Use case diagram adalah diagram yang digunakan pada tahap elisitasi dan analisis kebutuhan sebagai sebuah grafik yang merepresentasikan fungsional kebutuhan dari sistem (Williams, 2004). *Use case* merepresentasikan sebagian besar fungsional utama yang utuh dari awal hingga akhir pada sistem (Bruegge & Dutoit, 2000). *Use case* juga merupakan representasi visual hubungan antara aktor dan juga kumpulan skenario sebagaimana sistem bertindak atau bekerja. Terdapat beberapa kesalahan yang sering dilakukan pada pembuatan *use case* (Kurniawan, 2018), antara lain:

1. *Use case* yang dibuat berurutan sesuai tahapan eksekusi tiap fungsi, seharusnya *use case* dibuat dengan tidak menggambarkan urutan eksekusi oleh aktor,
2. Batasan sistem yang tidak konsisten,
3. Identifikasi aktor yang tidak tepat,
4. *Use case* yang digambarkan secara abstrak,
5. *Use case* tidak lengkap tergambarkan,
6. Generalisasi aktor yang tidak tepat, dan
7. Format yang tidak terstruktur.

Dengan mengetahui klasifikasi kesalahan-kesalahan yang sering terjadi, diharapkan dapat mengatasi kesalahan yang umumnya ditemui pada pembuatan *use case diagram*. Pada Tabel 2.1 menjelaskan notasi-notasi yang ada pada *use case diagram*, beserta nama notasi dan deskripsinya.

Tabel 2.1 Simbol pada *Use case Diagram*

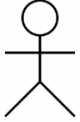

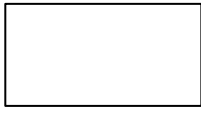

Notasi	Nama Notasi	Deskripsi
	<i>Use case</i>	Skenario untuk mendeskripsikan tentang cara kerja sistem pada sebuah situasi untuk mencapai tujuan.
	Aktor	Aktor adalah manusia atau sistem lain yang dapat berinteraksi dengan sistem yang akan dibangun.
	Asosiasi	Menggambarkan interaksi antara <i>use case</i> dengan aktor
	Generalisasi	Apabila <i>use case</i> yang mirip dengan <i>use case</i> lain atau aktor yang mirip dengan aktor lain. Arah panah mengarah

Notasi	Nama Notasi	Deskripsi
		pada <i>use case</i> atau aktor yang menggeneralisasinya.
	<i>Extend</i>	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan dapat berdiri sendiri. Arah panah menghadap pada <i>use case</i> yang ditambahkan.
	<i>Include</i>	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> atau aktor yang menjadi generalisasinya.

2.4.2 Sequence diagram

Sequence diagram atau diagram rangkaian merupakan diagram yang digunakan pada tahap analisis dan juga perancangan. Diagram ini seringkali digunakan untuk menggambarkan kronologi atau runtutan alur dari setiap kejadian atau objek pada *use case*. Diagram ini merepresentasikan tingkah laku sistem berdasarkan interaksi yang dibutuhkan oleh tiap set objek dalam bertukar pesan/informasi untuk menghasilkan hasil yang diinginkan (Williams, 2004). Berikut ini pada Tabel 2.2 berisi notasi-notasi yang terdapat pada *sequence diagram*.

Tabel 2.2 Simbol pada *Sequence Diagram*

Notasi	Nama Notasi	Deskripsi
	Aktor	Manusia atau sistem lain yang dapat berinteraksi dengan sistem yang akan dibangun.
	Message/Pesan	Menyatakan suatu obyek memanggil operasi yang ada pada objek lain atau objek itu sendiri.
	Objek	Objek yang berinteraksi.
	<i>Return</i>	Menyatakan nilai kembalian dari sebuah operasi yang dijalankan pada suatu objek

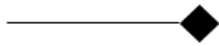
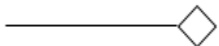
Notasi	Nama Notasi	Deskripsi
		dan juga menyatakan pembentukan objek baru.
-	<i>Lifeline</i>	Kehidupan dari suatu objek.

2.4.3 Class diagram

Class diagram atau diagram kelas merupakan kelas yang digunakan pada tahap analisis dan perancangan. Kelas diagram yang dibuat pada tahap analisis digunakan untuk mendeskripsikan kelas-kelas dan relasi-relasi di dalam *problem domain*, namun tidak mensugesti bagaimana suatu sistem diimplementasikan. Namun pada tahap perancangan, kelas ini digunakan untuk mendeskripsikan bagaimana seharusnya sistem yang ingin di implementasikan untuk dikembangkan (Williams, 2004). Pada Tabel 2.3 ini berisi notasi-notasi pada *class diagram*, beserta nama notasi, dan deskripsinya.

Tabel 2.3 Simbol pada *Class Diagram*

Notasi	Nama Notasi	Deskripsi						
<table border="1" style="margin: auto;"> <tr> <td style="text-align: center;"><i>Nama Class</i></td> </tr> <tr> <td style="text-align: center;">+ atribut</td> </tr> <tr> <td style="text-align: center;">+ atribut</td> </tr> <tr> <td style="text-align: center;">+ atribut</td> </tr> <tr> <td style="text-align: center;">+ <i>method</i></td> </tr> <tr> <td style="text-align: center;">+ <i>method</i></td> </tr> </table>	<i>Nama Class</i>	+ atribut	+ atribut	+ atribut	+ <i>method</i>	+ <i>method</i>	Nama Kelas	Objek pada sistem yang memiliki atribut dan operasi.
	<i>Nama Class</i>							
	+ atribut							
+ atribut								
+ atribut								
+ <i>method</i>								
+ <i>method</i>								
	Atribut	Untuk memberikan karakteristik atau ciri yang mendeskripsikan kelas.						
	Method	Perilaku yang mencerminkan suatu kelas.						
—	Asosiasi	Hubungan antarkelas yang di mana suatu kelas memanggil objek dari kelas lainnya.						
- - - - ->	Dependensi	Hubungan antarkelas yang di mana <i>method</i> suatu kelas membutuhkan objek dari kelas lain untuk dijadikan parameter.						
—>	Generalisasi	Hubungan <i>inheritance</i> antara anak kelas dengan induk kelas.						

Notasi	Nama Notasi	Deskripsi
	Komposisi	Hubungan antar kelas yang saling bergantung satu sama lain.
	Agregasi	Hubungan antar kelas yang di mana kelas tersebut merupakan bagian dari kelas lain tetapi tidak saling bergantung.

2.5 Analisis berorientasi objek

Analisis berbasis objek atau *Object Oriented Analysis (OOA)* adalah metode untuk memeriksa dan menganalisis kebutuhan-kebutuhan dari perspektif kelas dan objek yang ditemukan dalam domain masalah, walaupun perbedaan batasan antara analisis dan perancangan berbasis objek sangat kabur (Booch, 1994). Namun, dapat dibedakan dengan fokus analisis yang terfokus pada apa yang harus dikerjakan, bukan bagaimana mengerjakan sebuah sistem. Dalam analisis berbasis objek, terdapat beberapa pendekatan yang digunakan dan relevan dengan sistem berbasis *objek* (Booch, 1994), yaitu:

1. Pendekatan klasik, yaitu dengan penurunan kelas dari kebutuhan-kebutuhan yang ada dalam domain masalah.
2. *Behavior analysis*, pendekatan ini terfokus dengan interaksi dinamis sebagai sumber utama dari kelas maupun objek.
3. *Domain analysis*, pendekatan ini terfokus dengan salah satu aplikasi yang spesifik dan mengidentifikasi kelas dan objek yang umum digunakan untuk semua aplikasi dengan domain yang spesifik.
4. *Use case analysis*, berbeda dengan pendekatan klasik, *behavior*, maupun *domain*, pendekatan ini dapat digabungkan dengan ketiga pendekatan di awal, untuk mendorong proses analisis ke arah yang lebih berguna.
5. *CRC Card* atau *Class Responsibility Collaboration Card*, adalah salah satu cara yang efektif untuk menganalisis skenario.
6. *Informal English Description*, adalah suatu cara alternatif dari pendekatan klasik. Dalam pendekatan ini dimulai dengan menuliskan deskripsi dari masalah dalam bahasa Inggris. Namun, pendekatan ini tidak dapat menyelesaikan dengan baik masalah yang besar.

2.6 Perancangan berorientasi objek

Object Oriented Design (OOD) atau perancangan berbasis objek adalah sebuah metode untuk merancang yang meliputi penguraian proses berbasis objek dan

sebuah notasi untuk menggambarkan model logis, fisik, statis dan dinamis pada sistem yang sedang dirancang (Booch, 1994). Dalam perancangan objek ini, terdapat dua macam bagian, yaitu bagian yang mengarah ke dekomposisi berbasis objek dan menggunakan notasi berbeda untuk menunjukkan model berbeda dari perancangan logika dan fisik dari sistem. Berbeda dengan analisis berbasis objek, dalam tahap ini terfokus dengan bagaimana membangun sistem.

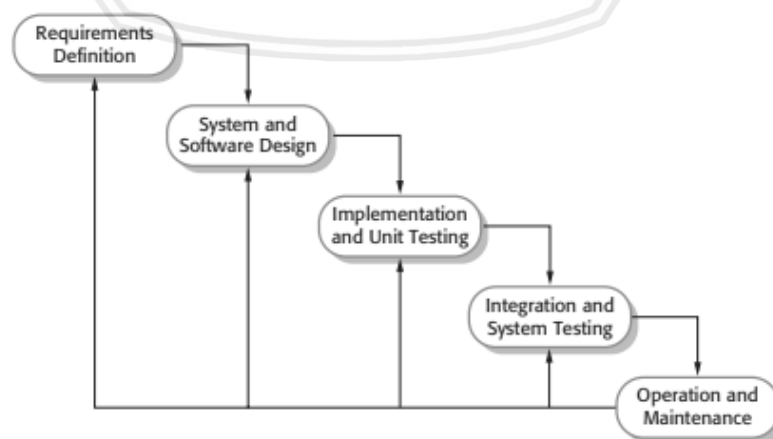
2.7 Model pengembangan perangkat lunak *waterfall*

Model pengembangan *waterfall* adalah model yang bersifat sistematis dan sangat klasik yang digunakan dalam membangun suatu perangkat lunak. Dengan menggunakan model ini, pengaplikasian yang cenderung lebih mudah dan kebutuhan sistem dapat didefinisikan secara utuh, eksplisit, dan benar dari awal proyek. Namun, menggunakan model ini sangat sulit apabila terjadi suatu perubahan, karena akan merubah atau merombak dari tahap awal sistem.

2.7.1 Tahap-tahap *waterfall*

Dalam metode *waterfall*, terdapat beberapa tahapan seperti yang terdapat pada Gambar 2.10. Tahap-tahap yang ada antara lain:

1. Mencari kebutuhan, tahap analisis terhadap kebutuhan perangkat lunak, dan tahap untuk mengumpulkan data dengan wawancara kepada *user*, maupun pengumpulan data dengan *survey* atau observasi.
2. Perancangan sistem perangkat lunak, merupakan lanjutan dari proses menggali kebutuhan, untuk menghasilkan dokumen *user requirement*.
3. Implementasi dan *unit testing*.
4. Integrasi dan *system testing*.
5. Pengoperasian dan perawatan sistem, pada tahap ini perangkat lunak atau sistem yang sudah melewati tahap-tahap sebelumnya sudah dapat digunakan oleh *user* dan akan terus dilakukan pemeliharaan secara berkala.



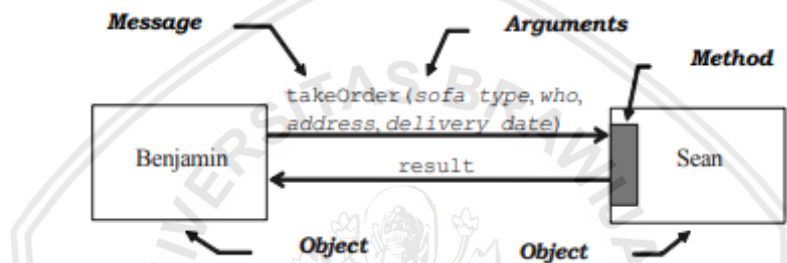
Gambar 2.10 Langkah-Langkah Metode *Waterfall*

Sumber: (Sommerville, 2011)

2.8 Pemrograman berorientasi objek

Pemrograman berbasis objek berbeda dengan pendekatan pemrograman terdahulunya yang berdasarkan alur prosedur. Pada pendekatan ini, menghasilkan pemodelan objek, dan interaksi di dalam masalahnya berdasarkan objek-objek beserta interaksinya. Dengan menggunakan pendekatan ini menghasilkan sistem yang lebih dekat dengan representasi dari domain masalah pada dunia nyata dibandingkan dengan pendekatan terstruktur (Poo, Kiong, & Ashok, 2008).

Seperti pada Gambar 2.11, dapat dilihat bahwa pendekatan yang dilakukan lebih menyerupai dengan keadaan aslinya apabila Benjamin memesan (*take order*) kepada Sean, dengan memberikan pesan beserta argumen, atau dalam dunia nyata berisi nama pemesan, alamat, tanggal, maupun tipe sofa yang akan dipesan kemudian diterima oleh Sean. Nantinya Sean pun akan menghasilkan *result* sesuai dengan apa yang diminta Benjamin.



Gambar 2.11 Interaksi Objek pada Pemrograman berorientasi Objek

Sumber: (Poo et al., 2008)

2.9 Framework CodeIgniter

Framework atau kerangka kerja adalah suatu teknik bersifat *object oriented* yang bersifat *reusable* dari seluruh bagian dari sistem yang direpresentasikan dengan abstrak-abstrak kelas dan bagaimana mereka berinteraksi (Johnson, 1997). Sebuah kelas abstrak tidak memiliki instansiasi, maka digunakan sebagai *template* untuk membuat *subclass* daripada membuat *template* untuk membuat objek baru. *Framework* menggunakan mereka sebagai rancangan komponennya. *Framework* menggunakan keuntungan dari karakteristik *object oriented*, yaitu kelas abstrak, *polymorphism*, dan *inheritance*.

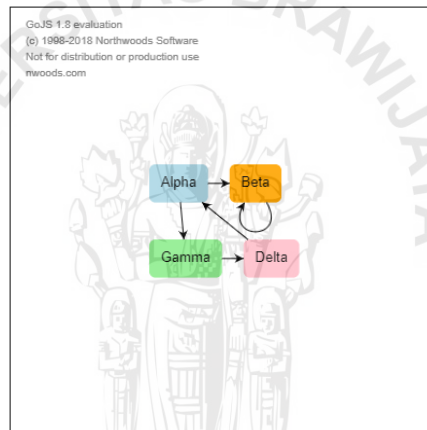
CodeIgniter adalah sebuah *framework* atau kerangka kerja yang dapat mempercepat *developer* untuk membuat sebuah aplikasi berbasis *web*. *CodeIgniter* adalah salah satu *framework* yang mudah dan lebih diketahui oleh orang banyak dalam hal instalasi bahkan penggunaannya, sehingga mempermudah apabila dilakukan perbaikan sistem oleh orang lain, dan dapat *maintenance* tanpa kesulitan oleh orang lain. *CodeIgniter* merupakan aplikasi *open source* yang berupa *framework* dengan model MVC (*Model, View, Controller*) untuk membangun *website* dinamis dengan menggunakan PHP.

Keuntungan dari penggunaan *framework CodeIgniter* antara lain menggunakan *pattern* MVC yang mempermudah dan meningkatkan fleksibilitas,

menghasilkan url yang *search engine friendly*, mudah untuk membuat *library* dan *helpernya*, *support* banyak *database*, MySQL (4.1+), MySQLi, MS SQL, Postgres, Oracle, SQLite, dan ODBC. Selain itu *CodeIgniter* yang sifatnya *opensource* atau gratis juga masuk ke dalam berbagai kalangan sehingga otomatis dukungan komunitasnya pun sangat baik.

2.10 GoJS

GoJS adalah suatu interaktif *JavaScript* diagram dalam bentuk HTML dan diproses menggunakan *JavaScript*. *Library* ini membantu untuk membuat diagram secara interaktif dalam *browser* modern. GoJS mendukung *template* dan *data-binding* dari objek-objek grafis hingga ke dalam data model. Properti dari objek-objek ini disimpan dalam *JavaScript* yang sederhana (Northwoods Software Corporation, 1998). Dalam Gambar 2.12 ditunjukkan contoh penggunaan GoJS dalam pembuatan diagram editor yang dibuat secara sederhana menggunakan html untuk tampilan dan *JavaScript* untuk menyimpan dan menampilkan data pada layar yang tersedia.

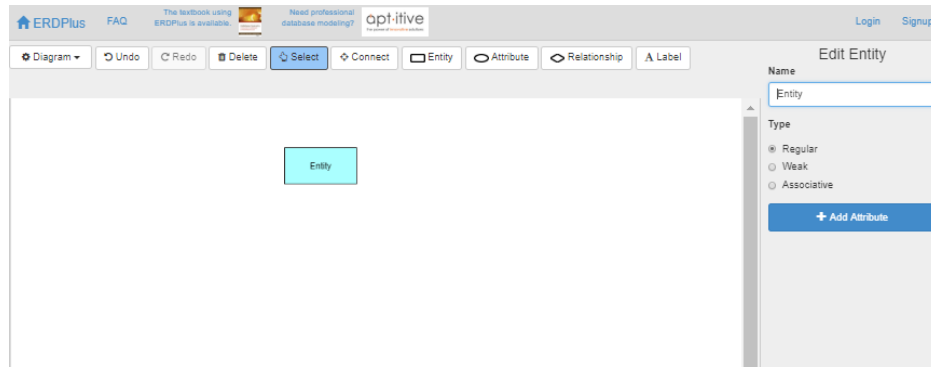


Gambar 2.12 Contoh Pembuatan *Diagram Editor* pada GoJS

Sumber : <https://gojs.net/latest/samples/minimal.html>

2.11 ERDPlus

ERDPlus adalah suatu perangkat lunak berbasis *web* yang digunakan untuk alat pemodelan *database* seperti *Entity Relationship Diagram*, *Relational Schema*, dan juga *Star schema* (ERDPlus, 2015). ERD Plus dapat digunakan dengan RDBMS atau sistem manajemen basis data relasional seperti Oracle, MySQL, Microsoft SQL Server, PostgreSQL, Teradata, IBM DB2, maupun Microsoft Access. Perangkat lunak ini berupa *open source* atau tidak berbayar dan dapat digunakan secara *online*. Seperti pada Gambar 2.13, dapat dilihat bahwa perangkat lunak ini dapat digunakan untuk membuat diagram, menambah objek-objek pada diagram, menghapus, maupun mengubah nama dan tipe dari objek. File yang dihasilkan dari perangkat lunak ini berekstensi “.erdplus”.



Gambar 2.13 Tampilan Aplikasi Web ERD Plus

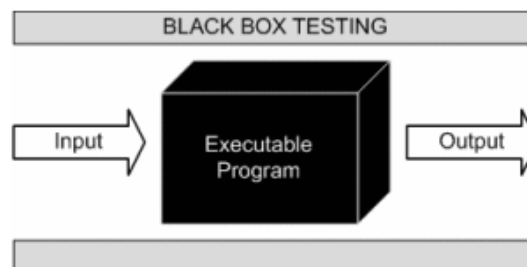
Sumber : <https://erdplus.com/#/standalone>

2.12 Pengujian

Sebagai perancang perangkat lunak pada dasarnya harus bersifat konstruktif dengan tidak menggunakan asas praduga terhadap kualitas suatu perangkat lunak. Karena itu dibutuhkan *testing* atau pengujian untuk menguji coba perangkat lunak yang dikembangkan dan menyelesaikan masalah-masalah atau *error* yang terdapat pada perangkat lunak (Sommerville, 2011).

2.12.1 Pengujian *black box*

Black box testing adalah pengujian di mana *tester* mendefinisikan kumpulan kondisi *input* dan melakukan pengecekan pada hasil *output* sistem, tanpa melihat atau memiliki pengetahuan bagaimana *output* dihasilkan atau bagaimana internal dari aplikasi yang diuji (Kumar, Singh, & Dwivedi, 2015). Pengujian ini digambarkan seperti menguji pada kotak hitam yang tidak tembus pandang seperti yang digambarkan pada Gambar 2.14, yang menggambarkan bahwa pengujian tidak mengetahui isi dari internal sistem. *Black box testing* bukanlah solusi alternatif dari *white box testing* namun, merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *white box testing*. *Black box testing* bertujuan untuk mengidentifikasi *bug-bug* yang ada pada hasil, kinerja, dan juga perilaku sistem. Melakukan simulasi terhadap bagaimana proses berjalan dan menghasilkan output yang diinginkan.



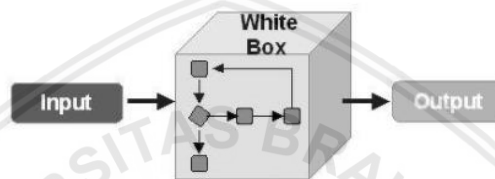
Gambar 2.14 Black Box Testing

Sumber: (Software testing fundamentals 2010)

Dalam pengujian ini, *black box* akan berusaha menemukan kesalahan pada kategori fungsi yang hilang atau tidak benar, kesalahan antarmuka, kesalahan dalam struktur data, kesalahan kinerja, maupun kesalahan inisialisasi.

2.12.2 Pengujian *white box*

White box testing adalah salah satu pengujian yang dilakukan pada pengujian unit, dan dilakukan oleh *programmer* yang sangat mengenal *source code* dengan baik. Pengujian ini dilakukan oleh *tester* yang sudah mengenal dan paham bagaimana sistem yang diuji diimplementasikan (Kumar et al., 2015) untuk menguji implementasi tiap kelas internal dengan lebih mendalam yaitu dengan menguji kemungkinan jalur yang akan dilalui.



Gambar 2.15 White Box Testing
(software testing genius 2011)

Alasan dinamakan *white box testing* dikarenakan tidak hanya melakukan pengecekan terhadap *input-output* namun juga lebih ke dalam setiap prosesnya. Kemudahan yang didapat dengan menggunakan *white box testing* ini adalah apabila diketahui terjadi suatu kesalahan/*error*, maka akan lebih mudah melacak sumber kesalahan. Seperti yang digambarkan pada Gambar 2.15, *white box* akan memproses *input* dengan pengecekan proses, tidak hanya mencocokkan output seperti pada pengujian *black box*.

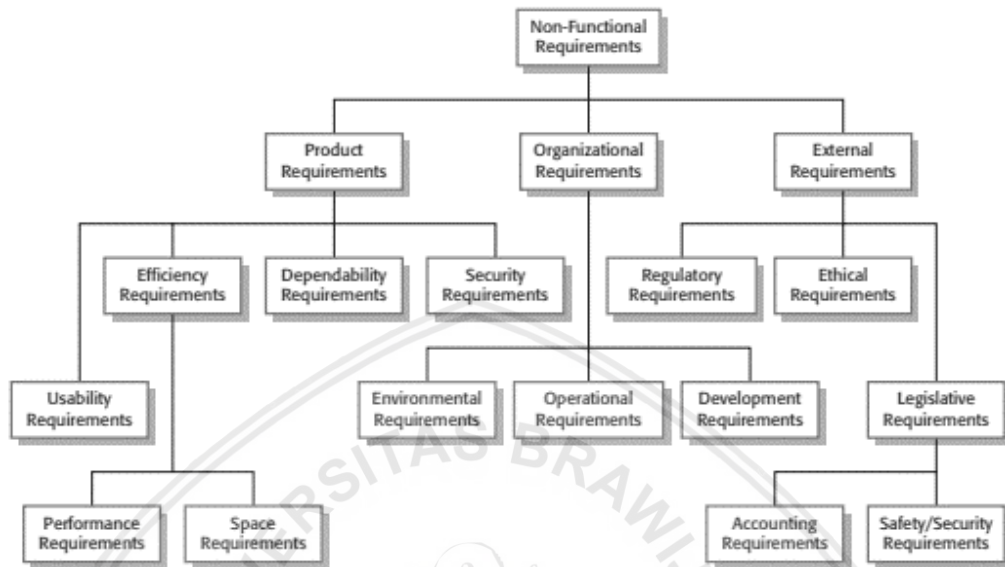
2.12.3 Pengujian non-fungsional (*Compatibility*)

Definisi dari pengujian non-fungsional adalah sebuah atribut maupun batasan pada sebuah sistem yang bukan merupakan bagian dari kebutuhan fungsional dari sistem (Chung & Leite, 2009).

Pada kebutuhan non-fungsional, terdapat berbagai macam jenis kebutuhan, seperti pada Gambar 2.16. Karena untuk melakukan pengujian harus dilakukan secara kuantitatif. Untuk melakukan pengujian, terdapat beberapa metrik yang dapat digunakan untuk merincikan properti dari kebutuhan non-fungsional sistem. Dalam penelitian ini, akan diambil kebutuhan produk atau *Product Requirements* untuk dijadikan kebutuhan non-fungsional yang diuji. Dari kebutuhan produk ini, akan diambil kebutuhan kesesuaian sistem atau *compatibility*.

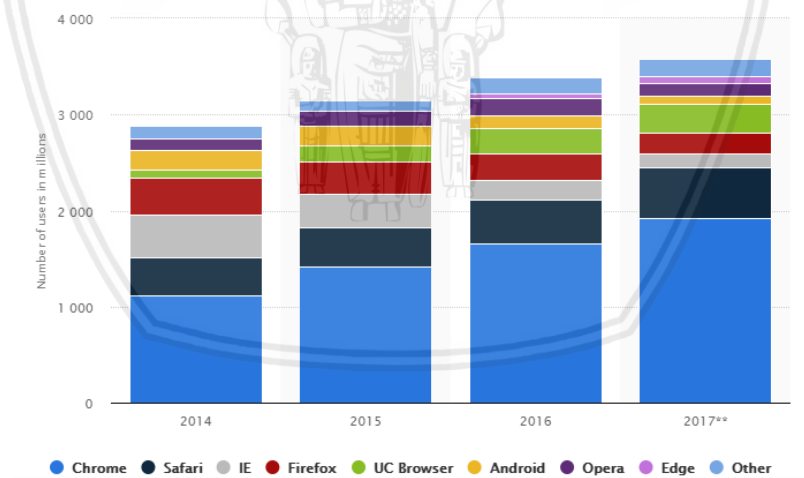
Pada penelitian ini, menggunakan pengujian *compatibility*. Pengujian *compatibility* adalah pengujian untuk mengevaluasi apakah sebuah perangkat lunak berjalan dengan lancar dalam sebuah lingkungan berbeda. Dalam hal perangkat lunak berbasis *web*, maka akan dievaluasi apakah dapat berjalan

dengan *browser* yang berbeda. Pengujian ini untuk meminimalisir kesalahan atau kegagalan yang akan terjadi pada saat dijalankan dengan lingkungan *browser* berbeda. Untuk melakukan pengujian *compatibility*, sebuah perangkat lunak berbasis *web*, digunakan oleh *browser PC*.



Gambar 2.16 Jenis-Jenis Kebutuhan Non-Fungsional

Sumber: (Sommerville 2011)



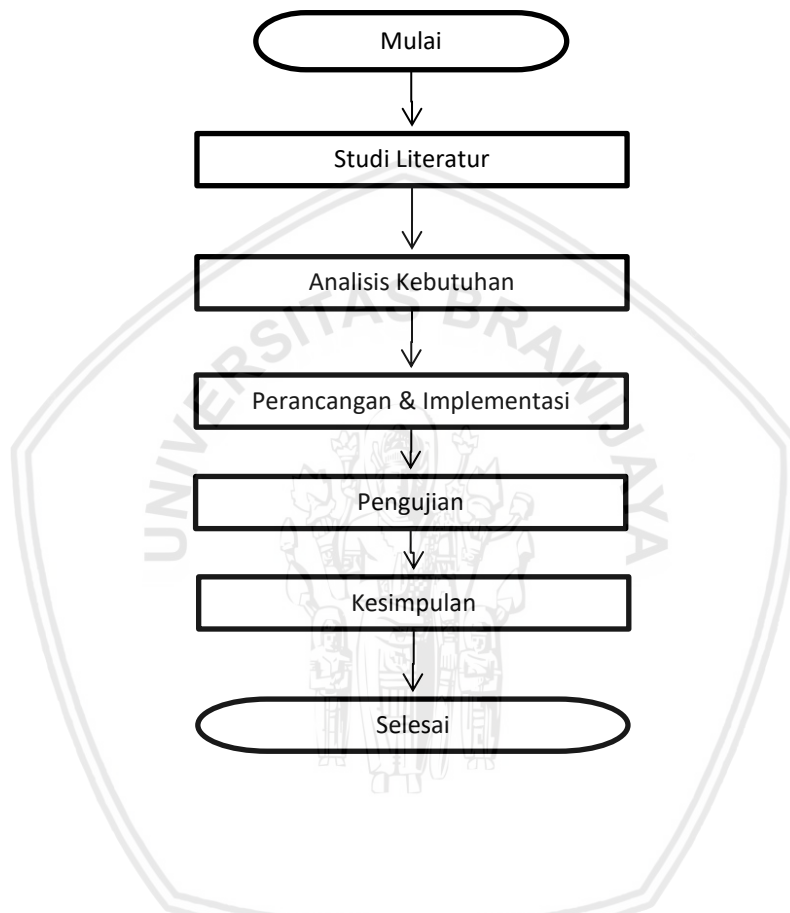
Gambar 2.17 Hasil Survey

Sumber: (Statista, 2017)

Dari hasil *survey* oleh Statista, pada Gambar 2.17 dapat dilihat bahwa pengguna *browser* dengan lingkup seluruh dunia, hingga pada tahun 2017 terbanyak dipegang oleh Chrome, yang kemudian diikuti Safari, Internet Explorer, Firefox dan yang kelima adalah UC Browser. Dari data-data yang ada, maka akan dilakukan pengujian *compatibility* dengan kelima *browser* terpopuler yang ada pada gambar 2.17.

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan mengenai langkah-langkah yang akan dilakukan untuk mengembangkan “Sistem Pemetaan Otomatis Entity Diagram ke dalam Database”. Seperti yang terdapat pada Gambar 3.1, beberapa tahapan tersebut yaitu studi literatur, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis.



Gambar 3.1 Diagram Alur Metodologi Penelitian

Sumber: Penulis

3.1 Studi literatur

Studi literatur mengandung landasan teori yang terkait dengan penelitian. Hal ini penting dilakukan agar pengetahuan dasar untuk membangun suatu sistem dapat terpenuhi dengan baik. Pustaka dan teori yang berkaitan dengan penelitian ini adalah:

1. Pemetaan *Entity Relationship Diagram* (ERD),
2. Rekayasa Perangkat Lunak,
3. UML (*Unified Modelling Language*),

4. Analisis berbasis objek,
5. Perancangan berbasis objek,
6. Model pengembangan *waterfall*,
7. Pengembangan berbasis objek,
8. *Framework CodeIgniter*,
9. ERD Plus,
10. *GoJS Library*, dan
11. Pengujian *black box*, *white box* dan non-fungsional.

3.2 Analisis kebutuhan

Analisis kebutuhan berupa identifikasi kebutuhan dari Sistem Pemetaan Otomatis Entity Diagram ke dalam Database. Aktor yang akan berperan dalam sistem dan kebutuhan yang dibutuhkan dalam sistem akan didapatkan dari hasil observasi. Pemodelan ini bertujuan untuk mengetahui kebutuhan apa saja yang akan dibangun dalam sistem lalu siapa saja yang akan berinteraksi dalam sistem serta mengetahui cara kerja sistem.

3.3 Perancangan sistem

Perancangan sistem merupakan tahapan perancangan yang didapat dari kebutuhan fungsional sistem. Perancangan sistem pada penelitian ini dengan menggunakan pendekatan berorientasi objek. Pada tahapan perancangan awal akan dilakukan perancangan dengan menggunakan UML (*Unified Modelling Language*).

3.4 Implementasi

Pada tahap implementasi sistem, pembangunan sistem akan mengacu pada hasil perancangan sistem yang sudah dilakukan. Pada tahap implementasi sistem, perancangan sistem akan diubah menjadi bahasa pemrograman untuk menghasilkan suatu bentuk sistem yang dapat digunakan oleh pengguna. Tahapan yang dilalui dalam implementasi Sistem Pemetaan Otomatis Entity Diagram ke dalam Database antara lain:

1. Membangun sistem sesuai dengan kebutuhan yang sudah ada pada tahap perancangan.
2. Membuat ERD dengan program ERDPlus, dan mengekspor ke dalam *format* JSON untuk di impor ke dalam sistem.
3. Membangun fungsi utama sistem yang dapat mengolah pemetaan ERD secara otomatis dengan tepat sesuai kaidah-kaidah atau aturan pada Rekayasa Perangkat Lunak. Sistem akan dapat menghasilkan keluaran yang dapat diunduh dengan format SQL.

Pada tahapan implementasi akan digunakan metode *Object Oriented Programming (OOP)* dan berbasis *web* dengan bahasa pemrograman PHP, dan juga menggunakan *framework CodeIgniter*. Sedangkan *software* yang digunakan untuk mendukung tahapan implementasi ini adalah XAMPP, Notepad++, ERDplus dan juga *Browser*.

3.5 Pengujian dan analisis

Setelah implementasi telah selesai, maka akan dilakukan pengujian sistem. Tahap ini dilakukan untuk mengetahui apakah sistem yang dikembangkan terdapat kesalahan *error* atau *bug*. Selain itu juga untuk mengetahui apakah kebutuhan yang dianalisis diawal sudah terpenuhi atau belum. Pada tahap ini akan digunakan beberapa metode pengujian, yakni:

1. Pengujian *white box*, sebagai pengujian unit menggunakan basis *path testing* yang akan menguji kode program berdasarkan algoritme yang ada pada setiap metode dalam sebuah kelas.
2. Pengujian *black box* merupakan salah satu metode pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak. Pengujian *black box* atau pengujian validasi bertujuan untuk mengidentifikasi *bug-bug* yang ada pada hasil, kinerja dan juga perilaku sistem.
3. Pengujian non-fungsional sistem, yaitu pengujian *compatibility*.

Hasil dari pengujian kemudian dianalisis, yang mana akan didapatkan apa saja yang sudah valid dan apa saja yang harus diperbaiki dalam sistem. Hasil dari pengujian yang telah dilakukan akan dijadikan sebagai kesimpulan.

3.6 Pengambilan kesimpulan dan saran

Pada tahap akhir ini, akan dibuat kesimpulan berdasarkan ringkasan hasil penelitian dari tahap pengujian, kendala dan juga evaluasi apa saja yang didapat dari pengujian akhir dengan metode-metode yang digunakan. Selain itu juga untuk mengambil saran dengan tujuan memberi masukan kepada penelitian berikutnya untuk memperbaiki kelemahan dari sistem.

BAB 4 ANALISIS KEBUTUHAN

Analisis kebutuhan merupakan tahapan pertama yang dilakukan pada pengembangan sebuah sistem. Analisis kebutuhan merupakan tahapan untuk menentukan kebutuhan apa saja yang harus terdapat pada Sistem Pemetaan Otomatis Entity Diagram ke dalam Database. Penelitian ini menggunakan pemodelan SDLC waterfall, sehingga pada tahap ini, akan dilakukan proses elisitasi kebutuhan dan juga penentuan aktor untuk mendapatkan gambaran besar sistem yang dibuat.

4.1 Gambaran Umum Sistem

Sistem ini merupakan sebuah sistem yang memproses suatu ERD untuk dirubah menjadi sebuah *database* yang bisa langsung digunakan dalam pengembangan sistem perangkat lunak secara otomatis. Sistem nantinya akan memasukan *file* diagram *entity relationship* berupa *file* berkeistensi JSON (*Javascript Object Notation*) dan sistem akan mengolah *file* tersebut untuk dipecah-pecah menjadi kumpulan data. Kemudian, data yang didapat akan digunakan untuk membangun suatu *database* dengan berbentuk SQL *script* yang nantinya dapat diunduh dan di *impor* ke dalam *database*.

4.2 Identifikasi Aktor

Identifikasi aktor ini bertujuan untuk mengidentifikasi aktor-aktor yang akan berinteraksi dengan sistem. Berdasarkan permasalahan yang ada pada latar belakang, hasil analisis penulis sesuai Tabel 4.1, aktor pada sistem ini terdiri dari User, *Member* atau pengguna yang sudah terdaftar, dan juga *Administrator* sistem.

Tabel 4.1. Tabel Identifikasi Aktor

Aktor	Deskripsi
User	Aktor yang mengakses halaman awal sistem tanpa diberi hak akses.
<i>Member</i>	Aktor yang diberikan hak akses untuk mengelola ERD.
<i>Administrator</i>	Aktor yang diberi hak akses untuk mengelola data <i>member</i> .

4.3 Daftar Kebutuhan Fungsional

Berdasarkan dari permasalahan, didapatkan Kebutuhan fungsional atau kebutuhan yang harus disediakan oleh sistem dengan aturan penomoran sebagai berikut

Kode : EAM_F_XXYY

EAM : Inisial Sistem (ERD Automated Mapping)

F/NF : Fungsional / Non-Fungsional

XX : nomor kebutuhan utama

YY : nomor spesifikasi kebutuhan (hanya untuk kebutuhan fungsional)

Dari hasil analisis kebutuhan dengan observasi terhadap aplikasi-aplikasi sejenis yaitu visual paradigm, ERDplus dan luchidchart (Lampiran 5) penulis mengambil fungsi yang terdapat didalam ketiga aplikasi sebagai dasar kebutuhan yang dibutuhkan oleh pengguna aplikasi sejenis. Dari hasil analisis tersebut juga didapatkan bahwa aplikasi non-premium tidak memiliki fungsi 'export to sql' yang sangat memudahkan pengguna. Penulis juga menganalisa kebutuhan target *user* yang didapatkan dari wawancara dengan pengguna sebagai *database designer* (lampiran 2) dimana perubahan dalam analisa kebutuhan juga mempengaruhi perancangan sistem *database* yang cukup menyulitkan. Dari beberapa hasil analisis tersebut, maka dapat dianalisa kebutuhan yang dibutuhkan oleh pengguna adalah sebagai berikut.

pada Tabel 4.2 adalah tabel kebutuhan yang dibutuhkan oleh *user* atau pengguna yang tidak terdaftar atau belum mendapatkan hak akses, hanya bisa melakukan proses login, lupa *password*, dan juga registrasi. Sedangkan untuk Tabel 4.3 adalah daftar kebutuhan yang dibutuhkan oleh *member* atau pengguna terdaftar dan sudah diberikan hak akses ke dalam sistem. *Member* dapat membuat *project* di mana berisi diagram, dan dapat mengelolanya, termasuk melakukan pemetaan otomatis terhadap diagram. Dari Tabel 4.4 adalah daftar kebutuhan dari administrator selaku pengelola dari sistem, di mana dapat melakukan kelola *member* maupun melihat daftar *project* yang terdaftar.

Tabel 4.2. Tabel Kebutuhan Fungsional User

No	Kode	Deskripsi / Spesifikasi
1	EAM_F_0100	Sistem harus menyediakan fungsi login kepada <i>user</i> .
1.1	EAM_F_0101	Sistem harus menyediakan <i>form</i> login untuk menginput <i>username</i> dan <i>password</i>
1.2	EAM_F_0102	Sistem harus menyediakan tombol login untuk mengeksekusi proses login.
2	EAM_F_0200	Sistem harus menyediakan fungsi registrasi atau pendaftaran sebagai <i>member</i> kepada <i>user</i> .
2.1	EAM_F_0201	Sistem harus menyediakan tombol registrasi untuk mengakses halaman login kepada <i>user</i>

No	Kode	Deskripsi / Spesifikasi
2.2	EAM_F_0202	Sistem menyediakan halaman <i>form</i> login kepada <i>user</i> yang berisi <i>input email</i> , nama, <i>username</i> , <i>password</i> , dan tanggal lahir.
2.3	EAM_F_0203	Sistem menyediakan tombol registrasi untuk mengeksekusi pendaftaran <i>user</i> .
3	EAM_F_0300	Sistem harus menyediakan fungsi lupa <i>password</i> kepada <i>user</i> .
3.1	EAM_F_0301	Sistem harus menyediakan <i>form input email</i> kepada <i>user</i> .
3.2	EAM_F_0302	Sistem harus mengirim <i>password</i> kepada <i>email user</i> sesuai <i>email</i> yang diisi.
3.3	EAM_F_0303	Sistem harus menyediakan tombol kirim <i>password</i> untuk mengeksekusi proses lupa <i>password</i> .

Tabel 4.3. Tabel Kebutuhan Fungsional *Member*

No	Kode	Deskripsi / Spesifikasi
1.	EAM_F_0400	Sistem harus menyediakan fungsi membuat <i>project</i> baru kepada <i>member</i> .
1.1	EAM_F_0401	Sistem menyediakan tombol 'buat <i>project</i> baru' kepada <i>member</i> .
1.2	EAM_F_0402	Sistem menyediakan <i>form input</i> nama <i>project</i> , dan tombol buat <i>project</i>
1.3	EAM_F_0403	Sistem menyediakan tombol 'buat <i>project</i> ' untuk mengeksekusi buat <i>project</i> baru.
2.	EAM_F_0500	Sistem harus menyediakan fungsi lihat daftar <i>project</i> yang dimiliki oleh <i>member</i> .
2.1	EAM_F_0501	Sistem menyediakan data berupa tabel berisi nama <i>project</i> , dan tanggal pembuatan.
3.	EAM_F_0600	Sistem harus menyediakan fungsi untuk melihat detail <i>project</i> yang dipilih.
3.1	EAM_F_0601	Sistem menampilkan detail informasi <i>project</i> dan pilihan fungsi untuk mengelola <i>project</i> .
3.2	EAM_F_0602	Sistem menampilkan ERD apabila telah dibuat.
4.	EAM_F_0700	Sistem harus menyediakan fungsi upload ERD kepada <i>member</i> dan sekaligus melakukan proses mapping erd.
4.1	EAM_F_0701	Sistem menyediakan <i>form input upload</i> ERD dengan bentuk <i>file</i> ekstensi <i>.erdplus</i> dengan maksimal ukuran 2MB.
4.2	EAM_F_0702	Sistem menyediakan tombol 'upload dan mapping' <i>file</i> untuk mengeksekusi proses <i>upload</i> .
4.3	EAM_F_0703	Sistem menampilkan pesan apabila proses berhasil atau gagal dilakukan.

No	Kode	Deskripsi / Spesifikasi
5.	EAM_F_0800	Sistem harus menyediakan fungsi untuk memetakan hasil pembuatan ERD ke dalam <i>database</i>
5.1	EAM_F_0801	Sistem memproses ERD yang sudah dibuat untuk dipetakan menjadi <i>database</i> dalam bentuk sql.
5.2	EAM_F_0802	Sistem menampilkan pesan apabila proses berhasil atau gagal dilakukan.
6.	EAM_F_0900	Sistem harus menyediakan fungsi <i>download sql</i> kepada <i>member</i> .
6.1	EAM_F_0901	Sistem menyediakan tombol <i>download</i> untuk mengeksekusi proses <i>download sql</i> yang telah dipetakan.
7.	EAM_F_1000	Sistem harus menyediakan fungsi untuk membuat ERD baru.
7.1	EAM_F_1001	Sistem menyediakan <i>workspace</i> kosong.
7.2	EAM_F_1002	Sistem menyediakan bentuk-bentuk maupun konektor yang bisa digunakan dalam pembuatan diagram.
7.3	EAM_F_1003	Sistem menyediakan tombol untuk menyimpan diagram.
8.	EAM_F_1100	Sistem harus menyediakan fungsi edit ERD kepada <i>member</i> .
8.1	EAM_F_1101	Sistem menyediakan <i>workspace</i> yang berisi diagram ERD yang sudah dibuat atau yang diupload.
8.2	EAM_F_1102	Sistem menyediakan bentuk-bentuk maupun konektor yang bisa digunakan dalam pengeditan diagram.
8.3	EAM_F_1103	Sistem menyediakan tombol untuk menyimpan diagram.
9.	EAM_F_1300	Sistem harus menyediakan fungsi <i>mapping</i> untuk memetakan diagram yang telah dibuat pada <i>workspace</i> dan telah tersimpan.
9.1	EAM_F_1301	Sistem harus menyediakan fungsi untuk menghapus <i>project</i> .
9.2	EAM_F_1302	Sistem menyediakan menyediakan tombol hapus.
9.3	EAM_F_1303	Sistem menampilkan dialog untuk konfirmasi hapus
10	EAM_F_1400	Sistem harus menyediakan fungsi <i>logout</i> kepada <i>member</i> .
10.2	EAM_F_1401	Sistem menyediakan tombol <i>logout</i> untuk mengeksekusi proses keluar dari sistem
10.3	EAM_F_1402	Sistem menampilkan dialog untuk konfirmasi <i>logout</i>

Tabel 4.4. Tabel Kebutuhan Fungsional *Administrator*

No	Kode	Deskripsi / Spesifikasi
1	EAM_F_1500	Sistem harus menyediakan fungsi lihat daftar <i>member</i> kepada administrator.
1.1	EAM_F_1501	Sistem menampilkan tabel berisi daftar <i>member-member</i> yang terdaftar.
2	EAM_F_1600	Sistem harus menyediakan fungsi hapus <i>member</i> dari daftar <i>member</i> kepada administrator.
2.1	EAM_F_1601	Sistem menyediakan tombol hapus disetiap baris <i>member</i> dalam daftar <i>member</i> untuk mengeksekusi proses hapus <i>member</i> .
3	EAM_F_1700	Sistem harus menyediakan fungsi lihat daftar <i>project</i> kepada administrator.
3.1	EAM_F_1701	Sistem menampilkan tabel yang berisi data <i>project</i> .
4	EAM_F_1400	Sistem harus menyediakan fungsi <i>logout</i> kepada administrator.
4.1	EAM_F_1401	Sistem menyediakan tombol <i>logout</i> untuk mengeksekusi proses keluar dari sistem
4.2	EAM_F_1402	Sistem menampilkan dialog untuk konfirmasi <i>logout</i>

4.4 Daftar Kebutuhan Non-Fungsional

Seperti pada Tabel 4.5, didapatkan Kebutuhan non-fungsional atau kebutuhan yang tidak harus disediakan oleh sistem namun mempengaruhi kualitas sistem, yaitu dalam hal mudahnya perangkat lunak untuk beradaptasi dengan lingkungan *browser* yang berbeda-beda.

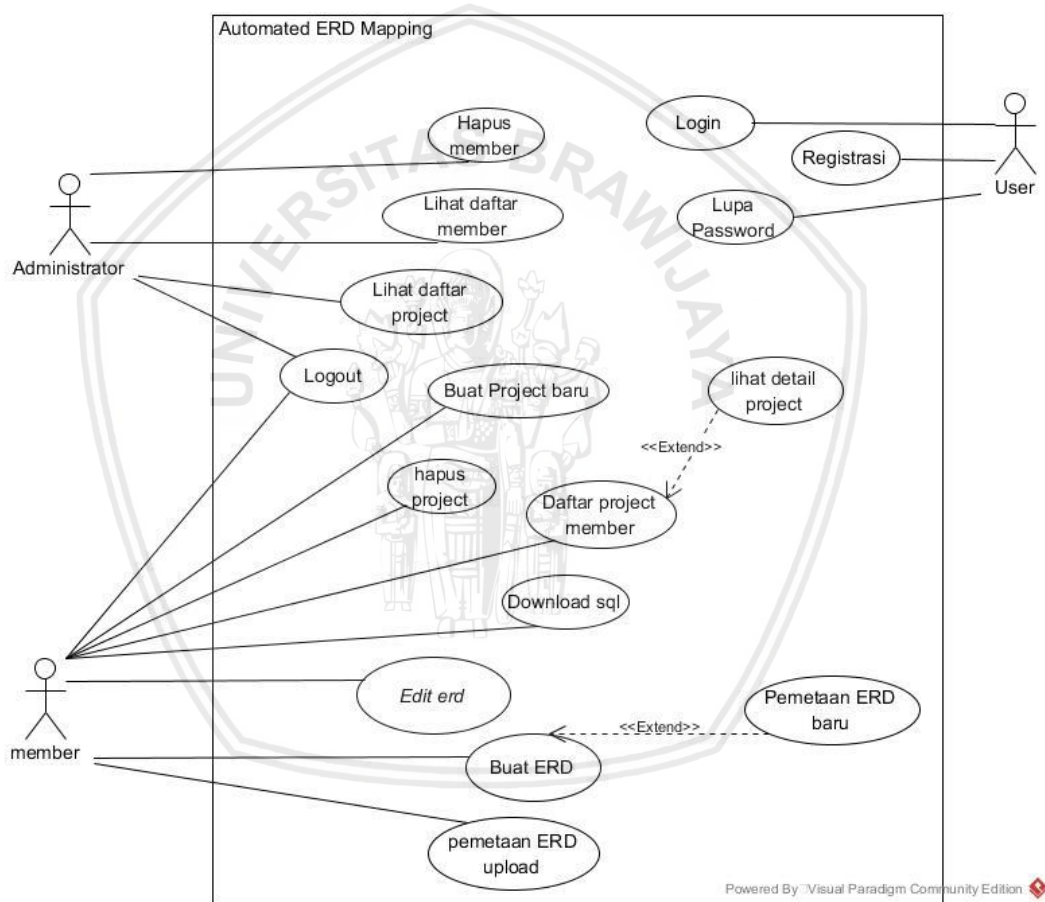
Tabel 4.5. Tabel Kebutuhan Non-fungsional

No	Kode	Parameter	Deskripsi
1.	EAM_NF_01	Compatibility	Sistem dapat dijalankan dengan baik pada berbagai <i>browser</i> pada PC, antara lain Google Chrome, Mozilla firefox, Opera, Internet explorer, Safari, dan UC browser

4.5 Use Case Diagram

Berdasarkan dari daftar kebutuhan-kebutuhan yang sudah didapatkan, dapat dibuat menjadi sebuah *Use case diagram*. *Use case diagram* adalah sebuah diagram untuk memodelkan apa yang dapat dilakukan oleh sistem dan aktor. Pada Gambar 4.1, dapat dilihat bahwa terdapat tiga macam actor, yaitu *user*, *member*, dan administrator. Pada *use case* juga dijelaskan bahwa *user* dapat melakukan

proses login, lupa *password*, dan juga registrasi. Aktor administrator pada gambar juga dijelaskan bahwa dapat melakukan empat macam hal yaitu *logout*, lihat daftar *project*, lihat daftar *member*, dan juga hapus *member* di mana hanya dapat dilakukan setelah melakukan lihat daftar *member*. Sedangkan untuk actor *member*, terdapat dua belas macam hal yang dapat dilakukan, yaitu *logout*, lihat daftar *project* dari *member* bersangkutan, membuat *project* baru, mengedit nama *project* yang sudah ada, dan menghapus *project* yang dipilih. Selain itu, dari daftar *project* juga dapat melihat detail salah satu *project* tertentu, yang akan dapat mengakses *upload* ERD, buat diagram baru, edit diagram, pemetaan diagram yang dapat dilakukan apabila sudah meng*upload* atau membuat erd, dan juga secara otomatis akan langsung melakukan simpan erd, dan dapat melakukan *download* sql di mana hanya akan bisa dilakukan apabila erd sudah tersimpan.



Gambar 4.1 Use Case Diagram

4.6 Usecase Scenario

Berdasarkan dari *use case* diagram yang telah dibuat, maka didapatkan *use case scenario* sebagai berikut,

4.6.1 Usecase Scenario Pemetaan ERD Upload

Pada Tabel 4.6 dijelaskan mengenai Usecase *scenario* pemetaan ERD oleh *member*. Dalam *scenario* ini, *member* dapat mengunggah *file* ERD berupa *file* berekstensi *erdplus* yang akan diproses pemetaan secara otomatis oleh sistem.

Tabel 4.6. Tabel Usecase *scenario* Pemetaan ERD *upload*

Pemetaan ERD upload	
Actor	<i>Member</i>
Objective	Mengunggah <i>file</i> ERD berupa <i>.erdplus</i> untuk langsung secara otomatis memproses pemetaan.
Pre condition	Halaman lihat detail <i>project</i> terbuka
Main flow	<ol style="list-style-type: none">1. Sistem menampilkan tombol '<i>upload ERD</i>',2. Sistem menampilkan pesan dialog windows untuk memilih <i>file</i> berekstensi <i>.erdplus</i> dan juga <i>file jpg</i> sebagai pembanding (optional),3. Aktor memilih <i>file</i>, kemudian memilih tombol '<i>pilih</i>',4. Sistem menampilkan nama <i>file</i> yang akan di<i>upload</i>.5. Aktor memilih tombol '<i>upload erd</i>',
Alternative flow	Kegagalan penyimpanan, <ol style="list-style-type: none">1. Jika ekstensi <i>file</i> salah, akan menampilkan pesan '<i>jenis file yang dipilih tidak tepat</i>' dan kembali ke halaman detail <i>project</i>.2. Jika ukuran <i>file</i> melebihi batas maksimum, akan menampilkan pesan '<i>ukuran file yang dipilih melebihi batas</i>' dan kembali ke halaman detail <i>project</i>
Post Condition	Mengaktifkan link sql download dan menampilkan ke halaman lihat detail <i>project</i>

4.6.2 Usecase Scenario Pemetaan ERD Baru

Use case scenario pemetaan ERD, sesuai pada Tabel 4.7, bertujuan untuk memetakan aatau mengolah *file* ERD berupa JSON, menjadi *database* berupa *file* berkestensi sql yang dapat langsung di*import* ke dalam *database*. Proses ini merupakan salah satu proses utama dari sistem. Pada *scenario* ini, nantinya *member* akan memilih tombol mapping diagram yang tersedia pada halaman lihat detail *project*.

Tabel 4.7. Tabel Usecase Scenario Pemetaan ERD baru

Pemetaan ERD baru	
Actor	Member
Objective	Memetakan <i>file</i> ERD yang telah dibuat menjadi script sql.
Pre condition	Halaman detail <i>project</i> terbuka, dan sudah ada ERD yang dibuat.
Main flow	<ol style="list-style-type: none"> 1. Sistem menampilkan tombol 'mapping ERD', 2. Sistem mengeksekusi proses mapping diagram ERD
Alternative flow	-
Post Condition	Halaman detail <i>project</i> akan terbuka, dan akan menampilkan tombol 'download sql' dan juga mengaktifkan link untuk mendownload sql

4.6.3 Usecase Scenario Buat ERD

Pada Tabel 4.8 dijelaskan mengenai Usecase *scenario* buat ERD oleh *member*. Dalam skenario ini, *member* dapat membuat diagram ERD baru pada *workspace* yang tersedia, yang akan tersimpan berupa *file* JSON untuk diproses pada tahap pemetaan diagram. Selain itu, nantinya setelah diagram tersimpan, akan mengaktifkan tombol yang mengarah ke proses pemetaan.

Tabel 4.8 Tabel Usecase Scenario Buat ERD

Buat ERD	
Actor	Member
Objective	Membuat ERD baru dan tersimpan dalam sistem
Pre condition	Halaman buat ERD terbuka
Main flow	<ol style="list-style-type: none"> 1. Sistem menampilkan <i>workspace</i> kosong untuk membuat diagram. 2. Aktor menyusun dan membuat diagram ERD pada <i>workspace</i> 3. Aktor memilih tombol 'simpan'.
Alternative flow	-
Post Condition	ERD tersimpan dan halaman detail <i>project</i> terbuka.

4.6.4 Usecase Scenario *download* Sql

Usecase *scenario* *download* sql pada Tabel 4.9 adalah mengenai proses untuk *member* dapat mengunduh *file* sql yang sudah terdapat dalam sistem.

Proses ini hanya dapat dilakukan apabila sudah terdapat diagram di dalam *database*, dan sudah melakukan pemetaan.

Tabel 4.9 Tabel Usecase *scenario download Sql*

Download sql	
Actor	<i>Member</i>
Objective	Mengunduh <i>file database</i> berupa sql yang sudah diproses oleh sistem.
Pre condition	Halaman detail <i>project</i> terbuka, dan menampilkan tombol <i>download sql</i> .
Main flow	1. Aktor memilih tombol " <i>download sql</i> ".
Alternative flow	1. Jika <i>file diagram</i> belum disimpan, maka tombol unduh akan berstatus tidak aktif.
Post Condition	<i>file sql</i> terunduh.

4.6.5 Usecase Scenario *edit erd*

Pada Tabel 4.10 dijelaskan mengenai Usecase *scenario edit ERD* oleh *member*. Dalam *scenario* ini, *member* dapat mengubah diagram ERD yang baru dibuat, mapun sudah *terupload*, pada *workspace* yang tersedia, yang nantinya akan menyimpan JSON untuk diproses pada tahap pemetaan diagram.

Tabel 4.10 Tabel Usecase *scenario Edit ERD*

Edit ERD	
Actor	<i>Member</i>
Objective	Mengedit diagram yang telah tersimpan
Pre condition	Halaman detail <i>project</i> terbuka, dan sudah terdapat diagram.
Main flow	<ol style="list-style-type: none"> 1. Aktor memilih tombol edit diagram 2. Sistem menampilkan <i>workspace</i> berisi erd yang telah tersimpan sebelumnya, dan menyediakan <i>workspace</i> yang dapat mengedit diagram erd. 3. Aktor memilih tombol 'simpan'.
Alternative flow	-
Post Condition	ERD akan tersimpan ke dalam <i>database</i> . Halaman detail <i>project</i> akan terbuka kembali.

4.6.6 Usecase Scenario Login

Pada Usecase *scenario* login Tabel 4.11 ini dijelaskan mengenai proses login oleh *user*. Di mana untuk mengizinkan sistem *memberikan* hak akses kepada *user* sebagai *member* maupun administrator apabila *username* dan *password* sudah divalidasi, dan langsung dialihkan menuju halaman dashboard.

Tabel 4.11 Tabel Usecase *scenario* login

Login	
Actor	<i>User</i>
Objective	Mengizinkan sistem <i>memberikan</i> hak akses kepada <i>user</i> sebagai <i>Member</i> ataupun <i>Administrator</i> .
Pre condition	Halaman login terbuka
Main flow	<ol style="list-style-type: none">1. Sistem menampilkan <i>form</i> isian data berupa <i>username</i>, dan <i>password</i>.2. Aktor mengisi <i>username</i> dan <i>password</i> dan menekan tombol 'login'3. Sistem melakukan validasi <i>user</i>
Alternative flow	<ol style="list-style-type: none">1. Kegagalan validasi : jika salah memasukan <i>username</i> atau <i>password</i>, sistem akan menampilkan pesan "<i>username</i> atau <i>password</i> salah" dan kembali ke halaman login
Post Condition	Halaman dashboard terbuka.

4.6.7 Usecase Scenario Registrasi

Pada Tabel 4.12 dijelaskan mengenai Usecase *scenario* registrasi, di mana proses untuk *user* mendaftar sebagai *member*. Proses ini dapat diakses melalui halaman registrasi, di mana *user* akan mengisi *form* pendaftaran yang tersedia, dan sistem akan menyimpan data tersebut untuk digunakan pada validasi pada saat melakukan login.

Tabel 4.12 Tabel Usecase *scenario* registrasi

Registrasi	
Actor	<i>User</i>
Objective	Menambahkan <i>member</i> baru ke dalam sistem.
Pre condition	Halaman registrasi terbuka.

Main flow	<ol style="list-style-type: none"> 1. Sistem menampilkan <i>form</i> isian data yang terdiri dari <i>email</i>, <i>username</i>, nama lengkap, <i>password</i> dan tanggal lahir. 2. Aktor mengisi <i>form</i> yang tersedia secara lengkap, dan menekan tombol 'registrasi'. 3. Sistem mengeksekusi proses penambahan <i>member</i>.
Alternative flow	<ol style="list-style-type: none"> 1. jika ada <i>input</i> yang belum lengkap, akan tetap di halaman registrasi, dan menampilkan pesan "silakan mengisi seluruh <i>form</i> secara lengkap"
Post Condition	Penambahan <i>member</i> baru diproses, kemudian akan dialihkan ke halaman login disertai pesan "berhasil terdaftar"

4.6.8 Usecase Scenario Lupa *password*

Pada *Use case scenario* dari lupa *password*, dijelaskan proses untuk membantu *user* ketika tidak mengingat *password*, di mana dijelaskan pada Tabel 4.13 bahwa sistem akan mengirim *password* ke *email member* yang sudah terdaftar.

Tabel 4.13 Tabel Usecase *scenario* lupa *password*

Lupa password	
Actor	<i>User</i>
Objective	Mengirim <i>password</i> ke <i>email</i> yang sudah didaftarkan.
Pre condition	Halaman lupa <i>password</i> terbuka.
Main flow	<ol style="list-style-type: none"> 1. Sistem menampilkan <i>form</i> isian yang terdiri dari <i>email member</i>. 2. Aktor mengisi <i>form email</i> dan memilih tombol 'kirim <i>password</i>'
Alternative flow	<ol style="list-style-type: none"> 1. jika <i>email</i> tidak terdaftar, sistem akan menampilkan pesan "<i>email</i> tidak terdaftar" dan kembali ke halaman lupa <i>password</i>.
Post Condition	<i>Password</i> dikirimkan kepada <i>email member</i> , dialihkan ke halaman home dan menampilkan pesan "silakan mengecek email"



4.6.9 Usecase Scenario Logout

Pada Tabel 4.14 menjelaskan *use case scenario* dari proses *logout* yang dilakukan oleh *member*, maupun administrator. Di mana pada proses ini, sistem akan mencabut hak akses *member* maupun administrator untuk mengakses halaman dashboard masing-masing, sehingga kembali memiliki hak akses *user*.

Tabel 4.14 Tabel Usecase *scenario* logout

Logout	
Actor	<i>Member</i> dan administrator
Objective	Mencabut hak akses <i>member</i> maupun administrator, menjadi hak akses <i>user</i> .
Pre condition	Mengakses sistem sebagai <i>member</i> maupun administrator.
Main flow	<ol style="list-style-type: none"> 1. Aktor memilih tombol <i>logout</i> 2. Sistem menampilkan pesan “apakah yakin ingin keluar?” 3. Aktor memilih “ya”
Alternative flow	-
Post Condition	Hak akses sebagai <i>user</i> dicabut, hak akses aktor menjadi hak akses <i>user</i> dan halaman home akan terbuka.

4.6.10 Usecase Scenario Lihat daftar *member*

Pada Tabel 4.15 dijelaskan mengenai Usecase *scenario* lihat daftar *member*, di mana proses untuk menampilkan daftar *member* yang terdaftar pada sistem dalam bentuk tabel kepada administrator.

Tabel 4.15 Tabel Usecase *scenario* lihat daftar *member*

Lihat daftar <i>member</i>	
Actor	<i>Administrator</i>
Objective	Menampilkan tabel daftar <i>member</i> .
Pre condition	Halaman daftar <i>member</i> terbuka
Main flow	<ol style="list-style-type: none"> 1. Sistem mengambil data, 2. Sistem menampilkan data berupa nama, username, email, jumlah project ke dalam tabel, dan menampilkan jumlah total member yang terdaftar.

Alternative flow	-
Post Condition	Halaman daftar <i>member</i> terbuka, dan menampilkan data <i>member</i> .

4.6.11 Usecase Scenario hapus *member*

Use case scenario hapus *member* pada Tabel 4.16 menjelaskan mengenai *scenario* yang terjadi pada saat administrator menghapus *member* dari daftar *member*. Proses ini dilakukan dengan menekan tombol hapus pada salah satu *member* yang ingin dihapus dari daftar *member*. Sebelum *file* terhapus, sistem akan menampilkan pesan dialog.

Tabel 4.16 Tabel Usecase *scenario* hapus *member*

Hapus <i>member</i>	
Actor	<i>Administrator</i>
Objective	Menghapus data terpilih dari tabel daftar <i>member</i> .
Pre condition	Halaman daftar <i>member</i> terbuka, dan terdapat data.
Main flow	<ol style="list-style-type: none"> 1. Sistem menampilkan tabel data, beserta tombol hapus disetiap barisnya. 2. Aktor memilih tombol hapus disalah satu baris. 3. Sistem menampilkan pesan “apakah yakin ingin menghapus?” 4. Aktor memilih “ya”
Alternative flow	<ol style="list-style-type: none"> 1. Jika aktor memilih “tidak”, maka akan kembali ke halaman daftar <i>member</i>.
Post Condition	Data <i>member</i> terpilih terhapus, dan kembali ke halaman daftar <i>member</i> .

4.6.12 Usecase Scenario Lihat daftar *project*

Pada Tabel 4.17 dijelaskan mengenai Usecase *scenario* lihat daftar *project*, di mana proses untuk menampilkan daftar *project* yang terdaftar pada sistem dalam bentuk tabel kepada administrator, beserta nama *member* yang membuat.

Tabel 4.17 Tabel Usecase *scenario* lihat daftar *project*

Lihat daftar <i>project</i>	
Actor	<i>Administrator</i>
Objective	Menampilkan tabel daftar <i>project</i> .
Pre condition	Halaman daftar <i>project</i> terbuka

Main flow	<ol style="list-style-type: none"> 1. Sistem mengambil data, 2. Sistem menampilkan data project berupa nama project, member yang membuat ke dalam tabel, dan juga jumlah total project yang ada.
Alternative flow	-
Post Condition	Halaman daftar <i>project</i> terbuka, dan menampilkan data <i>project</i> .

4.6.13 Usecase Scenario lihat daftar *project member*

Pada Tabel 4.18 dijelaskan mengenai Usecase *scenario* lihat daftar *project*, di mana proses untuk menampilkan daftar *project* yang telah dibuat oleh *member* itu kepada *member*. Pada proses ini, apabila tidak terdapat data *project*, maka akan menampilkan tulisan bahwa *project* belum ditemukan.

Tabel 4.18 Tabel Usecase *scenario* lihat daftar *project member*

Lihat daftar <i>project member</i>	
Actor	<i>Member</i>
Objective	Dapat melihat daftar <i>project</i> yang dimiliki oleh <i>member</i> tersebut.
Pre condition	Halaman daftar <i>project member</i> terbuka.
Main flow	<ol style="list-style-type: none"> 1. Sistem mengambil data berupa <i>project</i> yang dimiliki oleh aktor. 2. Sistem menampilkan data ke dalam tabel.
Alternative flow	Jika belum ada data, Sistem menampilkan tulisan “belum ada <i>project</i> ”
Post Condition	Halaman daftar <i>project member</i> terbuka, dan menampilkan data <i>project</i> yang dimiliki.

4.6.14 Usecase Scenario buat *project baru*

Pada Tabel 4.19 dijelaskan mengenai Usecase *scenario* buat *project baru* yang akan menambah *project baru* pada daftar *project* yang dimiliki oleh *member* tersebut. Pada saat membuat *project baru*, *member* sekaligus mengisi nama *project* yang dibuat. Setelah itu, sistem akan langsung mengarahkan kehalaman daftar *project member* dengan data yang sudah diupdate.



Tabel 4.19 Tabel Usecase *scenario* buat *project* baru

Buat <i>project</i>	
Actor	<i>Member</i>
Objective	Menambah data <i>project</i> baru ke dalam sistem.
Pre condition	Halaman daftar <i>project member</i> terbuka.
Main flow	<ol style="list-style-type: none"> 1. Aktor memilih tombol ‘tambah <i>project</i>’ 2. Sistem menampilkan <i>form</i> isian yang berisi nama <i>project</i>, 3. Aktor mengisi <i>form</i> yang tersedia, dan memilih tombol “buat <i>project</i>”
Alternative flow	-
Post Condition	<i>Project</i> baru dibuat, dan kembali ke halaman daftar <i>project member</i> .

4.6.15 Usecase Scenario hapus *project*

Use case scenario hapus *project*, sesuai pada Tabel 4.20, bertujuan untuk dapat menghapus *project* yang sudah ada, dengan menekan tombol hapus pada salah satu *member* yang ingin dihapus pada halaman daftar *project*. Pada saat sebelum sistem menghapus data *project*, sistem akan menampilkan pesan, karena nantinya tidak hanya *project* saja yang terhapus, namun juga isi diagram di dalamnya.

Tabel 4.20 Tabel Usecase *scenario* hapus *project*

Hapus <i>project</i>	
Actor	<i>Member</i>
Objective	Menghapus data <i>project</i> yang dimiliki.
Pre condition	Halaman daftar <i>project member</i> terbuka.
Main flow	<ol style="list-style-type: none"> 1. Sistem menampilkan data <i>project</i> beserta tombol hapus. 2. Aktor memilih salah satu tombol ‘hapus’ pada baris data. 3. Sistem menampilkan pesan “apakah yakin ingin menghapus <i>project</i> beserta diagram di dalamnya?” 4. Aktor memilih tombol “ya”
Alternative flow	Jika aktor memilih tombol ‘tidak’, Halaman daftar <i>project</i> akan terbuka kembali.
Post Condition	Data pilihan akan terhapus, dan kembali ke halaman daftar <i>project</i> .

4.6.16 Usecase Scenario lihat detail *project*

Pada Tabel 4.21 dijelaskan mengenai Usecase *scenario* lihat detail *project*, di mana adalah informasi detail *project* ketika dipilih dari daftar *project* oleh *member*. Di dalam detail *project* ini, nantinya akan menampilkan informasi *project* yang berupa diagram, mapun fungsi-fungsi utama yang terdapat pada sistem.

Tabel 4.21 Tabel Usecase *scenario* lihat detail *project*

Lihat detail <i>project</i>	
Actor	<i>Member</i>
Objective	Menampilkan detail <i>project</i> yang dipilih.
Pre condition	Halaman daftar <i>project member</i> terbuka.
Main flow	<ol style="list-style-type: none"> 1. Sistem menampilkan data <i>project member</i> 2. Aktor memilih salah satu <i>project</i> dari daftar <i>project</i> yang dimiliki oleh <i>member</i>.
Alternative flow	<ol style="list-style-type: none"> 1. Jika belum melakukan mapping, maka akan membuka halaman workspace dan tombol mapping. 2. Jika belum ada erd yang tersimpan, akan menampilkan form untuk menginput file dan dua tombol yaitu tombol 'buat erd' dan tombol 'upload erd'
Post Condition	Halaman detail <i>project</i> akan terbuka dan menampilkan detail erd yang telah dimapping, beserta table relasinya.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan

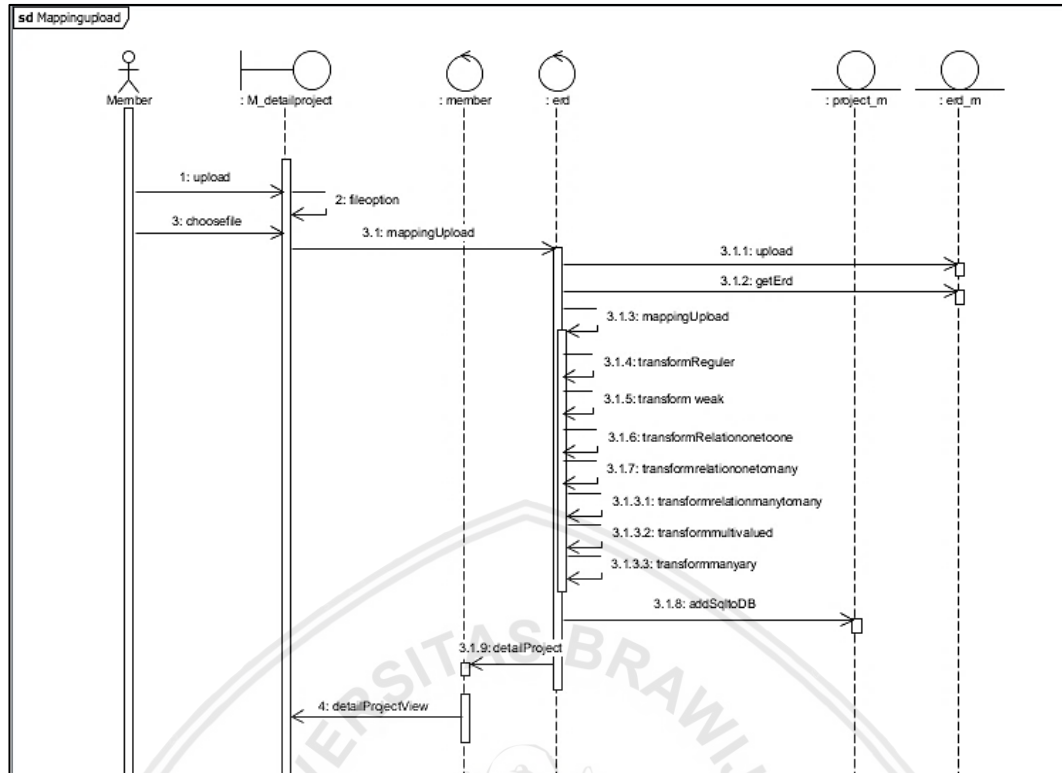
Pada tahap ini, setelah melakukan proses analisis kebutuhan untuk mengetahui kebutuhan-kebutuhan apa saja yang dibutuhkan, maka objek-objek akan diidentifikasi berdasarkan spesifikasi kebutuhan dan juga *use case scenario* yang sudah didapatkan. Akan dijelaskan dengan lebih detail lagi objek-objek yang dihasilkan, serta menentukan hubungan antar objek-objek tersebut. Pada sub-bab ini, akan berisi perancangan yang terdiri dari perancangan *sequence diagram*, *class diagram*, *database*, dan juga perancangan antarmuka.

5.1.1 Sequence Diagram

Sesuai dengan model pengembangan sistem yang digunakan yaitu model waterfall, setelah didapatkan hasil analisis kebutuhan. Kemudian akan menghasilkan perancangan dari sistem, yaitu runtutan setiap kebutuhan fungsional. Di sini akan dijelaskan *sequence diagram* dari lima fungsi utama yang diambil dari analisis kebutuhan sistem.

5.1.1.1 Sequence Diagram Pemetaan Upload ERD

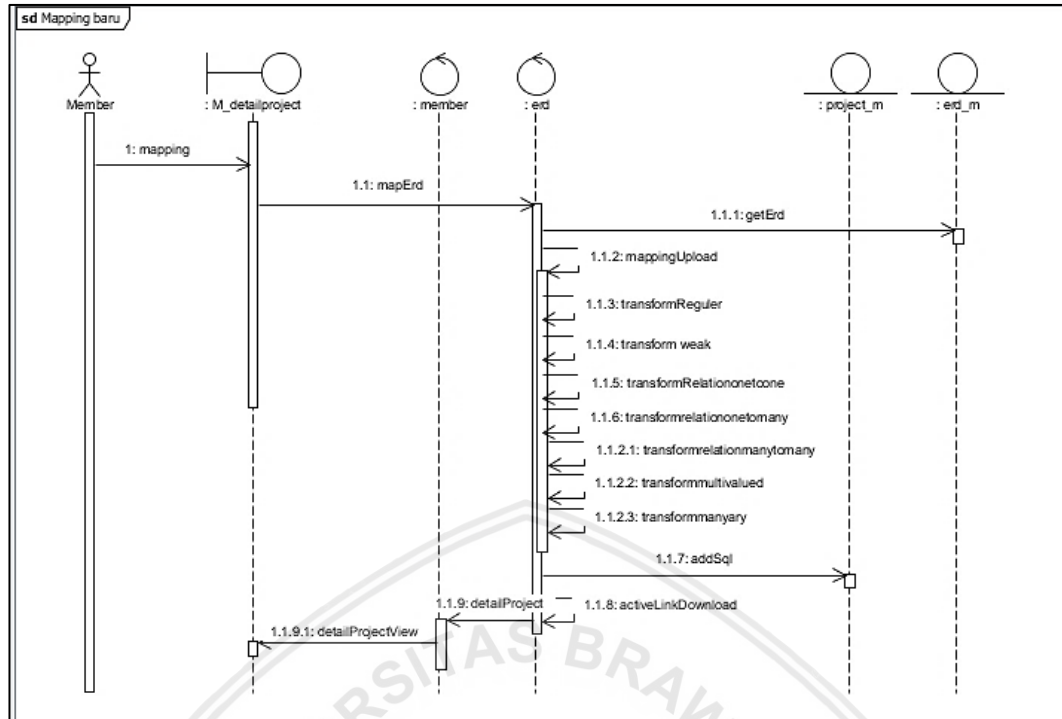
Pada *sequence diagram* Gambar 5.1, dapat diketahui bahwa yang berperan sebagai aktor adalah *member*. Pada proses Pemetaan *upload ERD*, *member* akan memanggil *uploadErd* pada boundary *M_detailProject* di mana akan menampilkan modal dialog. *Member* setelah itu akan memanggil *upload* pada boundary *M_detailProject* yang akan membuka pilihan *file* yang akan dipilih oleh aktor. Setelah itu akan memanggil pesan untuk mengupload pada controller *erd*. Controller *erd* nantinya akan memanggil proses *uploadErd* yang akan mengirim pesan kepada entity *erd_m* dan juga entity *project_m*. Setelah itu controller *erd* akan memanggil proses mapping upload, di mana akan langsung mengeksekusi ketujuh proses mapping yang nantinya akan menyimpan hasil sql. Kemudian controller akan memanggil kembali boundary *M_detailProject*.



Gambar 5.1 Sequence diagram Pemetaan Upload ERD

5.1.1.2 Sequence Diagram Pemetaan ERD baru

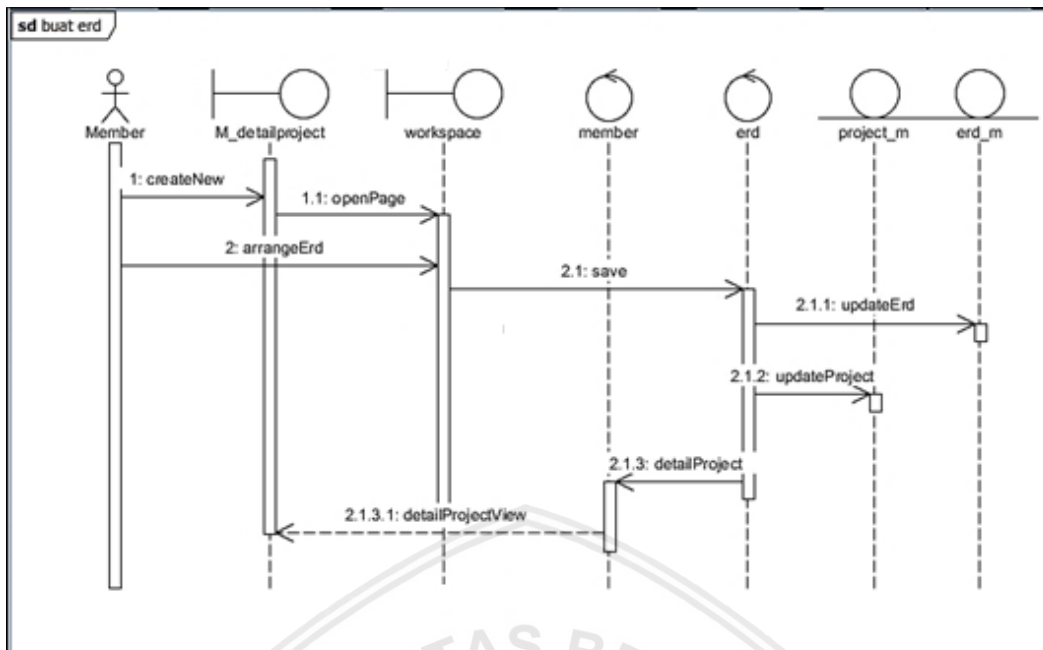
Pada *sequence* diagram pemetaan ERD pada Gambar 5.2 dapat dilihat bahwa *member* sebagai aktor memanggil proses mapping pada *M_detailProject*, yang kemudian memanggil *mapErd* pada controller erd, yang akan mengambil data dari entitas *erd_m*, kemudian mengolah data dengan pemanggilan *createSql*. Setelah itu, akan menambah data pada entitas *project_m*, dan memanggil proses dalam controller erd itu sendiri untuk proses *activelinkdownload*. Setelah semua selesai akan meredirect kembali ke boundary *M_detailProject*.



Gambar 5.2 Sequence diagram Pemetaan ERD baru

5.1.1.3 Sequence Diagram Buat ERD

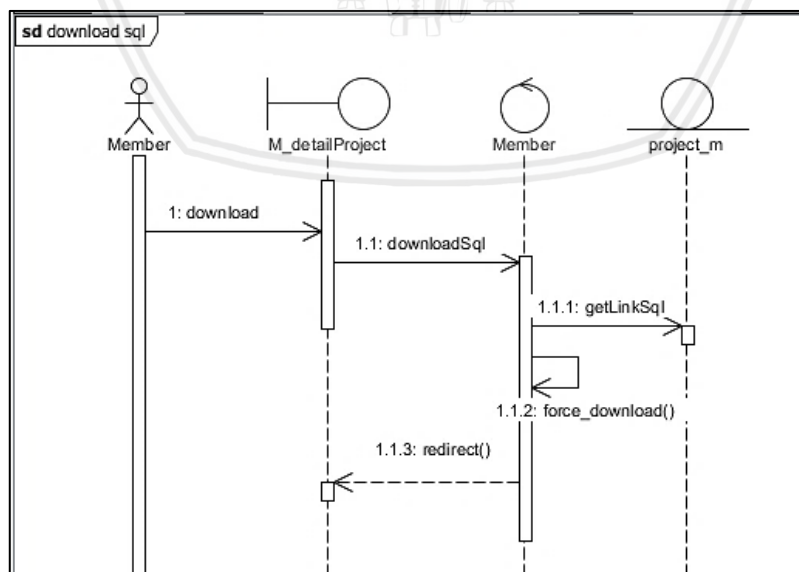
Pada *sequence* buat ERD, seperti yang dijelaskan pada Gambar 5.3, terdapat *member* sebagai aktor, yang akan memanggil *createErd* pada boundary *M_detailProject*. Boundary *M_detailProject* akan memanggil controller *member* untuk memanggil boundary buatERD. Setelah itu aktor akan menyusun erd pada boundary yang nantinya akan memanggil proses *save* pada controller *erd* dan akan mengupdate entity *erd_m* dan juga *project_m*. Setelah itu controller akan meredirect ke boundary *M_detailProject*.



Gambar 5.3 Sequence diagram buat ERD

5.1.1.4 Sequence Diagram Download SQL

Pada *sequence diagram download sql* dalam Gambar 5.4, *member* sebagai aktor akan memanggil proses *download* pada boundary *M_detailProject*. Setelah itu, akan memanggil proses *downloadSql* pada controller *erd*. Nantinya controller akan mengambil data dari entity *project_m* dan kemudian memanggil proses *download* pada controller itu sendiri dan kemudian meredirect ke boundary *M_detailProject*.



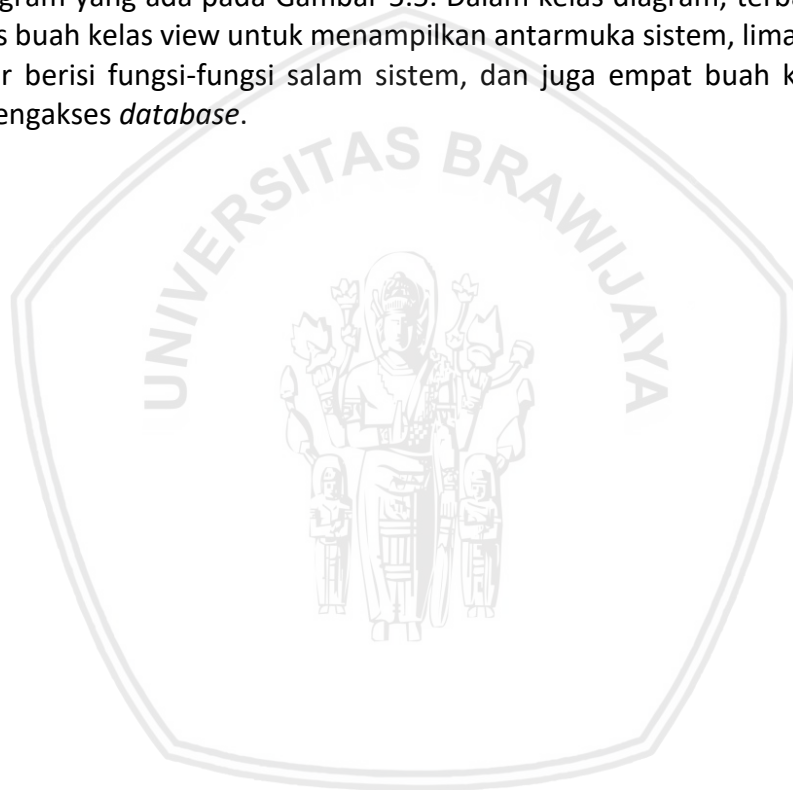
Gambar 5.4 Sequence diagram Download SQL

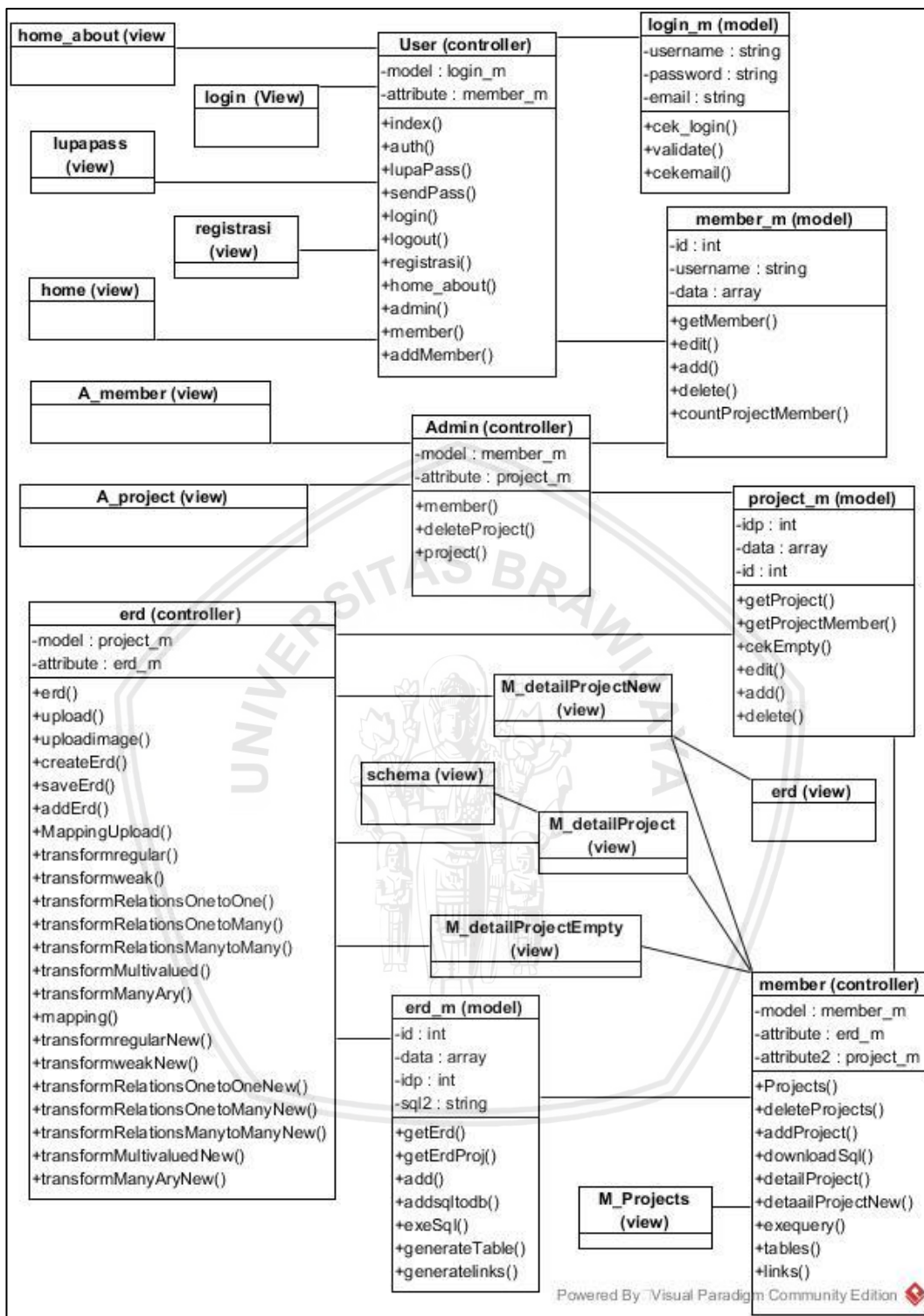
5.1.2 Class Diagram

Perancangan kelas diagram adalah penggambaran objek-objek yang dibutuhkan dan didapat dari hasil analisis kebutuhan dan perancangan *sequence* diagram. Pada perancangan kelas, akan digambarkan mengenai objek beserta relasi-relasi yang ada. Ditahap ini, juga dijelaskan mengenai perancangan detail dari kelas-kelas yang telah digambarkan pada perancangan umum.

5.1.2.1 Perancangan Umum

Perancangan umum dilakukan untuk mendeskripsikan objek-objek ada dan relasi di antara objek-objek. Pada perancangan umum akan digambarkan sebagai kelas diagram yang ada pada Gambar 5.5. Dalam kelas diagram, terbagi menjadi tiga belas buah kelas view untuk menampilkan antarmuka sistem, lima buah kelas controller berisi fungsi-fungsi sistem, dan juga empat buah kelas model untuk mengakses *database*.





Gambar 5.5 Class Diagram



5.1.2.2 Perancangan Detail

Perancangan detail akan berisi penjelasan mengenai operasi apa saja yang terdapat pada kelas-kelas yang telah digambarkan pada kelas diagram Gambar 5.6. Pada perancangan detail, akan dijelaskan mengenai algoritma setiap operasi yang ada, pada penelitian ini akan diambil 4 buah kelas utama.

5.1.2.2.1 Kelas member

Pada Tabel 5.1 dijabarkan mengenai detail kelas yang terdapat pada controller *member*. Terdapat tujuh buah method atau operasi.

Tabel 5.1 Perancangan detail kelas controller *member*

Nama Operasi	Level Akses	Tipe
Projects()	public	void
detailProject()	public	Void
detailProjectNew()	public	Void
Projects()	public	Void
deleteproject()	Public	Void
addProject()	public	void
downloadSql()	public	void

5.1.2.2.1 Kelas erd

Pada Tabel 5.2 dijabarkan mengenai detail kelas yang terdapat pada controller *erd* sebagai controller untuk segala macam fungsi yang berhubungan dengan *erd*. Terdapat 12 buah method atau operasi.

Tabel 5.2 Perancangan detail kelas controller *erd*

Nama Operasi	Level Akses	Tipe
<i>upload()</i>	<i>public</i>	Void
<i>Erd()</i>	<i>public</i>	void
<i>addErd()</i>	<i>public</i>	Void
<i>saveErd()</i>	<i>public</i>	Void
Uploadimage()	<i>public</i>	Void
MappingUpload()	<i>public</i>	Void
Transformrregular()	<i>public</i>	Void
Transformweak()	<i>public</i>	Void

Nama Operasi	Level Akses	Tipe
Transformrelationonetomany()	<i>public</i>	Void
Transformrelationmanytomany()	<i>public</i>	Void
Trasnformmultivalued()	<i>public</i>	Void
Transformmanyary()	<i>public</i>	Void
Mapping()	<i>public</i>	Void
TransformrregularNew()	<i>public</i>	Void
TransformweakNew()	<i>public</i>	Void
TransformrelationonetomanyNew()	<i>public</i>	Void
TransformrelationmanytomanyNew()	<i>public</i>	Void
TrasnformmultivaluedNew()	<i>public</i>	Void
TransformmanyaryNew()	<i>public</i>	Void

5.1.2.2.1 Kelas admin

Pada Tabel 5.3 dijabarkan mengenai detail kelas yang terdapat pada controller administrator sebagai controller untuk segala macam fungsi yang berhubungan dengan aktor administrator. Terdapat tiga buah method atau operasi.

Tabel 5.3 Perancangan detail kelas controller admin

Nama Operasi	Level Akses	Tipe
<i>Member()</i>	<i>public</i>	Void
<i>Project()</i>	<i>public</i>	Void
<i>deleteProject()</i>	<i>public</i>	Void

5.1.2.2.1 Kelas user

Pada Tabel 5.4 dijabarkan mengenai detail kelas yang terdapat pada controller *user* sebagai controller untuk segala macam fungsi yang berhubungan dengan *user* seperti login, registrasi, dan juga lupa *password*. Terdapat sembilan buah method.

Tabel 5.4 Perancangan detail kelas controller user

Nama Operasi	Level Akses	Tipe
Index()	<i>public</i>	Void
Auth()	<i>public</i>	Void
lupaPass()	<i>public</i>	Void
SendPass()	<i>public</i>	Void

Nama Operasi	Level Akses	Tipe
login()	public	Void
Registrasi()	public	Void
Homeabout()	public	Void
Logout()	public	Void
AddMember()	public	Void

5.1.2.2.1 Perancangan Algoritma operasi mapping ERD upload dari kelas erd

Pada perancangan operasi mapping dari kelas erd, didapatkan potongan algoritma seperti pada Tabel 5.5, di mana setelah tahap inialisasi variabel untuk mendapatkan file JSON, akan mendapatkan isi dari file, melakukan proses upload file erd, dan melakukan *decode* json ke dalam array. Nantinya akan melakukan setiap tahap yaitu tujuh tahap untuk mentransform data dari JSON menjadi sebuah script sql yang akan disimpan dalam file sql.

Tabel 5.5 Potongan algoritma operasi mapping ERD upload

NO	Source Code
1	START
2	Inialisasi file, jsonFile, link, array json,
3	Jika file_get_content berhasil
4	do
5	Decode file jsonFile
6	Jika write_file berhasil
7	do
8	Eksekusi proses transform regular entity,
9	Eksekusi proses transform weak entity ,
10	Eksekusi proses transform relation one to one,
11	Eksekusi proses transform relation one to many,
12	Eksekusi proses transform relation many to many,
13	Eksekusi proses transform atribut multivalue
14	Eksekusi proses transform many relations
15	Jika Simpan sql ke DB berhasil
16	set flashdata berhasil;

NO	Source Code
17	Jika gagal
18	Set flash data gagal
19	Jika file get content gagal, show error
20	END

5.1.2.2.1 Perancangan Algoritma operasi transform regular entity dari kelas erd

Pada perancangan operasi transform regular entity dari kelas erd, didapatkan potongan algoritma seperti pada Tabel 5.6. proses ini berada di dalam proses mapping upload. Pada tahap awal akan melakukan inialisasi array yang dibutuhkan, kemudian akan melakukan pencarian tiap data dari json yang sudah decode.pada tahap ini, akan mencari shape dengan bentuk regular, untuk dijadikan dalam bentuk sql beserta tiap atributnya yang didapatkan dari koneksi tiap data pada json.

Tabel 5.6 Potongan algoritma operasi transform regular entity

NO	Source Code
1	START
2	Inialisasi array atts;
3	Foreach shape
4	Jika tipe = "entity" dan detail -> type ="regular"
5	Write to file "CREATE TABLE".(detail->name).")"
6	Foreach connectors
7	Jika tipe ="connector" dan destination = shape->id
8	Foreach shape
9	If tipe ="attribute" dan id==connector->source
10	If attribute->isunique = 1
11	Save to array atts = attribute->name. "NOT NULL PRIMARY KEY"
12	Else
13	save to array atts = atribut->name.
14	end if
15	end if
16	end foreach

NO	Source Code
17	write to file = implode array atts
18	write to file “)”
19	end if
20	end foreach
21	end if
22	end foreach
23	return true;
24	

5.1.2.2.2 Perancangan Algoritma operasi transform relation one to one

Pada perancangan operasi transform relation one to one dari kelas erd, didapatkan potongan algoritma seperti pada Tabel 5.7. proses ini berada di dalam proses mapping upload. Pada tahap awal akan melakukan inialisasi array-array yang dibutuhkan, kemudian melakukan pencarian tiap data dari json yang sudah decode.pada tahap ini, akan mencari shape dengan tipe relationship, dan mengecek kardinalitas dan id connectornya.

Tabel 5.7 Potongan algoritma operasi transform relation one to one

NO	Source Code
1	START
2	Inialisasi array cardinality, idcon,
3	Foreach shape
4	if tipe = “relationship”
5	Foreach shape->slots
6	Save to array cardinality = shape->slots->cardinality
7	Save to array idcon = shape->slots->enityid
8	End foreach
9	if dalam array cardinality ada “many”
10	Continue
11	Else
12	Write to file “ALTER TABLE”
13	End if
14	Foreach shape
15	Foreach idcon

NO	Source Code
16	if shape->id == idcon
17	ambil index pertama
18	write to file shape->name. " ADD FOREIGN KEY ("
19	ambil index kedua,
20	cari connector yang destination = index kedua.
21	Foreach shape
22	If tipe ="attribute" dan id==connector->source
23	If attribute->isunique = 1
24	Save to array atts = attribute->name. "REFERENCE "
25	shape->name.
26	end if
27	end if
28	end foreach
29	end if
30	end foreach
31	end foreach
32	end foreach
33	end if
34	end foreach
35	return true;
36	end

5.1.2.2.3 Perancangan Algoritma operasi mapping ERD baru dari kelas erd

Pada perancangan operasi mapping erd baru dari kelas erd, didapatkan potongan algoritma seperti pada Tabel 5.8, di mana setelah tahap inialisasi variabel untuk mendapatkan file JSON, akan mendapatkan isi dari file, dan melakukan *decode* json ke dalam array. Nantinya akan melakukan setiap tahap yaitu tujuh tahap untuk mentransform data dari JSON menjadi sebuah script sql yang akan disimpan dalam file sql. Pada operasi ini sama dengan proses mapping upload, namun memanggil method mapping yang berbeda.

Tabel 5.8 Potongan algoritma operasi mapping ERD baru

NO	Source Code
1	START
2	Inisialisasi <i>file</i> , <i>jsonFile</i> , <i>link</i> , <i>array json</i> ,
3	Jika <i>file_get_content</i> berhasil
4	do
5	Decode <i>file jsonFile</i>
6	Jika <i>write_file</i> berhasil
7	do
8	Eksekusi proses transform regular entity new,
9	Eksekusi proses transform weak entity new,
10	Eksekusi proses transform relation one to one new,
11	Eksekusi proses transform relation one to many new,
12	Eksekusi proses transform relation many to many new,
13	Eksekusi proses transform atribut multivalued new
14	Eksekusi proses transform many relations new
15	Jika Simpan sql ke DB berhasil
16	set flashdata berhasil;
17	Jika gagal
18	Set flash data gagal
19	Jika file get content gagal, show error
20	END

5.1.2.2.4 Perancangan Algoritma operasi buat erd dari kelas erd

Pada perancangan operasi buat ERD dari kelas erd, didapatkan potongan algoritma seperti pada Tabel 5.9. Pada tahap awal akan melakukan proses inisialisasi variabel yang dibutuhkan, kemudian membuat file json dengan nama project dan menyimpan data di dalam file. Menyimpan variable-variabel ke dalam data untuk dilakukan proses adder dari model erd_m.

Tabel 5.9 Potongan algoritma operasi buat erd

NO	algoritma
1	Start
2	Inisialisasi data, iderd, idproject, nameproject
3	Write file nameproject.json = data
4	Save to Array info data, iderd, idproject
5	addErd(info) dari model erd_m
	end

5.1.2.2.5 Perancangan Algoritma operasi download erd dari kelas erd

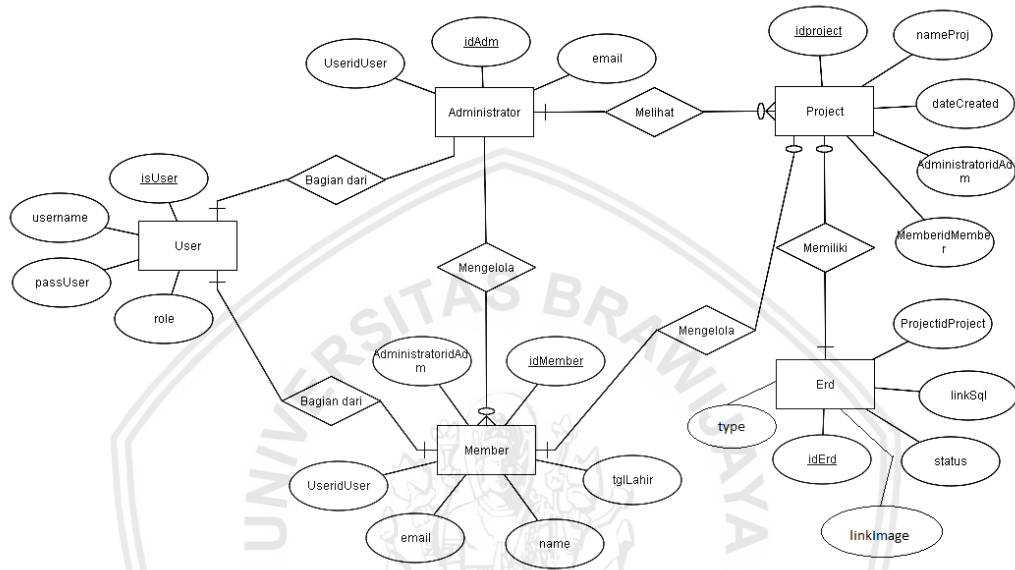
Pada perancangan operasi download ERD dari kelas erd, didapatkan potongan algoritma seperti pada Tabel 5.10. Pada tahap awal akan melakukan proses inisialisasi variabel yang dibutuhkan, kemudian akan mendapatkan path file, dan mengeksekusi download atau force_download file sesuai path yang tersedia di ci, dan apabila gagal akan menampilkan pesan error, sedangkan apabila berhasil akan memanggil view detail project.

Tabel 5.10 Potongan algoritma operasi download erd

NO	algoritma
1	Start
2	Inisialisasi file path
3	Get linksql dari model
4	Ekseskusi download dari ci.
5	if download linksql gagal
6	show error
7	else
8	open view detail project
9	end if
10	end

5.1.3 Perancangan Basis Data

Perancangan basis data akan dilakukan secara konseptual dan secara fisik. Perancangan ini menjelaskan struktur sistem yang terdiri dari entitas-entitas yang saling berelasi. Perancangan secara konseptual akan dilakukan menggunakan Entity Relationship Diagram (ERD) yang digambarkan pada Gambar 5.6. pada diagram konseptual digambarkan memiliki 5 buah kelas yang terdiri dari kelas *user*, *administrator*, *project*, *member*, dan *erd*.



Gambar 5.6 Conceptual Data Model

Dari hasil diagram konseptual dapat dikembangkan lagi menjadi diagram secara logical, di mana sudah *memberikan* primary key, tipe data maupun dimunculkan kolom-kolom yang membuat relasi antar kelas.

Dari perancangan konseptual di atas, dapat dibuat struktur tabel tiap entitas, yaitu pada Tabel 5.11 adalah struktur tabel user, berisi data-data user.

Tabel 5.11 Struktur tabel User

No	Nama Field	Tipe Data	Deskripsi
1.	idUser	Integer(10)	Nomor id data <i>user</i>
2.	passUser	Varchar(255)	Password tiap <i>user</i> yang digunakan untuk login
3	Username	Varchar(255)	Username tiap <i>user</i> yang digunakan untuk login
4.	role	Integer(10)	Peran <i>user</i> untuk login sebagai <i>member</i> atau administrator.

Pada Tabel 5.12 berisi struktur tabel member, berisi data member yang nantinya akan terkait dengan tabel user.

Tabel 5.12 Struktur tabel *Member*

No	Nama Field	Tipe Data	Deskripsi
1.	<i>Idmember</i>	Integer(10)	Nomor id data <i>member</i>
2.	<i>tglLahir</i>	Date	Tanggal lahir <i>member</i>
3.	Name	Varchar(255)	Data nama lengkap <i>member</i>
4.	<i>email</i>	Varchar(255)	Data <i>email member</i> untuk proses lupa <i>password</i>

Pada Tabel 5.13 adalah struktur dari tabel administrator, berisi data dari administrator. Data ini hampir sama dengan tabel member, yaitu saling berhubungan dengan tabel user.

Tabel 5.13 Struktur tabel *Administrator*

No	Nama Field	Tipe Data	Deskripsi
1.	<i>idAdm</i>	Integer(10)	Nomor id data administrator
2.	<i>email</i>	Varchar(255)	Data <i>email administrator</i> untuk proses lupa <i>password</i>

Dan untuk Tabel 5.14, adalah struktur tabel project, berisi data-data yang diperlukan untuk mengelola project oleh member maupun administrator.

Tabel 5.14 Struktur tabel *Project*

No	Nama Field	Tipe Data	Deskripsi
1.	<i>idProject</i>	Integer(10)	Nomor id data <i>project</i>
2.	<i>nameProj</i>	Varchar(255)	Data nama <i>project</i>
3.	<i>dateCreated</i>	Date	Tanggal pembuatan <i>project</i>

Dan tabel terakhir yaitu Tabel 5.15 adalah struktur tabel erd, berisi data erd yang nantinya akan digunakan untuk mengelola erd yang akan dibuat, diupload, maupun dipetakan oleh sistem. nantinya tabel ini akan berhubungan dengan tabel project.

Tabel 5.15 Struktur tabel Erd

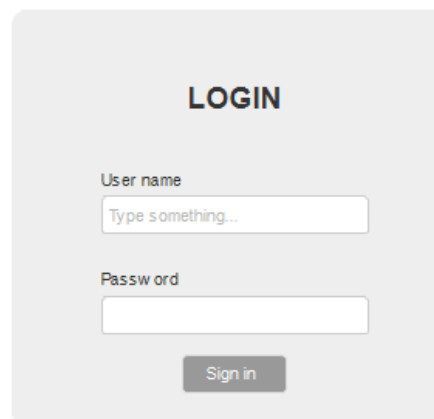
No	Nama Field	Tipe Data	Deskripsi
1.	idErd	Integer(10)	Nomor id data erd
2.	Status	Integer(10)	Status aktif link sql
3.	linkSql	Varchar(500)	Link untuk mendownload sql
4	linkErd	Varchar(500)	Link untuk mengupload erd.
5	Project.idProject	Integer(10)	Nomor id project
6	Type	Integer()	Untuk membedakan antara project yang di upload dan dibuat dengan diagram
7	linkImage	Varchar (500)	Link untuk membuka gambar pendukung

5.1.4 Perancangan Antarmuka

Pada perancangan antarmuka akan digambarkan antarmuka yang dibutuhkan dari hasil analisis kebutuhan dan juga hasil perancangan sistem. Antarmuka yang digambarkan akan menjadi dasar dalam pengimplementasian antarmuka sistem.

5.1.4.1 Perancangan antarmuka login

Pada perancangan antarmuka seperti pada Gambar 5.7 terdapat *form* isian yang terdiri dari isian untuk *username*, dan juga isian untuk *password*. Pada *form* login ini, akan disertakan juga sebuah tombol/tombol yang bertuliskan "sign in" untuk mengeksekusi proses login.

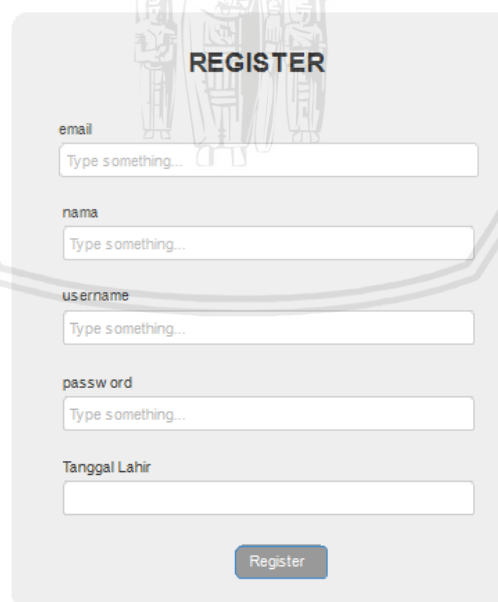


A login form titled "LOGIN" with a light gray background. It contains two input fields: "User name" with a placeholder "Type something..." and "Passw ord" (with a space between 'w' and 'o'). Below the fields is a "Sign in" button.

Gambar 5.7 Perancangan antarmuka login

5.1.4.2 Perancangan antarmuka Registrasi

Untuk perancangan antarmuka *form* Registrasi yang ditampilkan pada Gambar 5.8, terdapat *form* isian yang terdiri dari kolom isian untuk *email*, nama, *username*, *password* dan tanggal lahir. Pada *form* registrasi ini terdapat sebuah tombol/tombol bertuliskan "register" yang akan mengeksekusi proses registrasi setelah *user* mengisi semua kolom yang ada pada *form*.

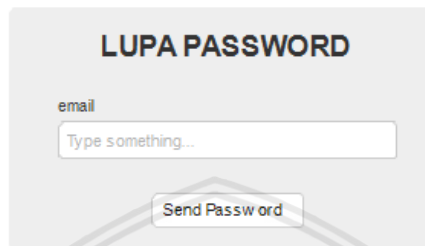


A register form titled "REGISTER" with a light gray background. It contains five input fields: "email", "nama", "username", "passw ord" (with a space between 'w' and 'o'), and "Tanggal Lahir". Below the fields is a "Register" button.

Gambar 5.8 perancangan antar muka registrasi

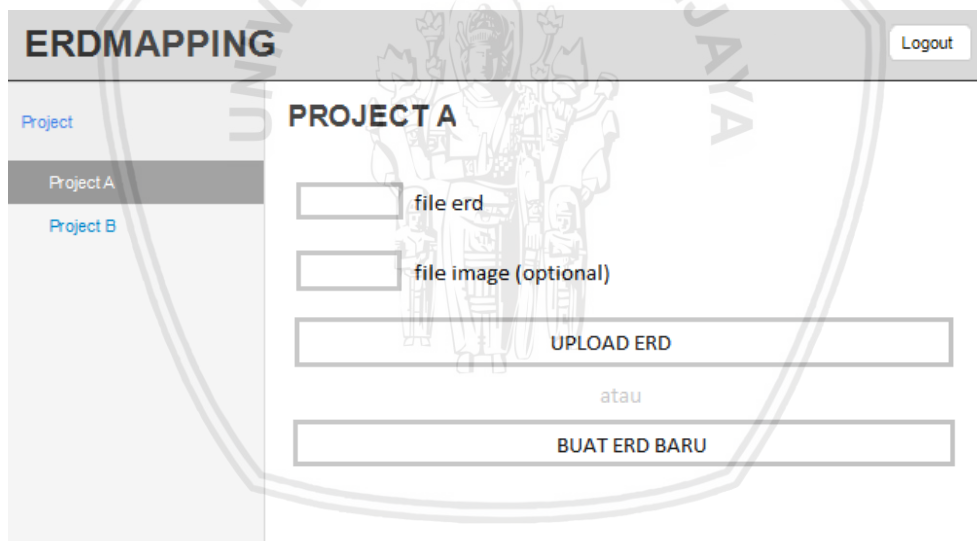
5.1.4.3 Perancangan antarmuka lupa password

Pada perancangan *form* Lupa Password yang dideskripsikan pada Gambar 5.9, terdapat kolom isian *email* di mana *user* akan mengisinya dengan alamat *email* yang terdaftar pada sistem. Form ini menyediakan tombol “Send Password” yang memiliki fungsi untuk mengeksekusi *email* yang diisi oleh *user* agar dapat diproses pada sistem.



Gambar 5.9 perancangan antar muka lupa *password*

5.1.4.4 Perancangan antarmuka Detail Project

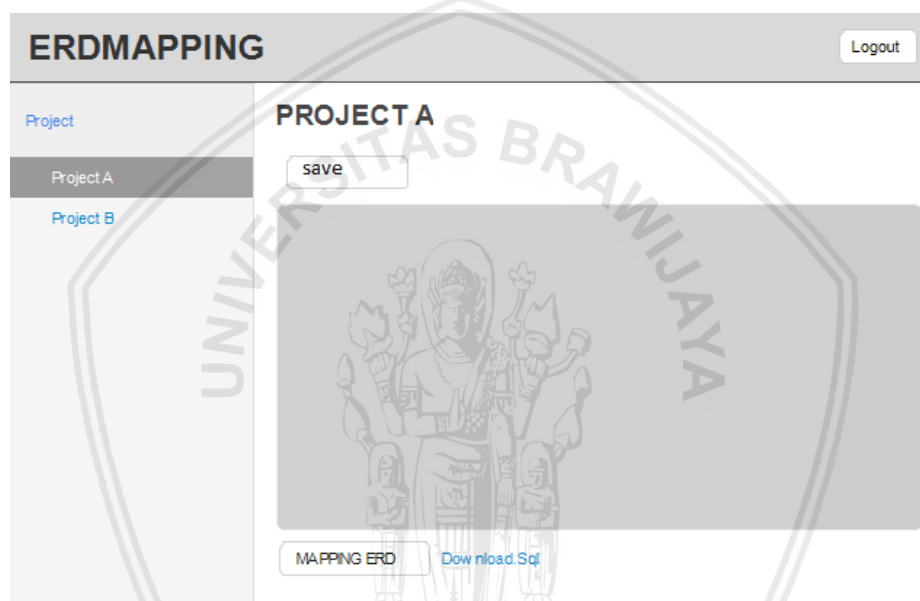


Gambar 5.10 perancangan antar muka detail *project* kosong

Pada perancangan antarmuka Detail *project* yang dapat diperhatikan pada Gambar 5.11 ini adalah terdapat beberapa tab pada sebelah kiri tampilan antarmuka yang akan menampilkan beberapa pilihan *project* yang telah dibuka oleh *user*, sehingga *user* dapat memilih *project* mana yang akan dikerjakan. Ketika *user* sudah memilih tab nama *project* yang diinginkan, antarmuka akan menampilkan tombol file input untuk memilih file erd dan juga gambar yang bersifat tidak wajib. Kemudian dibawahnya terdapat tombol “Upload ERD”. Fungsi tombol “Upload ERD” adalah untuk mengeksekusi perintah upload dan mapping file. Dibawahnya terdapat pilihan untuk membuat ERD baru. Pada sisi kanan atas

tampilan juga terdapat tombol “Logout” yang memiliki fungsi untuk *user* dapat keluar dari hak akses menggunakan sistem.

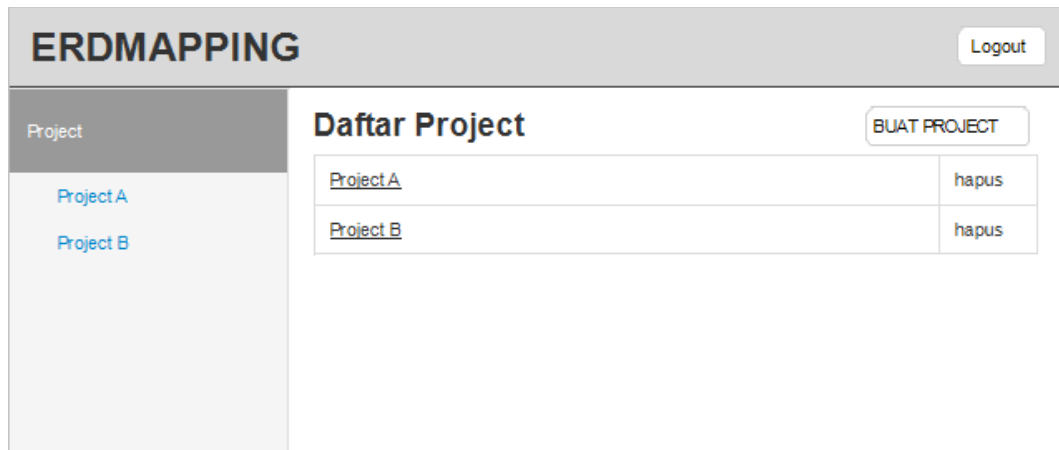
Pada tampilan antarmuka yang dapat diperhatikan di Gambar 5.11, terdapat tombol “Edit” yang berfungsi untuk perintah untuk menampilkan Form “Buat/Edit ERD”. Di bawah tombol “Edit” terdapat *workspace* tempat di mana sistem menampilkan kepada *user* diagram yang sudah dibuat pada *project* tersebut. Pada bawah bagian tampilan antarmuka terdapat tombol “Mapping ERD” yang berfungsi untuk menampilkan mapping ERD yang tersedia pada sistem untuk membantu *user* dalam pengerjaan membuat/mengubah ERD. Di sebelah kanan tombol “Mapping ERD” terdapat link “Download sql” yang berfungsi akan mengantar *user* ke *browser* untuk mendownload sql yang dibutuhkan.



Gambar 5.11 perancangan antar muka detail *project*

5.1.4.5 Perancangan antarmuka Daftar Project (Member)

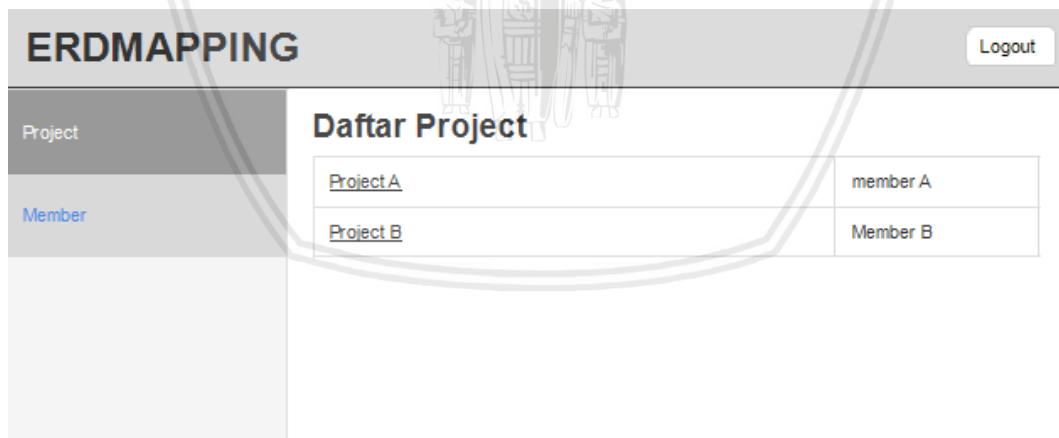
Untuk tampilan perancangan antarmuka “Daftar *Project*” untuk *user* dengan hak akses “*member*” dapat dilihat di Gambar 5.12, di mana pada tab bagian kiri tampilan antarmuka terdapat tab *Project* yang juga menampilkan sub-tab berupa *Project* apa saja yang sudah dibuat. Pada tampilan ini, terdapat tombol “Buat *Project*” yang berfungsi untuk mengeksekusi perintah untuk menampilkan *form* “Buat *Project* Baru”. Tampilan antarmuka ini juga menampilkan tabel “Daftar *Project*” yang menampilkan *project-project* yang sedang dikerjakan oleh *user*. Untuk setiap *project*, terdapat fungsi link yang bertugas untuk menjalankan perintah hapus *project* yang dipilih *user*. Pada sisi kanan atas tampilan juga terdapat tombol “Logout” yang memiliki fungsi untuk *user* dapat keluar dari hak akses menggunakan sistem.



Gambar 5.12 perancangan antar muka daftar *project member*

5.1.4.6 Perancangan antarmuka Daftar Project (Administrator)

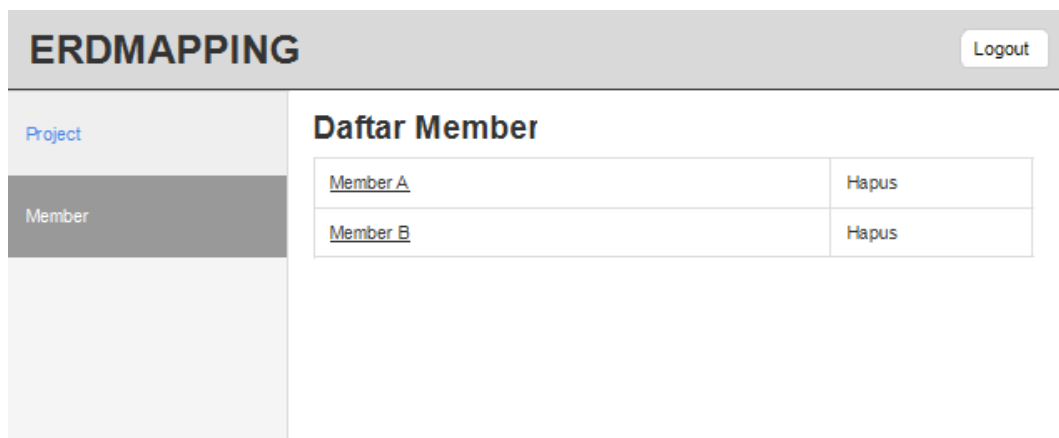
Untuk tampilan perancangan antarmuka “Daftar *Project*” untuk *user* dengan hak akses “administrator” dapat dilihat di Gambar 5.13, di mana pada tab bagian kiri tampilan antarmuka terdapat tab *Project* dan tab *Member*. Secara default, tampilan ini akan menampilkan isi dari tab *Project*. Pada tab *Project* ini terdapat tabel Daftar *Project* yang terdapat 2 kolom, yaitu Nama *Project* dan Nama *Member* yang berpartisipasi dalam pengerjaan *Project*. Sedangkan pada tab *Member*, dapat dilihat pada perancangan antarmuka yang bisa dilihat di Gambar 5.14.



Gambar 5.13 perancangan antar muka Daftar *project*

5.1.4.7 Perancangan antarmuka Daftar Member

Pada tampilan perancangan antarmuka “Daftar *Member*” yang ditampilkan pada Gambar 5.14 untuk *user* dengan hak akses “Administrator” terdapat pada tabel Daftar *Member* yang terdiri dari 2 kolom, yaitu Nama Mamber dan Hapus, di mana pada kolom Hapus terdapat link “Hapus” yang berfungsi untuk menghapus *member* yang dipilih.



Gambar 5.14 Perancangan Daftar Member

5.1.4.8 Perancangan antarmuka Buat Project Baru

Perancangan antarmuka yang ditampilkan pada Gambar 5.15 merupakan perancangan antarmuka *form project* baru untuk menambahkan *project* baru pada sistem. Terdapat kolom isian berupa "Nama Project" untuk tempat *user* mengisikan nama *project* yang diinginkan, serta tombol "Buat project baru" untuk mengeksekusi *form* tersebut.



Gambar 5.15 perancangan buat *project* baru

5.2 Implementasi

Pada tahap implementasi, akan dilakukan pembuatan perangkat lunak sesuai dengan hasil kebutuhan analisis, dan perancangan. Setiap Kebutuhan sistem akan menjadi fungsi yang ada pada setiap antarmuka sistem. Sistem akan dibuat berdasarkan kelas-kelas yang sudah dirancangan pada tahap perancangan sesuai dengan algoritma. Pada tahap ini, akan menghasilkan spesifikasi sistem untuk implementasi, batasan implementasi, implementasi kelas-kelas yang

dihasilkan, implementasi kode program, implementasi basis data, dan juga hasil implementasi antarmuka.

5.2.1 Spesifikasi Sistem

Dalam spesifikasi sistem akan dijelaskan mengenai spesifikasi dari perangkat lunak dan juga perangkat keras yang digunakan dalam pembuatan sistem.

5.2.1.1 Spesifikasi Perangkat Keras

Pada Tabel 5.16 berikut dijelaskan mengenai komponen perangkat keras apa yang digunakan pada pengimplementasian sistem.

Tabel 5.16 Tabel Spesifikasi perangkat keras

Nama Komponen	Spesifikasi
<i>System Model</i>	<i>Acer Aspire E5-471G</i>
<i>Processor</i>	<i>Intel Core i5 @1.70GHz</i>
<i>Memory</i>	<i>4GB</i>
<i>Display</i>	<i>Intel(R) HD Graphics Family</i>
<i>Hardisk size</i>	<i>465,76 GB</i>

5.2.1.2 Spesifikasi Perangkat Lunak

Pada Tabel 5.17 berikut dijelaskan mengenai komponen perangkat lunak apa saja yang digunakan pada saat implementasi sistem.

Tabel 5.17 Tabel Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi
<i>Operating System</i>	<i>Microsoft Windows 10 Pro 64-bit</i>
<i>Programming Language</i>	<i>PHP, Javascript</i>
<i>Programming Environment</i>	<i>Notepad++, Brackets, XAMPP v3.2.2</i>
<i>Database Management System</i>	<i>MySQL</i>

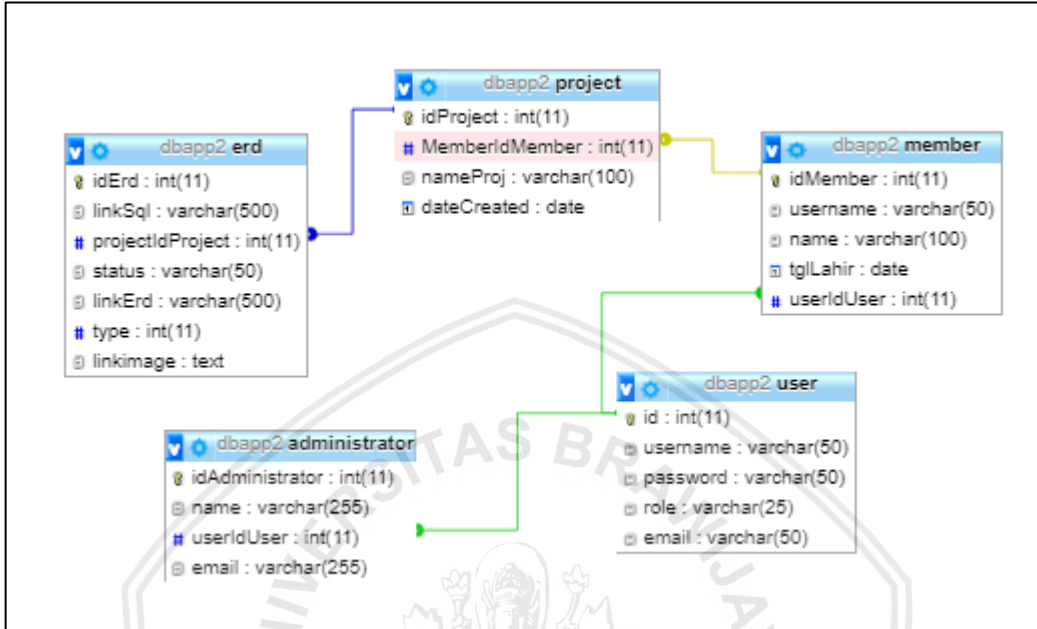
5.2.2 Batasan Implementasi

Terdapat batasan-batasan pada proses implementasi Sistem Pemetaan Otomatis Entity Diagram ke dalam Database, yaitu sebagai berikut :

1. Aplikasi dibangun berbasis *web*, menggunakan bahasa pemrograman PHP, *Javascript*.
2. Pembuatan Antarmuka aplikasi menggunakan bahasa pemrograman HTML, dan CSS.
3. *Database* dibangun menggunakan MySQL

- Pengembangan Sistem mapping dibuat dengan batasan 7 tahap mapping *Entity Relationship Diagram* menjadi Sql.

5.2.3 Implementasi Basis Data



Gambar 5.16 Hasil implementasi Basis data pada phpmyadmin

Setelah dilakukan proses perancangan basis data, kemudian mengimplementasikan sistem, hingga menghasilkan implementasi basis data seperti pada Gambar 5.16. Dapat diketahui bahwa sistem menghasilkan basis data yang terdiri dari lima buah tabel. Di mana masing-masing tabel saling terhubung atau berelasi.

5.2.4 Implementasi Kelas

Dalam implementasi kelas, akan dijabarkan mengenai kelas-kelas apa saja yang ada dalam sistem. Pada tabel 5.18 juga dijelaskan mengenai tipe kelas yang ada apakah berfungsi sebagai *controller*, *entitas*, atau *boundary*.

Tabel 5.18 implementasi kelas

No	Tipe Kelas	Nama Kelas	Nama File
1.	<i>Controller</i>	Admin	Admin.php
2.	<i>Controller</i>	<i>Member</i>	<i>Member.php</i>
3.	<i>Controller</i>	User	User.php
4.	<i>Controller</i>	Erd	Erd.php
5.	<i>Entity</i>	<i>Member_m</i>	<i>member_m.php</i>
6.	<i>Entity</i>	<i>Project_m</i>	<i>Project_m.php</i>

No	Tipe Kelas	Nama Kelas	Nama File
7.	<i>Entity</i>	Erd_m	erd_m.php
8.	<i>Entity</i>	login_m	Login_m.php
9.	<i>Boundary</i>	<i>A_member</i>	<i>A_member.php</i>
10.	<i>Boundary</i>	<i>A_project</i>	<i>A_project.php</i>
11.	<i>Boundary</i>	Erd	Erd.php
12.	<i>Boundary</i>	Home	Home.php
13.	<i>Boundary</i>	Home_about	Home_about.php
15.	<i>Boundary</i>	Login	Login.php
16.	<i>Boundary</i>	Lupa_pass	Lupa_pass.php
17.	<i>Boundary</i>	<i>M_detailProject</i>	<i>M_detailProject.php</i>
18.	<i>Boundary</i>	<i>M_detailProjectNew</i>	<i>M_detailProjectNew.php</i>
19.	<i>Boundary</i>	<i>M_detailProjectEmpty</i>	<i>M_detailProjectEmpty.php</i>
20.	<i>Boundary</i>	<i>M_Projects</i>	<i>M_Projects.php</i>
21.	<i>Boundary</i>	Registrasi	Registrasi.php
22.	<i>Boundary</i>	schema	Schema.php
23.	<i>Boundary</i>	Footer	Registrasi.php
24.	<i>Boundary</i>	Header	Registrasi.php

5.2.5 Implementasi Kode Program

Dalam implementasi Kode program akan ditampilkan *sample* dari empat method fungsi utama. Pada tabel 5.19 adalah potongan kode program mapping dari kelas erd. Pada fungsi ini nantinya akan berguna untuk memetakan atau mapping file yang sudah dibuat. Pada awalnya akan mengambil link file erdplus yang berbentuk json, kemudian melakukan overwrite kedalam file sql baru dengan nama yang sama. Setelah itu melakukan `json_decode()` yang nantinya mendapatkan hasil array untuk melakukan proses mapping. Pada program dilakukan 7 tahap mapping, setelah itu akan menyimpan link file sql kedalam database.

Tabel 5.19 Potongan kode program mapping upload

no	Source code
1	<code>function mappingUpload(\$id,\$idp){</code>
2	<code> \$data['erdp'] = \$this->erd_m->getErdProj(\$idp);</code>
3	<code> \$FILE = \$data['erdp']->linkErd;</code>
4	<code> \$filename = explode("/", \$FILE);</code>
5	<code> \$file_name = end(\$filename);</code>
6	<code> \$filejson= file_get_contents(\$FILE) or die("Error: Cannot create object");</code>
7	<code> \$namefile[0] = "ujicoba";</code>
8	<code> \$new_path = FCPATH.'assets/download/'.\$namefile[0].'.sql';</code>
9	
10	<code> \$json = json_decode(\$filejson);</code>
11	<code> \$txt = "DROP DATABASE IF EXISTS ".\$namefile[0]."; CREATE DATABASE ".\$namefile[0].";";</code>
12	<code> if (write_file(\$new_path, \$txt)){</code>
13	<code> \$this->transformregular(\$json,\$new_path);</code>
14	<code> \$this->transformweak(\$json,\$new_path);</code>
15	<code> \$this->transformrelationsonetoone(\$json,\$new_path);</code>
16	<code> \$this->transformrelationsonetomany(\$json,\$new_path);</code>
17	<code> \$this->transformrelationsmanytomany(\$json,\$new_path);</code>
18	<code> \$this->transformmultivalued(\$json,\$new_path);</code>
19	<code> \$this-> transformmanyary(\$json, \$new_path);</code>
20	<code> if(\$this->erd_m->addsqltodb(\$idp,\$new_path));</code>
21	<code> \$this->session->set_flashdata('msg','SUKSES');</code>
22	<code> redirect('Member/detailProject/'.\$idp);</code>
23	<code> // end of if write awal</code>
24	<code> }</code>
25	<code> else {</code>
26	<code> //jika gagal</code>
27	<code> \$this->session->set_flashdata('msg','unable to write file');</code>
28	<code> redirect('Member/detailProjectEmpty/'.\$idp);</code>
29	<code> }</code>
30	
31	<code> }</code>
32	
33	

Pada Tabel 5.20 ini adalah potongan kode program `transformregular` dari kelas erd. Pada fungsi ini nantinya akan berguna untuk mentransform erd yang nantinya akan dimapping, dengan melewati tahap 1, yaitu tahap merubah entitiity yang berjenis regular atau strong entity. Disini akan mengambil data array json yang sudah didapatkan dari proses sebelumnya untuk diolah. Pada baris 4 akan dicari bentuk entitas regular dari array. Kemudian membuat table baru dengan

data tersebut. Kemudian pada baris 20, akan mencari atribut dari bentuk yang sudah ditemukan. Setelah itu memeriksa tipe datanya. Setiap baris sql query yang dipanggil langsung disimpan kedalam file sql yang ada dengan `wite_file()`.

Tabel 5.20 Potongan kode program method transform regular

no	Source code
1	public function transformregular(\$json,\$new_path){
2	
3	//#1 table - entity regular + attr
4	foreach(\$json->shapes as \$shape){
5	
6	if(\$shape->type == 'Entity' && \$shape->details->type == 'regular'){
7	//create table
8	\$txt = "USE UJICOBA; CREATE TABLE ";
9	write_file(\$new_path, \$txt,'a');
10	\$txt= strtoupper(\$shape->details->name);
11	write_file(\$new_path, \$txt,'a');
12	\$txt="";
13	write_file(\$new_path, \$txt,'a');
14	\$atts= array();
15	//reg-entity atribut
16	foreach(\$json->connectors as \$cons){
17	if(\$cons->type == 'Connector' && \$cons->destination == \$shape->details->id){
18	//foreach untuk cari atribut
19	foreach(\$json->shapes as \$att){
20	if(\$att->type == 'Attribute' && \$att->details->id == \$cons->source){
21	//cek otomatis data type
22	if(strpos(\$att->details->name,"id")!= False strpos(\$att->details-
23	>name,"no")!= False){
24	\$type = " INT(11) ";
25	}
26	else if(strpos(\$att->details->name,"date")!= False){
27	\$type = " DATE";
28	}
29	else {
30	\$type = " VARCHAR(255)";
31	}
32	//cek unik
33	if (\$att->details->isUnique == 1){
34	\$atts = strtoupper(\$att->details->name).\$type.' NOT NULL PRIMARY KEY';
35	\$attss[]=\$atts;
36	
37	\$PKreg[] = strtoupper(\$att->details->name);
38	}
39	else{
40	\$atts = strtoupper(\$att->details->name).\$type;
41	\$attss[]=\$atts;
42	}
43	
44	}
45	}
46	}
47	}
48	\$txt=implode(",",\$attss);
49	write_file(\$new_path, \$txt,'a');
50	\$txt="";
51	write_file(\$new_path, \$txt,'a');
52	unset(\$attss);
53	

no	Source code
54	}
55	}
56	//END #1
57	return true;
58	}

Pada Tabel 5.21 ini adalah potongan kode program transform relations one to one dari kelas erd. Pada fungsi ini nantinya akan berguna untuk mentransform erd yang nantinya akan dimapping apabila memiliki relasi satu ke satu, dengan melewati tahap ke 3, yaitu tahap merubah entitiy yang memiliki relasi satu ke satu. . Disini akan mengambil data array json yang sudah didapatkan dari proses sebelumnya untuk diolah. Pada baris 3-7 akan dicari relasi dari tiap entitas. Kemudian memanggil 'ALTER TABLE' dengan data tersebut. Kemudian pada baris 38, akan mencari atribut dari bentuk yang sudah ditemukan. Setelah itu memeriksa tipe datanya. Karena membuat kolom baru, maka alter table akan digunakan dua kali pertama untuk membuat kolom baru, kemudian kedua untuk merubah kolom tersebut menjadi foreign key yang terhubung ke table lain. Setiap baris sql query yang dipanggil langsung disimpan kedalam file sql yang ada dengan wite_file().

Tabel 5.21 Potongan kode program method transform relation one to one

no	Source code
1	function transformrelationsonetoone(\$json,\$new_path){
2	//#3 Relation 1-1
3	foreach(\$json->shapes as \$shape){
4	
5	\$card = array();
6	\$sidecon = array();
7	if(\$shape->type == 'Relationship'){
8	foreach(\$shape->details->slots as \$side){
9	\$card[] = \$side->cardinality;
10	\$sidecon[] = \$side->entityId;
11	}
12	
13	if(in_array("many",\$card,TRUE)){
14	continue;
15	}
16	else{
17	
18	//create table 1-1
19	foreach(\$json->shapes as \$shape){
20	foreach(\$sidecon as \$id){
21	if(\$shape->details->id == \$id){
22	
23	//cari first
24	if(\$sidecon[1]==\$id){
25	\$txt = " ALTER TABLE ";
26	write_file(\$new_path, \$txt,'a');
27	\$txt= strtoupper(\$shape->details->name);
28	\$varacak[]=strtoupper(\$shape->details->name);
29	write_file(\$new_path, \$txt,'a');
30	}
31	}

no	Source code
32	foreach(\$json->shapes as \$sh){
33	if(\$sh->details->id == \$idcon[0]){
34	foreach(\$json->connectors as \$cons){
35	if(\$cons->type == 'Connector' && \$cons->destination == \$sh->details->id){
36	
37	//foreach untuk cari atribut
38	
39	foreach(\$json->shapes as \$att){
40	if(\$att->type == 'Attribute' && \$att->details->id == \$cons->source){
41	//cek otomatis data type
42	if(strpos(\$att->details->name,"id")!= False strpos(\$att->details-
43	>name,"no")!= False){
44	\$type = " INT(11) ";
45	}
46	else if(strpos(\$att->details->name,"date")!= False){
47	\$type = " DATE";
48	}
49	else {
50	\$type = " VARCHAR(255)";
51	}
52	
53	//cek unik TEST
54	if (\$att->details->isUnique == 1){
55	
56	foreach (\$json->shapes as \$slast){
57	if(\$slast->details->id == \$cons->destination){
58	\$atts = " ADD ``.strtoupper(\$att->details->name).``.\$type;
59	\$atts2 = " ADD FOREIGN KEY (``.strtoupper(\$att->details->name).``)
60	REFERENCES ``.strtoupper(\$slast->details->name).`` (``.strtoupper(\$att->details->name).``);";
61	\$attss[]=\$atts;
62	\$attss2[]=\$atts2;
63	\$PKreg[] = strtoupper(\$att->details->name);
64	}}
65	}
66	}
67	}
68	}
69	}
70	}
71	}
72	
73	
74	}
75	}
76	}
77	}
78	//end create table
79	\$txt=implode(",",\$attss);
79	write_file(\$new_path, \$txt,'a');
80	\$txt=" ";
81	write_file(\$new_path, \$txt,'a');
82	\$txt="ALTER TABLE ".\$varacak[0];
83	write_file(\$new_path, \$txt,'a');
84	\$txt=implode(",",\$attss2);
85	write_file(\$new_path, \$txt,'a');
86	
87	unset(\$attss);
88	unset(\$attss2);
89	unset(\$varacak);
90	}

no	Source code
91	
92	//end
93	unset(\$card);
94	unset(\$idcon);
95	}
96	}
97	//END #3
98	return true;
99	}

Pada tabel 5.22 ini adalah potongan kode program transform multivalued attriibut dari kelas erd. Pada fungsi ini nantinya akan berguna untuk mentransform erd yang nantinya akan dimapping, dengan melewati tahap ke 6, yaitu tahap merubah attribut yang memiliki nilai multivalued, untuk dijadikan entity yang berdiri sendiri.

Tabel 5.22 Potongan kode program method transform multivalued attribute

no	Source code
1	
2	function transformmultivalued(\$json,\$new_path){
3	
4	//#6 Multivalued Attribute
5	
6	foreach(\$json->shapes as \$shape){
7	if(\$shape->type == 'Attribute' && \$shape->details->isMultivalued == '1'){
8	\$Attribute_name=strtoupper(\$shape->details->name);
9	
10	//create table
11	\$txt = " CREATE TABLE ";
12	write_file(\$new_path, \$txt, 'a');
13	
14	
15	\$sattss= array();
16	\$satts= array();
17	//reg-entity atribut
18	foreach(\$json->connectors as \$cons){
19	if(\$cons->type == 'Connector' && \$cons->source == \$shape->details->id){
20	
21	//foreach untuk cari entity
22	foreach(\$json->shapes as \$sent){
23	
24	if(\$sent->type =='Entity' && \$sent->details->id == \$cons->destination){
25	//cek otomatis data type
26	
27	
28	if(strpos(\$sent->details->name,"id")!= False strpos(\$sent->details->
29	name,"no")!= False){
30	\$type = " INT";
31	}
32	else if(strpos(\$sent->details->name,"date")!= False){
33	\$type = " DATE";
34	}
35	else {
36	\$type = " VARCHAR(255)";
37	}

no	Source code
38	
39	\$txt= strtoupper(\$ent->details->name)."_" .strtoupper(\$shape->details->name);
40	write_file(\$new_path, \$txt,'a');
41	\$txt="";
42	write_file(\$new_path, \$txt,'a');
43	
44	//PK dari entitynya
45	foreach(\$json->connectors as \$cons){
46	if(\$cons->type == 'Connector' && \$cons->destination == \$ent->details->id){
47	//foreach untuk cari atribut
48	foreach(\$json->shapes as \$att){
49	if(\$att->type == 'Attribute' && \$att->details->id == \$cons->source){
50	//cek otomatis data type
51	if(strpos(\$att->details->name,"id")!= False strpos(\$att->details-
52	>name,"no")!= False){
53	\$type = " INT(11) ";
54	}
55	else if(strpos(\$att->details->name,"date")!= False){
56	\$type = " DATE";
57	}
58	else {
59	\$type = " VARCHAR(255)";
60	}
61	
62	//cek PK
63	if (\$att->details->isUnique == 1){
64	\$txt = strtoupper(\$att->details->name).\$type.' NOT NULL FOREIGN KEY';
65	write_file(\$new_path, \$txt,'a');
66	
67	}
68	}
69	}
70	}
71	}
72	
73	//+ nama atributnya
74	\$txt=",";
75	write_file(\$new_path, \$txt,'a');
76	\$txt= strtoupper(\$shape->details->name).\$type.'NOT NULL UNIQUE KEY';
77	write_file(\$new_path, \$txt,'a');
78	//Attribute ASAL
79	
80	//END ATT ASAL
81	
82	
83	//cek unik
84	if (\$att->details->isUnique == 1){
85	\$atts = strtoupper(\$att->details->name).\$type.' NOT NULL UNIQUE';
86	\$attss[]=\$atts;
87	}
88	}
89	}
90	}
91	}
92	\$txt=implode(",",\$attss);
93	write_file(\$new_path, \$txt,'a');
94	
95	
96	\$txt=");";
97	write_file(\$new_path, \$txt,'a');

no	Source code
98	unset(\$atts);
99	}
100	}
101	//END #6
102	}

Pada tabel 5.23 adalah potongan kode untuk menyimpan erd baru dari kelas erd (controller) ke dalam database melalui model erd_m. Setiap data yang dibutuhkan untuk disimpan disimpan ke dalam array \$data. Kemudian akan memanggil model erd_m() yang akan memproses database.

Tabel 5.23 Potongan kode program Adderd

no	Source code
1	function addErd(\$id, \$data){
2	\$data = array(
3	'idproject' => \$id,
4	'full_path'=> \$data,
5	'type'=>'0',
6);
7	\$this->erd_m->add(\$data);
8	print_r(\$data);
9	}

Pada tabel 5.24 adalah potongan kode program downloadsql dari kelas Member (controller). Pada fungsi ini nantinya akan berguna untuk mengunduh file sql yang telah dipetakan dan tersimpan dalam database.

Tabel 5.24 Potongan kode program download sql

no	Source code
1	public function downloadsql(){
2	//download helper
3	\$this->load->helper('download');
4	\$idp = \$this->session->userdata('project_idp');
5	\$idr = \$this->session->userdata('project_iderd');
6	//file path
7	\$data['sqlink'] = \$this->erd_m->getErdProj(\$idp);
8	\$filesql = file_get_contents(\$data['sqlink']->linkSql);
9	//download file from directory
10	\$filename = explode("/", \$data['sqlink']->linkSql);
11	\$file_name = end(\$filename);
12	force_download(\$file_name, \$filesql);
13	}

5.2.6 Implementasi Antarmuka

Dalam implementasi antarmuka, akan ditampilkan dan dijelaskan mengenai tampilan antarmuka dari sistem yang sudah diimplementasikan. Tampilan yang dibuat adalah hasil implementasi dari hasil perancangan yang dijelaskan pada sub-bab perancangan antarmuka.

5.2.6.1 Implementasi Antarmuka Home

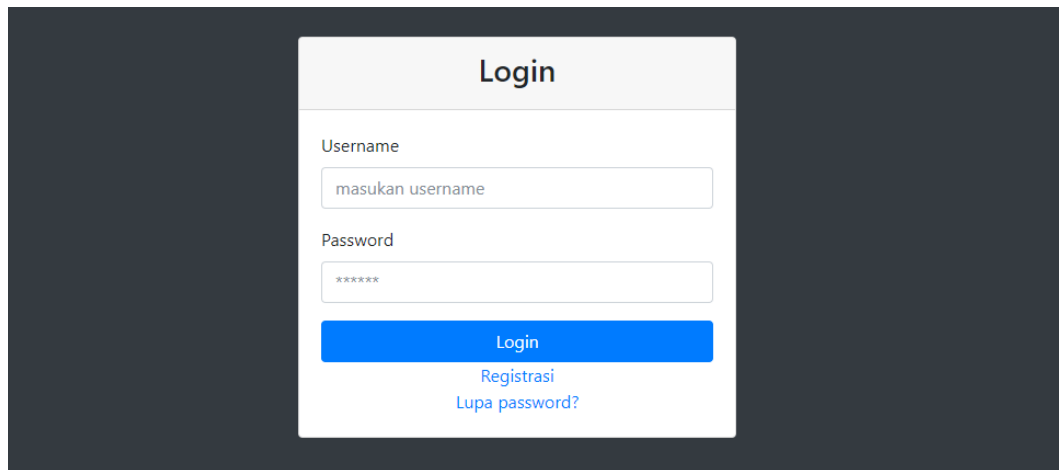
Pada antarmuka sistem yang ditampilkan pada Gambar 5.17, merupakan tampilan halaman utama dari sistem di mana terdapat 2 tombol yaitu tombol "Login" dan tombol "Sign up". Tombol "Login" berfungsi untuk mengantarkan *user* ke tampilan halaman/*form* Login agar *user* dapat akunnya. Sedangkan tombol "Sign up" berfungsi untuk mengantarkan *user* ke halaman/*form* "Registrasi" di mana *user* dapat melakukan registrasi untuk membuat akun baru pada sistem.



Gambar 5.17 Tampilan antarmuka home

5.2.6.2 Implementasi Antarmuka login

Pada antarmuka sistem yang ditunjukkan pada Gambar 5.18 merupakan halaman Login, yang terdapat *form* Login yang terdiri dari kolom isian "Username" dan "Password". Pada kolom isian "Username", *user* dapat mengisinya *username* yang sudah dibuat *user* serta kata sandi akun tersebut dapat diisikan pada kolom isian "Password". Pada antarmuka Login in terdapat sebuah tombol/tombol yang bertuliskan "sign in" untuk mengeksekusi proses login. Di bawah tombol Login terdapat 2 tautan, yaitu tautan "Registrasi" yang berfungsi untuk mengantarkan *user* ke halaman *form* Registrasi dan tautan "Lupa Password" untuk mengantarkan *user* ke halaman *form* Lupa Password.

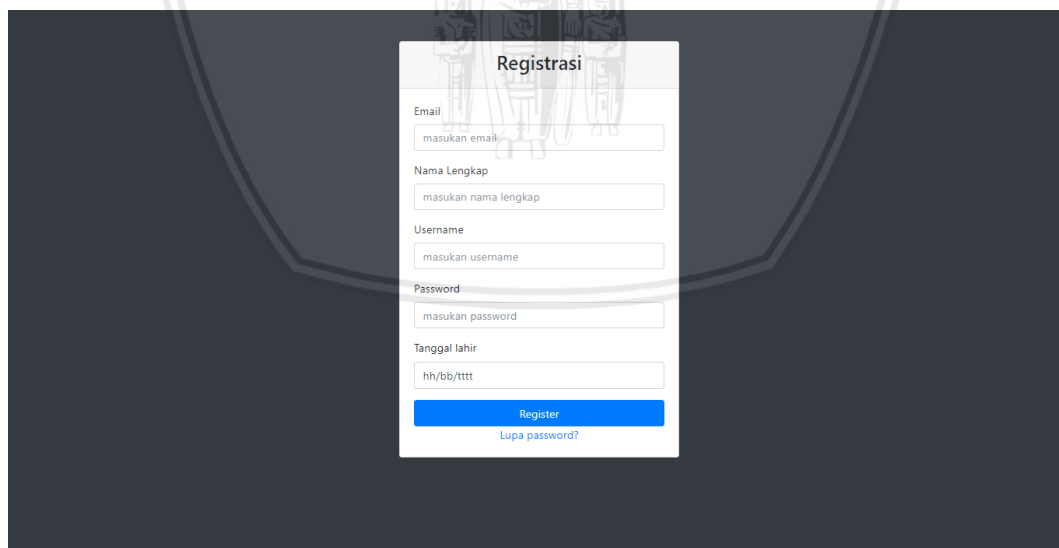


The image shows a login form with a white background and a dark grey border. At the top, the word "Login" is centered in a bold, black font. Below this, there are two input fields: "Username" with a placeholder "masukan username" and "Password" with a placeholder "*****". A blue button labeled "Login" is positioned below the password field. Underneath the button, there are two links: "Registrasi" and "Lupa password?", both in a smaller, blue font.

Gambar 5.18 Tampilan antarmuka login

5.2.6.3 Implementasi Antarmuka Registrasi

Pada antarmuka Registrasi yang ditampilkan pada Gambar 5.19, *form* isian yang terdiri dari kolom isian untuk *email*, nama, *username*, *password* dan tanggal lahir. Pada *form* registrasi ini terdapat sebuah tombol/tombol bertuliskan "Register" yang akan mengeksekusi proses registrasi setelah *user* mengisi semua kolom yang ada pada *form*. Di bawah tombol Register terdapat juga tautan "Lupa Password" yang berfungsi untuk mengantarkan *user* ke halaman *form* Lupa Password.



The image shows a registration form with a white background and a dark grey border. At the top, the word "Registrasi" is centered in a bold, black font. Below this, there are five input fields: "Email" with a placeholder "masukan email", "Nama Lengkap" with a placeholder "masukan nama lengkap", "Username" with a placeholder "masukan username", "Password" with a placeholder "masukan password", and "Tanggal lahir" with a placeholder "hh/bb/tttt". A blue button labeled "Register" is positioned below the password field. Underneath the button, there is a link "Lupa password?" in a smaller, blue font.

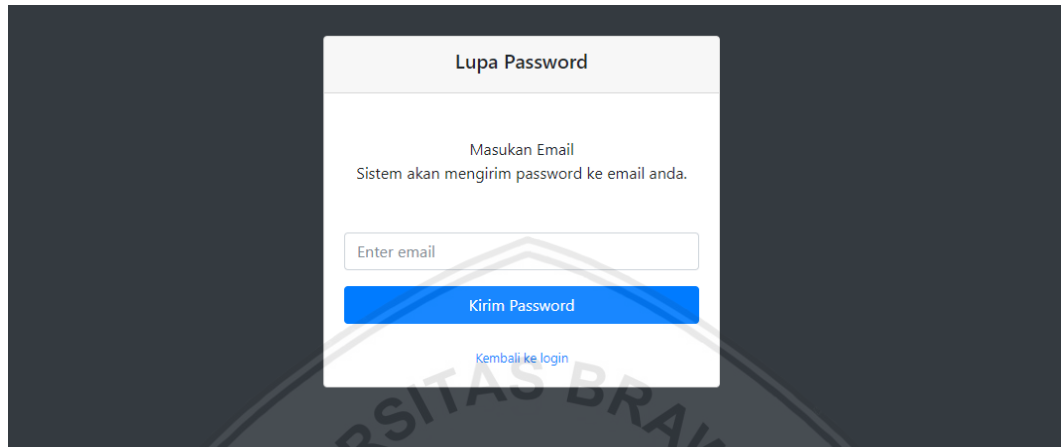
Gambar 5.19 Tampilan antarmuka registrasi

5.2.6.4 Implementasi Antarmuka Lupa password

Pada tampilan antarmuka yang ditunjukkan pada Gambar 5.20, terdapat *form* isian Lupa *password* yang terdapat kolom isian *email* di mana *user* akan

repository.ub.ac.id

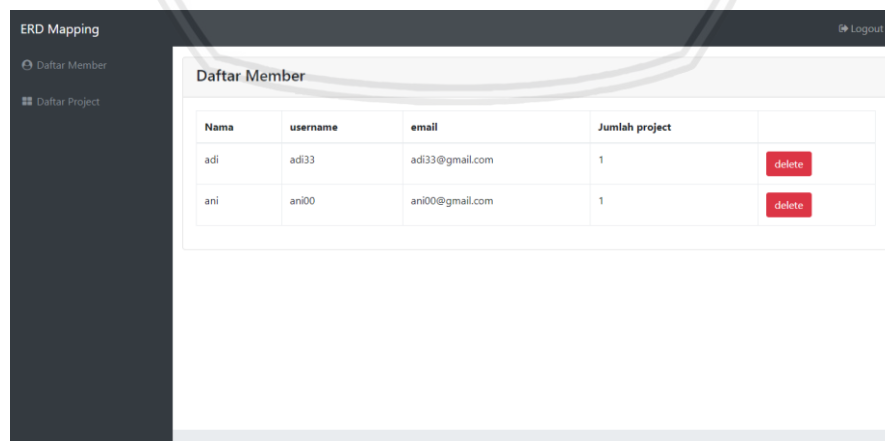
mengisinya dengan alamat *email* yang terdaftar pada sistem. Form ini menyediakan tombol “Kirim Password” yang memiliki fungsi untuk mengeksekusi *email* yang diisi oleh *user* agar dapat diproses pada sistem. Di bawah tombol “Kirim Password” terdapat tautan “Kembali ke Login” yang berfungsi untuk mengantar *user* kembali ke halaman Login.



Gambar 5.20 Tampilan antarmuka lupa *password*

5.2.6.5 Implementasi Antarmuka Daftar Member oleh Administrator

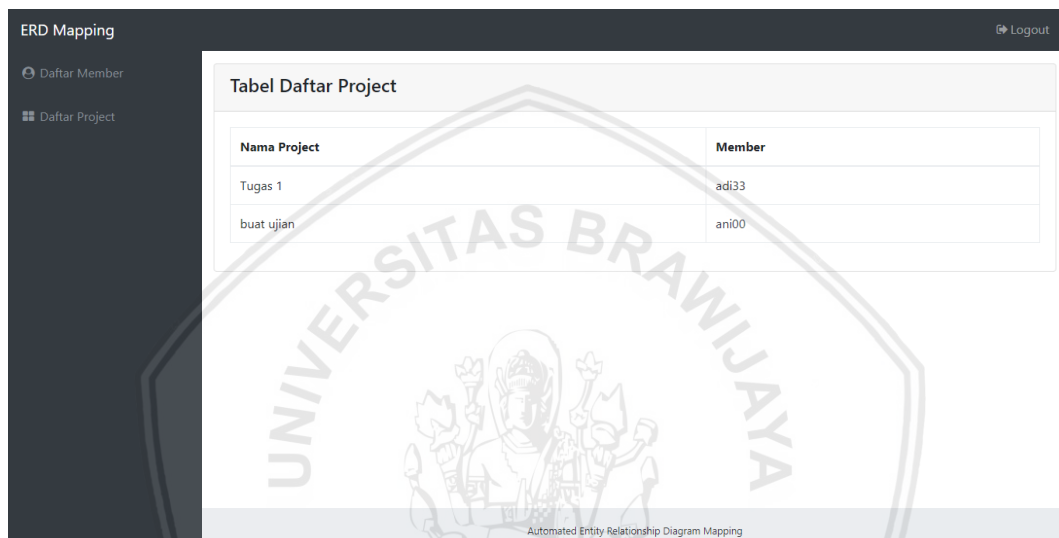
Pada tampilan antarmuka sistem yang ditunjukkan pada Gambar 5.21, merupakan tampilan halaman Daftar *Member* untuk akun yang memiliki hak akses sebagai *Administrator* ketika memilih opsi tab Daftar *Member*. Pada sisi kiri halaman ini terdapat tab terdapat tab *Project* dan tab *Member*. Pada halaman ini terdapat pada tabel Daftar *Member* yang terdiri dari 2 kolom, yaitu Nama Mamber dan Hapus, di mana pada kolom Hapus terdapat link “Hapus” yang berfungsi untuk menghapus *member* yang dipilih.



Gambar 5.21 Tampilan antarmuka daftar *member*

5.2.6.6 Implementasi Antarmuka Daftar Project oleh Administrator

Pada tampilan antarmuka sistem yang ditunjukkan pada Gambar 5.22 ini, merupakan tampilan halaman Daftar *Project* untuk akun yang memiliki hak akses sebagai *Administrator* di mana pada tab bagian kiri tampilan antarmuka terdapat tab *Project* dan tab *Member*. Secara default, tampilan halaman ini akan disajikan sistem setelah proses Login oleh akun *Administrator* tersebut. Pada tab *Project* ini terdapat tabel Daftar *Project* yang terdapat 2 kolom, yaitu Nama *Project* dan Nama *Member* yang berpartisipasi dalam pengerjaan *Project*. Sedangkan pada tab *Member*, dapat dilihat pada perancangan antarmuka yang bisa dilihat di Gambar 5.25.



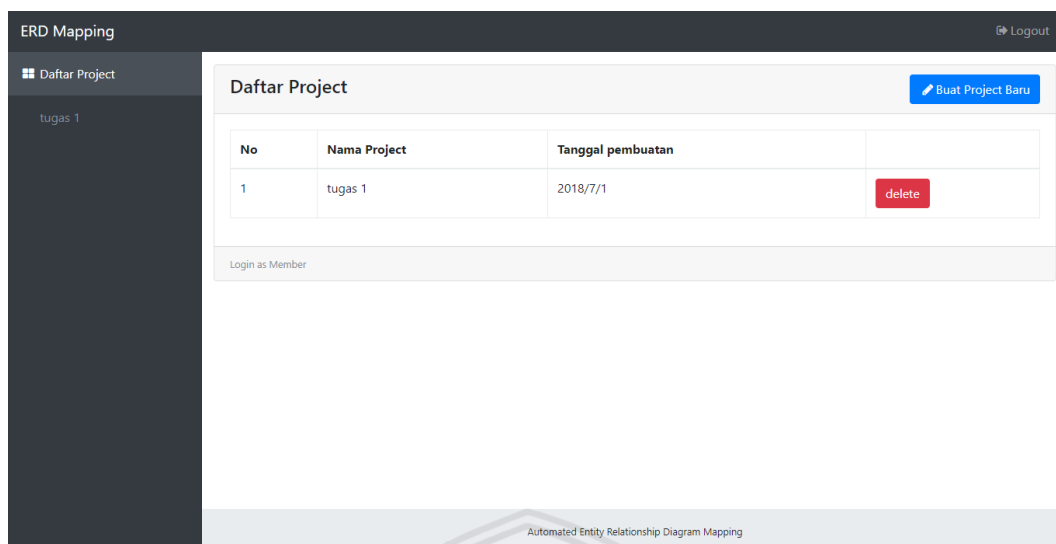
The screenshot shows a web application interface for 'ERD Mapping'. On the left is a dark sidebar with navigation options: 'Daftar Member' and 'Daftar Project'. The main content area displays a table titled 'Tabel Daftar Project'. The table has two columns: 'Nama Project' and 'Member'. It contains two rows of data. At the bottom of the interface, there is a watermark for 'UNIVERSITAS BRAWIJAYA' and the text 'Automated Entity Relationship Diagram Mapping'.

Nama Project	Member
Tugas 1	adi33
buat ujian	ani00

Gambar 5.22 Tampilan antarmuka daftar *project* oleh administrator

5.2.6.7 Implementasi Antarmuka Daftar Project oleh Member

Pada tampilan antarmuka yang ditunjukkan pada Gambar 5.23 ini merupakan tampilan halaman Daftar *Project* untuk akun yang memiliki hak akses sebagai *Member*, di mana pada tab bagian sisi kiri tampilan antarmuka terdapat tab *Project* yang juga menampilkan sub-tab berupa *Project* apa saja yang sudah dibuat. Pada tampilan ini, terdapat tombol “Buat *Project* Baru” yang berfungsi untuk mengeksekusi perintah untuk menampilkan *form* “Buat *Project* Baru”. Tampilan antarmuka ini juga menampilkan tabel “Daftar *Project*” yang menampilkan *project-project* yang sedang dikerjakan oleh *user*. Untuk setiap *project*, terdapat tombol “Delete” yang bertugas untuk menjalankan perintah hapus *project* yang dipilih *user*. Pada sisi kanan atas tampilan juga terdapat tombol “Logout” yang memiliki fungsi untuk *user* dapat keluar dari hak akses menggunakan sistem.



Gambar 5.23 Tampilan antarmuka daftar *project* oleh *member*

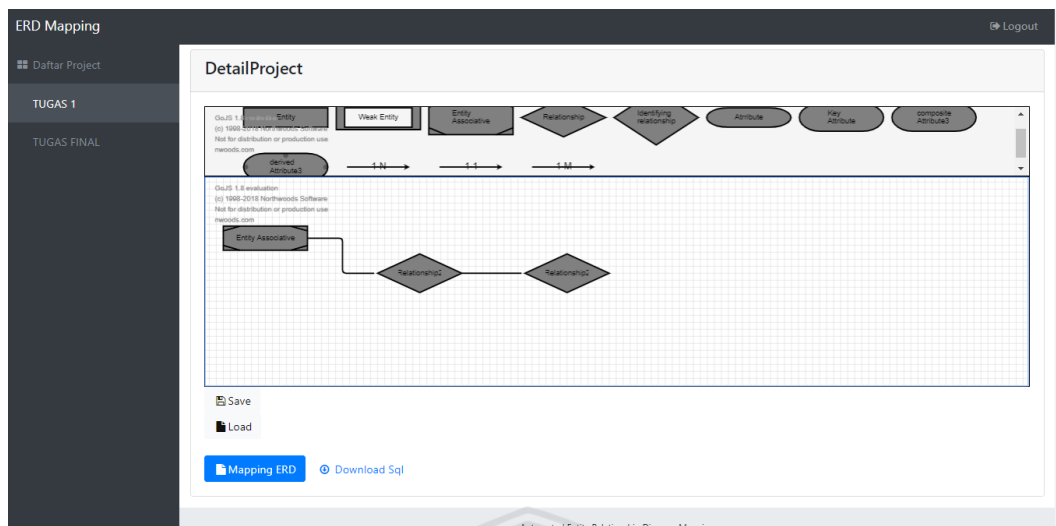
5.2.6.8 Implementasi Antarmuka Detail Project (jika sudah membuat ERD)

Pada tampilan antarmuka sistem “Detail *Project*” di Gambar 5.24 dan 5.25 ini menjelaskan bahwa terdapat beberapa tab pada sebelah kiri tampilan antarmuka yang akan menampilkan beberapa pilihan *project* yang telah dibuka oleh *user*, sehingga *user* dapat memilih *project* mana yang akan dikerjakan. Hasil keluaran apabila *user* sudah memilih tab nama *project* yang diinginkan akan berbeda tergantung pada jenis erd yang ada.

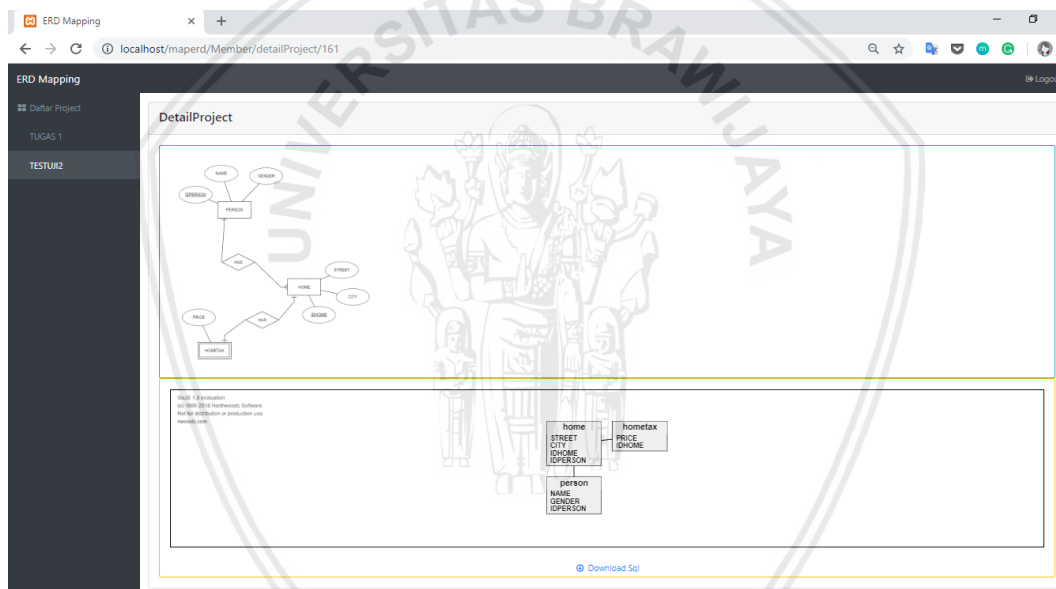
Pada 5.24 merupakan hasil keluaran apabila erd dibuat dengan diagram. antarmuka akan menampilkan 2 tombol, yaitu tombol “Save” dan tombol “Load” dan juga workspace. Fungsi tombol “Save” adalah untuk mengeksekusi perintah untuk menyimpan *project* yang sudah/sedang dikerjakan *user* ke dalam sistem. Sedangkan fungsi tombol “mapping” adalah untuk mengeksekusi perintah untuk melakukan mapping,

Pada 5.25 merupakan hasil keluaran apabila erd dibuat dengan cara upload file erdplus. Akan menampilkan gambar apabila menyertakan gambar, dan menampilkan table relasi yang didapatkan dari hasil eksekusi sql pada database. Keduanya ditampilkan untuk mengecek apakah sql sudah sesuai dengan yang diinginkan.

Pada sisi bawah terdapat tombol ‘download sql’ dan juga pada sisi kanan atas tampilan juga terdapat tombol “Logout” yang memiliki fungsi untuk *user* dapat keluar dari hak akses menggunakan sistem. Kedua fungsi ini sama-sama ditampilkan pada kedua hasil.



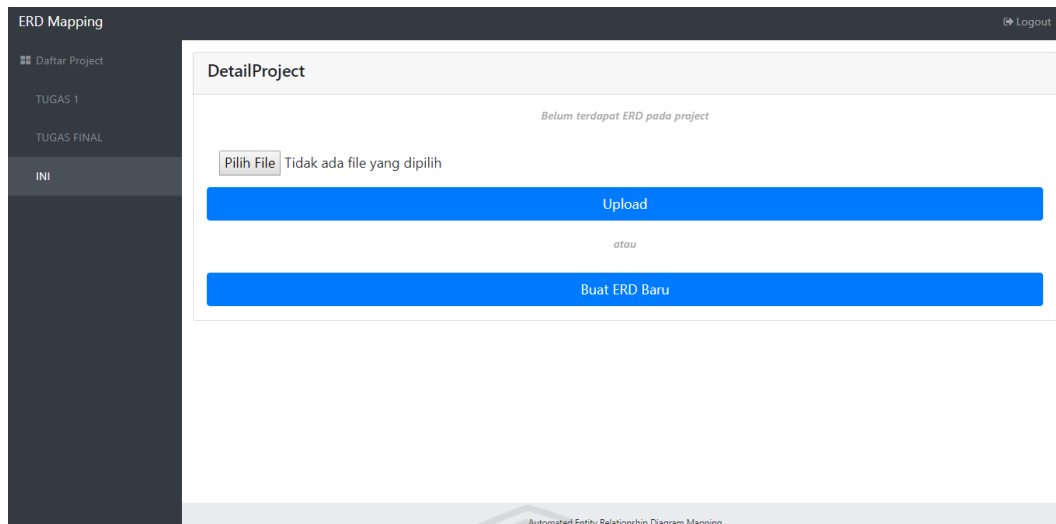
Gambar 5.24 Tampilan antarmuka detail *project* berisi erd yang dibuat



Gambar 5.25 Tampilan antarmuka detail *project* berisi erd yang diupload

5.2.6.9 Implementasi Antarmuka Detail Project (jika belum terdapat ERD)

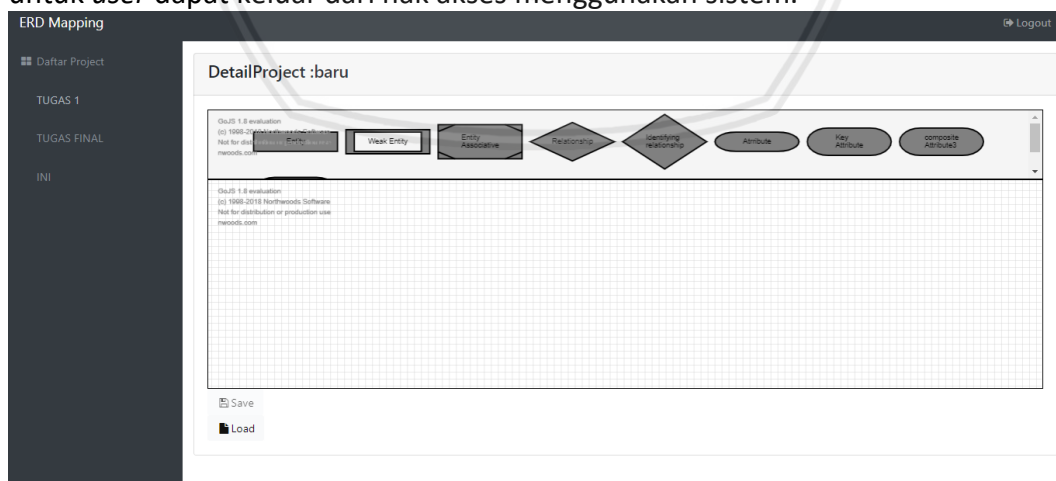
Pada tampilan antarmuka Detail *Project* yang ditunjukkan Gambar 5.26 ini merupakan tampilan halaman sistem disaat *user* sudah membuat *project* baru tetapi sistem mendeteksi bahwa *project* tersebut belum terdapat ERD yang dibuat. Pada halaman Detail *Project* ini akan menampilkan 2 tombol, yaitu tombol “Buat ERD Baru” dan tombol “Upload”. Sedangkan fungsi tombol “Upload” adalah untuk mengeksekusi perintah untuk menampilkan Form “Upload ERD”. Pada sisi kanan atas tampilan juga terdapat tombol “Logout” yang memiliki fungsi untuk *user* dapat keluar dari hak akses menggunakan sistem.



Gambar 5.26 Tampilan antarmuka detail *project* kosong

5.2.6.10 Implementasi Antarmuka Detail Project (buat ERD baru)

Pada tampilan antarmuka sistem “Detail *Project*” di Gambar 5.27 ini menjelaskan bahwa terdapat beberapa tab pada sebelah kiri tampilan antarmuka yang akan menampilkan beberapa pilihan *project* yang telah dibuka oleh *user*, sehingga *user* dapat memilih *project* mana yang akan dikerjakan. Ketika *user* sudah memilih tab nama *project* yang diinginkan, antarmuka akan menampilkan 2 tombol, yaitu tombol “Save” dan tombol “Load”. Fungsi tombol “Save” adalah untuk mengeksekusi perintah untuk menyimpan *project* yang sudah/sedang dikerjakan *user* ke dalam sistem sekaligus menyimpannya kedalam file json. tombol save hanya akan aktif ketika sudah melakukan perubahan pada workspace. Pada sisi kanan atas tampilan juga terdapat tombol “Logout” yang memiliki fungsi untuk *user* dapat keluar dari hak akses menggunakan sistem.



Gambar 5.27 Tampilan antarmuka detail *project* berisi erd

BAB 6 PENGUJIAN

Pada tahap pengujian, adalah tahap di mana hasil implementasi diperiksa atau diuji coba untuk menentukan apakah hasil implementasi sudah sesuai dengan hasil analisis kebutuhan serta perancangan atau belum. Pada tahap ini akan dilakukan pengujian unit untuk menguji tiap unit yang ada pada sistem, pengujian validasi untuk menguji apakah sistem sudah berjalan sesuai dengan apa yang dirancang, dan terakhir adalah pengujian non fungsional, yaitu pengujian *compatibility*.

6.1 Pengujian Unit

Pengujian unit adalah pengujian di mana akan menguji method-method yang ada dalam kelas yang sudah dihasilkan pada tahap perancangan. Pengujian unit adalah salah satu bagian dari pengujian *whitebox*, di mana akan menguji sistem dengan melihat keseluruhan algoritma sistem juga.

6.1.1 Pengujian Unit method *mappingupload* pada Kelas ERD

Pada Tabel 6.1 merupakan potongan algoritma method upload erd dari kelas controller erd. Method ini berfungsi untuk melakukan proses upload ke dalam database, yang nantinya akan langsung melakukan mapping secara otomatis.

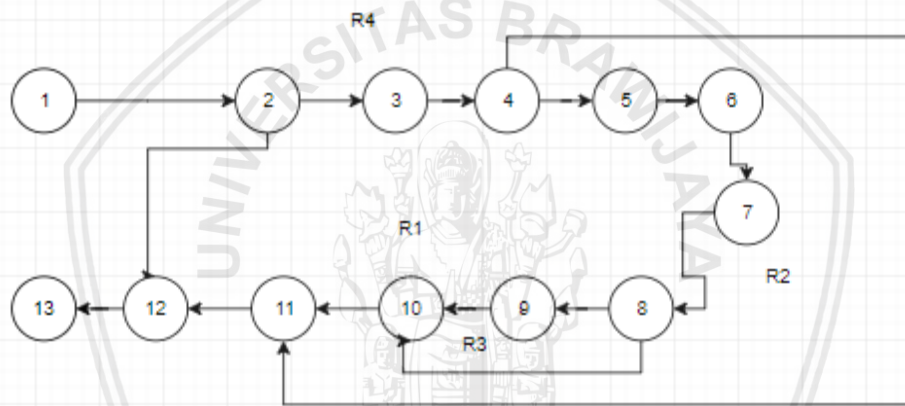
Tabel 6.1 Potongan algoritma method *mappingUpload*

NO	Source code/Algoritma
1	Start
2	<code>idProject = \$id, idp = \$idp</code>
3	<code>inisialisasi array \$data</code>
4	Inisialisasi <code>\$file, \$filename, \$pathjson,</code>
5	If (<code>File_get_contents jsonfile success</code>)
6	do <code>Json_decode save to filejson array</code>
7	if (<code>write new file sql</code>)
8	do
9	<code>create database</code>
10	<code>transformregular()</code>
11	<code>transformweak()</code>
12	<code>transformrelationonetoone()</code>
13	<code>transformreationonetomany()</code>
14	<code>transformrelationmanytomany()</code>
15	<code>transformmultivalued()</code>
16	<code>transformmanyary()</code>
17	if (<code>addsqltoDB()</code>)
18	<code>set_flashdata success</code>

NO	Source code/Algoritma
19	end if _____ 10
20	End if _____ 11
21	Else print error _____ 12
22	End if _____ 13
23	End

6.1.1.1 Flow graph

Dari algoritma yang ada pada operasi mapping *upload*, maka dibangunlah flow *graph* seperti pada Gambar 6.1 untuk menemukan cyclomatic complexity dari proses yang ada.



Gambar 6.1 Flowgraph operasi mapping Erdupload

6.1.1.2 Cyclomatic Complexity

$V(G) = \text{Jumlah Region} = 4$

$V(G) = E - N + 2 = 15 - 11 + 2 = 4$

$V(G) = P + 1 = 3 + 1 = 4$

6.1.1.3 Independent Path

Dari hasil penghitungan cyclomatic complexity, didapatkan bahwa proses memiliki tiga buah jalur, yaitu 1-2-3-4-5-6-7-8-9-10-11-12-13, 1-2-12-13, dan 1-2-3-4-11-12-13 dan 1-2-3-4-5-6-7-8-10-11-12-13. Pada Tabel 6.2 akan dijelaskan mengenai setiap prosedur uji, hasil yang diharapkan, hasil yang didapatkan, dan juga status apakah hasil sudah sesuai atau belum.



Tabel 6.2 Test Case algoritma proses mapping upload

No	Jalur	Data input	Expected Result	Result	Status
1.	1-2-3-4-5-6-7-8-9-10-11-12-13	Isi <i>file</i> ada, <i>database</i> yang bernama sama tidak ada.	Membuat <i>database</i> baru. Proses mapping berhasil.	<i>Database</i> baru dibuat. Proses mapping berhasil	Valid
2.	1-2-12-13	Isi <i>file</i> kosong.	Menampilkan pesan error	Menampilkan pesan error.	Valid
3.	1-2-3-4-11-12-13	<i>Gagal melakukan write_file</i>	Tidak melakukan apapun pada <i>database</i>	<i>Tidak menyimpan apapun pada database</i>	valid
4	1-2-3-4-5-6-7-8-10-11-12-13	<i>Jlka sqltoDB gagal</i>	Tidak menampilkan pesan sukses	Tidak menampilkan pesan sukses	valid

6.1.2 Pengujian Unit operasi mappingBaru Kelas erd

Pada Tabel 6.3 merupakan potongan algoritma method mapping dari kelas controller erd. Method ini berfungsi untuk melakukan proses pemetaan.

Tabel 6.3 Potongan source code dari method mappingBaru

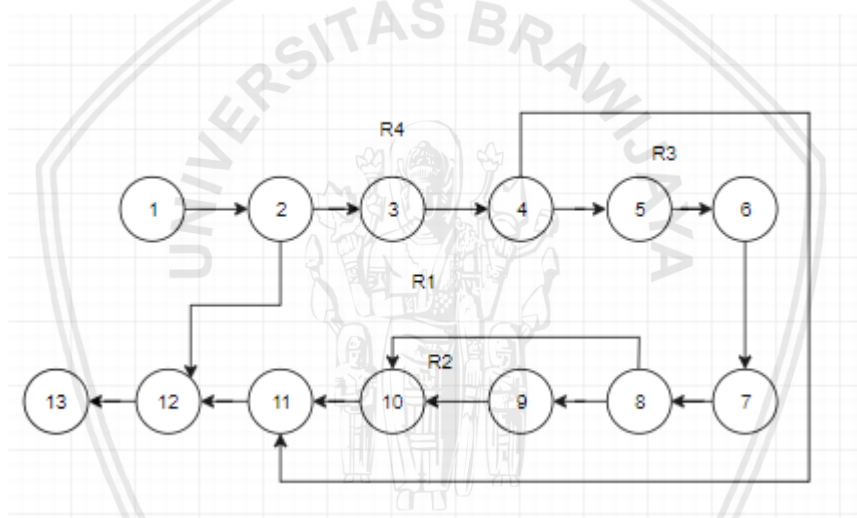
NO	Source code/Algoritma
1	Start
2	<code>idProject = \$id, idp = \$idp</code>
3	<code>inisialisasi array \$data</code>
4	<code>Inisialisasi \$file, \$filename, \$pathjson,</code>
5	<code>If (File_get_contents jsonfile success)</code>
6	<code>do Json_decode save to filejson array</code>
7	<code>if (write new file sql)</code>
8	<code>do</code>
9	<code>create database</code>
10	<code>transformregularNew()</code>
11	<code>transformweakNew()</code>
12	<code>transformrelationonetooneNew()</code>
13	<code>transformreationonetomanyNew()</code>
14	<code>transformrelationmanytomanyNew()</code>
15	<code>transformmultivaluedNew()</code>
16	<code>transformmanyaryNew()</code>



NO	Source code/Algoritma
17	if (addsqltoDB()) _____ 8
18	set_flashdata success _____ 9
19	end if _____ 10
20	End if _____ 11
21	Else print error _____ 12
22	End if _____
23	End _____ 13

6.1.2.1 Flow graph

Dari algoritma yang ada pada operasi mapping, maka dibangunlah flow graph seperti pada Gambar 6.2 untuk menemukan cyclomatic complexity dari proses yang ada.



Gambar 6.2 Flowgraph operasi mappingbaru

6.1.2.2 Cyclomatic Complexity

$$V(G) = \text{Jumlah Region} = 4$$

$$V(G) = E - N + 2 = 15 - 11 + 2 = 4$$

$$V(G) = P + 1 = 3 + 1 = 4$$

6.1.2.3 Independent Path

Dari hasil penghitungan cyclomatic complexity, didapatkan bahwa proses memiliki tiga buah jalur, yaitu 1-2-3-4-5-6-7-8-9-10-11-12-13, 1-2-12-13, dan 1-2-3-4-11-12-13 dan 1-2-3-4-5-6-7-8-10-11-12-13. Pada Tabel 6.4 akan dijelaskan mengenai setiap prosedur uji, hasil yang diharapkan, hasil yang didapatkan, dan juga status apakah hasil sudah sesuai atau belum.

Tabel 6.4 Test Case algoritma proses mappingbaru

No	Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1-2-3-4-5-6-7-8-9-10-11-12-13	Isi <i>file database</i> ada, yang bernama sama tidak ada.	Membuat <i>database</i> baru. Proses mapping berhasil.	<i>Database</i> baru dibuat. Proses mapping berhasil	Valid
2.	1-2-12-13	Isi <i>file</i> kosong.	Menampilkan pesan error	Menampilkan pesan error.	Valid
3.	1-2-3-4-11-12-13	<i>Gagal melakukan write_file</i>	Tidak melakukan apapun pada database	<i>Tidak menyimpan apapun pada database</i>	valid
4	1-2-3-4-5-6-7-8-10-11-12-13	<i>Jlka sqltoDB gagal</i>	Tidak menampilkan pesan sukses	Tidak menampilkan pesan sukses	valid

6.1.3 Pengujian Unit downloadSql

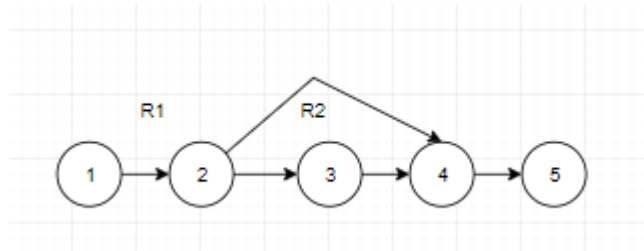
Pada Tabel 6.5 merupakan potongan algoritma method download sql dari kelas controller Member. Method ini berfungsi untuk melakukan proses download file sql hasil dari mapping.

Tabel 6.5 Potongan source code dari method downloadSql

NO	Source code/Algoritma
11	<pre> Start Load helper <i>download</i> Load model erd_m getErd Inialisasi linksql, filename, GetEr() If (Force_ <i>download</i> dari linksql = gagal) Display error <i>download</i> End if Load view detailProject end </pre>

6.1.3.1 Flow graph

Dari algoritma yang ada pada operasi *downloadsql*, maka dibangunlah flow graph seperti pada Gambar 6.3 untuk menemukan cyclomatic complexity dari proses yang ada.



Gambar 6.3 Flowgraph operasi *downloadSql*

6.1.3.2 Cyclomatic Complexity

$$V(G) = \text{Jumlah Region} = 2$$

$$V(G) = E - N + 2 = 5 - 5 + 2 = 2$$

$$V(G) = P + 1 = 1 + 1 = 2$$

6.1.3.3 Independent Path

Dari hasil penghitungan cyclomatic complexity, didapatkan bahwa proses memiliki dua buah jalur, yaitu 1-2-3-4-6-7 dan 1-2-3-5-6-7. Pada Tabel 6.6 akan dijelaskan mengenai setiap prosedur uji, hasil yang diharapkan, hasil yang didapatkan, dan juga status apakah hasil sudah sesuai atau belum.

Tabel 6.6 Test Case algoritma proses *download sql*

No	Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1-2-3-4-5	File yang <i>download</i> tersedia	File <i>terdownload</i> , kembali ke halaman detailproject	File <i>terdownload</i> , kembali ke halaman detail project	valid
2.	1-2-4-5	File yang <i>download</i> tidak tersedia	Display error	Menampilkan pesan bahwa proses <i>download</i> tidak berhasil.	valid

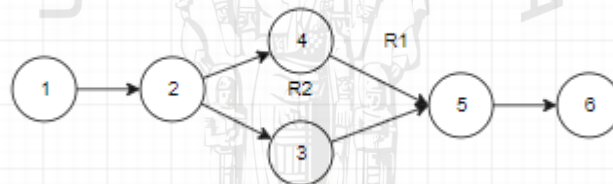
6.1.4 Pengujian Unit operasi tambah ERD Kelas ERD

Tabel 6.7 Potongan source code dari method addERD

NO	Source code/Algoritma
1	Start
2	Inialisasi linkjson,json,idproject, iderd _____ 1
3	if (json file is empty) _____ 2
4	load view detailproject _____ 3
5	else adderd(iderd) _____ 4
6	End if _____ 5
7	Load view detailProject _____ 6
8	end
9	

6.1.4.1 Flow graph

Dari algoritma yang ada pada operasi *addERD*, maka dibangunlah flow graph seperti pada Gambar 6.4 untuk menemukan cyclomatic complexity dari proses yang ada.



Gambar 6.4 Flowgraph operasi addERD

6.1.4.2 Cyclomatic Complexity

$V(G) = \text{Jumlah Region} = 2$

$V(G) = E - N + 2 = 6 - 6 + 2 = 2$

$V(G) = P + 1 = 1 + 1 = 2$

6.1.4.3 Independent Path

Dari hasil penghitungan cyclomatic complexity, didapatkan bahwa proses memiliki dua buah jalur, yaitu 1-2-3-5-6 dan 1-2-4-5-6. Pada Tabel 6.8 akan dijelaskan mengenai setiap prosedur uji, hasil yang diharapkan, hasil yang didapatkan, dan juga status apakah hasil sudah sesuai atau belum.

Tabel 6.8 Test Case Algoritma proses tambah ERD

No	Jalur	Prosedur Uji	Expected Result	Result	Status
1.	1-2-3-5-6	Isi file json kosong. Tidak membuar erd apapun.	Menampilkan halaman detailproject	Menampilkan halaman detail project	valid
2.	1-2-4-5-6	Membuat erd pada workspace. Isi file json berisi.	Menyimpan data, kembali ke halaman detail project	Menyimpan data, kembali ke halaman detail project	valid

6.2 Pengujian Validasi

Dalam tahap pengujian validasi, akan diketahui apakah pembangunan sistem sudah sesuai dengan hasil analisis kebutuhan dan spesifikasi sistem yang telah dirancang di tahap awal. Pada tahap pengujian ini, akan mengacu pada kebutuhan yang ada pada tahap perancangan, fungsional maupun non fungsional. Pengujian ini akan menggunakan metode *black box testing*, di mana akan menguji sistem tanpa melihat algoritma di dalamnya.

6.2.1 Pengujian Pemetaan ERD Upload

Pada pemetaan ERD upload pada Tabel 6.9, didapatkan hasil valid pada *main flow*, dan juga *alternative flow* nya. Pemetaan ERD upload atau mapping upload ini melaalui prosedur pengujian *main flow* yaitu apabila file upload yang dipilih berekstensi selain dari erdplus. Hasil output yang keluar sama dengan hasil yang diharapkan yaitu sistem menampilkan halaman detail project dan link download sql aktif.

Tabel 6.9 hasil pengujian pemetaan ERD upload

Test Case upload ERD			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> memilih <i>file</i> berekstensi erdplus, dan mengupload <i>file</i> .	Sistem mengolah file menjadi sql. File tersimpan. Sistem menampilkan halaman detail <i>project</i> dan mengaktifkan link download sql	Sistem mengolah file menjadi sql. File tersimpan. Sistem menampilkan halaman detail project dan link download sql aktif.	valid

(Alternative flow)			
<i>Member</i> memilih <i>file</i> berekstensi .jpeg	Menampilkan pesan “pilih <i>file</i> dengan tipe erdplus”	Sistem menampilkan pesan “pilih <i>file</i> bertipe erdplus”	valid
<i>Member</i> memilih <i>file</i> berekstensi 100mb	Menampilkan pesaan “pilih <i>file</i> dengan batas ukuran 5mb”	Sistem menampilkan pesan “pilih <i>file</i> dengan batas ukuran 5mb”	valid

6.2.2 Pengujian Pemetaan ERD baru

Pada pengujian ERD baru mendapatkan hasil pengujian valid pada Tabel 6.10. Hasil yang diharapkan adalah menampilkan dan mengaktifkan tombol link download sql. Pada saat dijalankan, sistem menghasilkan seperti yang diharapkan.

Tabel 6.10 hasil pengujian pemetaan ERD baru

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> memilih tombol ‘mapping erd’	Menampilkan dan mengaktifkan tombol link <i>download sql</i>	Sistem menampilkan dan mengaktifkan tombol link <i>download sql</i>	valid

6.2.3 Pengujian Buat ERD

Pada pengujian buat ERD upload pada Tabel 6.11, hasil pengamatan sistem untuk mainflow nya adalah sistem dapat menyimpan diagram kedalam file json dalam database. Hasil ini sudah sesuai dengan hasil yang diharapkan. Sehingga mendapat hasil valid. Sedangkan untuk alternatif flow nya, apabila member ingin melakukan buat erd, namun tidak melakukan perubahan apapun pada erd, sistem tidak akan menyimpan atau menjalankan apapun.

Tabel 6.11 hasil pengujian buat ERD

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> menyusun diagram dan menyimpannya	Diagram tersimpan.	Sistem menyimpan diagram.	valid
(Alternative flow)			
<i>Member</i> tidak melakukan perubahan pada <i>workspace</i>	Sistem tidak melakukan apapun	Sistem tidak melakukan apapun	valid

6.2.4 Pengujian Edit ERD

Pada pemetaan edit erd pada Tabel 6.12, mendapatkan hasil pengujian valid untuk testcase utamanya. Yaitu pada saat member melakukan perubahan pada diagram dan memilih simpan, sistem akan menyimpan perubahan.

Tabel 6.12 hasil pengujian edit ERD

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> melakukan perubahan pada diagram, dan menyimpan.	Sistem menyimpan erd yang telah diedit	Sistem menyimpan diagram yang telah diedit	valid

6.2.5 Pengujian *download* Sql

Pada pemetaan *download* Sql pada Tabel 6.13, untuk main flownya, ketika member memilih tombol *download* maka diharapkan browser melakukan proses *download* file. Setelah dilakukan pengujian didapatkan hasil yang sama oleh sistem. Begitu juga dengan alternatif flownya, apabila file belum tersimpan, maka tidak akan melakukan apapun saat menekan tombol *download* sql.

Tabel 6.13 hasil pengujian *download sql*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> memilih tombol <i>download</i>	Sistem melakukan proses <i>download</i> melalui <i>browser</i> .	Sistem melakukan proses <i>download</i> melalui <i>browser</i> .	valid
(Alternative flow)			
<i>file</i> belum tersimpan	Tombol <i>download</i> tidak bisa digunakan, Sistem tidak melakukan apa-apa	Tombol <i>download</i> tidak bisa digunakan, Sistem tidak melakukan apa-apa	Valid

6.2.6 Pengujian login

Pada pengujian login pada Tabel 6.14, mendapatkan status valid, karena hasil pengamatan sama dengan hasil yang diharapkan. Ketika user mengisi form dengan lengkap maka halaman dashboard masing-masing member akan terbuka. Namun, apabila salah memasukan, maka akan kembali ke halaman login dan menampilkan pesan gagal

Tabel 6.14 hasil pengujian login

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
User mengisi <i>username</i> dan <i>password</i> yang benar	Halaman dashboard sesuai hak akses <i>member</i> atau administrator terbuka	Halaman dashboard terbuka (sesuai role dari <i>user</i>)	Valid

(Alternative flow)			
User mengisi <i>username</i> atau <i>password</i> salam	Tetap berada dihalaman login, Menampilkan pesan bahwa <i>password</i> atau <i>username</i> salah	Tetap berada dihalaman login, Menampilkan pesan bahwa <i>password</i> atau <i>username</i> salah	Valid

6.2.7 Pengujian Registrasi

Pada pengujian registrasi pada Tabel 6.15, mendapatkan status valid, karena hasil pengamatan sama dengan hasil yang diharapkan. Ketika user mengisi form dengan lengkap maka sistem akan menyimpan data user dan membawa ke halaman login. Namun, apabila tidak lengkap menginputkan form, maka akan kembali ke halaman registrasi dan menampilkan pesan untuk melengkapi.

Tabel 6.15 hasil pengujian registrasi

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
User mengisi data diri secara lengkap.	Sistem menyimpan data, dan menampilkan halaman login	Sistem menyimpan data, dan menampilkan halaman login	valid
(Alternative flow)			
User mengisi data diri, namun masih ada input yang dikosongi	Sistem menampilkan pesan untuk mengisi form dengan lengkap	Sistem menampilkan pesan untuk mengisi form dengan lengkap	valid

6.2.8 Pengujian Lupa *password*

Pada pengujian lupa *password*, mendapatkan hasil valid karena hasil pengamatan sesuai dengan hasil yang diharapkan. Seperti pada Tabel 6.16 dapat dilihat bahwa hasil pengamatan sesuai dengan hasil yang diharapkan. Sama dengan main flow, alternatif flownya pun juga mendapatkan hasil valid dengan menampilkan pesan bahwa email tidak terdaftar apabila menginputkan email yang salah.

Tabel 6.16 hasil pengujian lupa *password*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
User mengisi <i>email</i> yang telah didaftarkan	Mengirim <i>password</i> kepada <i>email</i> tersebut, dan kembali ke halaman login	Mengirim <i>password</i> kepada <i>email</i> tersebut, dan kembali ke halaman login	valid
(Alternative flow)			
Salah menginputkan email	Sistem menampilkan pesan " <i>email tidak terdaftar</i> "	Sistem menampilkan pesan " <i>email tidak terdaftar</i> "	valid

6.2.9 Pengujian Logout

Pada pengujian logout pada Tabel 6.17 didapatkan hasil valid. Apabila member atau administrator memilih tombol logout, maka sistem akan kembali ke halaman home, sesuai dengan hasil yang diharapkan.

Tabel 6.17 Hasil pengujian *logout*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> atau administrator memilih tombol <i>logout</i> , dan memilih 'ya' dari pesan dialog	Sistem kembali ke halaman home.	Sistem kembali ke halaman home.	valid

6.2.10 Pengujian Lihat Daftar *Member*

Pada pengujian lihat daftar member pada Tabel 6.18 memiliki keluaran sesuai dengan hasil yang diharapkan pada sistem. sistem akan menampilkan data member yang terdaftar dalam bentuk tabel.

Tabel 6.18 Hasil pengujian lihat daftar *member*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Administrator</i> membuka halaman dashboard atau lihat daftar <i>member</i>	Sistem menampilkan data <i>member</i> yang terdaftar dalam bentuk tabel	Sistem menampilkan data <i>member</i> yang terdaftar dalam bentuk tabel	valid

6.2.11 Pengujian hapus *member*

Pada pengujian hapus *member* pada Tabel 6.19, didapatkan hasil valid karena memiliki keluaran sesuai dengan hasil yang diharapkan dihasilkan sistem. ketika administrator menghapus salah satu *member*, maka akan membuka kembali halaman daftar *member*, dengan data yang sudah berubah.

Tabel 6.19 Hasil pengujian hapus *member*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Administrator</i> memilih tombol delete pada salah satu <i>member</i> . Dan memilih 'ya' pada dialog yang terbuka	Membuka kembali halaman daftar <i>member</i> , dengan perubahan data <i>member</i> yang sudah terhapus.	Membuka kembali halaman daftar <i>member</i> , dengan perubahan data <i>member</i> yang sudah terhapus.	valid
(Alternative flow)			
<i>Administrator</i> memilih tombol delete pada salah satu <i>member</i> . Dan memilih 'tidak' pada dialog yang terbuka	Akan kembali membuka halaman lihat data <i>member</i> tanpa ada data yang terhapus	Akan kembali membuka halaman lihat data <i>member</i> tanpa ada data yang terhapus	valid

6.2.12 Pengujian lihat daftar *project*

Pada pengujian lihat daftar member pada Tabel 6.20 memiliki keluaran sesuai dengan hasil yang diharapkan pada sistem. sistem akan menampilkan data *project* yang terdaftar dalam bentuk tabel.

Tabel 6.20 Hasil pengujian lihat daftar *project*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Administrator</i> membuka halaman daftar <i>project</i>	Sistem menampilkan halaman daftar <i>project</i> beserta data <i>project</i> dalam tabel.	Sistem menampilkan halaman daftar <i>project</i> beserta data <i>project</i> dalam tabel.	valid

6.2.13 Pengujian lihat daftar *project member*

Pada pengujian lihat daftar *project member* pada Tabel 6.21, mendapatkan hasil valid karena hasil pengamatan sesuai dengan hasil yang diharapkan keluar dari sistem. dimana sistem menampilkan halaman daftar *project* yang dimiliki oleh member.

Tabel 6.21 Hasil pengujian lihat daftar *project member*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> membuka halaman daftar <i>project</i>	Sistem menampilkan halaman daftar <i>project</i> beserta data <i>project</i> dalam tabel.	Sistem menampilkan halaman daftar <i>project</i> beserta data <i>project</i> dalam tabel.	valid
(Alternative flow)			
Belum terdapat <i>project</i> yang dibuat oleh <i>member</i> , dan <i>Member</i> membuka halaman daftar <i>project</i>	Sistem menampilkan halaman daftar <i>project</i> dengan tulisan “ belum terdapat <i>project</i> ”	Sistem menampilkan halaman daftar <i>project</i> dengan tulisan “ belum terdapat <i>project</i> ”	valid

6.2.14 Pengujian buat *project member*

Pada Tabel 6.22, pengujian buat *project member* mendapatkan hasil pengamatan apabila member mengisi nama *project* dan memilih tombol buat *project* baru, akan membawa ke halaman daftar *project member* dengan data yang sudah terupdate.

Tabel 6.22 hasil pengujian buat *project member*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> mengisi nama <i>project</i> dan memilih tombol buat <i>project</i> baru.	Kembali ke halaman daftar <i>project member</i> dengan data yang baru	Kembali ke halaman daftar <i>project member</i> dengan data yang baru	valid

6.2.15 Pengujian hapus *project*

Pada pengujian hapus *project member* pada Tabel 6.23 mendapatkan hasil pengamatan yang valid yaitu apabila member memilih tombol delete pada salah satu data, akan menampilkan halaman daftar *project member* dengan data yang sudah diperbaharui, dimana file terpilih terhapus.

Tabel 6.23 hasil pengujian hapus *project*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> memilih tombol delete pada salah satu <i>project</i> pada daftar <i>project</i> . Dan memilih 'ya' pada pesan dialog yang keluar	Sistem menampilkan daftar <i>project</i> dengan perubahan <i>project</i> terpilih telah terhapus	Sistem menampilkan daftar <i>project</i> dengan perubahan <i>project</i> terpilih telah terhapus	valid
(Alternative flow)			
<i>Member</i> memilih tombol delete pada salah satu <i>project</i> pada daftar <i>project</i> . Dan memilih 'tidak'.	Kembali kehalaman daftar <i>project</i> , tanpa perubahan.	Kembali kehalaman daftar <i>project</i> , tanpa perubahan.	valid

6.2.16 Pengujian lihat detail *project*

Pada pengujian lihat detail *project* pada Tabel 6.24, menghasilkan pengujian yang valid, yaitu apabila member memilih salah satu *project* pada navigasi bar yang sudah mengupload file erd, akan menampilkan halaman detail *project* yang berisi tabel dan data-data pendukung erd. Sedangkan untuk alternatif flownya, ketika member memilih salah satu *project* yang berupa erd hasil pembuatan dengan diagram tool, maka sistem akan menampilkan halaman detail *project* beserta workspace yang tersimpan. Selain itu juga apabila member memilih salah satu *project* yang belum memiliki erd, maka akan menampilkan halaman detail *project* kosong yang terdapat fungsi untuk upload maupun membuat erd baru.

Dari ketiga flow yang ada, hasil pengamatannya sudah sesuai dengan hasil yang diharapkan, sehingga menghasilkan nilai valid.

Tabel 6.24 hasil pengujian lihat detail *project*

Test Case			
(Main flow)			
Prosedur pengujian	Hasil yang diharapkan	Hasil pengamatan	Status
<i>Member</i> memilih salah satu <i>project</i> pada navigasi bar yang sudah mengupload file erd, melakukan mapping, dan sudah terdapat file sql	Sistem menampilkan halaman detail <i>project</i> yang berisi table relasi, dan tombol untuk mendownload sql	Sistem menampilkan halaman detail <i>project</i> yang berisi table relasi, dan tombol untuk mendownload sql	valid
(Alternative flow)			
<i>Member</i> memilih salah satu <i>project</i> pada navigasi bar yang sudah terdapat erd hasil pembuatan dengan diagram.	Sistem menampilkan halaman detail <i>project</i> beserta workspace yang tersimpan, dan tombol mapping.	Sistem menampilkan halaman detail <i>project</i> beserta workspace yang tersimpan, dan tombol mapping.	valid
<i>Member</i> memilih salah satu <i>project</i> pada navigasi bar yang belum ada erd yang tersimpan.	Sistem menampilkan dua tombol yaitu tombol 'buat erd' dan tombol 'upload erd'	Sistem menampilkan dua tombol yaitu tombol 'buat erd' dan tombol 'upload erd'	valid

6.3 Pengujian Non-fungsional (Pengujian Compatibility)

Dalam pengujian *compatibility* ini digunakan lima *browser* berbeda pada PC, dan dijalankan dengan bantuan localhost dari xampp. Dari Tabel 6.25 dapat dilihat bahwa hasil empat *browser* pada desktop PC untuk menjalankan sistem berhasil, kecuali pada UC *browser* di mana tidak dapat menampilkan *workspace* untuk membuat ERD.

Tabel 6.25 Hasil pengujian Compatibility

Jenis <i>browser</i>	Chrome	Safari	Internet Explorer	Firefox	UC Browser
PHP	√	√	√	√	√
Javascript	√	√	√	√	√
CSS & Html	√	√	√	√	√
API GOJs	√	√	√	√	-

6.4 Analisis Pengujian

Dari hasil pengujian yang telah dilakukan, dapat diketahui bahwa dalam pengujian unit dengan empat buah *sample kelas*, menghasilkan nilai valid atau fungsional yang diujicoba berhasil. Untuk pengujian validasi atau blackbox, dari semua fungsional yang diujikan dan juga pengujian validasi untuk keluaran dari proses sistem, didapatkan hasil valid. Dan untuk pengujian nonfungsional, dengan menggunakan pengujian compatibility browser atau kemampuan sistem untuk digunakan dilingkungan browser yang berbeda-beda, mendapatkan hasil tidak *compatible* dengan satu dari lima buah browser yang diujicobaakan. Kegagalan salah satu browser ini, diperkirakan karena browser yang tidak dapat mengeksekusi canvas dan element dari library GOJs. Browser UC browser yang tidak dapat menjalankan GOJs dapat menjalankan fungsi lain, namun tidak dapat menjalankan fungsi yang berkaitan dengan pembuatan ER

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil analisis Sistem Pemetaan Otomastis Entity Relationship Diagram ke dalam *database*, dihasilkan 16 buah kebutuhan fungsional beserta spesifikasinya dari tiga macam aktor, sebuah kebutuhan non-fungsional, dan 16 buah usecase. Analisis didapatkan dari observasi program serupa yang ada untuk memetakan diagram.

Dalam tahap perancangan diambil empat buah contoh sequence diagram yang dari 16 buah sequence, dan sebuah class diagram yang terdiri dari empat buah kelas model, empat buah kelas controller dan 13 buah kelas view. Dalam perancangan juga merancang detail tiap operasi yang ada didalam setiap kelas, algoritma. Selain menghasilkan perancangan sistem, juga didapatkan hasil dari perancangan database berupa conceptual data model yang berisi lima buah entitas beserta relasi, dan atribut-atribut yang dibutuhkan, dan juga perancangan antarmuka atau *user interface* dari sistem. Pada tahap implementasi, didapatkan bahwa sistem dapat dikembangkan dengan bahasa pemrograman php, dan javascript. Sistem dikembangkan dengan bantuan manajemen basis data MySQL, dan *framework* codeigniter. *Library* yang digunakan dalam sistem adalah library Gojs untuk mengakses diagram *workspace* dan dapat merubah diagram menjadi berbentuk *file* berekstensi JSON.

Pengujian sistem dilakukan dengan tiga macam pengujian, yaitu pengujian *white box*, *black box*, maupun pengujian *compatibility*. Pada pengujian whitebox yang telah dilakukan terhadap 12 buah jalur menghasilkan 100% valid. Pada pengujian blackbox, dilakukan pada 16 buah test case menghasilkan 100% valid. Sedangkan untuk pengujian *compatibility* pada lima buah browser yaitu Google Chrome, Safari, Internet Explorer, Mozilla firefox, dan UC browser mendapatkan nilai 95% valid. Sistem dapat dijalankan dengan baik pada empat buah *browser*, kecuali pada UC *browser* yang tidak bisa mengakses library GoJS.

7.2 Saran

Berdasarkan hasil penelitian dan kesimpulan yang telah di tuliskan sebelumnya, maka penulis dapat menuliskan saran untuk penelitian selanjutnya di antaranya adalah fungsi *mapping* erd yang masih terpisah antara proses *upload* dan buat baru, sehingga hasil dari *upload* belum bisa langsung di *edit* di dalam sistem. Pada pengembangan berikutnya bisa ditambahkan fungsi sistem yang dapat melakukan hingga tahap pemetaan EER menjadi sebuah *database*. Penelitian selanjutnya juga dapat menambahkan fungsi kolaborasi, sehingga dalam satu *project* dapat dikerjakan lebih dari satu *member* dan, pengembangan selanjutnya dapat mengembangkan sistem ke berbasis mobile atau berbasis android.

DAFTAR PUSTAKA

- A.S.Gaikwad, F.A.Kadri, S.S.Khandagle, dan N.I.Tava. 2017. "Review on Automation Tool for ERD Normalization." *International Research Journal of Engineering and Technology (IRJET)* 4 (2).
- Al-Masree, Hala Khaled. 2015. "Extracting Entity Relationship Diagram (ERD) From Relational." *International Journal of Database Theory and Application* 8: 15-26.
- Bassett, Lindsya. 2015. *Introduction to JavaScript Object Notation*. 1. O'Reilly Media, Inc.
- Booch, Grady. 1994. *Object Oriented Analysis and Design with applications*. 2nd. Addison-wesley.
- Bruegge, B, dan A. H. Dutoit. 2000. *Object-Oriented Software Engineering*. 3rd. Prentice Hall.
- Chung, L, dan JCS do Prado Leite. 2009. "On Non-Functional Requirements in Software." *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos* (Springer-Verlag Berlin Heidelberg) 363-379.
- CS Odessa. 2017. *Design Element: Chen for entity relationship diagram - ERD*. Diakses Mei 2018. Conceptdraw.com.
- Edrawsoft. t.thn. *Simple Chen ERD Template*. <https://www.edrawsoft.com/template-simple-chen-erd.php>.
- Elmasri, Ramez, dan Shamkant B. Navathe. 2016. *Fundamentals of Database Systems*. 7. Pearson Education Limited.
- ERDPlus. 2015-2016. *ERDPlus*. Diakses 5 2018. <https://erdplus.com/>.
- Hay, David C. 1999. "A comparison of Data Modeling Techiques." *Essential Strategies, Inc.* Diakses April 2018. <https://pdfs.semanticscholar.org/e50e/870b81f7767208aa6f913f575938cac5a920.pdf>.
- Johnson, Ralph E. 1997. "Frameworks = (Components + patterns)." *Communications of the ACM* 40 (10): 39-42.
- JSON.org. 2002. *Introducing JSON*. Diakses 4 2018. <https://www.json.org/>.
- Kumar, M, SK Singh, dan RK Dwivedi. 2015. "A Comparative Study of Black Box Testing and White Box." *International Journal of Advance Research in Science (IJARCSMS)* 3 (10): 32-44. www.ijarcsms.com.
- Kurniawan, Tri Astoto. 2018. "Pemodelan Use Case (UML): Evaluasi Terhadap beberapa Kesalahan dalam Praktik." *Jurnal Teknologi Informasi dan Ilmu Komputer* 5.
- Lethbridge, Timothy, dan Robert Laganriere. 2005. *Object-Oriented Software Engineering*. 2nd. NY, USA.

- Mohammed, Mohammed Anwar, Danial Abdul kareem Muhammed, dan Jaza Mahmood Abdullah. 2015. "Practical Approaches of Transforming ER Diagram into Tables." *International Journal of Multidisciplinary and Scientific Emerging Research* 4.
- Nishadha. 2012. *UML Diagram Types Examples*. Diakses January 28, 2018. <http://creately.com/blog/diagrams/uml-diagram-types-examples/>.
- Northwoods Software Corporation. 1998. *Introduction to GoJS Diagramming Components*. Diakses Juni 2018. <https://gojs.net/latest/intro/index.html>.
- Poo, Danny, Derek Kiong, dan Swarnalatha Ashok. 2008. *Object-Oriented Programming and Java*. 2nd. Singapore: Springer.
- Pressman, Roger S. 2001. *Software Engineering : a practitioners approach*. 5th. New York: McGraw-Hill Book Company.
- Rumbaugh, James, Ivar Jacobson, dan Grady Booch. 2004. *Unified Modeling Language Reference Manual*. 2nd.
- Santoso, Leo Willyanto. 2004. "PERANCANGAN DAN PEMBUATAN APLIKASI ERD GENERATOR NOTASI ORM DARI SKRIP BASIS DATA ORACLE BERBASIS J2EE." *Jurnal Informatika Universitas Petra* 5. Diakses 3 2018. <http://jurnalinformatika.petra.ac.id/index.php/inf/article/view/15437>.
- Software testing fundamentals. 2010. *Black Box Testing*. Diakses November 2018. <http://softwaretestingfundamentals.com/black-box-testing/>.
- software testing genius. 2011. *White box unit testing a bottom up approach*. Diakses November 2018. Sumber: <https://www.softwaretestinggenius.com/white-box-unit-testing-a-bottom-up-approach-of-software-testing/>.
- Sommerville, Ian. 2011. *Software Engineering*. 9. Boston.
- Song, Il-Yeol, Marry Evans, dan E. K Park. 1995. "A comparative Analysis of Entity Relationship Diagrams." *Journal of computer and software Engineering* 3 (4): 427-459.
- Statista. 2017. *User population of selected internet browsers worldwide from 2012 to 2017 (in millions)**. July. Diakses July 2018. <https://www.statista.com/statistics/543218/worldwide-internet-users-by-browser/>.