TUGAS AKHIR - KI141502

PENERAPAN FRAMEWORK METEOR JS DAN NOSQL DATABASE DALAM PEMBUATAN APLIKASI ROOM CHAT

ADIVITA CHANDRA MUTIA
NRP 5109 100 033

Dosen Pembimbing I
Dwi Sunaryono, S.Kom, M.Kom.
Dosen Pembimbing II
Sarwosri, S.Kom, M.T

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR - KI141502

PENERAPAN FRAMEWORK METEOR JS DAN NOSQL DATABASE DALAM PEMBUATAN APLIKASI ROOM CHAT

ADIVITA CHANDRA MUTIA
NRP 5109 100 033

Dosen Pembimbing I
Dwi Sunaryono, S.Kom, M.Kom.
Dosen Pembimbing II
Sarwosri, S.Kom, M.T.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

THE APPLICATIONS OF FRAMEWORK METEOR JS AND NOSQL DATABASE IN BUILDING ROOM CHAT APPLICATION

ADIVITA CHANDRA MUTIA
NRP 5109100033

Supervisor I
Dwi Sunaryono, S.Kom, M.Kom

Supervisor II
Sarwosri, S.Kom, M.T

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENERAPAN FRAMEWORK METEOR JS DAN NOSQL DATABASE DALAM PEMBUATAN APLIKASI ROOM CHAT

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
ADIVITA CHANDRA MUTIA
NRP: 5109 100 033

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dwi Sunaryono, S.Kom, M.Kom
NIP: 197205281997021001 ..... (pembimbing 1)

Sarwosri, S.Kom, M.T
NIP: 197608092001122001 ..... (pembimbing 2)

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI INFORMASI
JURUSAN TEKNIK INFORMATIKA

SURABAYA
Agustus, 2016

[Halaman ini sengaja dikosongkan]

PENERAPAN FRAMEWORK METEOR JS DAN NOSQL DATABASE DALAM PEMBUATAN APLIKASI ROOM CHAT

Nama Mahasiswa	**: Adivita Chandra Mutia**
NRP	**: 5109100033**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dwi Sunaryono, S.Kom, M.Kom.**
Dosen Pembimbing 2	**: Sarwosri, S.Kom, M.T.**

Abstrak

Penggunaan framework dalam pengembangan sebuah aplikasi khususnya web bisa sangat membantu developer. Berbagai macam framework tersedia dengan menawarkan library atau kelebihan masing-masing baik yang berbasis PHP, Java, JavaScript dll. Namun yang saat ini semakin banyak dikenal dan digunakan webdesigner ataupun webdeveloper adalah JavaScript framework. Pada awalnya, framework JavaScript berkembang saat memasuki era web 2.0. Generasi kedua dari layanan berbasis web dimana lebih menitikberatkan pada kolaborasi online, sharing content antar pengguna dan lebih terarah ke User Content Generated. Salah satu framework berbasis JavaScript yang baru saja dikembangkan adalah Meteor. Meteor menawarkan cara yang lebih sederhana dalam pengembangan suatu aplikasi web real-time serta sudah didukung langsung oleh database NoSQL yaitu MongoDb.

Pada tugas akhir ini akan menerapkan framework meteor js dan penggunaan NoSQL database dalam pembuatan aplikasi room chat. Meteor memberikan kemudahan dalam pengembangan aplikasi dengan 1 bahasa yaitu JavaScript di berbagai lingkungan atau environments baik aplikasi server, web browser dan perangkat mobile.

Kemudian hasil tugas akhir ini akan diuji berdasarkan fungsionalitasnya. hasil pengujian menunjukkan aplikasi chat yang dibangun menggunakan framework meteor js lebih sederhana namun tetap responsif. Rata-rata waktu yang dibutuhkan untuk pengiriman dan penerimaan pesan yaitu 0.0154 detik.

Kata kunci: Meteor, JavaScript, Database NoSQL, MongoDB, Room Chat

THE APPLICATIONS OF METEOR JS AND NOSQL DATABASE IN BUILDING ROOM CHAT APPLICATION

Nama Mahasiswa	**: Adivita Chandra Mutia**
NRP	**: 5109100033**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dwi Sunaryono, S.Kom, M.Kom.**
Dosen Pembimbing 2	**: Sarwosri, S.Kom, M.T.**

Abstract

The use of the framework in the development of a particular application can be very helpful web developer. Various kinds of frameworks available by offering a library or the advantages of each well is based on PHP, Java, JavaScript etc. But today more and more known and used webdesigner or webdeveloper is a JavaScript framework. At first, the JavaScript framework develops when entering the era of web 2.0. The second generation of web-based service which is more focused on online collaboration, sharing content among users and more targeted to the User Generated Content. One of the JavaScript-based framework recently developed is Meteor. Meteor offers a simpler way in the development of a real-time web applications that can run multiplatform and has been supported directly by that MongoDB NoSQL Database.

In this final project will implement a framework and NoSQL Database Meteor Database in building room chat application. Meteor provides ease of application development with only 1 language of JavaScript in various environments both application servers , web browsers and mobile devices.

Then the results of this final project will be tested based on the functionality. test results show a chat application that

was built using the framework meteor js more modest but still responsive. Average time required for sending and receiving messages is 0.0154 seconds.

Keywords: Meteor, JavaScript, NoSQL Database, MongoDB, Room Chat

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur Alhamdulillah kepada Allah Yang Maha Kuasa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul: **"PENERAPAN FRAMEWORK METEOR JS DAN NOSQL DATABASE DALAM PEMBUATAN APLIKASI ROOM CHAT"**

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pekerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Bapak Machmud Muhammad dan Ibu Supatmi, orang tua penulis yang tak henti-hentinya memberikan doa, dukungan dan semangat setiap saat.
2. Kakak Yunita Purwasingsih, Kakak Vivi Sarikayanti dan keluarga penulis yang selalu memberikan dukungan semangat dan doa.
3. Bapak Dwi Sunaryono, S.Kom, M.Kom. dan Ibu Sarwosri, S.Kom, M.T. selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan bimbingan selama proses pengerjaan Tugas Akhir ini.
4. Ibu Isye Arieshanti, S.Kom, M.Phil. dan Bapak Dr.Eng. Radityo Anggoro, sebagai Dosen Wali penulis, yang telah membantu penulis selama menempuh kuliah S1.
5. Fadilatuzzahra dan sahabat-sahabat penulis Haqiqi, Yuda, Anggry, Rein, Jafier, Jarwo, Trdz, Ocu, Taufik, WTP, Harun, kawan-kawan TC angkatan 2009 dan

Viruz yang secara tidak langsung membantu jalannya tugas Akhir ini.

6. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
7. Pak Yudi, Pak Sugeng, Pak Pri dan segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.
8. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu -persatu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Agustus 2016

Adivita Chandra Mutia

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini dipaparkan mengenai garis besar Tugas Akhir, meliputi latar belakang, tujuan, rumusan permasalahan, batasan permasalahan, metodologi penyelesaian Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Meteor adalah sebuah *platform* yang dibangun di atas Node.js untuk membuat aplikasi web *real-time.* Karena dibangun di atas Node.js, meteor menggunakan *JavaScript* baik pada sisi *client* maupun *server*. Selain itu meteor juga menyediakan *free testing server* untuk melakukan testing aplikasi secara live serta terintegrasi dengan *Cordova* sehingga membantu dalam proses *deploy* ke *mobile*.

Framework meteor juga sudah mendukung NoSQL *database*. NoSQL atau singkatan dari *Not Only* SQL adalah jenis basis data yang tidak menggunakan perintah SQL dalam memanipulasi (menyimpan maupun mengambil data) basis data tersebut. Dari cara penyimpanan, noSQL *database* tersebar dari cara penyimpanan dalam *key-Value based, Document based, Column based* dan *Graph based.* Jenis *database* noSQL yang berjalan pada ekosistem meteor adalah MongoDB yang *document based* karena disimpan dalam dokumen-dokumen. MongoDb menyimpan data-datanya dalam suatu dokumen *JSON* yang disebut *BSON (Binary JSON).* Representasi *front-end* dari *MongoDB* disebut *Minimongo,* yang seluruhnya ditulis dalam *JavaScript*. Selain itu meteor memiliki API *Mongo* yang lancar mengintegrasikan *MongoDb* di *back-end* dan *Minimongo* pada *front-end*. Hal inilah yang menghasilkan *reload* halaman lebih cepat dan meringankan *update latency* halaman web.

Aplikasi ini dibangun untuk melakukan Penerapan *Framework Meteor* berbasis *javascript* dan penggunaan *database*

yang sudah terkoneksi langsung dengan *Meteor yaitu MongoDB* dalam pengembangan aplikasi *room chat. Meteor* menawarkan cara yang sederhana dalam membangun aplikasi web dan dapat berjalan *multiplatform* baik *desktop* maupun *mobile* secara *realtime* serta penggunaan *data on the wire* yang artinya server hanya akan mengirimkan data, bukan HTML dan client yang memproses data tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini sebagai berikut:

1. Bagaimana membuat aplikasi *room chat* menggunakan *meteor js*?
2. Bagaimana penggunaan *NoSQL Database* dalam pengembangan aplikasi *room chat*?

1.3 Batasan Masalah

Beberapa batasan Tugas Akhir ini adalah sebagai berikut:

1. Aplikasi ini hanya melakukan percakapan *text*.
2. Aplikasi ini hanya melakukan percakapan umum antar pengguna yang *online* pada jaringan *local.*

1.4 Tujuan

Tujuan dari pembuatan aplikasi ini adalah melakukan *eksporasi* pada *framework* yang baru saja berkembang yaitu meteor js dan penggunaan noSQL *database* jenis *MongoDB* dalam pengembangan aplikasi *room chat.*

1.5 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Studi Literatur

Pada tahap ini dilakukan pengumpulan informasi mengenai hal-hal pendukung pengerjaan Tugas Akhir. Pengumpulan ini dimaksudkan untuk melakukan analisis dan perancangan sistem.

Adapun literatur yang dipakai adalah:

a. Teori konsep *Java Script.*
b. Teori konsep Meteor js.
c. Teori konsep *MongoDB.*
d. Teori pengembangan aplikasi menggunakan Meteor js.
e. Teori penggunaan *MongoDB* pada Meteor js.
f. Teori konsep aplikasi Chat.

2. Analisis dan Desain Aplikasi

Pada tahapan ini, penulis melakukan analisis dan desain aplikasi. Analisis dan perancangan aplikasi dilakukan untuk merumuskan spesifikasi kebutuhan aplikasi dan mendapatkan kebutuhan-kebutuhan yang sesuai dengan konsep Meteor js. Adapun tahap desain dilakukan untuk memodelkan hasil analisis dalam bentuk desain arsitektur, desain antarmuka dan desain-desain pendukung lain dalam pembangunan aplikasi.

3. Implementasi

Pada tahap ini dilakukan implementasi perangkat lunak ke dalam bentuk kode program. Adapun perincian adalah sebagai berikut:

a. Implementasi pembuatan *project* menggunakan Meteor js;
b. Implementasi Meteor js pada aplikasi Group Chat;

4. Pengujian dan Evaluasi

Tahapan ini digunakan untuk melakukan pengujian dan evaluasi pada aplikasi. Tahapan ini bertujuan untuk mengetahui

performance dan kesalahan-kesalahan yang ada pada aplikasi sehingga dapat dilakukan perbaikan terhadap aplikasi itu sendiri.

5. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.6 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk *pseudocode*.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Pada bab ini membahas mengenai teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Konsep Dasar

Pada perkembangan saat ini teknologi komputer menjadi hal penting yang dapat menyokong kelancaran segala aktifitas kehidupan manusia. Pada saat ini teknologi paling berkembang adalah internet. Internet menjadi sebuah media penyampaian informasi melalui sebuah website. Di beberapa negara maju internet bahkan menjadi tulang punggung dalam memperoleh informasi dan berkomunikasi.

Chatting adalah suatu pesan instant ataupun instant messaging di sebuah teknologi jaringan komputer yang mengijinkan pemakainya untuk mengirimkan pesan ke pengguna lain yang tersambung dalam sebuah jaringan komputer ataupun internet. Chatting tidak hanya populer dikalangan kaum remaja atau anak muda saja tetapai jaman sekarang ini, sudah merambah kekalangan orang tua sekalipun. Dengan adanya chatting, kita dapat dengan bebas mengobrol mengenai apa saja mulai dari persahabatan, pekerjaan, pelajaran disekolah, mata kuliah bahkan sampai dengan hal-hal bersifat pribadi sekali pun.

Aplikasi untuk melakukan aktifitas chatting disebut IM atau Instan Messenger. IM merupakan aplikasi pengolah pesan cepat yang memfasilitasi aktifitas komunikasi antara dua orang atau lebih secara realtime[**1**].

Pembuatan aplikasi *chat* dapat dilakukan dengan beragam teknologi yang semakin berkembang. Pada tugas akhir ini akan menerapkan *framework* berbasi *JavaScript* yaitu *Meteor JS* dan penggunaan *NoSQL Database* dalam pembuatan aplikasi *room chat.* Aplikasi *room chat* ini nantinya akan berjalan pada jaringan *local* pada perangkat yang digunakan dalam pembuatan aplikasi.

2.2 Teknologi yang Digunakan

Pada sub-bab ini membahas mengenai teknologi yang digunakan dalam pengembangan perangkat lunak.

2.2.1 *Framework*

Framework adalah sekumpulan perintah/fungsi dasar yang dapat membantu dalam menyelseikan proses-proses yang lebih kompleks. Sehingga seorang programmer tidak perlu lagi membuat fungsi-fungsi (biasanya disebut kumpulan *library*) dari awal. Programmer tinggal memanggil kumpulan *library* atau fungsi yang sudah ada didalam *framerwork*. Dengan adanya *framework*, akan sangat membantu proses penyelesaian program didukung oleh analisa sistem yang baik dan pertimbangan sumberdaya yang ada. Secara umum *framework* menggunakan struktur MVC (Model, *View, Controller).* Model mencakup semua proses yang terkait dengan pemanggilan struktur data baik pemanggilan fungsi, *input processing* atau mencetak *output* ke dalam *browser*. *Controller* mencakup semua proses yang terkait dengan pemanggilan *database* dan *kapsulisasi* proses-proses utama. *View* mencakup semua proses yang terkait *layout output* atau tempat untuk template interface web atau aplikasi[**2**].

2.2.2 *JavaScript*

JavaScript adalah bahasa pemrograman tingkat tinggi dan dinamis. JavaScript populer di internet dengan dapat bekerja di sebagian besar penjelajah web populer seperti *Internet Explorer* (IE), *Mozilla Firefox, Netscape* dan *Opera.* Kode *JavaScript* dapat disisipkan dalam halaman web menggunakan tag *SCRIPT* [**3**].

Sejarah *JavaScript* dimulai sekitar tahun 1994, ketika *internet* dan *website* sedang mengalami perkembangan yang pesat. *Website* pada saat itu umumnya dibuat menggunakan bahasa pemograman *PERL* yang pemrosesannya hanya bisa dilakukan di sisi *web server*.

Kelemahan pemrosesan di sisi *web server* adalah, setiap instruksi dari user harus dikirim terlebih dahulu kepada *web server*, baru kemudian ditampilkan lagi di dalam *web browser*. Karena kecepatan rata-rata koneksi internet yang terbatas, hal ini dipandang tidak efisien. Programmer *web* membutuhkan bahasa pemograman *client-side* yang bisa berjalan di *web browser* tanpa harus dikirim ke *server*.

Hal ini lah yang membuat Brendan Eich seorang programmer dari *Netscape* mulai mengembangkan sebuah bahasa pemograman *script* yang dinamakan *Mocha*. Bahasa script Mocha ini ditujukan untuk *client-side* dan juga *server-side*. Dalam perkembangan selanjutnya, nama *Mocha* diubah menjadi *LiveScript* untuk versi *client-side*, dan *LiveWire* untuk versi *server-side* hingga akhirnya berganti nama menjadi *JavaScript* dengan tujuan bahasa baru ini akan populer seperti bahasa *Java* yang saat itu sedang booming di kalangan programmer. Versi *JavaScript* ini dinamakan dengan *JavaScript* 1.0[**4**].

2.2.3 *Meteor*

Meteor adalah sebuah *platform* yang dibangun di atas *Node.js* untuk membuat aplikasi web *real-time.* Meteor bekerja di antara database aplikasi yang dibangun dan antarmuka pengguna dan memastikan bahwa keduanya senantiasa selaras. Karena dibangun di atas *Node.js*, meteor menggunakan *JavaScript* baik pada sisi *client* maupun *server*. Terlebih lagi, meteor juga dapat men-*share* kode antar-*environment.* Hasil dari semua ini adalah sebuah *platform* yang sangat sederhana namun *kapabel*, yang menangani secara otomatis kerumitan-kerumitan dan kesulitan umum pengembangan aplikasi web[**5**]**.**

Beberapa kelebihan yang dimiliki oleh Meteor adalah sebagai berikut:

1. *Meteor* memberikan kemudahan dalam pengembangan dengan 1 bahasa yaitu *JavaScript* di berbagai lingkungan atau *environments* baik aplikasi *server*, *web browser* dan perangkat *mobile*.
2. Penggunaan *data on the wire* yang artinya *server* hanya akan mengirimkan data, bukan *HTML* dan *client* yang memproses data tersebut.
3. Meteor memiliki ekosistem yang terdiri dari komunitas *JavaScript* sangat aktif dan terbaik pada bagiannya sehingga selalu membantu dalam setiap pengembangan pada Meteor.
4. Meteor memberikan dukungan reaktivitas yang memungkinkan *UI (User interface)* berjalan lancar sesuai dengan keadaan sebenarnya dengan upaya pengembangan yang minimal.

2.2.4 *MongoDB*

MongoDB adalah sistem basis data berorientasi dokumen lintas *platform*. Diklasifikasikan sebagai basis data "NoSQL". *MongoDB* menghindari struktur basis data relasional tabel berbasis tradisional yang mendukung *JSON* seperti dokumen dengan skema dinamis (*MongoDB* menyebutnya sebagai format *BSON*), membuat integrasi data dalam beberapa jenis aplikasi lebih mudah dan lebih cepat. Dirilis dibawah kombinasi dari *GNU Affero General Public License* dan lisensi *Apache*, *MongoDB* adalah perangkat lunak *free* dan *open source*[**6**].

MongoDB hadir dengan beberapa kelebihan yaitu :

1. Sudah secara langsung terkoneksi dengan *Metor.*
2. Performa yang ditawarkan MongoDB lebih cepat dibandingkan MySQL, ini disebabkan oleh memcached dan format dokumennya yang berbentuk seperti JSON.

3. *Replikasi*, adalah fitur yang sangat bermanfaat untuk *backup* data secara realtime. *MongoDB* sangat cocok digunakan untuk portal berita ataupun blog, namun belum cocok untuk digunakan pada sistem informasi yang berkaitan dengan keuangan karena *MongoDB* tidak mendukung *transaction SQL*.
4. *Auto-sharding*, merupakan fitur untuk memecah *database* yang besar menjadi beberapa bagian demi optimalisasi performa *database*. Penggunaannya sendiri sangat berguna ketika Anda memiliki website dengan *database* yang jutaan baris, *sharding* akan membantu memecahnya menjadi beberapa bagian.
5. *Cross-platform*, sehingga dapat digunakan di *Windows*, *Linux*, *OS X* dan *Solaris*.
6. Proses *CRUD* (Create, Read, Update, Delete) terasa sangat ringan
7. Map/*Reduce*, akan sangat membantu ketika melakukan operasi *agregasi*. Dimana semua entry datangnya dari *collection* dan outputnya pun akan menjadi *collection* juga. Kalau di *MySQL* biasanya kita menggunakan *query GROUP BY*.

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai hal-hal yang berkaitan dengan perancangan sistem yang akan dibuat. Perancangan tersebut mencakup deskripsi umum aplikasi, arsitektur sistem, model fungsional aplikasi, spesifikasi kebutuhan aplikasi, *use case* aplikasi, serta diagram alir aplikasi.

3.1 Deskripsi Umum

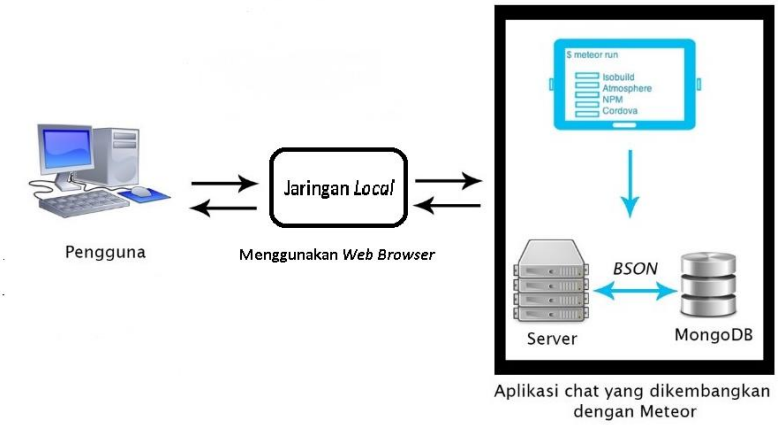
Penggunaan *framework* dalam pengembangan sebuah aplikasi khususnya web bisa sangat membantu *developer*. Berbagai macam *framework* tersedia dengan menawarkan *library* atau kelebihan masing-masing baik yang berbasis *PHP*, *Java*, *JavaScript* dll. Namun yang saat ini semakin banyak dikenal dan digunakan *webdesigner* ataupun *webdeveloper* adalah *JavaScript framework*. Pada awalnya, *framework JavaScript* berkembang saat memasuki era *web* 2.0. Generasi kedua dari layanan berbasis web dimana lebih menitikberatkan pada kolaborasi *online*, *sharing content* antar pengguna dan lebih terarah ke *User Content Generated***.** Berdasarkan kebutuhan akan konsep dan paradigma baru dalam pengembangan tersebut maka dalam pengembangannya diperlukan suatu sistem alur kerja yang bisa memudahkan dalam menciptakan aplikasi berbasis web yang canggih dan interaktif. Selain itu, ketika *JavaScript* digabungkan dengan *XML* yeng menghasilkan *AJAX* (*Ashynchonous JavaScript And XML*). Sebuah teknik pemanggilan cepat yang lebih interaktif dan bisa menghubungi *server* dengan proses dibelakang layar atau tanpa melakukan *reload* terhadap suatu *page* [**7**]**.**

Salah satu *framework* berbasis *JavaScript* yang baru saja dikembangkan adalah Meteor. Meskipun masih tergolong cukup muda, popularitas dari meteor semakin meningkat dari hari kehari. Hal ini dapat dilihat dari keaktifan komunitas dan beragam situs yang muncul yang dibuat dengan Meteor. Penyebab lainnya juga karena meteor menawarkan cara yang lebih sederhana dalam

pengembangan suatu aplikasi web *real-time.* Berdasarkan hal tersebut, akan dilakukan *eksplorasi* bagaimana penerapan meteor js dalam pengembangan aplikasi *room chat* yang bisa digunakan kapan saja dan dimana saja.

3.2 Arsitektur Umum Sistem

Langkah pertama adalah penerapan *framework* meteor js dan *MongoDB* dalam pengembangan aplikasi *room chat*. Aplikasi ini dapat berjalan pada *web browser* dengan jaringan *local* pada *destktop* dimana aplikasi ini dibangun seperti **gambar 3.1**.



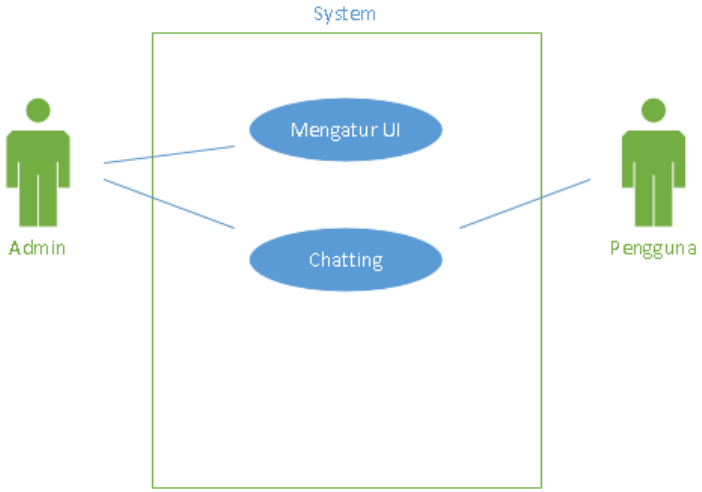
Gambar 3. 1. Arsitektur sistem

Pada proses awal dari sistem, pengguna harus melakukan pendaftaran identitas dan data tersebut akan disimpan di dalam *database* sehingga pengguna dapat melakukan *login* dan *logout*. Secara *default*, pendaftar pertama ketika aplikasi telah di bangun akan menjadi *admin. Admin* memiliki hak untuk mengatur tampilan UI pada aplikasi dan dapat melihat data *User* yang telah mendaftar. Setelah itu *Admin* dan *User* yang telah *online* dapat

melakukan percakapan umum. Percakapan yang dapat dilakukan hanya percakapan text.

3.3 Analisis Kebutuhan Fungsional

Use Case diagram pada gambar 3.2 akan menggambarkan kebutuhan *fungsionalitas* sistem beserta aktor yang terlibat.



Gambar 3. 2. Diagram *use case*

Gambar 3. menunjukkan proses jalannya aplikasi yang akan dibangun, dimana penjelasan tiap-tiap *Use Case* akan dijelaskan dalam *diagram* alur aplikasi.

Use case skenario

1. *Use Case* Mengatur *UI*
   Deskripsi :Proses yang hanya dapat dilakukan dengan *role admin.*
   Pre-Condition :*Admin* memilih masuk ke dalam sistem *role admin.*
   Post-Condition :*Admin* mengatur *UI* dari aplikasi.

2. *Use case Chatting*

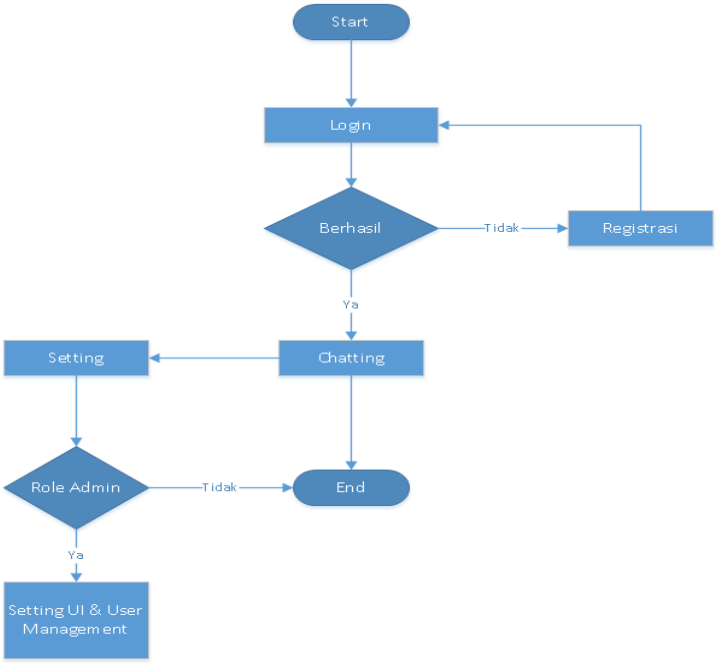
Deskripsi : Proses pengguna melakukan *chatting* dengan pengguna lainnya dalam chat umum.

Pre-Condotion : Pengguna mengirimkan pesan ke pengguna lain.

Post-Condition : Pengguna bertukar pesan dengan pengguna lain.

3.4 Diagram Alir Aplikasi Sistem

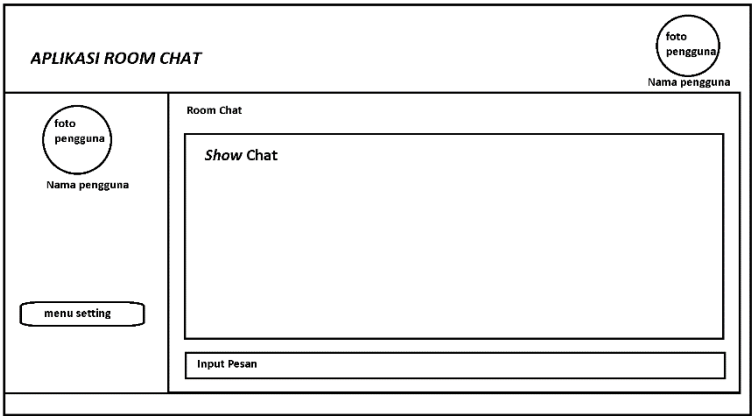
Alur setiap proses yang terdapat pada aplikasi digambarkan pada diagram alir, untuk memudahkan pemahaman secara garis besar proses yang dilakukan oleh sistem seperti yang ditunjukkan pada gambar 3.3.



Gambar 3. 3. Diagram Alir Aplikasi Chat

Aplikasi berjalan atau kondisi *Start* apabila sudah masuk atau pada halaman utama. Proses yang selanjutnya dilakukan adalah Login atau bila baru pertama kali menjadi pengguna diharuskan untuk melakukan registrasi dengan memberikan *Username*, *Password* dan *Email*. Penggunaan pendaftaran *Email* memudahkan pengguna bila tidak mengingat *Username* atau *Password* sehingga bisa di *reset*. Setelah melakukan proses *Login* atau registrasi, nama dan informasi pengguna akan terlihat dan statusnya menjadi *online*. Pengguna kemudian dapat melakukan percakapan umum dengan pengguna lain. Proses tambahan yang dapat dilakukan adalah pengaturan tampilan aplikasi dan dapat melihat data dari pengguna namun hanya dapat dilakukan oleh *admin.* Proses berakhir ketika pengguna melakukan *Logout.*

3.5 Perancangan Antarmuka



Gambar 3. 4. Perancangan antarmuka

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem pada perangkat keras dan perangkat lunak dari perancangan sistem yang telah dibahas pada Bab 3.

4.1 Lingkungan Implementasi

Dalam merancang perangkat lunak ini digunakan perangkat pendukung sebagai berikut.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan aplikasi adalah laptop, Spesifikasi dari perangkat-perangkat tersebut adalah sebagai berikut:

- Laptop Asus intel core-i3 processor T3400(2.16 GHz, 667 MHz FSB, 1 MB L2 cache)

4.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat spesifikasi perangkat lunak yang digunakan dalam pengembangan sistem adalah sebagai berikut:

- Linux Ubuntu 14.04 sebagai sistem operasi laptop.
- Meteor js sebagai *framework.*
- Atom sebagai *javaScript* editing.
- *Mozilla Firefox* dan *Chromium* sebagai web browser.
- *JDK* dan Android Studio untuk pengembangan pada *aplikasi mobile*.

4.2 Implementasi Proses

Pada sub bab ini terdiri dari penjelasan tentang proses yang terjadi pada pembuatan *project* ini meliputi persiapan aplikasi dan perancangan sistem aplikasi.

4.2.1 Implementasi persiapan aplikasi

Untuk implementasi persiapan aplikasi, tahap awal yang dilakukan adalah installasi Meteor. Meteor bisa dijalankan pada *sistem* operasi OS X, *Windows* dan Linux. Untuk instalasi pada *Windows* dapat di download langsung di *https://www.meteor.com/* dan pada OS X atau Linux dapat di install versi terbaru dari Meteor dengan melakukan perintah pada terminal:

```
curl https://install.meteor.com/ | sh
```

Karena merupakan sebuah *Framework*, Meteor tidak menjadi aplikasi setelah di install sehingga untuk menggunakan *framework* ini cukup dengan perintah-perintah melalui *Command* Line atau *cmd* pada Windows dan terminal pada *Ubuntu*. Namun *source* dari aplikasi yang telah dibuat akan otomatis masuk pada folder *Home* pada *Linux* dan pada folder *AppData* pada *Windows.* Langkah selanjutnya adalah membuat *project* baru pada Meteor dengan perintah:

```
Meteor create "(nama aplikasi)"
```

Pada project ini, nama yang digunakan adalah *appChat*. Bila sudah berhasil membuat sebuah *project*, masuklah dalam *directory* dari *project* tersebut dan coba jalankan dengan memberikan perintah.

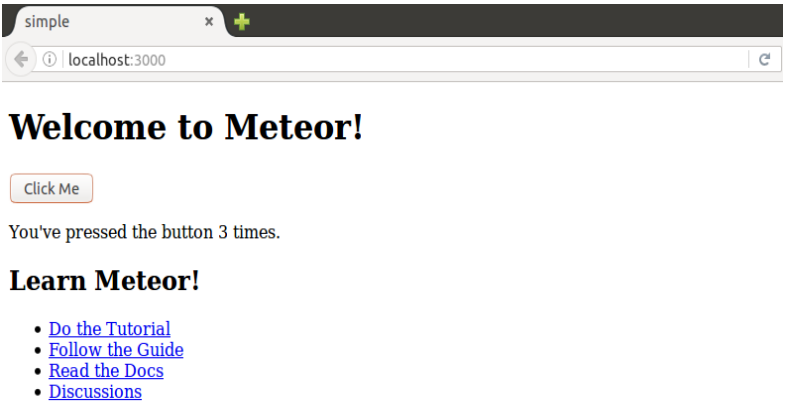
```
Cd chatDemo
meteor
```

Tunggulah sampai proses pembuatan selesai atau setelah *project* memberikan informasi berikut

```
# Meteor server running on:
http://localhost:3000/
```

Meteor memberikan demonstrasi *frameworknya* dengan menampilkan contoh kerja dari aplikasi yang dapat dibangun dengan Meteor yaitu dengan *code* yang sederhana, bisa

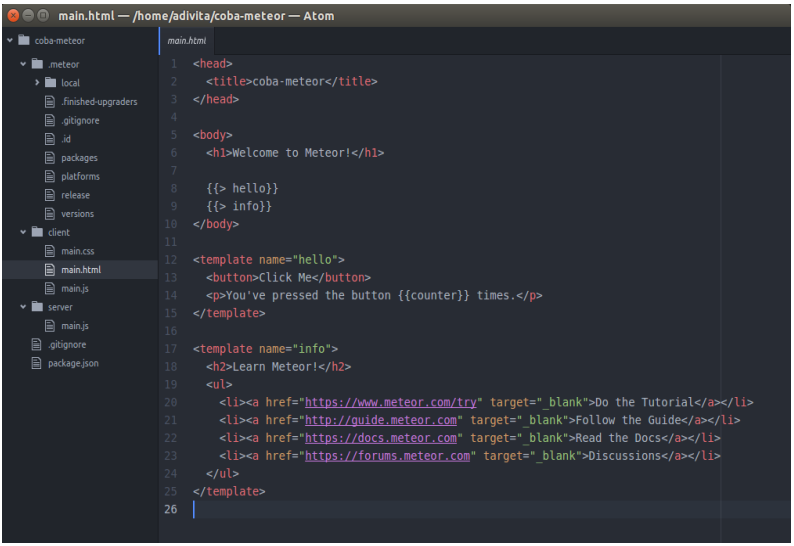
memberikan sebuah aplikasi yang cepat, responsif dan *realtime* seperti pada gambar 4.1.



Gambar 4. 1. ***Tampilan project baru pada meteor***

Dalam tampilan project baru tersebut berisi informasi-informasi dasar dan *link* yang berkaitan dengan Meteor. Selain itu tersedia juga tombol yang akan melakukan perhitungan otomatis setiap pengguna menekan tombol tersebut. Fungsi ini memperlihatkan kemampuan responsif dan *realtime* dari Meteor.

Pada gambar 4.2 dibawah ini merupakan *source default* dari aplikasi Meteor yang telah dibangun. Folder meteor berisi informasi-informasi dari *package* sampai informasi versi dari meteor yang digunakan. Pada folder *client* terdiri dari file, css, html dan js yang berisi *code* yang digunakan untuk mengatur sistem pada sisi *client* dan tampilan pada aplikasi. Folder *server* diperlukan untuk pengaturan sistem pada sisi *server*.



Gambar 4. 2. *Tampilan source project*

Penambahan *module* atau *package* ke dalam *project* agar lebih memudahkan dalam pengembangan aplikasi *Chat* dengan perintah *"add"* dan menghapus *package* yang tidak digunakan dengan perintah *"remove"*. Pada aplikasi ini, digunakan *module* atau *package* tambahan dari Meteoris seperti yang ditunjukkan Tabel 4.1.

Tabel 4.1 Daftar module atau package yang digunakan dan tidak digunakan

meteor add	Meteor remove
meteoris:core	Insecure
meteoris:theme-admin	Autopublish
meteoris:user	
Meteoris:Role	

4.2.2 Implementasi Perancangan Aplikasi

Pada subbab ini menjelaskan tentang perancangan aplikasi *room chat* yang dikembangkan dengan *framework* Meteor.

Untuk tampilan navigasi bar pada aplikasi dibuat *file* pada *appChat/client/hook/navbar.html*.

```
<template
name="meteoris_themeAdmin_hookNavbar">
    <div class="navbar-custom-menu">
        <ul class="nav navbar-nav">
            {{#if currentUser}}
            <!-- User Account: style can be found in
dropdown.less -->
            <li class="dropdown user user-menu">
                <a href="#" class="dropdown-toggle" data-
toggle="dropdown">
                    <img          src="/images/user.png"
class="user-image" alt="User Image">
                    <span                  class="hidden-
xs">{{currentUser.profile.name}}</span>
                </a>
                <ul class="dropdown-menu">
                    <!-- User image -->
                    <li class="user-header">
                        <img      src="/images/user.png"
class="img-circle" alt="User Image">
                        <p>
                            {{currentUser.profile.name}}
                        </p>
                    </li>
                    <!-- Menu Footer-->
                    <li class="user-footer">
                        <div class="pull-left">
                            <a  href="#"  class="btn  btn-
default btn-flat">Profile</a>
                        </div>
                        <div class="pull-right">
                            <a  href="#"  id="btnLogout"
class="btn btn-default btn-flat">Logout</a>
                        </div>
                    </li>
                </ul>
```

```
            </li>
            {{else}}
            <li>
                <a href="/meteoris/user/login">Login</a>
            </li>
            {{/if}}
        </ul>
    </div>
</template>
```

Gambar 4. 3. ***Pseudocode navbar.html***

Untuk fungsi-fungsi pada navigasi bar dibuat *file* pada *appChat/client/hook/navbar.js*

```
var ctrl = new Meteoris.UserController();


Template.meteoris_themeAdmin_hookNavbar.events = {
    'click #btnLogout': function(){
        ctrl.logout();
    }
};
```

Gambar 4. 4. ***Pseudocode navbar.js***

Pada *sidebar* terdapat beberapa informasi dan tambahan fitur khususnya untuk admin sehingga dibuatkan *file* pada *appChat/client/hook/sidbar.html.*

```
<template name="meteoris_themeAdmin_hookSidebar">
    <!-- Left side column. contains the logo and
sidebar -->
    <aside class="main-sidebar">
        <!-- sidebar: style can be found in
sidebar.less -->
        <section class="sidebar">
            {{#if currentUser}}
            <!-- Sidebar user panel -->
            <div class="user-panel">
                <div class="pull-left image">
                    <img       src="/images/user.png"
class="img-circle" alt="User Image">
```

```
                </div>
                <div class="pull-left info">

<p>{{currentUser.profile.name}}</p>
                    <a href="#"><i class="fa fa-circle
text-success"></i> Online</a>
                </div>
            </div>
            {{/if}}

            <ul class="sidebar-menu">
                <li                class="header">MAIN
NAVIGATION</li>
                <li><a   href="/"><i   class="fa   fa-
home"></i> Home</a></li>
                <!--Uncomment this if you want to hide
this menu using the power of meteoris:role-->
                <!--{{#if   meteoris_roleUserIsInGroup
"admin"}}-->
                <li class="header">ADMIN AREA</li>
                <li class="treeview">
                    <a href="#">
                        <i class="fa fa-gears"></i>
                        <i   class="fa   fa-angle-left
pull-right"></i>
                        <span>Setting</span>
                    </a>
                    <ul class="treeview-menu">
                        <li><a  href="/meteoris/theme-
admin/setting"><i   class="fa   fa-laptop"></i>   Theme
Admin Setting</a></li>
                        <li><a
href="/meteoris/user"><i    class="fa    fa-users"></i>
Users Management</a></li>
                        <li><a
href="/meteoris/user/settings"><i     class="fa     fa-
user"></i> User Settings</a></li>
                        <li><a
href="/meteoris/role"><i   class="fa   fa-flag-o"></i>
Role Management</a></li>
                    </ul>
                </li>
```

```
                <!--{{/if}}-->
            </ul>
        </section>
        <!-- /.sidebar -->
    </aside>

</template>

SIDEBAR.HTML
<template name="meteoris_themeAdmin_hookSidebar">
    <!-- Left side column. contains the logo and
sidebar -->
    <aside class="main-sidebar">
        <!-- sidebar: style can be found in
sidebar.less -->
        <section class="sidebar">
            {{#if currentUser}}
```

Gambar 4. 5. ***Pseudocode sidebar.html***

Pembuatan fungsi *Router* untuk halaman utama pada aplikasi. *appChat/adivita.chat/lib/router.js*

```
//routing as home page (/), rendering
meteoris_themeAdminMain as the template and
adivita_chatIndex as the content
FlowRouter.route('/', {
    action: function() {
        BlazeLayout.render('meteoris_themeAdminMain',
{content:"adivita_chatIndex"});
    }
});
```

Gambar 4. 6. ***Pseudocode router.js***

Untuk pembuatan *basic views file* pada halaman utama, dibutuhkan *html file* pada *module* adivita.chat dan didalam folder *client/views* dengan nama *index.html* dan *template* yang akan digunakan adalah *adivita_chatIndex* seperti pada **gambar 4.5**.

```
<template name="adivita_chatIndex">
    <div class="content">
```

```
        <div class="box box-danger direct-chat direct-
chat-danger">
            <div class="box-header with-border">
                <h3 class="box-title">Room Chat</h3>
            </div><!-- /.box-header -->
            <div class="box-body">
                <!-- Conversations are loaded here -->
                <div class="direct-chat-messages">
                    <!--Show this message if models/chats
still empty-->
                    {{#if isEmpty}}
                    <div     class="alert     alert-info"
role="alert">
                        Selamat   Datang   &   Mulailah
percakapan dengan menulis pesan pada kolom di bawah... :)
                    </div>
                    <!--Else show chats list and loop it
through-->
                    {{else}}
                    {{#each models}}
                    <div           class="direct-chat-msg
{{messageClass}}"> <!-- Use messageClass to know whether
it right style or left style -->
                        <div      class="direct-chat-info
clearfix">
                            <span     class="direct-chat-
name pull-left">{{user.profile.name}}</span> <!-- belongs
to relationship from Collections helper we already make
before -->
                            <span     class="direct-chat-
timestamp   pull-right">{{meteoris_formatter   'dateTime'
createdAt}}</span>   <!--   using   meteoris_formatter   to
easily format date -->
                        </div><!-- /.direct-chat-info -->
                        <img      class="direct-chat-img"
src="images/user.png"   alt="message   user   image"><!--
/.direct-chat-img -->
                        <div class="direct-chat-text">
                            {{message}}  <!--  show  the
message -->
                        </div><!-- /.direct-chat-text -->
                    </div><!-- /.direct-chat-msg -->
```

```
                    {{/each}}
                    {{/if}}

                    <!--Show     spinner/loading     when
template subscriptions is not ready yet-->
                    {{#unless
Template.subscriptionsReady}}
                    {{> spinner}}
                    {{/unless}}

                </div><!--/.direct-chat-messages-->
            </div><!-- /.box-body -->
            <div class="box-footer">
                <form>
                    <div class="input-group">
                        <!-- add id message to form input
-->
                        <input  id="message"  type="text"
placeholder="Type Message ..." class="form-control">
                        <span class="input-group-btn">
                            <!-- add id btnSend to Send
Button -->
                            <button          id="btnSend"
type="submit"       class="btn       btn-danger       btn-
flat">Send</button>
                        </span>
                    </div>
                </form>
            </div><!-- /.box-footer-->
        </div><!--/.direct-chat -->
    </div>
</template>
```

Gambar 4. 7. *Pseudocode index.html*

Selanjutnya adalah pembuatan *collection* dan *server publish.* File yang dibuat adalah C*hat.js* dan diletakkan di *appChat/adivita.chat/lib/colllection/Chat.js*

```
<template name="adivita_chatIndex">
    <div class="content">
```

```
        <div class="box">
            <div class="box-body">
                <h1>Welcome to Meteoris Chat</h1>
            </div>
        </div>
    </div>
</template>
```

Gambar 4. 8. *Pseudocode chat.*js

Untuk pengaturan pada sisi server diperlukan *create file ChatServer.js* yang berfungsi melakukan publikasi *collection* ke *client*. File yang dibuat disimpan di *appChat/adivita.chat/server/ChatServer.js.*

```
Meteor.publishComposite('adivita_chat',     function(doc,
sort) {
    console.log("subscribing   some   Chat   with   it's
relation");
    var doc = doc || {};
    var sort = sort || {};
    return{
        find: function() {
            return Adivita.Chat.find(doc, sort);
        },
        children: [
            /* return all related Users */
            {
                find: function(collection) {
                    return
Meteor.users.find(collection.userId);
                }
            },
        ],
    }
});
```

Gambar 4. 9. *Pseudocode server.js*

Diluar dari sisi *client* dan *server* diberikan *file model js.* pada *model.js* ini digunakan untuk membuat fungsi *Meteor collection* yaitu penyimpanan pada *database* yang memanfaatkan *MongoDB.*

```
ChatRooms = new
Meteor.Collection("chatrooms");
```

Gambar 4. 10. ***Pseudocode model.js***

Setelah memberikan pengaturan pada *server*, langkah selanjutnya adalah membuat *controller* untuk memberikan metode kontrol pada aplikasi. Pembuatan file *ChatController* pada *appChat/adivita.chat/lib/controllers/ChatController.js*

```
Namespace('Adivita.ChatController');

/**
 * our controller extends Meteoris.Controller to get the
inheritance method
 * from Meteoris.Controller which is very useful, and
reusable
 */
Adivita.ChatController = Meteoris.Controller.extend({
    /*
     * for now we are Hard coded the channel id to general
     * you can change this later by Flow Router uri/query
parameter or Session
     * as you wish
     */
    channelId: "general",
    /*
     * passing data to index helpers template
     * because our template has index suffix
     */
    index: function() {
        var models = this.getAll();
        return {
            isEmpty: models.count() === 0 ? true : false,
            models: models,
        };
    },
    /* action getAll data from Chat collection */
    getAll: function() {
        return      Adivita.Chat.find(this.getCriteria(),
this.getSortLimit());
    },
```

```
    /* get sortLimit for limit & sorting collection */
    getSortLimit: function() {
        //sort by createdAt ascending
        var sort = {
            sort: {
                'createdAt': 1
            }
        };
        //set limit message to 100 (hardcoded for now)
        sort.limit = 100;
        return sort;
    },
    /*
     * get criteria for searching collection
     * we are giving channelId as general (hardcoded)
     */
    getCriteria: function() {
        var criteria = {channelId: this.channelId};
        return criteria;
    },
    /* private get user input docs */
    _getDoc: function(t) {
        return {
            channelId: this.channelId, //insert channelId
from hardcoded value
            message:   t.find('#message').value,   //find
message value from t/from user input
        };
    },
    /* action inserting data/sending message in this case
*/
    post: function(t) {
        var doc = this._getDoc(t);

        Adivita.Chat.insert(doc, function(err, _id) {
            if (err) {
                Meteoris.Flash.set('danger',
err.message);
                throw new Meteor.Error(err);
            }
            //using  meteoris  Flash  to  show  Toast/Flash
message
```

```
            Meteoris.Flash.set('success',        "Message
Sent!");

            //empty the value of message text
            t.find('#message').value = "";
        });
    },
});
```

Gambar 4. 11. ***Pseudocode ChatController.js***

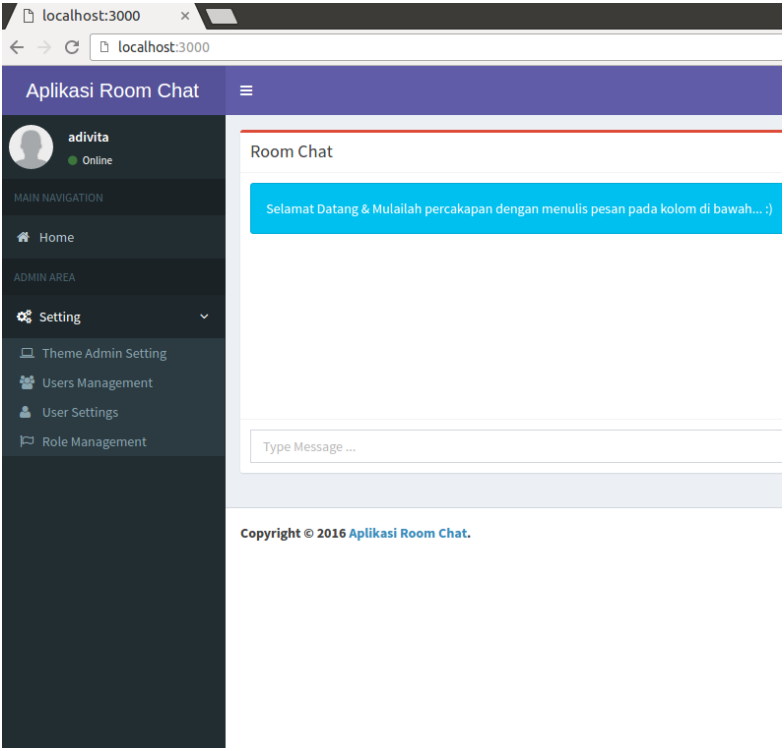
Langkah terakhir yang dilakukan adalah pembuatan file *index* pada *client.* File yang dibuat diletakkan pada *appChat/adivita.chat/client/views/index.js*

```
Meteor.publishComposite('adivita_chat', function(doc,
sort) {
    console.log("subscribing  some  Chat  with  it's
relation");
    var doc = doc || {};
    var sort = sort || {};
    return{
        find: function() {
            return Adivita.Chat.find(doc, sort);
        },
        children: [
            /* return all related Users */
            {
                find: function(collection) {
                    return
Meteor.users.find(collection.userId);
                }
            },
        ],
    }
});
```

Gambar 4. 12. ***Pseudocode*** **index*.js***

Untuk menghindari kesalahan pada penulisan fungsi-fungsi atau tampilan pada aplikasi, proses pembuatan file dilakukan dengan status aplikasi berjalan pada *localhost.* Sehingga

bila terjadi error akan ada warning pada jendela *terminal*. Bila tidak terjadi error, maka server akan melakukan *restart* aplikasi dengan otomatis. Tampilan awal aplikasi *room chat* yang dibangun bisa dilihat pada gambar 4.13.



Gambar 4. 13. *Tampilan home aplikasi*

4.2.3. Implementasi deploy aplikasi

Pada subbab ini menjelaskan tentang proses *deploy* dari aplikasi yang dibangun menggunakan Meteor. Sejak awal kemunculannya, *framework Meteor.js* ini memberikan sarana untuk mendeploy aplikasi yang dibangun di berbagai *platform* baik *desktop* yang berjalan melalui *browser* maupun aplikasi *mobile*.

Untuk membangun aplikasi pada perangkat *mobile*, dibutuhkan penambahan *platform* pada meteor dan melakukan konfigurasi pada perangkat. Pengembangan pada *mobile* memanfaatkan integrasi *meteor* dengan *cordova*. Perintah untuk menambah platform ios atau android dapat dilakukan sperti pada **gambar 4.14** dibawah ini.

```
meteor add-platform ios

Meteor add-platform android

meteor remove-platform ios android
```

Gambar 4. 14. ***Kodesumber menambah dan menghapus platform ios dan android***

```
meteor list-platform ios android
```

Gambar 4. 15. ***Kodesumber perintah pengecekan list platform yang terintegrasi dengan project***

Langkah selanjutnya untuk membangun aplikasi *mobile* pada *Meteor* adalah melakukan instalasi dan konfigurasi pada perangkat atau *local machine*.

Untuk pengembangan aplikasi yang berjalan pada *platform* ios, membutuhkan perangkat yang berjalan pada OS X atau *Mac* yang telah terinstall *Xcode* versi 7.2 atau lebih. Namun bila akan

mengembangkan dan menjalankan aplikasi pada android, harus melakukan instalasi atau menyiapkan beberapa syarat sebagai berikut:

- Melakukan instalasi *Java Development Kit* (JDK)

```
• $ sudo add-apt-repository
  ppa:webupd8team/java
• $ sudo apt-get update
• $ sudo apt-get install oracle-java8-
  installer
```

Gambar 4. 16. *Kodesumber* instalasi *Java* 8 pada *Linux*

- Melakukan instalasi *Java development Kit* (JDK) dan *Android Studio* menggunakan *Ubuntu Make*.

```
$ sudo add-apt-repository ppa:ubuntu-
desktop/ubuntu-make

$ sudo apt-get update

$ sudo apt-get install ubuntu-make

$ umake android
```

Gambar 4. 17. *Kodesumber* instalasi *Ubuntu Make*

- Melakukan konfigurasi *ANDROID_HOME* dan menambah *tools directories* di dalam PATH.

```
#Android
export
ANDROID_HOME="/Users/<username>/Library/Android/s
dk"
Export
PATH=$PATH:$ANDROID_HOME/tools:$ANDROID_HOME/plat
form-tools
```

Gambar 4. 18. *Kodesumber* pengaturan Android Home

- Langkah selanjutnya adalah optional (pilihan) yaitu membuat *Android Virtual Device* (AVD) untuk menjalankan aplikasi pada emulator.

Cara untuk menjalankan aplikasi pada perangkat Android adalah dengan menghubungkan perangkat menggunakan kabel *USB.* Setelah itu pastikan terhubung dengan jaringan internet agar mudah melakukan komunikasi dengan server. Pada perangkat Android dilakukan konfigurasi "*Allow USB debugging*" dan yang terakhir adalah membangun aplikasi dengan pemanggilan memberikan perintah pada terminal seperti gambar.

```
meteor run android-device
```

Gambar 4. 19. *Kodesumber run* aplikasi pada android

Untuk memeriksa apakah perangkat sudah terhubung dan terkonfigurasi dengan benar, jalankan perintah seperti gambar dibawah untuk mendapatkan daftar dari perngkat-perangkat.

```
adb devices
```

Gambar 4. 20. *Kodesumber* melihat daftar perangkat yang terhubung

Meteor memberikan kemudahan bagi pengembang aplikasi *desktop* dengan menyediakan server untuk melakukan *hosting* atau *deploy* aplikasi pribadi secara gratis. Namun promosi tersebut berakhir pada tanggal 25 Maret 2016 bertepatan dengan perilisan Meteor secara resmi dan peluncuran versi terbaru dari Meteor sehingga untuk melakukan *deploy* pada server milik meteor diharuskan untuk membayar biaya sewa. Selain pada server Meteor, *hosting project* yang mendukung *Meteor* juga tersedia seperti pada *Heroku, AWS, Google Cloud, Digital Ocean* dan lain-lain. Untuk *mendeploy* aplikasi chat ini, digunakan server dari *Scalingo.* Berikut adalah langkah-langkah untuk melakukan *deploy* aplikasi dari *Meteor* menggunakan *Scalingo*.

- Melakukan instalasi *Scalingo*. Bagi pengguna *Linux* & *MacOS X* dapat melakukan download dan instalasi melalui terminal, namun pada *Windows* dapat di download di situs resmi *Scalingo*.

```
curl -O https://cli-dl.scalingo.io/install &&
bash install
```

Gambar 4. 21. *Kodesumber instalasi scalingo pada Linux*

- Melakukan instalasi *Git*

```
apt-get install git
```

Gambar 4. 22. *Kodesumber instalasi git pada Linux*

- Membuat *SSH Key*. "Laptop" merupakan nama *SSH* yang di tambahkan

```
scalingo keys-add Laptop ~/.ssh/id_rsa.pub
```

Gambar 4. 23. *Kodesumber Menambah SSH Key*

- Pada direktori *project* yang telah di buat sebelumnya. Tambahkan file pada *Git Repository*.

```
git init .
```

```
git commit -m "Init meteor application"
```

Gambar 4. 24. ***Kodesumber Menambah project meteor ke dalam git repository***

- Membuat project *Scalingo* atau melakukan eksport project *Meteor* pada server *Scalingo.*

```
scalingo create chat-app
Git repository detected: remote scalingo added
→ 'git push scalingo master' to deploy your app
```

Gambar 4. 25. ***Kodesumber Membuat Project Scalingo***

- Membuat ketentuan dari database *MongoDB. Karena MongoDB* sudah menjadi pusat data dari *Meteor*, maka perlu di tambahkan atau didaftarkan ketersedian data pada project *Scalingo* .

```
scalingo -a chat-app addons-add scalingo-mongodb
free
-----> Addon scalingo-mongodb has been pro
Modified          variables:          [MONGO_URL
SCALINGO_MONGO_URL]
       Message  from  addon  provider:  Database
successfully created
```

Gambar 4. 26. ***Kodesumber Menambah addons Scalingo***

- Deploy aplikasi

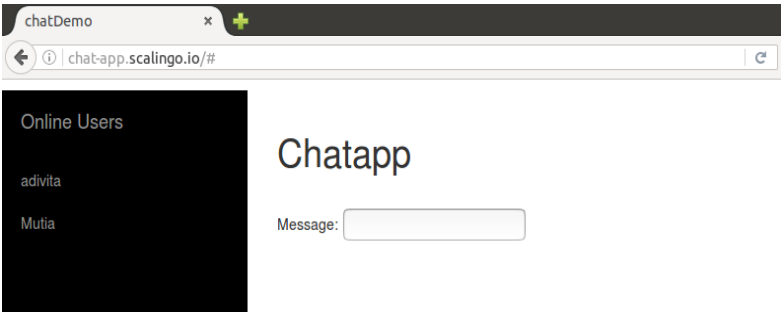
```
git push scalingo master
```

Gambar 4. 27. ***Kodesumber Perintah melakukan deploy***

- Setelah Scalingo selesai membuat project, aplikasi dapat digunakan pada alamat yang di berikan *Scalingo*.

```
…
Waiting for your application to boot...
<-- https://chat-app.scalingo.io -->
```

Gambar 4. 28. ***Tampilan pada terminal ketika aplikasi sudah di deploy***



Gambar 4. 29. ***Tampilan aplikasi setelah di deploy***

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN ANALISA

Pada bab ini akan membahas uji coba dan evaluasi dari aplikasi yang dibuat. Aplikasi akan diuji coba fungsionalitas dan performa dengan menjalakan skenario yang sudah ditentukan. Uji coba dilakukan untuk mengetahui hasil dari aplikasi ini sehingga dapat menjawab rumusan masalah pada Bab I.

5.1 Lingkungan Uji Coba

Pada sub bab ini dijelaskan mengenai gambaran lingkungan yang digunakan untuk melakukan uji coba aplikasi. Karena aplikasi *room chat* ini bekerja pada jaringan *local* pada perangkat, uji coba aplikasi ini juga dilakukan dengan menggunakan laptop yang digunakan untuk membangun apliasi dengan spesifikasinya sebagai berikut:

- Laptop Asus K43SJ
  - Spesifikasi perangkat keras
    - Intel® Core™ i3-2330M CPU @ 2.20GHz (4 CPUs), ~2.2GHZ
    - RAM 6144 MB
  - Spesifikasi perangkat lunak
    - Linux Ubuntu 14.04 64-bit sebagai sistem operasi yang digunakan
    - Mozila Firefox sebagai *browser* yang digunakan
    - Chromonium sebagai *browser* tambahan

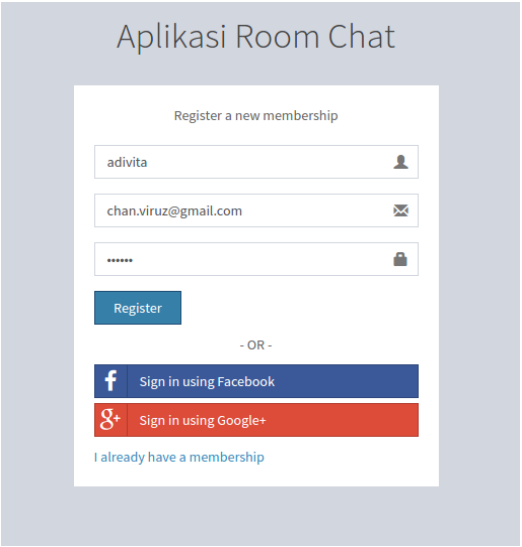
5.2 Skenario Uji Coba

Uji coba ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasi pada tahap kebutuhan benar-benar diimplementasikan dan bekerja seperti yang seharusnya.

5.2.1 Uji Coba Fungsionalitas

Pada uji coba fungsionalitas akan diuji bagaimana aplikasi dapat berjalan dengan baik dan dapat digunakan pada *web browser* berbeda namun jaringan *local*. Hasil uji coba ditunjukkan dengan *screenshot* dari berbagai pengguna termasuk admin pada *browser* berbeda.

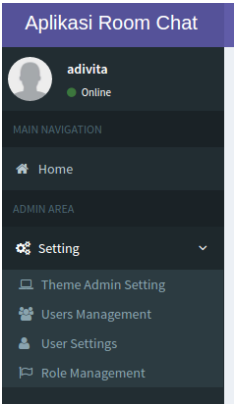
Tahap awal diuji dengan melakukan registrasi. Untuk pengguna pertama yang melakukan registrasi akan otomatis menjadi *admin*. Tampilan registrasi awal dapat dilhat pada gambar 5.1.



Gambar 5.1 Tampilan Halaman Registrasi

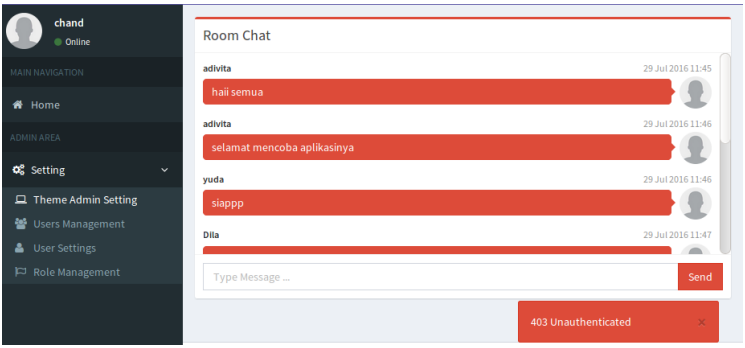
Pada *Form* registrasi di atas, data yang dibutuhkan untuk mendaftar adalah *username*, *email* dan *password.* Karena aplikasi berjalan pada jaringan *local*, fungsi registrasi menggunakan *Facebook* dan *Google+* tidak dapat digunakan. Setelah melakukan registrasi, pengguna akan langsung masuk kedalam aplikasi tanpa menunggu persetujuan *Admin*.

Admin dapat melakukan pengaturan aplikasi dengan masuk ke *setting*. Pada menu *setting*, *admin* dapat mengatur tampilan aplikasi dan dapat melihat data dari pengguna yang telah melakukan registrasi pada aplikasi.



Gambar 5. 2. Tampilan *sidebar Admin*

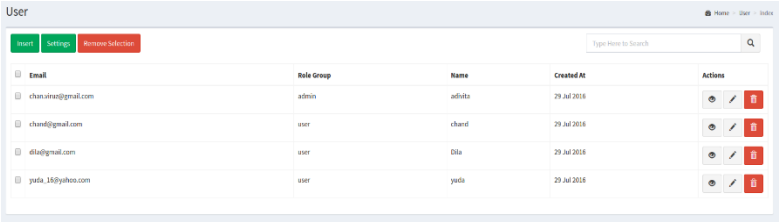
Hanya *admin* yang memiliki akses pada *setting*, sehingga pengguna biasa hanya dapat melakukan chat pada room general pada menu utama. Apabila pengguna memilih menu *setting*, akan keluar pemberitahuan akses ditolak sperti pada gambar 5.3 dibawah.



Gambar 5. 3. Tampilan pengguna biasa mencoba masuk menu *setting*

Gambar 5. 4. Tampilan *Admin* mengatur *UI*

Pada gambar 5.4 diatas, *admin* dapat mengganti tulisan yang terdapat pada tampilan halaman utama aplikasi yaitu tulisan nama aplikasi yang terletak di bagian atas *sidebar* hingga informasi mengenai aplikasi pada bagian bawah *form chat. Admin* juga disediakan beberapa pilihan warna atau *skin* aplikasi agar tampilan aplikasi lebih menarik.



Gambar 5. 5. Tampilan *Admin* melihat data *pengguna*

Pada pilihan *User Management* pada menu *setting*. *Admin* akan diberikan daftar pengguna yang telah melakukan registrasi pada aplikasi termasuk data *admin.* Data yang ditampilkan adalah alamat *email*, *role* dalam aplikasi, *username* yang digunakan serta waktu melakukan registrasi.

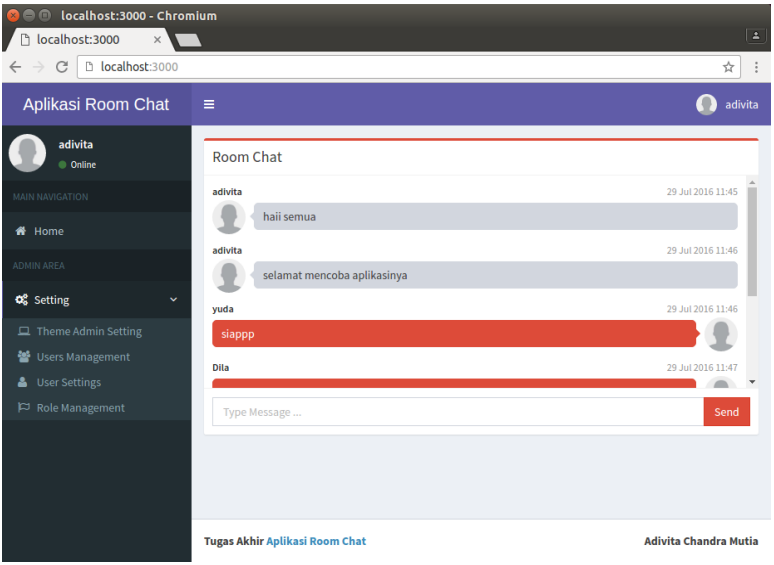
Pada aplikasi juga diberikan notifikasi yang memudahkan *Admin* dan pengguna bila transaksi proses yang dilakukan berhasil atau tidak.

Tabel 5.1 Informasi Pengguna

Pengguna	Role
Adivita	Admin
Chand	User
Dila	User
Yuda	User

Selanjutnya adalah uji coba fungsionalitas utama dari aplikasi yaitu melakukan *chat* dengan pengguna online lain dalam room *chat*. Tabel 5.1 merupakan informasi dari data pengguna yang telah melakukan registrasi dan akan melakukan uji coba pada aplikasi.

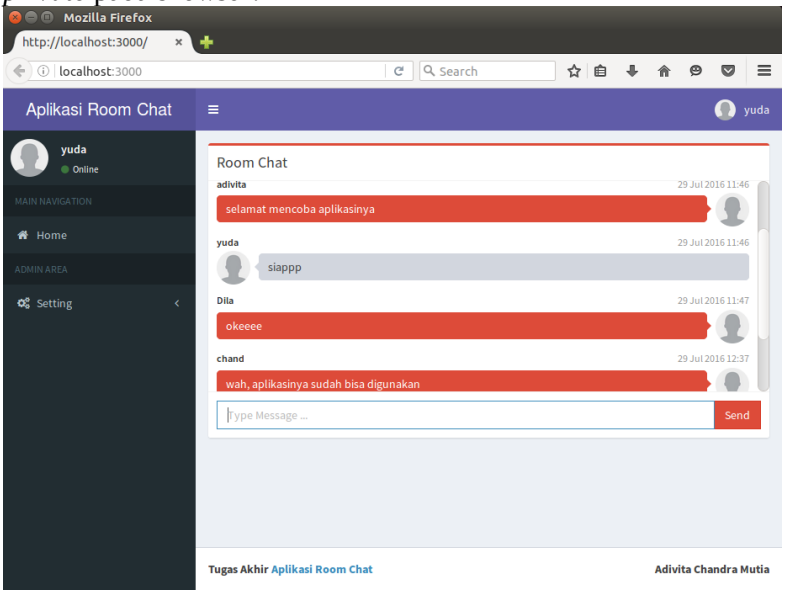
Gambar 5.6 merupakan tampilan dari pengguna Adivita yang menjadi *admin* pada aplikasi. Tsmpilan dari *chat* dibawah adalah *admin* Adivita mengirim pesan yang menjadi awal dari percapakan dengan pengguna yang lainnya.



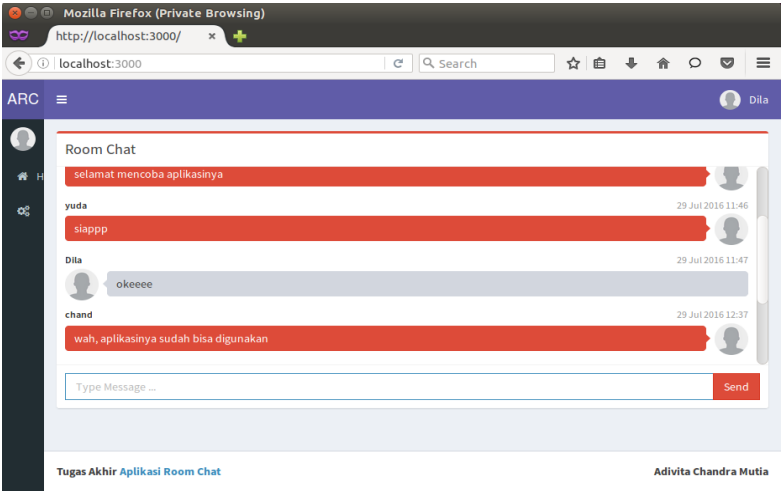
Gambar 5. 6. Tampilan pengguna Adivita

Bila pengguna mengirim pesan, maka nama dan foto pengguna terdapat di bagian kiri dari form *chat.* Warna *box* dari pesan pengguna yang mengirim dan penerima berbeda, dimana pengirim pesan akan berwarna abu-abu sedangkan bila menerima pesan akan berwanra *orange* dan aka nada data pengirim baik nama dan foto pada bagian kanan form *chat*. Keterangan waktu juga di tampilkan yang selalu terletak pada bagian kanan dari *form chat* dan berada di atas *box chat* dari pengguna.

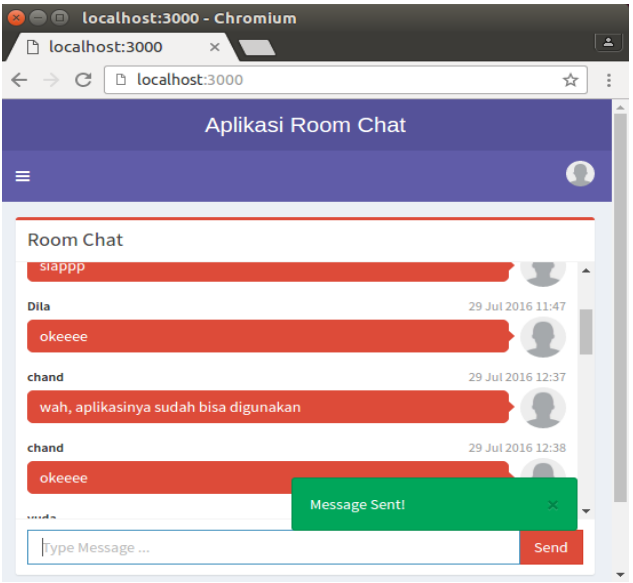
Pada gambar 5.7 dibawah menampilan *screenshot* dari pengguna Yuda yang pertama kali membalas *chat* dari Adivita. Perbedaan dari penerima dan pengirim pesan akan terlihat karena pengguna Yuda yang mengirim 1 pesan dan berada di antara pengguna yang lain. Begitupula pada gambar 5.7 dan gambar 5.8 yang menampilkan *screenshot* dari pengguna Dila dan Chand. Karena aplikasi berjalan pada jaringan *local*, pengujian dilakukan menggunakan *browser* yang berbeda dan juga menggunakan mode *private* pada *browser*.



Gambar 5. 7. Tampilan pengguna Yuda



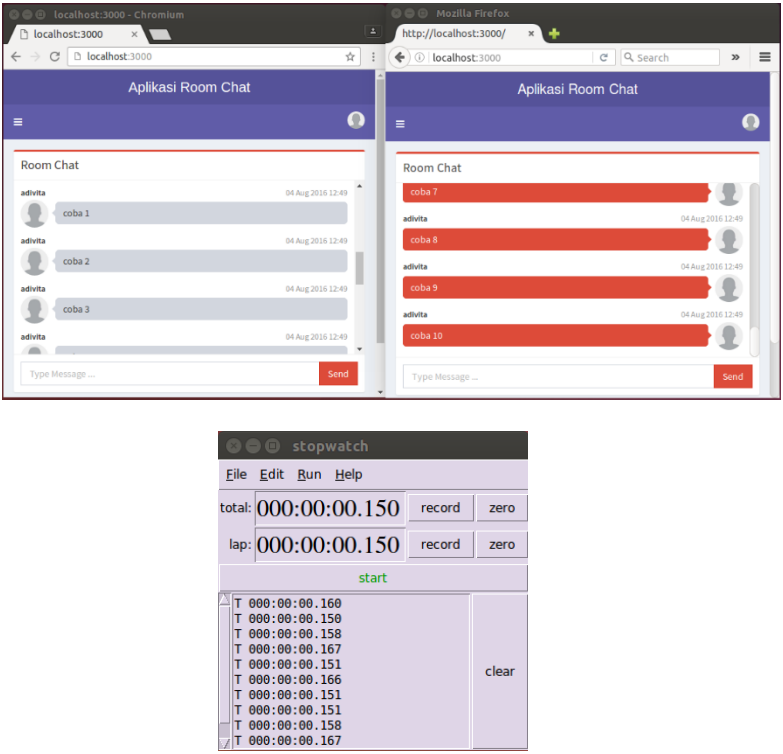
Gambar 5. 8. Tampilan pengguna Dila



Gambar 5. 9. Tampilan pengguna Chand

5.2.2 Uji Coba Performa

Pada bagian ini dilakukan uji coba performa untuk mengetahui kecepatan dari aplikasi chat yang dikembangkan menggunakan *Meteor*. Uji coba ini dilakukan dengan bantuan pengukur waktu atau *Stopwatch.* Uji coba dilakukan untuk mengetahui waktu yang diperlukan dari setiap pesan yang di kirim. Uji coba dilakukan dengan cara menghitung kecepatan pesan pengirim masuk ke dalam sistem sehingga bisa langsung tampil pada *form chat.*



Gambar 5. 10. Tampilan Uji Coba Performa

Tabel 5. 2. Hasil uji coba penghitungan waktu pengiriman pesan

uji coba ke-	Durasi yang dibutuhkan(detik)
1	0.16
2	0.15
3	0.15
4	0.15
5	0.16
6	0.15
7	0.16
8	0.15
9	0.15
10	0.16
Rata-rata	0.0154

Uji coba yang dilakukan adalah mengirim pesan ke pengguna yang lain dan menghitung waktu ketika pesan masuk dan diterima pengguna lain. Percobaan penghitungan waktu dilakukan 10 percobaan. Berdasarkan percobaan ini diketahui kecepatan rata-rata dari pengiriman pesan untuk masuk kedalam sistem sangat cepat yaitu 0.0154 detik.

[Halaman ini sengaja dikosongkan]

BAB VI
PENUTUP

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil penerapan *framework* yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

5.3 Kesimpulan

Dari hasil penerapan *framework Multiplatform* Meteor js dalam pembuatan aplikasi chat, dapat diambil kesimpulan sebagai berikut:

1. Aplikasi *room chat* berhasil dibuat menggunakan *framework Meteor js* yang berjalan pada jaringan *local*.
2. Pemanfaatan *Meteor js* sudah berhasil dilakukan dengan membuat aplikasi berjalan responsif.
3. Pemanfaatan *NoSQL database* sudah berhasil dilakukan dalam penyimpanan data pengguna dan *history* percakapan.

5.4 Saran

Adapun saran yang ingin disampaikan penulis untuk pengembangan lebih lanjut dari tugas akhir antara lain:

1. Aplikasi *chat* yang dapat melakukan percakapan pribadi antar pengguna yang online.
2. *Auto Logout* dari aplikasi bila pengguna *stanby* dalam waktu yang ditentukan.
3. Tersedia notifikasi untuk pesan yang belum dibaca.
4. Tersedia fitur untuk mengirim file selain text.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] "Pengertian Chat adalah," [Online] Available: http://adaadalah.blogspot.co.id/2016/01/pengertian-chat-adalah-chatting.html [Diakses 01 Agustus 2016].

[2] “Pengertian Framework,” [Online]. Available: https://rahmadikhsan.wordpress.com/2012/03/11/penjelasan-arti-dari-framework-fungsi-serta-kegunaannya-dalam-programming/ menurut-ahli.html [Diakses 05 Maret 2016].

[3] “Wikipedia,” [Online]. Available: https://id.wikipedia.org/wiki/JavaScript [Diakses 05 Maret 2016].

[4] "Sejarah dan Perkembangan Versi JavaScript," [Online] Available: http://www.duniailkom.com/tutorial-belajar-javascript-sejarah-dan-perkembangan-versi-javascript/ [Diakses 01 Agustus 2016].

5] “Tom Coleman dan Sacha Greif,” [Online]. Available: http://id.discovermeteor.com/chapters/introduction/ . [Diakses 05 Maret 2016].

[6] “Wikipedia” [Online]. Available: https://id.wikipedia.org/wiki/MongoDB [Diakses 05 maret 2016].

[7] “Pengertian framework dalam pemrograman,” [Online]. Available: http://www.dcc-dp.org/berita498-pengertian-framework-dalam-pemrograman.html [Diakses 18 juni 2016].

BIODATA PENULIS

Adivita Chandra Mutia, lahir di Sumbawa Besar, pada tanggal 26 April 1991. Penulis menempuh pendidikan mulai dari SDN 14 Sumbawa (1997-2003), SMPN 1 Sumbawa (2003-2006), SMAN 1 Sumbawa (2006-2009) dan selanjutnya pada akhir tahun 2009 penulis melanjutkan pendidikan sarjana di jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya (2009-2016). Banyak hal yang penulis peroleh saat menempuh bangku pendidikan terutama pada saat pengerjaan tugas Akhir ini.

Selama masa kuliah, penulis mengambil bidang minat Management Informasi (MI). Berkaitan dengan tugas akhir ini, komunikasi dengan penulis dapat melalui email: chan.viruz@gmail.com**.**