



**RANCANG BANGUN *APPLICATION PROGRAMMING INTERFACE*
(API) MENGGUNAKAN GAYA ARSITEKTUR *REPRESENTATIONAL
STATE TRANSFER* (REST) UNTUK PENGEMBANGAN SISTEM
INFORMASI *CHATting***



KERJA PRAKTIK

**Program Studi
S1 Sistem Informasi**

**UNIVERSITAS
Dinamika**

Oleh :

SEBASTIANUS SEMBARA

17.41010.0054

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2020

**RANCANG BANGUN *APPLICATION PROGRAMMING INTERFACE*
(API) MENGGUNAKAN GAYA ARSITEKTUR *REPRESENTATIONAL
STATE TRANSFER (REST)* UNTUK PENGEMBANGAN SISTEM
INFORMASI *CHATting***

Diajukan sebagian salah satu syarat untuk menyelesaikan

Program Sarjana Komputer



UNIVERSITAS
Dinamika

Disusun Oleh :

Nama : Sebastianus Sembara

NIM : 17410100054

Program : S1 (Strata Satu)

Jurusan : Sistem Informasi

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2020

LEMBAR PENGESAHAN

RANCANG BANGUN *APPLICATION PROGRAMMING INTERFACE* (API) MENGGUNAKAN GAYA ARSITEKTUR *REPRESENTATIONAL* *STATE TRANSFER* (REST) UNTUK PENGEMBANGAN SISTEM INFORMASI *CHATting*

Laporan Kerja Praktik oleh

Sebastianus Sembara

NIM : 17410100054

Telah diperiksa, diuji dan disetujui



Surabaya, 20 Juli 2020

UNIVERSITAS
Dinamika

Disetujui :

Pembimbing

**I Gusti Ngurah
Alit Widana Putra**

I Gusti Ngurah Alit Widana Putra, S.T., M.Eng.
NIDN. 0805058602

Digitally signed by I Gusti Ngurah Alit
Widana Putra
DN: cn=I Gusti Ngurah Alit Widana Putra,
o=Universitas Dinamika, ou=S1 Sistem
Informasi, email=alit@dinamika.ac.id, c=ID
Date: 2020.07.28 09:11:48 +07'00'

Penyelia



Fachrul Dani Prasetya

Mengetahui,

Ketua Program Studi S1 Sistem Informasi

Anjik

Sukmaaji

Dr. Anjik Sukmaaji, S.Kom., M.Eng.
NIDN. 0731057301

Digitally signed by
Anjik Sukmaaji
Date: 2020.07.28
11:49:55 +07'00'

It's okay Rocky, you can go when you feel like it

- Patrick Star -



UNIVERSITAS
Dinamika

SURAT PERNYATAAN
PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa Universitas Dinamika, saya:

Nama : Sebastianus Sembara
NIM : 17410100054
Program Studi : SI Sistem Informasi
Fakultas : Fakultas Teknologi dan Informatika
Jenis Karya : Laporan Kerja Praktik
Judul Karya : **RANCANG BANGUN *APPLICATION PROGRAMMING INTERFACE (API)* MENGGUNAKAN GAYA ARSITEKTUR *REPRESENTATIONAL STATE TRANSFER (REST)* UNTUK PENGEMBANGAN SISTEM INFORMASI *CHATting***

Menyatakan dengan sesungguhnya bahwa:

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, saya menyetujui memberikan kepada Universitas Dinamika Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalti Free Right*) atas seluruh isi/ sebagian karya ilmiah saya tersebut di atas untuk disimpan, dialihmediakan dan dikelola dalam bentuk pangkalan data (*database*) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta
 2. Karya tersebut di atas adalah karya asli saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya atau pendapat orang lain yang ada dalam karya ilmiah ini adalah semata hanya rujukan yang dicantumkan dalam Daftar Pustaka saya
 3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiat pada karya ilmiah ini, maka saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada saya.
- Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, 17 Juli 2020

Yang menyatakan



Sebastianus Sembara
NIM: 17410100054

ABSTRAK

PT. Vascomm Solusi Teknologi adalah sebuah perusahaan yang bergerak pada bidang IT solutions dan startup factory. Pada kesempatan melakukan kerja praktik di PT. Vascomm Solusi Teknologi selama kurang lebih 2 bulan, penulis diberi sebuah *project* untuk melakukan riset tentang teknologi *Application Programming Interface* (API) dengan gaya arsitektur *Representational State Transfer* (REST) dan penggunaan *socket.io* pada pengembangan *back-end* sistem informasi *chatting*. Berdasarkan hal tersebut maka akan dilakukan rancang bangun API menggunakan gaya arsitektur REST untuk pengembangan sistem informasi *chatting*. REST adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data. Perancangan *software* berbasis API menggunakan gaya arsitektur REST akan memungkinkan sebuah *back-end* dimanfaatkan secara lebih luas karena pembuatan *logic* pada sistem akan dilakukan secara terpisah dengan *logic* pada tampilan antarmuka pengguna.

Berdasarkan hasil penelitian didapat kesimpulan bahwa, perancangan sistem menggunakan API dengan gaya arsitektur REST dapat diimplementasikan pada perancangan sistem berbasis API dengan gaya arsitektur REST. Hal ini bertujuan untuk menjadikan sebuah sistem memiliki performa yang baik, cepat dan mudah untuk di kembangkan terutama dalam pertukaran dan komunikasi data.

Kata kunci: *Application Programming Interface, Representational State Transfer, Chatting, Socket.io, back-end*

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala rahmat yang diberikan sehingga penulis dapat melaksanakan kerja praktik dan menyelesaikan pembuatan laporan dari kerja praktik ini dengan baik. Laporan ini disusun berdasarkan kerja praktik dan hasil studi yang dilakukan selama dua bulan yang dilaksanakan pada tanggal 17 Februari 2019 hingga 17 April 2019 di PT. Vascomm Solusi Teknologi.

Dalam penyelesaian laporan kerja praktik ini, tidak terlepas dari segala dukungan material maupun non material dari berbagai pihak, maka dari itu penulis ingin menyampaikan rasa terimakasih yang sebesar-besarnya kepada:

1. Orang tua ku yang selalu memberikan segala dukungan sehingga penulis dapat menyelesaikan kerja praktik dengan baik.
2. Bapak Dr. Anjik Sukmaaji, S.Kom., M.Eng, selaku ketua Program Studi S1 Sistem Informasi Universitas Dinamika yang telah memberikan izin kepada penulis untuk melakukan kerja praktik.
3. Bapak I Gusti Ngurah Alit Widana Putra, S.T., M.Eng selaku dosen pembimbing yang telah membimbing dengan sabar, mengayomi, dan memberikan arahan kepada penulis mulai dari proses administrasi dari awal hingga laporan kerja praktik ini terselesaikan.
4. Bapak Aan Setianto, selaku CEO dari PT. Vascomm Solusi Teknologi yang telah memberikan kesempatan dan memperbolehkan penulis untuk melakukan kerja praktik
5. Mas Dani, Mas Nizar, Mas Rofiq selaku pembimbing dari PT. Vascomm Solusi Teknologi yang telah membantu, membimbing, dan

memberikan kesempatan kepada penulis dalam melakukan kerja praktik di PT. Vascomm Solusi Teknologi

6. Teman teman S1 Sistem Informasi Universitas Dinamika terkhusus untuk angkatan 17 yang telah hadir, menemani, serta membantu dalam proses pengerjaan kerja praktik ini.
7. *Stackoverflow.com* , *Dev.to*, *Github.com*, penggiat *open source*, dan semua forum developer yang telah memberikan potongan – potongan kode ajaib yang sangat bermanfaat sekali dalam penyelesaian pembuatan program pada kerja praktik ini tanpa kalian aku hanyalah butiran debu.
8. Pihak-pihak lain yang tidak disebutkan satu-persatu yang telah memberikan bantuan dan dukungan kepada penulis.

Semoga Tuhan Yang Maha Esa memberikan balasan yang setimpal kepada semua pihak yang telah membantu dan memberikan bimbingan serta nasehat dalam proses Kerja Praktik ini.

Penulis menyadari bahwa kerja praktik yang dikerjakan ini masih banyak terdapat kekurangan sehingga kritik yang bersifat membangun dan saran dari semua pihak sangatlah diharapkan agar aplikasi ini dapat diperbaiki menjadi lebih baik lagi. Semoga laporan Kerja Praktik ini dapat diterima dan bermanfaat bagi penulis dan semua pihak.

Surabaya, 20 Juli 2020

Penulis

DAFTAR ISI

	Halaman
ABSTRAK	vi
KATA PENGANTAR	vii
DAFTAR ISI	ix
DAFTAR TABEL	xii
DAFTAR GAMBAR	xiii
1. BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan	3
1.5 Manfaat	3
1.6 Sistematika Penulisan	4
2. BAB II GAMBARAN UMUM PERUSAHAAN	6
2.1 Profil Perusahaan	6
2.2 Identitas Perusahaan	6
2.3 Sejarah Perusahaan	7
2.4 Logo Perusahaan dan Filosofi	8
2.5 Visi Perusahaan	9
2.6 Misi Perusahaan	9
2.7 Struktur Organisasi	10
3. BAB III LANDASAN TEORI	12
3.1 <i>Chatting</i>	12
3.2 HTTP	12
3.3 <i>Application Programming Interface (API)</i>	13

3.4	<i>Representational State Transfer (REST)</i>	13
3.5	<i>JavaScript Object Notation (JSON)</i>	14
3.6	Node JS.....	15
3.7	Express JS.....	15
3.8	Socket IO.....	16
3.9	Mongo DB.....	16
3.10	<i>JSON Web Token (JWT)</i>	16
3.11	<i>POSTMan</i>	17
4.	BAB IV DESKRIPSI PEKERJAAN	18
4.1	Analisis Sistem	18
4.1.1	Wawancara.....	18
4.1.2	Studi Literatur	18
4.1.3	Identifikasi Masalah.....	19
4.1.4	Identifikasi Aktor.....	19
4.1.5	Identifikasi Data.....	20
4.1.6	Analisis Kebutuhan Fungsional	20
4.1.7	Analisis Kebutuhan Pengguna	21
4.1.7	Analisis Kebutuhan Data	22
4.2	Perancangan Sistem	23
4.2.1	Desain Alur Proses.....	23
4.2.2	Desain Basis Data	36
4.2.3	Desain API	41
4.3	Implementasi Sistem	44
4.4	Pembahasan Sistem	44
4.4.1	Struktur API	45
4.4.2	Struktur Kembalian.....	50

4.4.3 Implementasi <i>Socket.io</i>	51
5. BAB V KESIMPULAN DAN SARAN.....	57
5.1 Kesimpulan.....	57
5.2 Saran.....	57
DAFTAR PUSTAKA	58

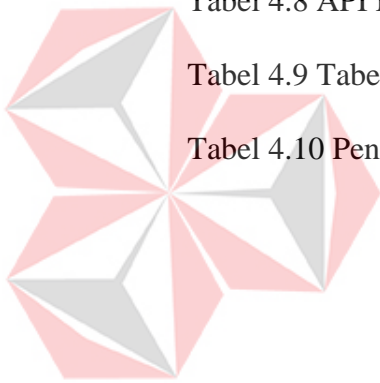


UNIVERSITAS
Dinamika

DAFTAR TABEL

Halaman

Tabel 4.1 Identifikasi Masalah.....	19
Tabel 4.2 Identifikasi Aktor	20
Tabel 4.3 Identifikasi Kebutuhan Fungsional	21
Tabel 4.4 Analisis Kebutuhan Pengguna	21
Tabel 4.5 API Model <i>User</i>	41
Tabel 4.6 API Model Kontak	42
Tabel 4.7 API Model <i>Conversations</i>	42
Tabel 4.8 API Model <i>Group Chats</i>	43
Tabel 4.9 Tabel API Model <i>Messages</i>	44
Tabel 4.10 Penamaan URL	47



UNIVERSITAS
Dinamika

DAFTAR GAMBAR

Halaman

Gambar 2.1 Logo PT. Vascomm Solusi Teknologi	8
Gambar 2.2 Struktur Organisasi PT. Vascomm Solusi Teknologi	11
Gambar 4.1 <i>System Flow Diagram</i> Fungsi <i>Login</i>	25
Gambar 4.2 <i>System Flow Diagram</i> Fungsi <i>Register</i>	27
Gambar 4.3 <i>System Flow Diagram</i> Fungsi <i>Pengelolaan Profile User</i>	29
Gambar 4.4 <i>System Flow Diagram</i> Fungsi <i>Pengelolaan Data Kontak</i>	31
Gambar 4.5 <i>System Flow Diagram</i> Fungsi <i>One to One Chat/ Personal Chat</i>	33
Gambar 4.6 <i>System Flow Diagram</i> Fungsi <i>One to Many Chat/ Group Chat</i>	35
Gambar 4.7 <i>Users Collection</i>	36
Gambar 4.8 <i>Contacs Collection</i>	37
Gambar 4.9 <i>Conversations Collection</i>	38
Gambar 4.10 <i>Group Chats Collection</i>	39
Gambar 4.11 <i>Messages Collection</i>	40
Gambar 4.12 <i>Error Message Missing Parameter</i>	49
Gambar 4.13 <i>Error Message Invalid Parameter</i>	49
Gambar 4.14 <i>Error Message Server</i>	49
Gambar 4.15 <i>Token Based Authentication</i>	50
Gambar 4.16 <i>Dependency</i> pada <i>File Socket.js</i>	51
Gambar 4.17 <i>Script Function Socket.io</i>	52
Gambar 4.18 Cara Kerja <i>Socket.io</i>	53
Gambar 4.19 <i>User Joined Request Socket.io</i>	54
Gambar 4.20 <i>Send Messages Request Socket.io</i>	54

Gambar 4.21 <i>User Typing Request Socket.io</i>	55
Gambar 4.22 <i>User Stop Typing Request Socket.io</i>	55



UNIVERSITAS
Dinamika

BAB I

PENDAHULUAN

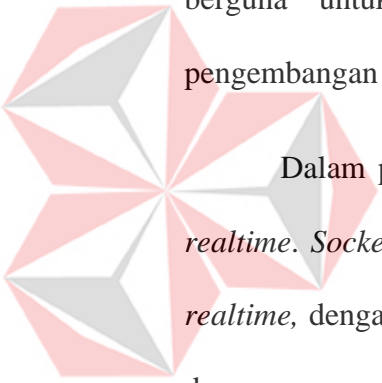
1.1 Latar Belakang

PT. Vascomm Solusi Teknologi adalah sebuah perusahaan yang bergerak pada bidang IT solutions dan startup factory. PT. Vascomm Solusi Teknologi berdiri sejak April 2006 dan hingga saat ini mempunyai tiga kantor cabang yang berlokasi di Sidoarjo, Jakarta, dan Yogyakarta. Sebagai perusahaan IT yang selalu memberikan solusi terbaik pada client, Vascomm berfokus pada digital banking solution dan smart ecosystem. PT. Vascomm Solusi Teknologi ini juga bermitra dengan beberapa perusahaan yang terkenal seperti BNI, BNI Syariah, Bank BTN, Bank Lampung, PT. Jayabrix Indonesia, PT. Samator dan Telkomsel. Sebagai perusahaan IT yang selalu memberikan solusi terbaik pada *client* PT. Vascomm Solusi Teknologi selalu senantiasa untuk mengembangkan teknologi pada semua produk nya. Pada kesempatan kerja praktik di PT. Vascomm Solusi Teknologi selama kurang lebih 2 bulan, penulis diberi sebuah *project* untuk melakukan riset tentang teknologi *Application Programming Interface* (API) dengan gaya arsitektur *Representational State Transfer* (REST) dan penggunaan *socket.io* pada pengembangan *back-end* pada aplikasi *chatting*.

Perancangan *software* berbasis API akan memungkinkan sebuah *back-end* dimanfaatkan secara lebih luas karena pembuatan *logic* pada sistem akan dilakukan secara terpisah dengan *logic* pada tampilan antarmuka pengguna. Penggunaan API juga mempermudah sebuah sistem untuk berkolaborasi dengan sistem lain, karena dengan menggunakan API adalah sebuah “penghubung” yang

memungkinkan suatu sistem untuk berinteraksi dengan sistem lain. Hal ini juga berguna jika pada proses yang akan datang proses bisnis yang ada membutuhkan interaksi dengan sistem lain, atau dalam masa pengembangan modul- modul baru pada sistem terkait.

Begitu pula dengan gaya arsitektur REST yang akan diterapkan pada pengembangan API. REST adalah sebuah *standart* arsitektur berbasis web yang menggunakan *protocol* HTTP untuk berkomunikasi data. Arsitektur REST dapat digunakan oleh berbagai macam bahasa pemrograman dan *platform*, selain itu juga struktur kerja pada arsitektur ini juga sangat mudah untuk dipahami. Hal ini berguna untuk para pengembang lain dalam proses pemeliharaan dan pengembangan sistem.



Dalam pembuatan aplikasi *chatting*, juga diperlukan transfer data secara *realtime*. *Socket.io* adalah sebuah *library javascript* untuk pengembangan sistem *realtime*, dengan menggunakan *socket.io* kita dapat menghubungkan antara *client* dan *server* secara *bidirectional* (dua arah). *Socket.io* sangat cocok untuk pengembangan *realtime-analytics*, *messaging and chat apps*, *streaming apps*, dan *document collaboration*. Maka dari itu dalam riset ini penulis menggunakan *Socket.io* untuk melakukan transfer data secara *real-time* pada aplikasi *chatting* yang akan dibuat.

Gagasan-gagasan diatas menjadi alasan sangat perlunya melakukan riset untuk merancang dan membangun *application programming interface* (api) menggunakan gaya arsitektur *representational state transfer* (rest) dengan *socket.io* untuk pengembangan aplikasi *chatting*.

1.2 Rumusan Masalah

Berdasarkan uraian diatas maka rumusan masalah laporan ini adalah bagaimana merancang *application programming interface* (API) menggunakan gaya arsitektur *representational state transfer* (REST) untuk pengembangan sistem informasi *chatting*?

1.3 Batasan Masalah

Berdasarkan uraian di atas, maka penulis membatasi pokok permasalahan yang akan dibahas sebagai berikut:

1. Keluaran data yang dihasilkan dari REST API berupa data JSON
2. Perancangan REST API meliputi manajemen *user*, manajemen kontak, *person to person chat*, dan *group chat*

1.4 Tujuan

Berdasarkan latar belakang dan rumusan masalah, maka tujuan dari kerja praktik ini adalah merancang *application programming interface* (API) menggunakan gaya arsitektur *representational state transfer* (REST) untuk pengembangan sistem informasi *chatting*.

1.5 Manfaat

Berdasarkan uraian di atas, manfaat yang akan diperoleh dari penelitian ini adalah sebagai berikut :

1. Dapat dijadikan bahan rujukan literatur dan obyek pertimbangan secara umum dalam pengembangan sistem informasi dengan kasus sejenis.
2. Dapat dijadikan sebagai bahan evaluasi dalam penelitian selanjutnya, khususnya yang berhubungan dengan *chatting apps* dan pengembangan sistem menggunakan REST API

1.6 Sistematika Penulisan

Untuk memberikan gambaran menyeluruh terhadap masalah yang dibahas,

maka sistematika penulisan dibagi ke dalam beberapa bab yaitu:

BAB I PENDAHULUAN

Pada bab ini menjelaskan tentang latar belakang dari hal-hal yang berhubungan dengan perusahaan, rumusan masalah, batasan masalah,

tujuan yang ingin dicapai, manfaat yang diperoleh dengan adanya REST API yang telah dibuat, serta sistematika penulisan dari proposal.

BAB II GAMBARAN UMUM PERUSAHAAN

Bab ini menjelaskan tentang PT. Vascomm Solusi Teknologi, mulai dari visi & misi perusahaan, dan stuktur organisasi.

BAB III LANDASAN TEORI

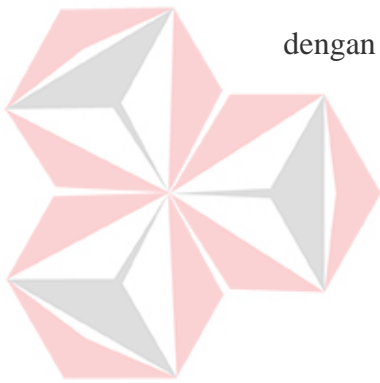
Pada bab ini membahas tentang teori-teori yang dianggap berhubungan dengan kerja praktik yang dilakukan, dimana teori-teori tersebut akan menjadi acuan untuk penyelesaian masalah.

BAB IV DESKRIPSI PEKERJAAN

Bab ini menguraikan tentang langkah-langkah yang digunakan untuk pembuatan sistem yang digunakan untuk penyelesaian masalah yang membahas keseluruhan desain *input*, proses, dan *output* dari sistem. Pada bab ini juga membahas tentang implementasi dari perancangan yang telah dilakukan dalam pembuatan REST API pada aplikasi *chatting*.

BAB V PENUTUP

Pada bab ini dibahas mengenai kesimpulan dari pembuatan REST API pada aplikasi *chatting* terkait dengan tujuan dan permasalahan, beserta dengan saran yang bermanfaat untuk pengembangan REST



UNIVERSITAS
Dinamika

BAB II

GAMBARAN UMUM PERUSAHAAN

2.1 Profil Perusahaan

PT. Vascomm Solusi Teknologi adalah sebuah perusahaan yang bergerak pada bidang *IT solutions* dan *startup factory*. PT. Vascomm Solusi Teknologi berdiri sejak April 2006 dan hingga saat ini mempunyai tiga kantor cabang yang berlokasi di Sidoarjo, Jakarta, dan Yogyakarta. Sebagai perusahaan *IT* yang selalu memberikan solusi terbaik pada *client*, Vascomm berfokus pada *digital banking solution* dan *smart ecosystem*. Vascomm mempunyai berbagai macam produk antara lain : *E – Money/ E – Wallet, branchless banking, virtual account, autodebet, remittance, onboarding, internet of things (IoT), big data, community apps/ platform, smart devices*. Pada tahun 2018 Vascomm mengembangkan sebuah model bisnis baru dengan membangun ekosistem digital. Mereka memberikan berbagai macam *resources* berupa tim, pembinaan dan modal untuk menciptakan dan menginkubasi para pengusaha *startup* untuk menciptakan atau mengembangkan perusahaan hingga mempunyai nilai bisnis. Hingga saat ini setidaknya terdapat dua *startup* yang berada dalam naungan Vascomm yaitu Branchless.id (<https://branchless.id>) dan Sitamoto (<https://sitamoto.id>).

2.2 Identitas Perusahaan

Nama Instansi : PT. Vascomm Solusi Teknologi

Alamat : Perkantoran Gateway Blok A No. 12, Jl. Raya Waru,

Dusun Sawo, Sawotratap, Kec. Gedangan, Kabupaten
Sidoarjo, Jawa Timur 61254

No. Telepon : +6231 85589763
Website : www.vascomm.co.id
Email : hello@vascomm.co.id

2.3 Sejarah Perusahaan

Perjalanan bisnis Vascomm berkecimpung di industri teknologi sudah dimulai sejak 13 tahun lalu, tepatnya pada April 2006. Mulanya, perusahaan bernama CV Alcomindo Jaya (CV. AJ).

Penggunaan *mobile phone* di Indonesia saat itu mengalami pertumbuhan positif. Peningkatan penggunaan ponsel pintar rupanya berdampak pada permintaan pembelian pulsa yang juga mengalami pertumbuhan signifikan. Perusahaan lalu melihat itu sebagai peluang untuk mengembangkan bisnis di industri telekomunikasi. Yaitu dengan berperan sebagai dealer pulsa untuk menjual kartu perdana, pulsa, paket data, PLN, BPJS, PDAM, hingga multi finance. CV. AJ dipercaya oleh para sub dealer atau OPJ (Outlet Pinggir Jalan) sebagai jalur distribusi resmi mereka.

Dari sisi operasional, jumlah karyawan CV. AJ awalnya hanya empat orang. Mereka adalah tim bisnis, teknis, sales, dan support. Berlanjut hingga tahun 2015, perusahaan melihat adanya peluang bisnis lain yang lebih prospektif di dunia perbankan berbasis digital. Alasan ini juga yang akhirnya membuat salah satu karyawan CV. AJ yang saat ini menjadi Chief Technology Officer (CTO)

untuk melakukan proses *rebranding* organisasi dari sisi internal dan bisnis. Yaitu dengan merumuskan model bisnis baru perusahaan yang fokus pada layanan Financial Technology (Fintech) dan Community.

Hingga saat ini, perusahaan yang telah mengganti namanya menjadi PT. Vascomm Solusi Teknologi ini mencoba pendekatan baru dalam merencanakan, mengembangkan konsep dan gagasan, serta mengoperasionalkannya. Sebagai sebuah *learning organization*, Vascomm terus berbenah diri mengembangkan model bisnisnya dengan dukungan talenta muda, generasi millennial yang punya segudang karakteristik. Ini terwujud melalui tujuan jangka panjang Vascomm selain sebagai *IT solution*, juga menjadi *startup factory* yang akan melahirkan banyak *startup* berbasis teknologi.

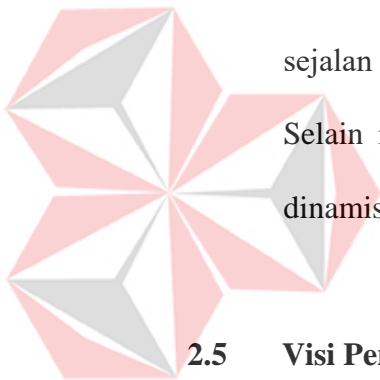
2.4 Logo Perusahaan dan Filosofi



Gambar 2.1 Logo PT. Vascomm Solusi Teknologi

- Rangkaian dari huruf V-A-S-C-O-M-M, ketika digabungkan membentuk sebuah pola yang kini dipakai menjadi lambang perusahaan.
- Gambarnya menyerupai bentuk hati yang bisa diartikan cinta, spiritualitas, atau moral. Logo Vascomm berbentuk hati ini merepresentasikan lingkungan kerja yang dinamis, bersahabat, dan punya attitude unggul.

- Diamond, kilauan putih hasil gradasi warna biru sekaligus pola gambarnya berasosiasi dengan batuan dari keluarga permata yang begitu indah. Layaknya sebuah berlian, logo Vascomm mengartikulasikan kokoh dan tidak terkalahkan. Salah satunya diartikulasikan lewat rumusan model strategi perusahaan dan eksistensinya yang kuat, tidak mudah tergoyahkan.
- Biru Tua, melambangkan teknologi. Vascomm adalah perusahaan yang punya core bisnis di bidang teknologi informasi.
- Biru Muda, melambangkan Vascomm sebagai sebuah entitas bisnis. Hanya saja, profit bukanlah fokus utama dalam pengembangan bisnisnya, melainkan sebuah impact dari solusi yang bermanfaat bagi siapapun. Ini sejalan dengan tagline Vascomm “Bright Solution for Your Business”. Selain itu, warna biru muda menandakan ketenangan, lingkungan kerja dinamis, dan sesuatu yang fresh.



2.5 Visi Perusahaan

PT. Vascomm Solusi Teknologi memiliki visi perusahaan yang merupakan tujuan yang akan dicapai dimasa yang akan datang yaitu: *“Deliver Bright Solution to Everyone”*

2.6 Misi Perusahaan

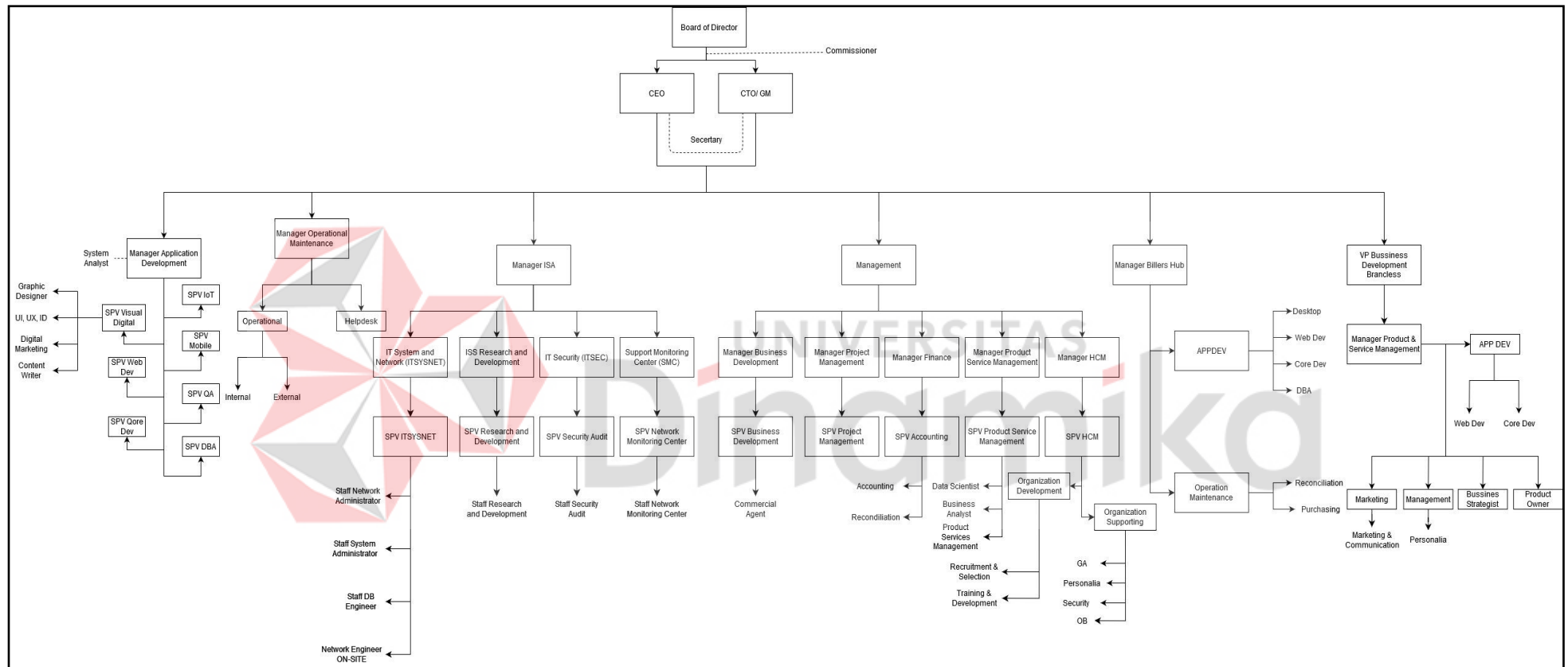
Misi dari PT. Vascomm Solusi Teknologi adalah: *“Enhance people’s life experience providing simple-safe and smart service”*

2.7 Struktur Organisasi

Struktur organisasi adalah sebuah susunan dari berbagai komponen atau unit unit kerja yang berada dalam sebuah organisasi/perusahaan. Pembentukan Struktur organisasi sendiri memiliki tujuan untuk menjalankan organisasi/perusahaan sesuai dengan tugas dan fungsi masing masing jabatan. Berikut ini adalah struktur organisasi dari PT. Vascomm Solusi Teknologi:



UNIVERSITAS
Dinamika



Gambar 2.2 Struktur Organisasi PT. Vascomm Solusi Teknologi

BAB III

LANDASAN TEORI

3.1 *Chatting*

Chatting adalah sebuah aktivitas komunikasi yang dilakukan dua orang atau lebih dengan memanfaatkan aplikasi *chatting* dan jaringan internet. *Chatting* adalah bentuk komunikasi yang cukup efektif karena tidak mengenal waktu dan ruang.

3.2 HTTP

Hyper Text Transfer Protocol (HTTP) merupakan sebuah protokol komunikasi antara *client* dan *server* dengan konsep *request-response*. Sebagai sebuah protokol, HTTP menentukan prosedur-prosedur komunikasi baik dalam format dan cara komunikasi hingga aksi dan reaksi antara *web server* dan *browser*

(Hidayatullah dan Jauhari, 2015).

Menurut Pratama (2014), protokol yang paling umum digunakan oleh pengguna jaringan komputer adalah HTTP, khususnya dalam mengakses suatu situs (*website*). Dalam komunikasi antara *server* dan *client*, protokol HTTP sedikitnya menjalankan tiga buah fungsi :

1. Menentukan reaksi *web server* (*server*) terhadap aksi dari *web browser* (*client*).
2. Membantu *web browser* menyajikan data dan informasi yang dikirimkan oleh *web server* berdasarkan permintaan dari *client*.

3. Membantu penerjemahan pesan dan perintah yang berasal dari *client* dan ditujukan ke *server*, serta tanggapan yang dikirimkan dari *server* ke *client* (berdasarkan permintaan dari *client*).

3.3 *Application Programming Interface (API)*

API (*Application Programming Interface*). API adalah suatu “penghubung” yang memungkinkan suatu aplikasi untuk berinteraksi dengan aplikasi lainnya dan berbagi data. Banyaknya sistem, aplikasi, kebutuhan pengguna, mekanisme yang berbeda tetapi memerlukan data yang sama. Mekanisme pengiriman data dapat distandarisasi melalui API. Dengan cara ini, *developer* dapat menawarkan berbagai macam data yang dapat *developer* lain mengerti, serta mereka dapat menggunakan sistem mereka sendiri. (Jiri Hradil, Vilém Sklenak, 2017)

3.4 *Representational State Transfer (REST)*

REST (*Representational State Transfer*) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data. Metode ini sering diterapkan dalam pengembangan aplikasi, dimana tujuannya adalah untuk menjadikan sebuah sistem yang memiliki performa yang baik, cepat dan mudah untuk di kembangkan terutama dalam pertukaran dan komunikasi data. (Doglio, 2018). REST adalah arsitektur standar *web* yang menggunakan protokol HTTP dalam komunikasi data. Arsitektur tersebut didirikan berdasarkan sumber data dimana masing-masing komponen merupakan sumber data. Sumber data diakses oleh antarmuka yang sama dengan menggunakan metode standar HTTP. Dalam

arsitektur REST, *server* yang mengikuti arsitektur REST menyediakan akses ke sumber data dan klien yang mengambil data. Setiap sumber data diidentifikasi menggunakan link URI. REST menggunakan berbagai format untuk menyajikan data, seperti teks, JSON dan XML. Berikut adalah metode HTTP yang umumnya digunakan dalam arsitektur REST:

1. GET untuk menyediakan akses untuk membaca sumber data
2. POST untuk menambah data baru
3. PUT untuk memperbaharui data yang tersedia
4. DELETE untuk menghapus data

3.5 *JavaScript Object Notation (JSON)*

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. JSON terbuat dari dua struktur yaitu kumpulan pasangan nama atau nilai dan daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list), atau urutan (sequence).

3.6 Node JS

Node.js adalah perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web dan ditulis dalam sintaks bahasa pemrograman JavaScript. Bila selama ini kita mengenal JavaScript sebagai bahasa pemrograman yang berjalan di sisi *client / browser* saja, maka Node.js ada untuk melengkapi peran JavaScript sehingga bisa juga berlaku sebagai bahasa pemrograman yang berjalan di sisi server, seperti halnya PHP, Ruby, Perl, dan sebagainya. Node.js dapat berjalan di sistem operasi Windows, Mac OS X dan Linux tanpa perlu ada perubahan kode program. Node.js memiliki pustaka server HTTP sendiri sehingga memungkinkan untuk menjalankan server web tanpa menggunakan program server web seperti *Apache* atau *Nginx*.

3.7 Express JS

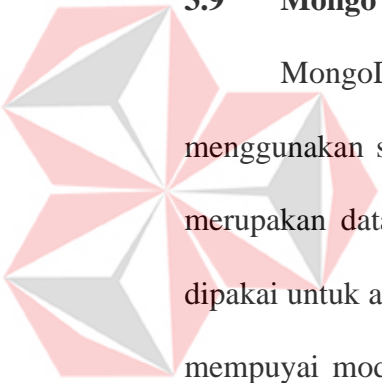
Express.js adalah satu *web framework* paling populer di dunia Node.js.

Dokumentasinya yang lengkap dan penggunaannya yang cukup mudah, dapat membuat kita mengembangkan berbagai produk seperti aplikasi web ataupun RESTful API. Express.js pun dapat digunakan menjadi pijakan untuk membangun *web framework* yang lebih kompleks seperti, Sails.js, MEAN (MongoDB, Express.js, Angular.js, Node.js) dan MERN (MongoDB, Express.js, React.js, Node.js). Express.js dibuat oleh TJ Holowaychuk dan sekarang dikelola oleh komunitas.

3.8 Socket IO

Socket.io merupakan sebuah *library javascript* yang membantu dalam pembuatan aplikasi web yang realtime, dengan menggunakan socket.io kita dapat menghubungkan antara client dan server dapat terjadi secara *bidirectional* (dua arah). Maksudnya yaitu kita dapat menghubungkan *client* dan *server* sehingga dapat berperan sebagai pengirim dan sekaligus penerima data, komponen yang terdapat pada socket.io terdiri dari dua bagian yang pertama client-site yaitu yang berjalan pada *browser*, dan *server-site* yang dapat digunakan sebagai modul untuk node.js.

3.9 Mongo DB



MongoDB adalah salah satu produk database noSQL *Open Source* yang menggunakan struktur data JSON untuk menyimpan datanya. MongoDB adalah merupakan database noSQL yang paling populer di internet. MongoDB sering dipakai untuk aplikasi berbasis Cloud, Grid Computing, atau Big Data. MongoDB mempunyai model penyimpanan data *document database*, jadi setiap satu *object* data disimpan dalam satu dokumen. *Document* sendiri bisa terdiri dari *key-value*, dan value sendiri bisa berupa array atau *key-value* bertingkat.

3.10 JSON Web Token (JWT)

JSON Web Token (JWT) adalah sebuah token berbentuk string panjang yang sangat random yang berguna untuk melakukan sistem *authentification* dan *authorization* dalam pertukaran data. Pada umumnya untuk melakukan hal itu

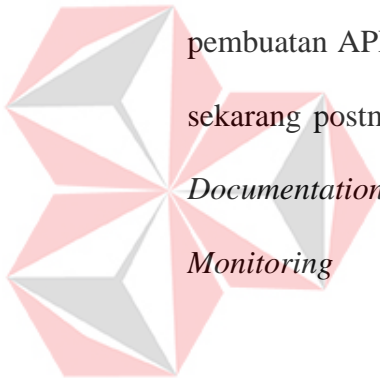
diperlukan *session*. Namun dalam pengembangan sistem menggunakan API hal ini akan dilakukan oleh JWT atau “jot”.

JWT terdiri dari tiga struktur yang dipisahkan oleh tanda titik (.), yaitu:

1. *Header* untuk memuat jenis *encoding* yang digunakan.
2. *Payload* untuk memuat nilai-nilai informasi yang ditransaksikan.
3. *Signature* untuk memuat nilai *hash* untuk memverifikasi *payload*

3.11 *POSTMan*

Postman merupakan *tool* wajib bagi para *developer* yang berkecukupan pada pembuatan API, fungsi utama postman ini adalah sebagai GUI API Caller namun sekarang postman juga menyediakan fitur lain yaitu Sharing Collection API for *Documentation* (free), *Testing* API (free), *Realtime Collaboration* *Team* (paid), *Monitoring* API (paid), *Integration* (paid).



UNIVERSITAS
Dinamika

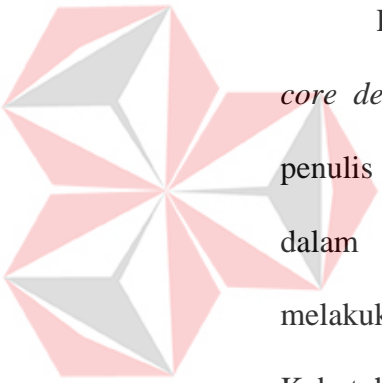
BAB IV

DESKRIPSI PEKERJAAN

4.1 Analisis Sistem

Pada tahap analisa, peneliti melakukan analisa kebutuhan yang menjadi dasar persiapan dalam pengembangan sistem. Hal ini dilakukan supaya dalam proses pengembangan sistem benar benar sesuai dengan apa yang akan direncanakan dan meminimalisir sebuah kesalahan atau bug dalam sistem

4.1.1 Wawancara



Pada tahap ini penulis melakukan wawancara kepada kepala divisi *core development* PT. Vascomm Solusi Teknologi untuk memudahkan penulis dalam menganalisis dan mendesain REST API dan *Socket.io* dalam pengembangan aplikasi *chatting*. Topik yang diangkat saat melakukan proses wawancara adalah seputar REST API, *Socket.io*, dan Kebutuhan Fungsional/ Fitur yang akan dikembangkan pada aplikasi *chatting*

4.1.2 Studi Literatur

Pada tahap ini dilakukan pembelajaran dari berbagai macam literatur, mengenai permasalahan yang ada, seperti algoritma pembuatan aplikasi *chatting*, metode pengembangan sistem berbasis API menggunakan arsitektur REST, Node JS, Express JS, JSON , JSON Web Token (JWT), *Socket.io*, Mongo DB. Literatur yang dipelajari dapat berupa jurnal ilmiah, artikel, buku referensi, aplikasi lain, dan halaman web.

4.1.3 Identifikasi Masalah

Identifikasi masalah bertujuan untuk mengetahui masalah serta dampak yang ditimbulkan sehingga dapat memastikan solusi yang harus dipecahkan. Identifikasi masalah ini juga membantu agar penulis tidak terlalu melebar dari pokok penelitian sehingga penulis dapat lebih fokus untuk membuat solusi sesuai permasalahan yang ada

Tabel 4.1 Identifikasi Masalah

PERMASALAHAN	DAMPAK	SOLUSI
Pengembangan sistem informasi <i>chatting</i> belum menggunakan <i>application programming interface</i> (API)	<ul style="list-style-type: none"> • Kesulitan saat melakukan pengembangan sistem terutama untuk cross-platform • Kesulitan saat melakukan <i>maintenance</i> karena <i>logic</i> pada sisi <i>server</i> masih bergabung dengan sisi <i>client</i> 	Membuat <i>application programming interface</i> (API) dengan gaya arsitektur <i>representational state transfer</i> (REST)

4.1.4 Identifikasi Aktor

Identifikasi aktor bertujuan untuk mengetahui pengguna yang ada dalam sebuah sistem. Identifikasi aktor dapat memudahkan dalam

pembuatan kebutuhan fungsional. Berikut ini adalah analisis aktor pada aplikasi *chatting* yang akan dibuat:

Tabel 4.2 Identifikasi Aktor

AKTOR	DESKRIPSI
<i>User</i>	<i>User</i> adalah pengguna sistem informasi <i>chatting</i> yang dapat berkomunikasi atau memulai obrolan dengan <i>user</i> lain

4.1.5 Identifikasi Data

Identifikasi data ini melibatkan kebutuhan data apa saja yang digunakan dalam proses *chatting*. Pada proses sistem informasi *chatting* data yang diperlukan adalah

- Data *User*
- Data Kontak/ Teman
- Data Percakapan
- Data *Group Chat*
- Data Pesan

4.1.6 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional bertujuan untuk mengetahui kebutuhan proses yang dibutuhkan dalam sebuah sistem. Kebutuhan fungsional bisa berangkat dari pertanyaan: “Apa yang harus sistem lakukan?” Berikut ini adalah analisis kebutuhan fungsional pada aplikasi *chatting* yang akan dibuat:

Tabel 4.3 Identifikasi Kebutuhan Fungsional

NO	AKTOR	FUNGSI	DESKRIPSI
1	<i>User</i>	<i>Authentication</i>	Merupakan proses untuk <i>user</i> melakukan otentikasi masuk atau pendaftaran kedalam sistem
2	<i>User</i>	Pengelolaan data <i>profile user</i>	Merupakan proses untuk <i>user</i> melakukan pengelolaan data <i>profile</i> mereka
3	<i>User</i>	Pengelolaan data kontak	Merupakan proses untuk <i>user</i> melakukan pengelolaan kontak, meliputi tambah, ubah, dan hapus teman.
4	<i>User</i>	<i>One to one chat/ Personal chat</i>	Merupakan proses untuk <i>user</i> melakukan <i>chatting</i> dengan satu <i>user</i> lain
5	<i>User</i>	<i>One to many chat/ Group chat</i>	Merupakan proses untuk <i>user</i> melakukan <i>chatting</i> dengan satu atau lebih <i>user</i> lain

4.1.7 Analisis Kebutuhan Pengguna

Analisis kebutuhan pengguna berfungsi untuk mengetahui kebutuhan dari masing masing pengguna yang berhubungan langsung dengan sistem yang akan dibuat. Kebutuhan pengguna pada sistem informasi *chatting* adalah

Tabel 4.4 Analisis Kebutuhan Pengguna

KEBUTUHAN FUNGSI	KEBUTUHAN DATA	OUTPUT
<i>Authentication</i>	Data <i>user</i>	Master <i>user</i>

Mengelola <i>profile user</i>	Data <i>user</i>	Data <i>user</i>
Mengelola data kontak	1. Data <i>user</i> 2. Data kontak	1. Master <i>user</i> 2. Master kontak
<i>One to one chat/ Personal chat</i>	1. Data <i>user</i> 2. Data kontak 3. Data percakapan 4. Data pesan	1. Master <i>user</i> 2. Master kontak 3. Master percakapan 4. Master pesan
<i>One to many chat/ Group chat</i>	Data <i>user</i> Data kontak Data <i>group chat</i> Data pesan	1. Master <i>user</i> 2. Master kontak 3. Master <i>group chat</i> 4. Master pesan

4.1.7 Analisis Kebutuhan Data

Analisis kebutuhan data dilakukan setelah menyusun analisis kebutuhan pengguna. Data yang dibutuhkan guna menunjang sistem informasi yang akan dibuat meliputi

- Data *User*

Data *user* merupakan data master yang digunakan untuk menyimpan data *user* pada sistem chatting. Data *user* yang diperlukan adalah *ID_USER*, *USERNAME*, *EMAIL*, *PASSWORD*, *ABOUT*, *AVATAR*

- Data Kontak

Data kontak merupakan data master yang digunakan untuk menyimpan data kontak dari setiap *user* pada sistem *chatting*. Data kontak yang diperlukan adalah *ID_KONTAK*, *ID_USER*, *LIST_CONTACT*

- Data Percakapan

Data percakapan merupakan data master yang digunakan untuk menyimpan data *personal chatting*. Data percakapan yang diperlukan adalah *ID_CONVERSATION, PARTICIPANTS*

- Data Group Chat

Data *group chat* merupakan data master yang digunakan untuk menyimpan data *group chatting*. Data percakapan yang diperlukan adalah *ID_GROUP, NAME_GROUP, AVATAR_GROUP, DESCRIPTION_GROUP, PARTICIPANTS*

- Data Pesan

Data pesan merupakan data master yang digunakan untuk menyimpan data pesan dari *personal* atau *group chat*. Data Pesan yang dibutuhkan adalah *ID_MESSAGE, ID_CONVERSATION, ID_GROUP, SENDER, CONTENT, DATE*

4.2 Perancangan Sistem

Setelah melakukan tahap analisis sistem, tahap selanjutnya adalah merancang sistem. Proses tahapan ini adalah membentuk suatu sistem informasi *chatting* menggunakan *application programming interface* (API) dengan gaya arsitektur *representational state transfer* (REST)

4.2.1 Desain Alur Proses

Desain alur proses bertujuan untuk mengetahui alur kerja sistem. Desain alur proses akan dianalisis berdasarkan kebutuhan fungsional.

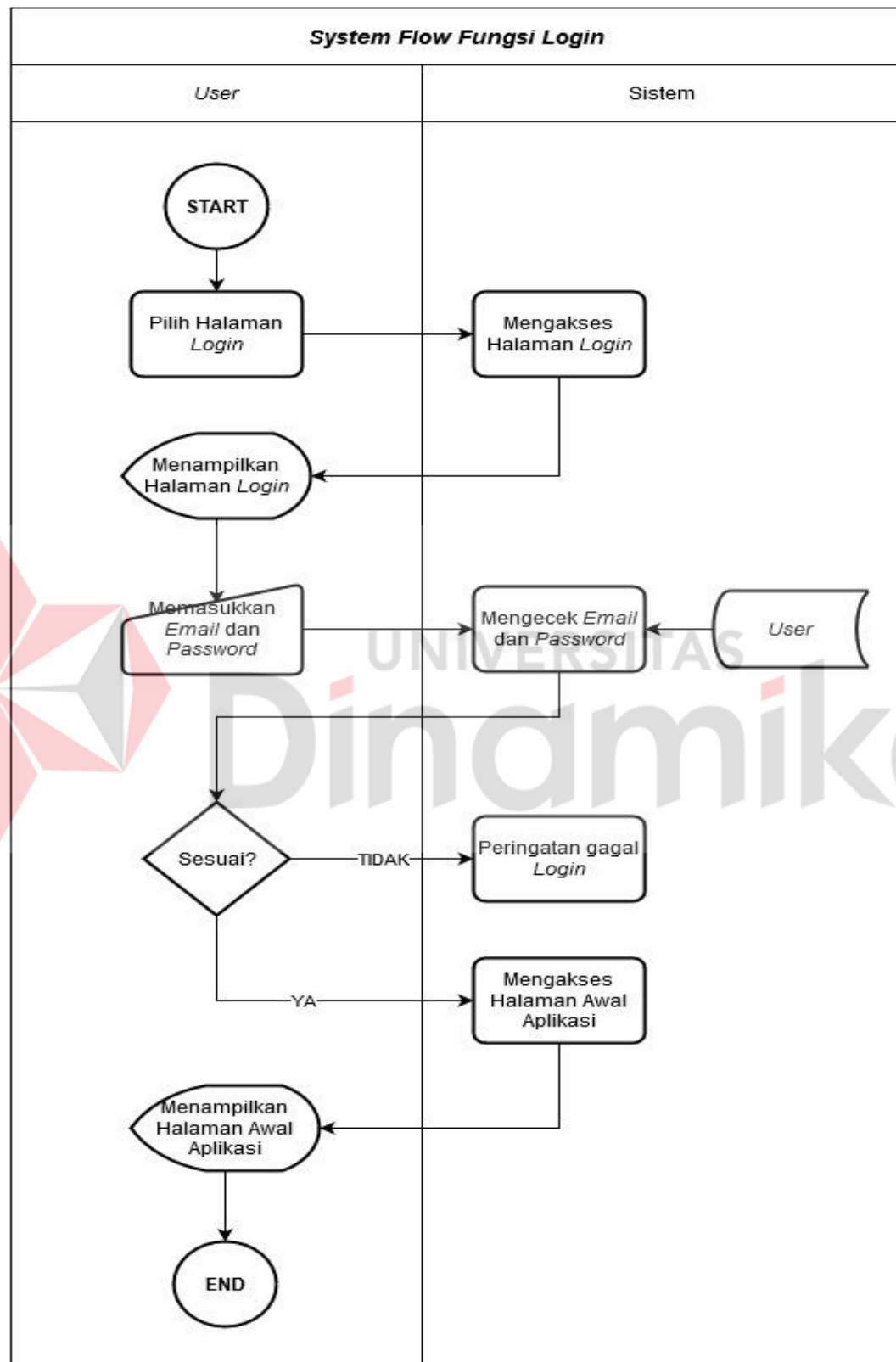
Desain alur proses akan digambarkan melalui *system flow diagram*. Berikut ini adalah *system flow diagram* dari aplikasi *chatting* yang akan dibuat:

- *System Flow Diagram Fungsi Login*

System flow login merupakan proses dimana *user* diberi hak akses untuk dapat masuk kedalam sistem aplikasi. Fungsi *login* dimulai ketika *user* mengakses halaman *login*. *User* akan diminta untuk memasukkan *email* dan *password* untuk melakukan proses *authentication*. Setelah itu sistem akan mencocokkan data *email* dan *password* yang diberikan oleh *user* dengan database. Jika data cocok, maka *user* dapat masuk kedalam sistem dan diarahkan pada halaman awal aplikasi. Jika data tidak cocok, maka sistem akan memberikan peringatan bahwa data yang diberikan tidak sesuai.



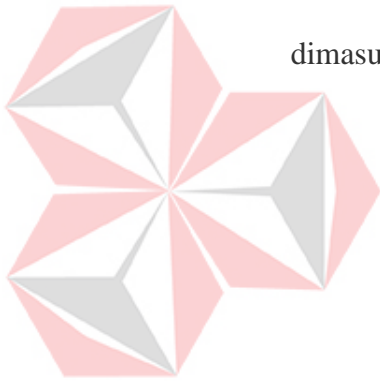
UNIVERSITAS
Dinamika



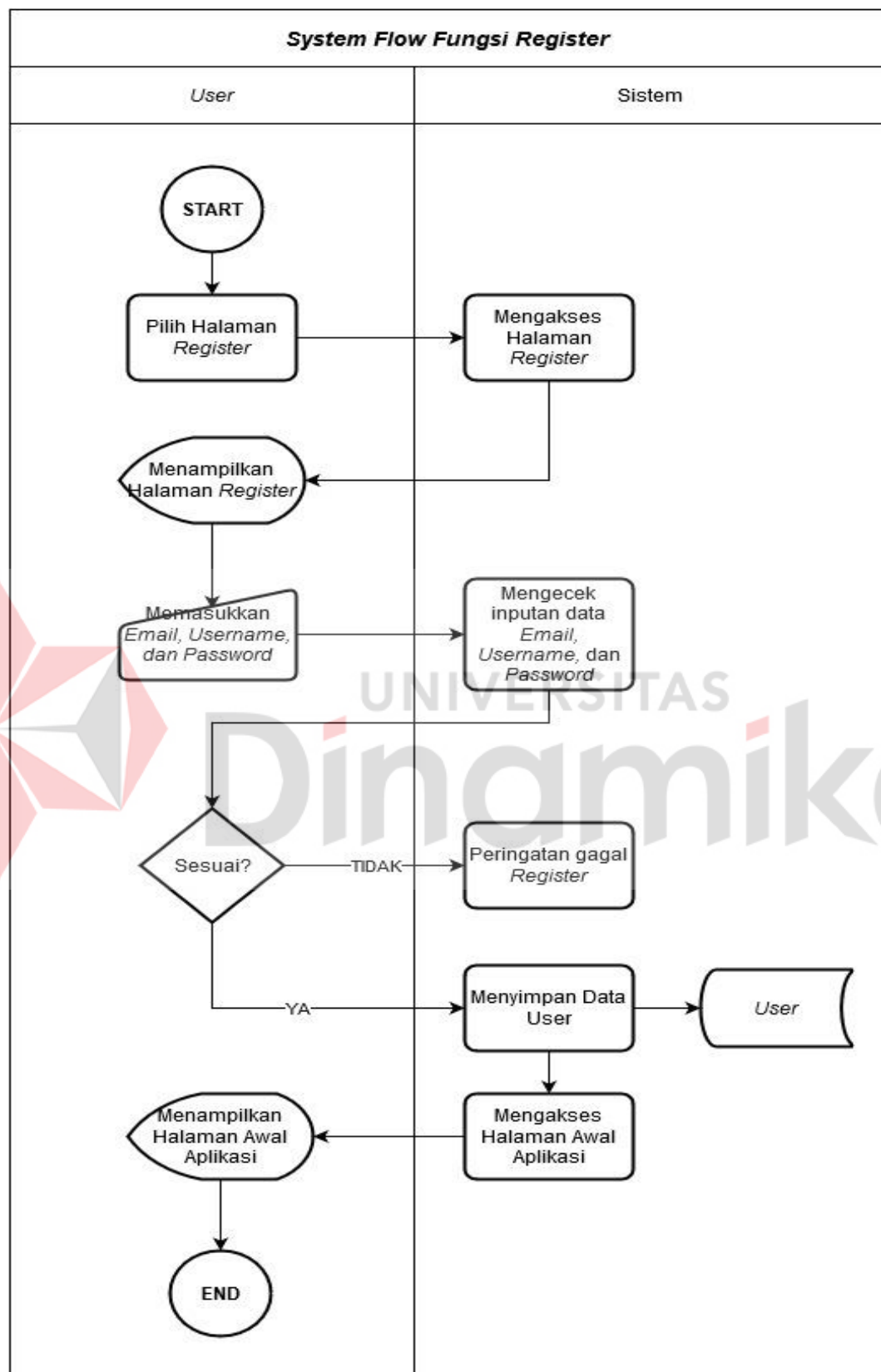
Gambar 4.1 System Flow Diagram Fungsi Login

- *System Flow Diagram Fungsi Register*

System flow register merupakan proses dimana *user* dapat mendaftar kedalam sistem. Sebelum *user* dapat masuk kedalam sistem mereka harus mempunyai sebuah informasi *login* untuk sistem dapat mengenali *user* tersebut. Jika *user* tidak mempunyai informasi *login*, maka mereka akan diarahkan pada halaman *register*. Setelah itu mereka harus memasukkan *username*, *email*, dan *password*. Sistem akan mengecek data masukan. Jika masukan data sesuai, maka data akan disimpan kedalam *database* dan *user* dapat melakukan *login*/ masuk kedalam sistem. Jika masukan data tidak sesuai sistem akan memberikan peringatan bahwa data yang dimasukkan tidak sesuai.



UNIVERSITAS
Dinamika



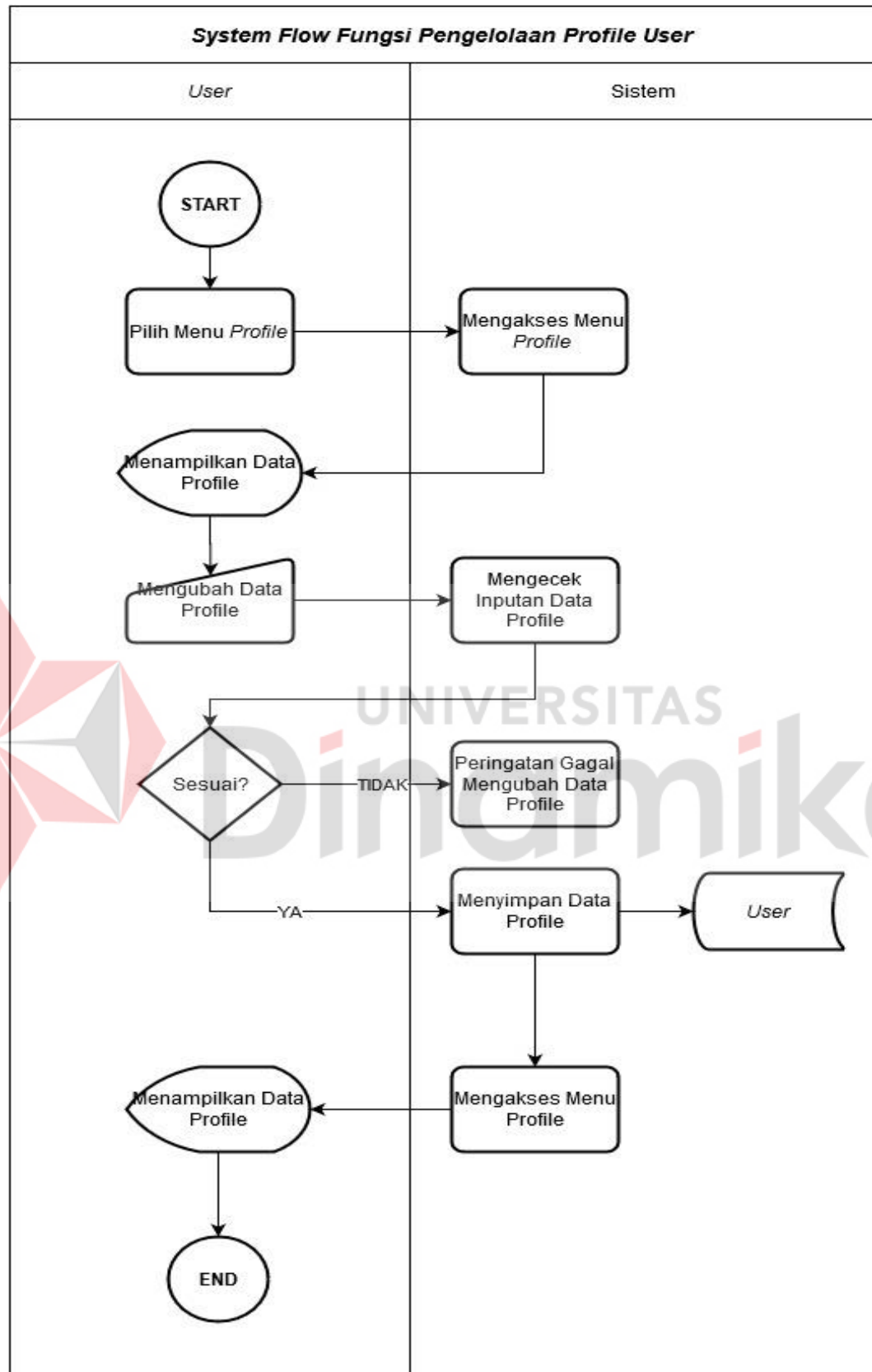
Gambar 4.2 System Flow Diagram Fungsi Register

- *System Flow Diagram Fungsi Pengelolaan Profile User*

System flow pengelolaan *profile user* merupakan proses dimana user dapat mengupdate data diri mereka. Data tersebut terdiri dari *username*, *about*, dan *avatar*. Fungsi pengelolaan *profile user* dimulai ketika *user* mengakses halaman *profile*. Setelah itu *user* dapat mengubah data diri mereka. Sistem akan mengecek data masukan. Jika data masukan sesuai, maka data *profile* lama pada database akan diubah sesuai data masukan. Jika data masukan tidak sesuai, maka sistem akan memberikan peringatan bahwa data yang dimasukkan tidak sesuai



UNIVERSITAS
Dinamika



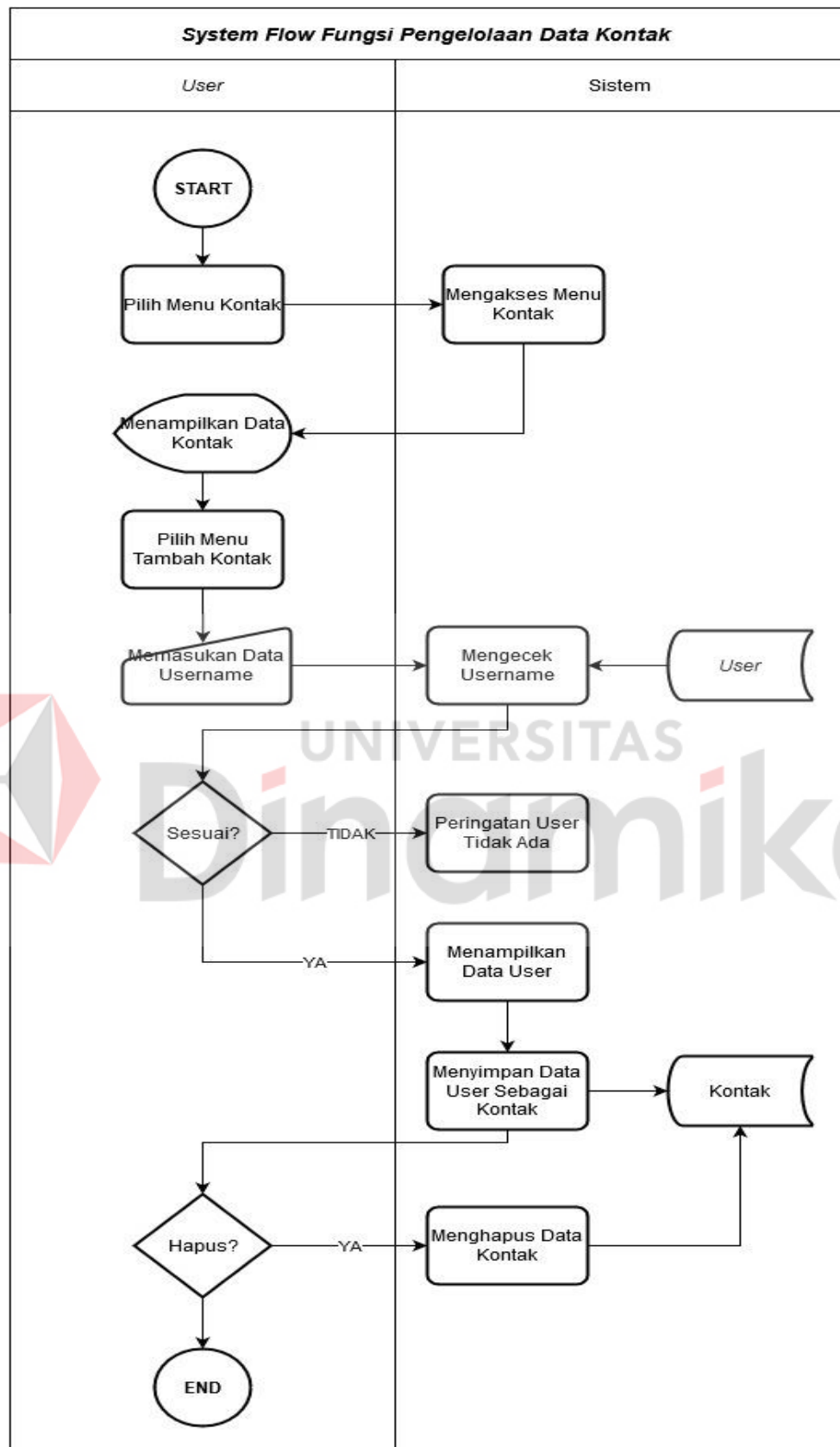
Gambar 4.3 System Flow Diagram Fungsi Pengelolaan Profile User

- *System Flow Diagram* Fungsi Pengelolaan Data Kontak

System flow pengelolaan data kontak merupakan proses dimana *user* dapat melakukan tambah dan hapus teman. Sebelum melakukan obrolan *user* harus menjadikan *user* lain sebagai teman pada daftar kontak mereka. Proses ini dimulai dari *user* memilih menu kontak, sistem akan menampilkan daftar teman. Jika ingin menambah teman, maka pilih menu tambah teman, lalu masukkan *username* teman yang akan ditambahkan pada kontak, maka sistem akan mengecek pada database. Jika data sesuai, maka sistem akan menampilkan data teman yang dicari. Jika data tidak sesuai, maka akan muncul peringatan user tidak ada.



UNIVERSITAS
Dinamika



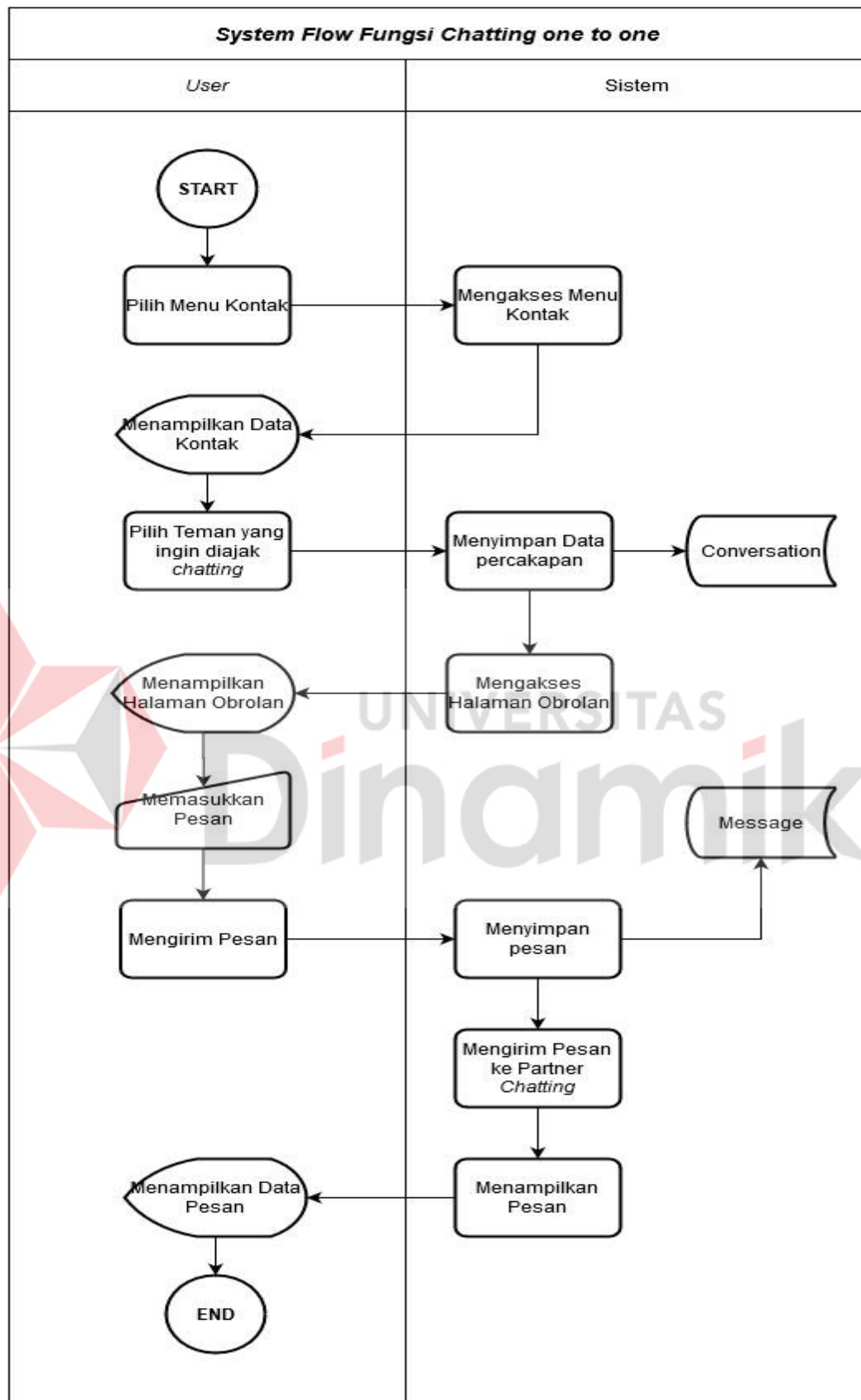
Gambar 4.4 *System Flow Diagram* Fungsi Pengelolaan Data Kontak

- *System Flow Diagram Fungsi Chatting one to one*

System flow chatting one to one merupakan proses dimana *user* dapat melakukan *chatting* dengan satu user lain. Proses ini berawal dari *user* memilih teman untuk dijadikan lawan bicara. Setelah itu *user* menulis pesan dan sistem kemudian akan menyimpan data pesan ke *database* dan mengirimkan data pesan ke lawan bicara secara langsung.



UNIVERSITAS
Dinamika



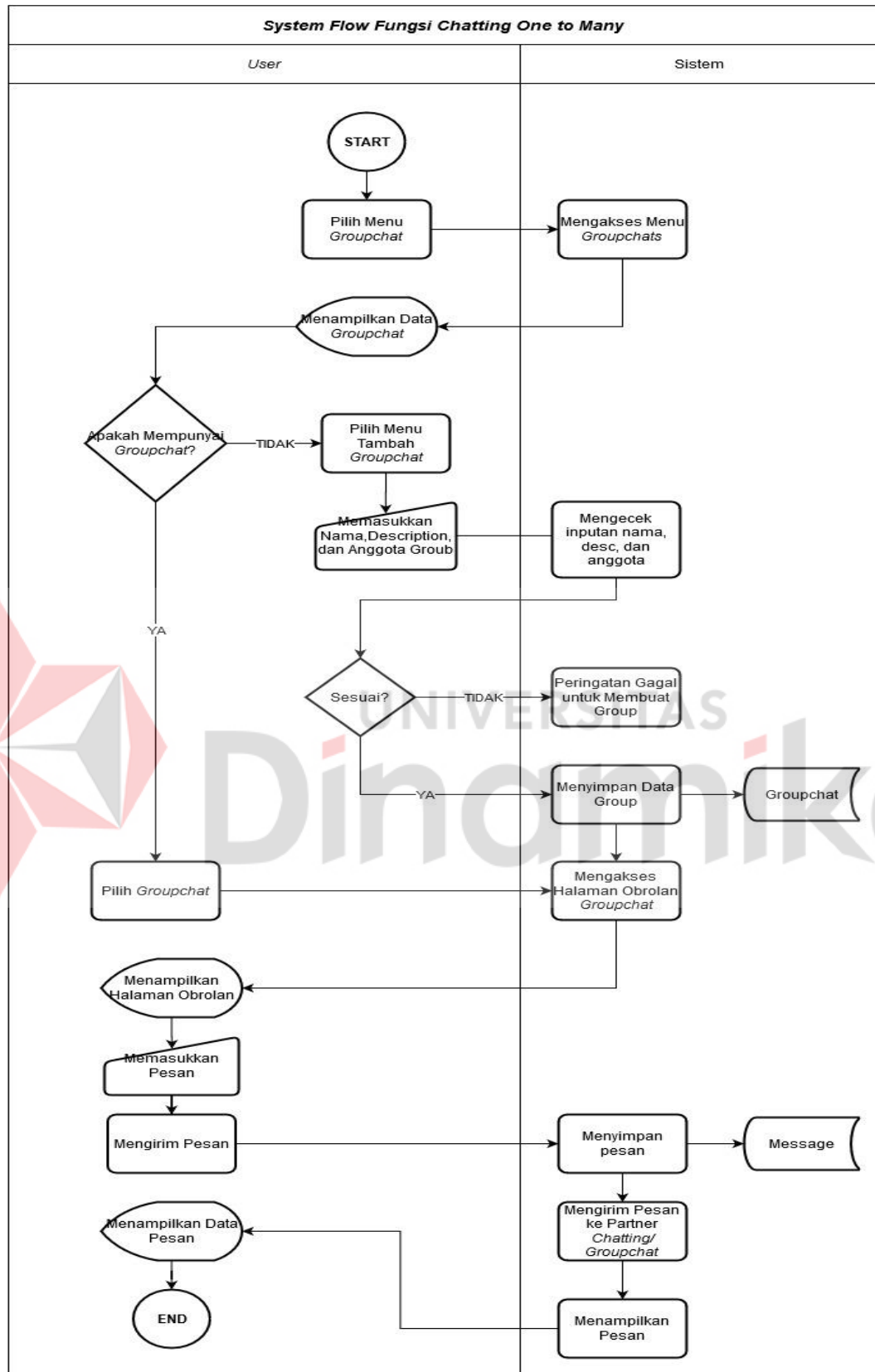
Gambar 4.5 System Flow Diagram Fungsi One to One Chat/ Personal Chat

- *System Flow Diagram Fungsi Chatting One to Many*

System flow chatting one to many merupakan proses dimana *user* dapat melakukan *chatting* dengan satu atau lebih *user* lain. Proses ini berawal dari *user* membuat *groubchat* dan menambahkan teman pada *groupchat*. Setelah itu *user* menulis pesan dan sistem kemudian akan menyimpan data pesan ke *database* dan mengirimkan data pesan ke *groupchat* secara langsung.



UNIVERSITAS
Dinamika




Gambar 4.6 System Flow Diagram Fungsi One to Many Chat/ Group Chat

4.2.2 Desain Basis Data

Desain basis data bertujuan untuk mengetahui kebutuhan data yang akan diperlukan dalam pembuatan sebuah sistem informasi. Pada aplikasi *chatting* yang akan dibuat penulis menggunakan mongoDB. Mongo DB adalah *database* noSQL yang menggunakan struktur data JSON dalam penyimpanan data. Berikut ini adalah desain basis data dari aplikasi *chatting* yang akan dibuat:

- *User Collection*

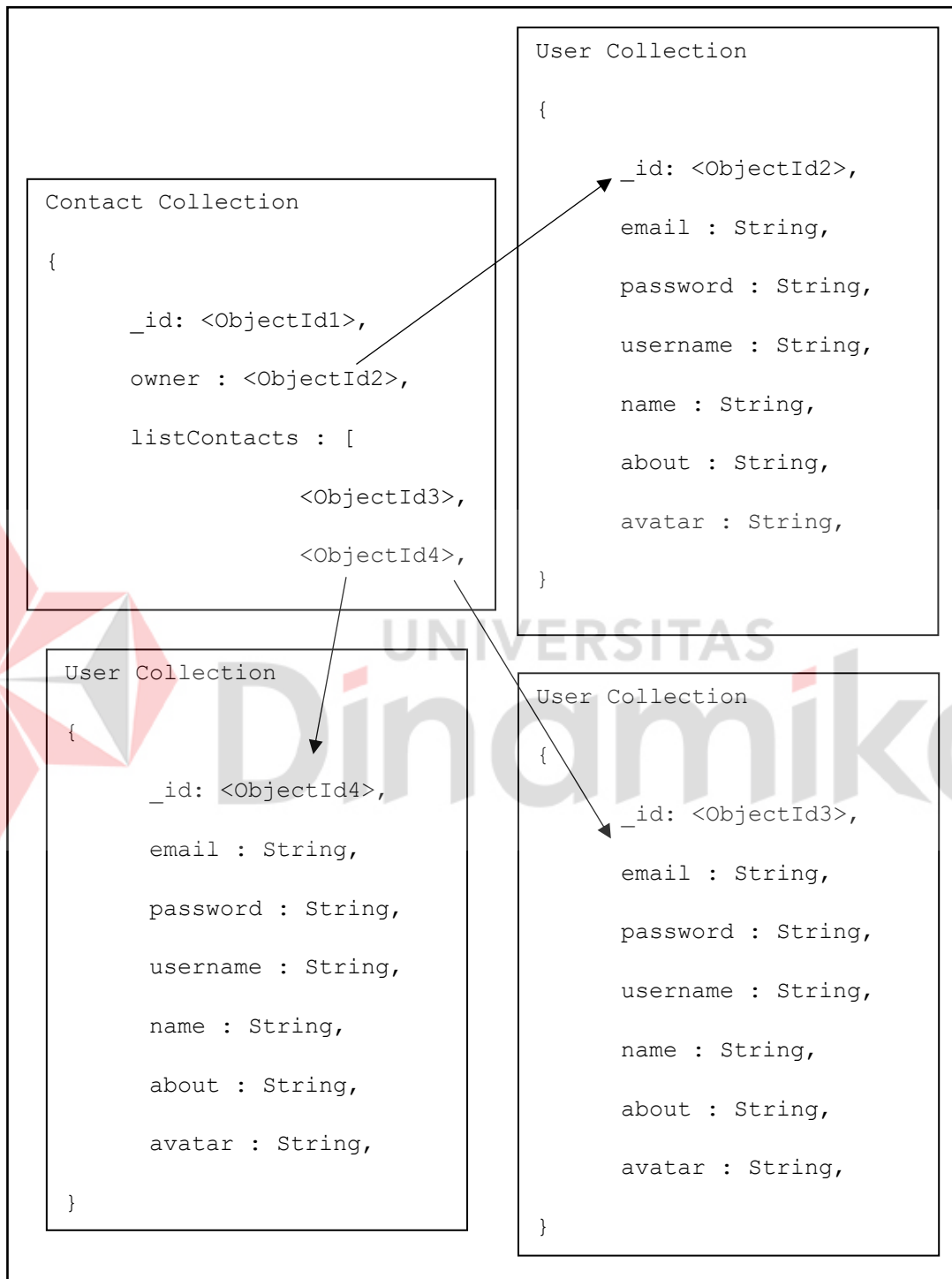


```
{  
  
  _id: <ObjectId>,  
  email : String,  
  password : String,  
  username : String,  
  name : String,  
  about : String,  
  avatar : String,  
  
}
```

Gambar 4.7 *Users Collection*

Users collection berisi data user. Terdiri dari beberapa *key-value*, yaitu *_ID*, *EMAIL*, *PASSWORD*, *USERNAME*, *NAME*, *ABOUT*, *AVATAR*.

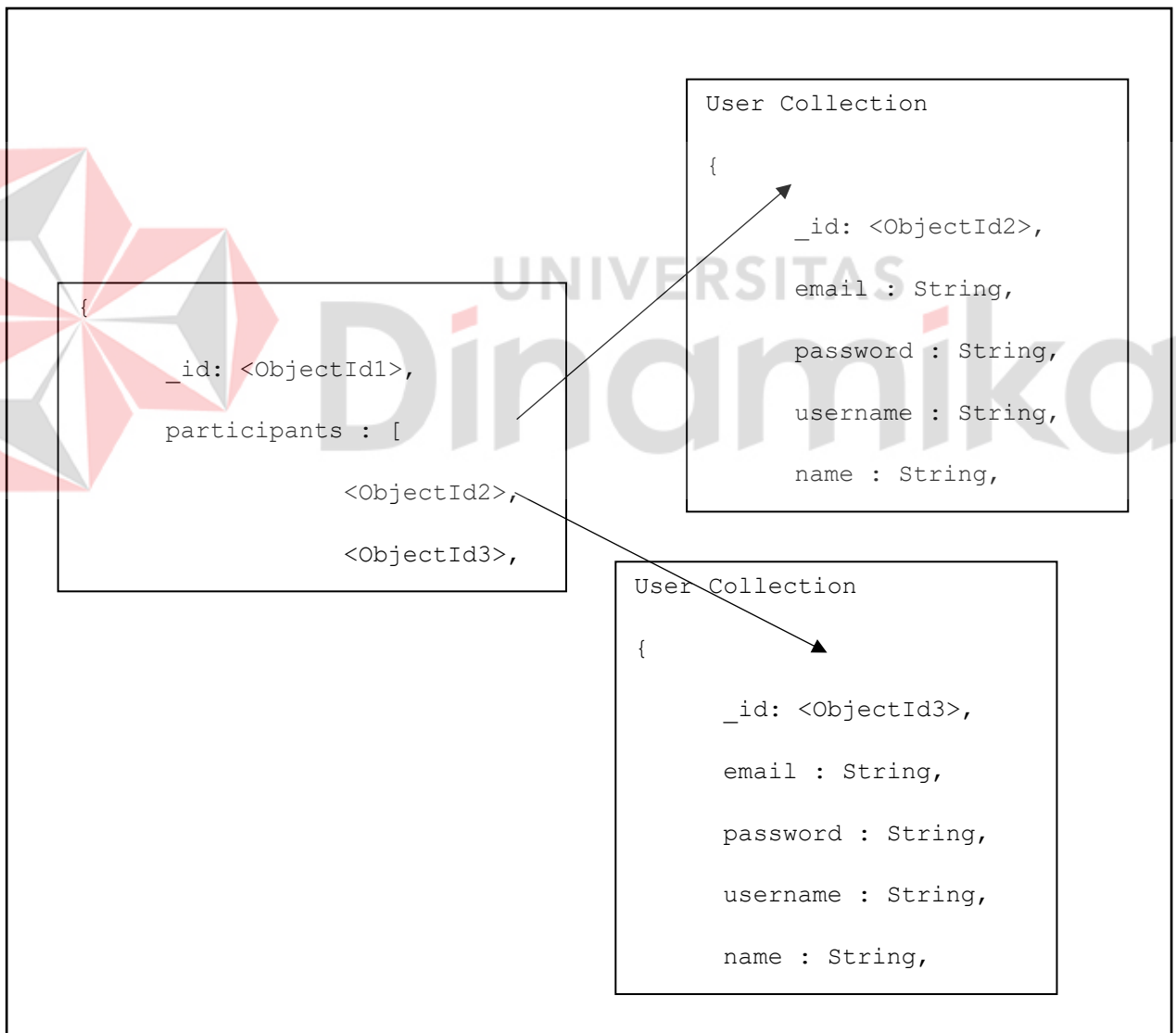
- *Contact Collection*



Gambar 4.8 Contacs Collection

Contacts collection berisi data contact semua user. Terdiri dari beberapa *key-value*, yaitu *_ID*, *OWNER*, dan *LISTCONTACTS*. *Object owner* menjelaskan tentang kepemilikan data *contact*, *object owner* akan melakukan refrensi pada *user collection*. Sedangkan *object listContact* adalah data semua *contact* yang dimiliki oleh seorang *user*, *object* ini juga akan merefrensi pada *user collection*

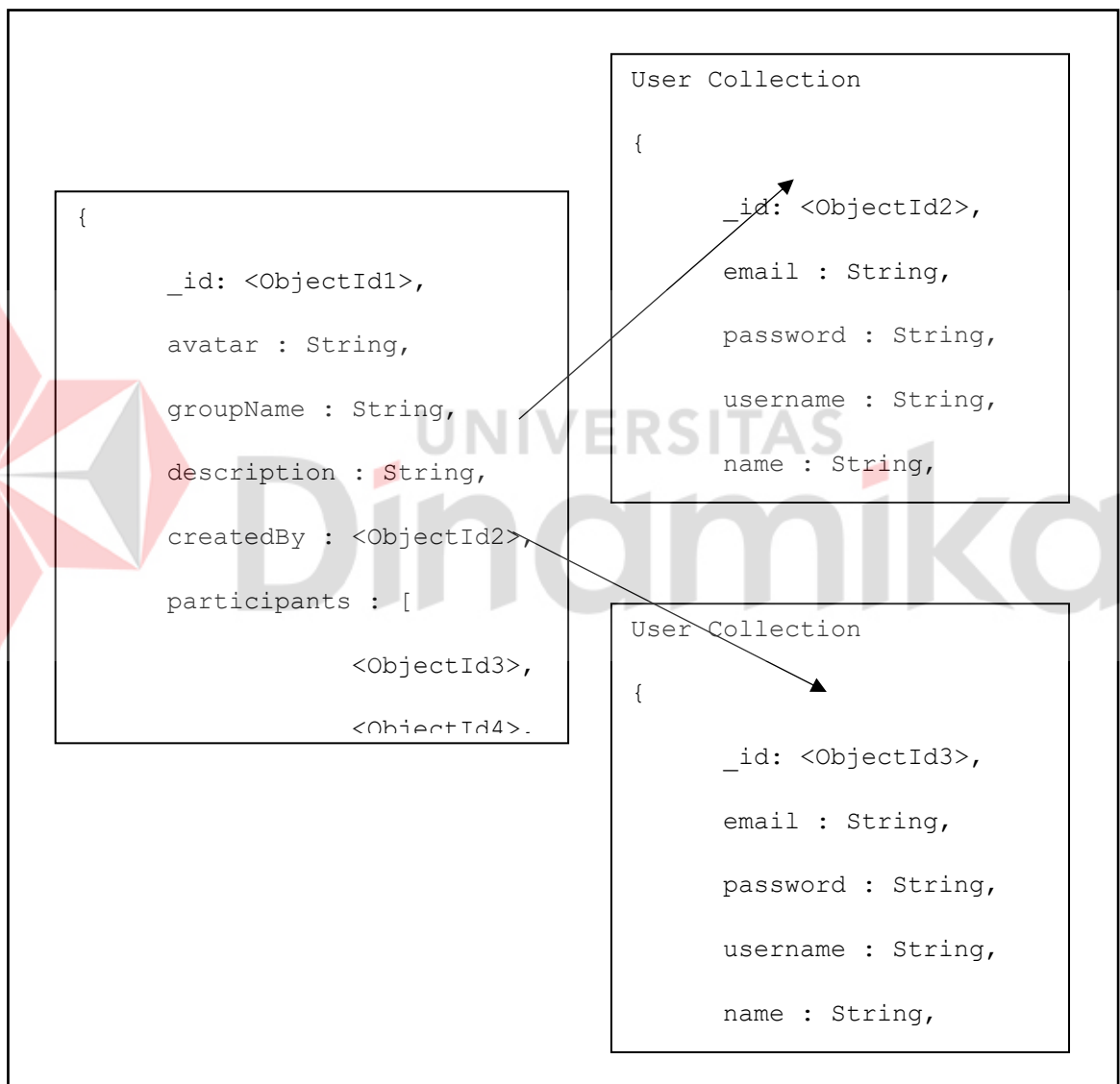
- *Conversations Collection*



Gambar 4.9 *Conversations Collection*

dari dua *key-value*, yaitu *_ID*, dan *PARTICIPANTS*. *Object participants* menjelaskan tentang siapa saja yang sedang melakukan percakapan, *object participants* akan melakukan refrensi pada *user collection*.

- *Group Chats Collection*

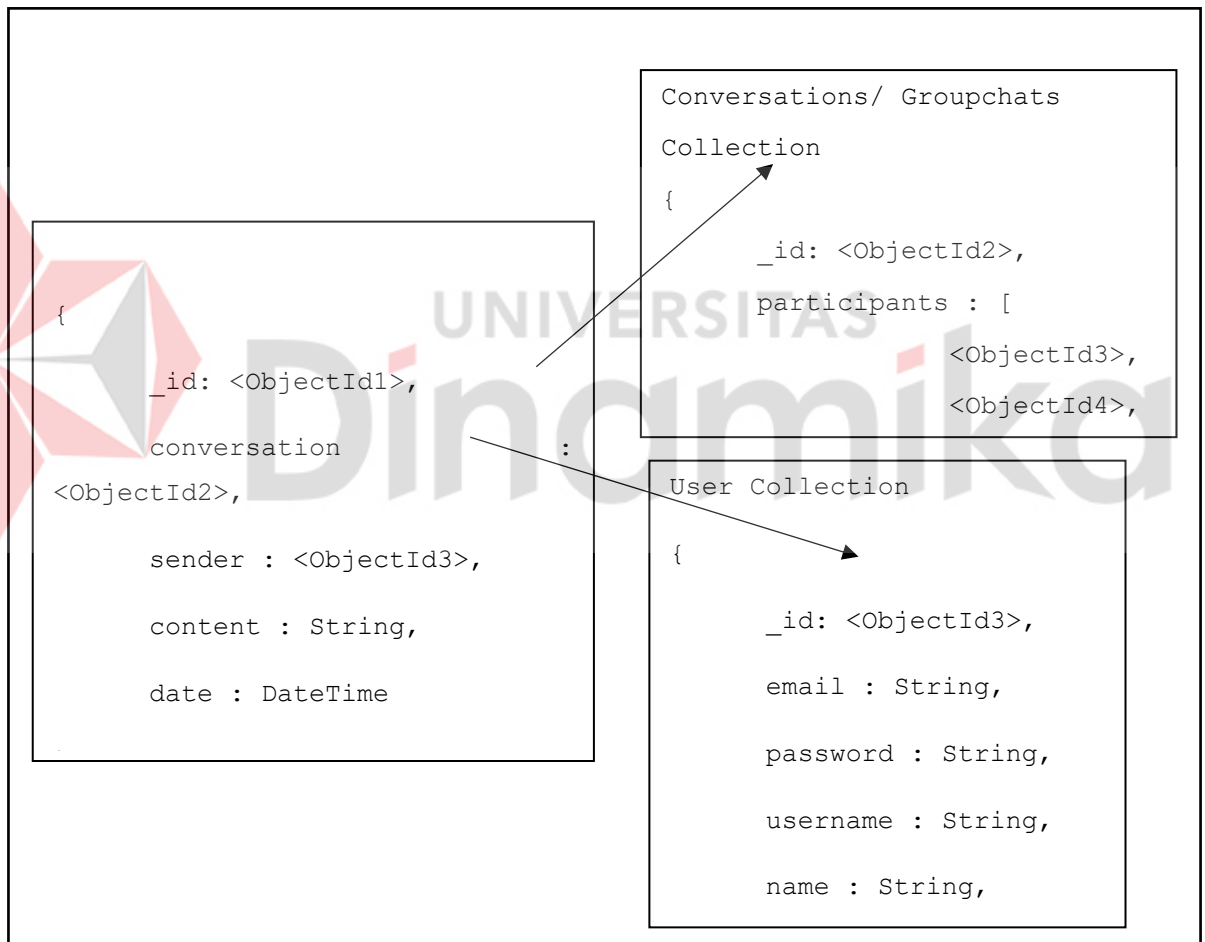


Gambar 4.10 *Group Chats Collection*

Groupchats collection berisi data percakapan *one to many* antara dua atau lebih *user/ groupchat*. Terdiri dari beberapa *key-value*, yaitu *_id*,

avatar, groupName, description, createdBy dan participants. Object createdBy menjelaskan tentang siapa yang membuat grupchat tersebut, object createdBy akan melakukan refrensi pada user collection. Object participants menjelaskan tentang siapa saja yang menjadi anggota grupchat tersebut, object participants akan melakukan refrensi pada user collection.

- *Messages Collection*



Gambar 4.11 *Messages Collection*

4.2.3 Desain API

Desain API akan ditentukan fungsi fungsi yang akan menjadi basis dari penerapan API dengan metode REST berdasarkan hasil dari analisis sistem. Desain ini bertujuan untuk memfokuskan pengembangan sistem supaya tidak keluar dari lingkup pengembangan API yang telah direncanakan. Untuk mempermudah desain API, penulis akan membagi desain menjadi lima modul, yaitu

1. Modul *user*, pada modul *user* telah ditentukan API berdasarkan hal hal yang berhubungan dengan *user*. Hal ini mencakup seluruh proses manajemen *user*, mulai dari *authentifikasi* user dan manajemen user.

Tabel 4.5 API Model *User*

NO	PROSES	API	
		METHOD	PATH
1	Signup	POST	/api/v1/users/signup
2	Signin	GET	/api/v1/users/signin
3	Menampilkan data <i>user</i>	GET	/api/v1/users/:username
4	Menampilkan data <i>user</i> yang sedang login (<i>Profile User</i>)	GET	/api/v1/users/profile
5	Mengunggah avatar	POST	/api/v1/users/profile/avatar

2. Modul kontak, pada modul kontak telah ditentukan API berdasarkan hal hal yang berhubungan dengan kontak. Hal ini mencakup seluruh proses manajemen kontak, mulai dari menambah dan menghapus kontak.

Tabel 4.6 API Model Kontak

NO	PROSES	API	
		METHOD	PATH
1	Menampilkan data kontak	GET	/api/v1/contacts
2	Menambah data kontak	POST	/api/v1/contacts
3	Menampilkan detail data kontak	GET	/api/v1/contacts/:id
4	Menghapus data kontak	DELETE	/api/v1/contacts/:id

3. Modul *conversation*, pada modul *conversation* telah ditentukan API berdasarkan hal hal yang berhubungan dengan *conversation*. Hal ini mencakup seluruh proses manajemen *conversation*, mulai dari menambah dan menghapus *conversation*.

Tabel 4.7 API Model Conversations

NO	PROSES	API	
		METHOD	PATH
1	Menampilkan data <i>conversation</i>	GET	/api/v1/conversations
2	Menambah data conversation	POST	/api/v1/conversations
3	Menampilkan detail data conversation	GET	/api/v1/conversations/:id
4	Menghapus data conversation	DELETE	/api/v1/conversations/:id

4. Modul *groupchat*, pada modul *groupchat* telah ditentukan API berdasarkan hal hal yang berhubungan dengan *groupchat*. Hal ini mencakup seluruh proses manajemen *groupchat*, mulai dari menambah, mengubah dan menghapus *groupchat*.

Tabel 4.8 API Model *Group Chats*

NO	PROSES	API	
		METHOD	PATH
1	Menampilkan data <i>group chat</i>	GET	/api/v1/groupchats
2	Menambah data <i>group chat</i>	POST	/api/v1/ groupchats
3	Menampilkan detail data <i>group chat</i>	GET	/api/v1/ groupchats /:id
4	Mengubah data <i>group chat</i>	PUT	/api/v1/ groupchats /:id
5	Menghapus data <i>group chat</i>	DELETE	/api/v1/ groupchats /:id
6	Menampahkan data partisipan kedalam <i>group chat</i>	POST	/api/v1/ groupchats/:id/participants
7	Menghapus data partisipan kedalam <i>group chat</i>	DELETE	/api/v1/ groupchat /:id/participants

5. Modul pesan, pada modul pesan telah ditentukan API berdasarkan hal hal yang berhubungan dengan pesan. Hal ini mencakup seluruh proses manajemen pesan, mulai dari menambah, mengubah dan menghapus pesan

Tabel 4.9 Tabel API Model *Messages*

NO	PROSES	API	
		METHOD	PATH
1	Menampilkan data pesan	GET	/api/v1/messages
2	Mengirim data pesan	POST	/api/v1/ messages

4.3 Implementasi Sistem

Tahapan selanjutnya setelah merancang sistem adalah mengimplementasikan sistem. Application programming interface chatting ini diimplementasikan menggunakan *virtual private server* (VPS) yang telah dipersiapkan sebelumnya.

Adapaun spesifikasi nya adalah sebagai berikut

- Provider Digitalocean server Singapura
- Plan Price \$5 per Bulan dengan spesifikasi server : 1GB/ 1 CPU, 25 GB SSD Disk, 1000 GB transfer

Perangkat Lunak yang digunakan antara lain :

- Web Server Nginx
- Node
- Text Editor (VS Code)
- Post Man

4.4 Pembahasan Sistem

Pada tahap hasil dan pembahasan, peneliti akan menjelaskan hasil pembuatan *Application Programming Interface* (API) menggunakan gaya arsitektur

Representational State Transfer (REST) dalam pengembangan sistem informasi *chatting*

4.4.1 Struktur API

Pembuatan sistem berbasis *Application Programming Interface* (API) harus memiliki sebuah standar agar dapat mengurangi kesalahan – kesalahan dalam proses *developing*. Bayangkan ketika sebuah API didesign dengan kembalian data yang tidak konsisten, nama path yang tidak merepresentasikan dari fungsi API, data yang dikembalikan terlalu berbelit belit dalam artian seharusnya bisa 1x panggil tapi harus 2x panggil dengan path API yang berbeda. Maka dari itu sebuah API harus didesain dengan beberapa persyaratan.

- ***Restful API***

Dalam penentuan metode transfer data API yang baik, penulis menggunakan gaya arsitektur REST dimana memanfaatkan metode HTTP *verb* untuk mengartikan suatu perintah API, yaitu :

1. GET untuk menyediakan akses untuk membaca sumber data
2. POST untuk menambah data baru
3. PUT untuk memperbaharui data yang tersedia
4. DELETE untuk menghapus data

- ***Penamaan Uniform Resource Locator (URL)***

Dalam penentuan nama URL pada API, penulis membuat beberapa aturan dalam penamaan. Ada beberapa macam aturan yang dapat membantu untuk membangun sistem API yang baik, yaitu :

1. Presentasi dari Fungsi API

Presentasi Fungsi API berguna untuk penulis agar tidak menimbulkan kebingungan ketika ingin menggunakan API.

2. *Use Plural Forms*

Penggunaan kata jamak pada penamaan URL. Hal ini berguna agar menjadi generalisasi penamaan URL pada pengembangan sistem API.

3. *Use Prefix*

Prefix adalah sebuah penamaan *statis* yang wajib ada pada semua URL API yang akan dikembangkan. Hal ini bertujuan untuk mengetahui URL yang kita punya adalah sebuah API.

4. *Versioning*

Versioning digunakan ketika pengembang ingin mengembangkan API ke versi yang baru dan masih support untuk API yang lama. Teknik *versioning* API adalah dengan menambahkan path pada API /v2 untuk API terbaru dan /v1 untuk API lama dan seterusnya. Hal ini dapat diatur melalui route di level aplikasi.

5. Menggunakan dua URL pada setiap *resource*

/users → menunjukkan semua data *user*

- *Resource* dengan banyak data
- *Resource* dengan satu data

/users/:id → menunjukkan satu data *user* berdasarkan

6. **Menjaga nama URL sependek mungkin**, dalam artian tidak boleh ada tiga atau lebih penamaan *resource* pada URL.
7. **Menggunakan nama benda untuk nama URL**, hindari penggunaan kata kerja

Tabel 4.10 Penamaan URL

NO	URL	MODUL
1	api/v1/users	<i>User</i>
2	api/v1/contacts	Kontak
3	api/v1/conversations	<i>Conversation</i>
4	api/v1/groupchats	<i>Group Chat</i>
5	api/v1/messages	<i>Message</i>

• HTTP Status Code

HTTP *status code* adalah sebuah kode untuk membantu kita mengenali response balik dari API. Pengembangan sistem API yang baik haruslah mencantumkan HTTP *status code* pada kembalian data API mereka. Berikut ini adalah contoh HTTP *status code* yang sering digunakan dalam pengembangan REST API

1. 200 OK

Perintah yang dikirim ke server benar dan berhasil dijalankan

2. 201 *Created*

Perintah yang dikirim ke server berhasil membuat *resource* baru

3. 400 *Bad Request*

Perintah yang dikirim ke server berisi isian yang salah

4. 401 *Unauthorized*

Pengirim perintah kehilangan *token authentication*, sehingga membuat mereka tidak mempunyai hak akses kedalam *resource* yang dituju

5. 403 *Forbidden*

Pengirim perintah tidak mempunyai hak akses ke dalam *resource* yang dituju

6. 404 *Not Found*

Resource yang dituju tidak ditemukan dalam server

7. 409 *Conflict*

Perintah yang dikirim ke server mengalami duplikasi atau isian yang salah.

8. 500 *Internal Server Error*


Server atau potongan kode dalam *resource* mengalami kesalahan.

- **Error Messages**

Error Message membantu dan memudahkan para pengembang untuk mengetahui dan mengenali kesalahan seperti *missing parameter*, *error parameter validation*, *error server*, dll. Berikut ini adalah contoh *error message* yang digunakan dalam pengembangan API

```
{
  "errors": [
    {
      "msg": "Invalid value",
      "param": "username",
      "location": "body"
    }
  ]
}
```

Gambar 4.12 *Error Message Missing Parameter*



```
{
  "errors": [
    {
      "value": "bastiangmail.com",
      "msg": "Invalid value",
      "param": "email",
      "location": "body"
    }
  ]
}
```

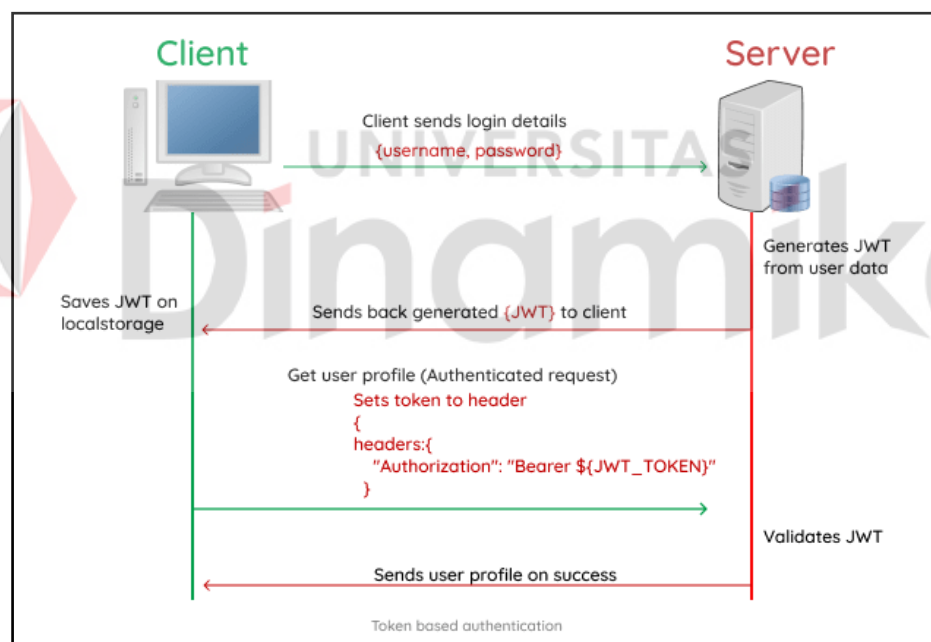
Gambar 4.13 *Error Message Invalid Parameter*

```
{
  "status" : 404,
  "success" : false,
  "message" : Server not Found
}
```

Gambar 4.14 *Error Message Server*

- **Authentication**

Pada pengembangan sebuah sistem, biasanya *authentication* dan *authorization* akan menggunakan sebuah session untuk memastikan siapa yang masuk dalam sistem. Namun dalam pengembangan sistem berbasis API, untuk melakukan hal ini dibutuhkan metode yang berbeda. Pengecekan pada API menggunakan sebuah token, dimana token ini akan dikirim melalui header request. Dalam penelitian ini penulis menggunakan JSON



Web Token “JWT” atau biasa disebut “jot”.

4.4.2 Struktur Kembali

Pada pengembangan berbasis API pasti akan mengembalikan sebuah *resource* kembali. Penulis juga akan mendefinisikan standar

struktur kembalian API agar dapat mudah dipahami oleh orang lain. Pada penelitian ini tipe data yang akan digunakan dalam kembalian resource data API adalah Javascript Object Notation (JSON), struktur kembalian akan berupa:

- a) HTTP status code
- b) Status success fungsionalitas (True OR False)
- c) Pesan request

Jika fungsionalitas berhasil akan menampilkan data yang diminta oleh client.

4.4.3 Implementasi *Socket.io*

Socket.io merupakan sebuah *library javascript* yang membantu dalam pembuatan aplikasi web yang *realtime*, dengan menggunakan *socket.io* kita dapat menghubungkan antara client dan server dapat terjadi secara *bidirectional* (dua arah). Implementasi *socket.io* pada sistem informasi *chatting* akan digunakan pada pengiriman dan penerimaan pesan, baik *personal chat* atau *group chat*.

```
var express = require('express');
var app = express();
var server = require('http').createServer(app);
var io = require('socket.io').listen(server);
global.io = io;
```

Gambar 4.16 *Dependency* pada File *Socket.js*

Untuk menggunakan *Socket* diperlukan sebuah dependency yaitu *socket.io*, dependency ini akan berguna untuk memanggil beberapa perintah socket yang kita butuhkan.



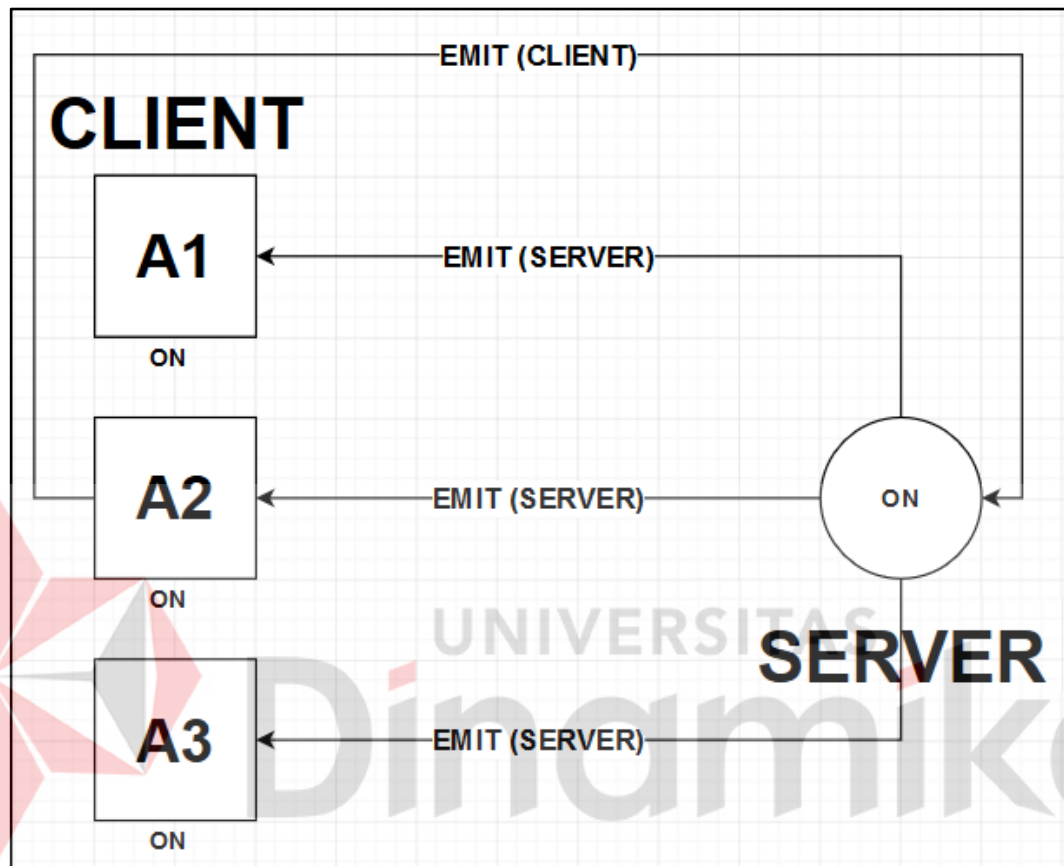
```
io.on('connection', (socket) => {  
  
  console.log(`Socket ${socket.id} connected.`);  
  
  // User Joined Function  
  
  // Send Message Function  
  
  // User Typing Function  
  
  // User Stop Typing Function  
  
  // User Left Function  
  
})
```

Gambar 4.17 *Script Function Socket.io*

Penggunaan *Socket.io* akan melibatkan sisi server dan client. Secara sederhana, *Socket.io* memiliki dua function inti yaitu on dan emit.

- a) On()
 - Sisi *server*: untuk mendeteksi adanya aksi tertentu dari client
 - Sisi *client*: untuk siap menerima response dari server
- b) Emit()

- Sisi *server*: untuk mengirim response ke client
- Sisi *client*: untuk mengirim request ke server



Gambar 4.18 Cara Kerja *Socket.io*

Secara sederhana cara kerja *Socket.io* adalah sebagai berikut:

emit dari *client* akan mengirim *request* lalu *on* di *server* akan menerima *request* dari *client* setelah itu *emit* dari *server* mengirim *response* ke *client* dan *on* di *client* akan menerima *response* dari *client*. Pada penelitian ini penulis membuat beberapa *request* di *server* untuk dikirim ke *client* menggunakan *Socket.io*, yaitu:

- **User Joined**

```
// User Joined Function

socket.on('userJoined', (user) => {

    socket.broadcast.emit('userJoined', user);

    console.log(user + " joined");

});
```

Gambar 4.19 *User Joined Request Socket.io*

Pada *request* ini jika *client* “A1” masuk, maka pada waktu yang bersamaan *client* lain akan mendapatkan status bahwa *client* “A1” telah masuk kedalam ruangan

- **Send Messages**

```
// Send Message Function

socket.on('newMessage', (sender, content,
conversationId, date) => {

    io.emit('newMessage', {

        sender: sender,

        content: content,

        conversationId: conversationId,

        date: date

    })

});
```

Gambar 4.20 *Send Messages Request Socket.io*

Pada *request* ini jika *client* “A1” mengirim sebuah pesan, maka pada waktu yang bersamaan *client* lain akan menerima pesan tersebut.

- **User Typing**

```
// User Typing Function
socket.on('typing', (username) => {
  socket.broadcast.emit('typing', {
    username: username
  });
  console.log(username + " typing...");
});
```

Gambar 4.21 *User Typing Request Socket.io*

Pada *request* ini jika *client* “A1” mengetik sesuatu pada kotak *text-box*, maka pada waktu yang bersamaan *client* lain akan melihat status “*client* ‘A1’ sedang mengetik”

- **User Stop Typing**

```
// User Stop Typing Function
socket.on('stop typing', (username) => {
  socket.broadcast.emit('stop typing', {
    username: username
  });
});
```

Gambar 4.22 *User Stop Typing Request Socket.io*

Pada *request* ini jika *client* “A1” berhenti mengetik sesuatu pada kotak *text-box*, maka pada waktu yang bersamaan *client* lain tidak akan melihat status “*client* ‘A1’ sedang mengetik”

- **User Left**

```
// User Left Function
socket.on('disconnect', (user) => {
  console.log(`Socket${socket.id}
  disconnected.`);
});
```

Pada *request* ini jika *client* “A1” keluar, maka pada waktu yang bersamaan *client* lain akan mendapatkan status bahwa *client* “A1” telah keluar ruangan



UNIVERSITAS
Dinamika

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dapat disimpulkan beberapa hal sebagai berikut :

1. Pengembangan sistem berbasis API dapat mempercepat pembuatan sebuah sistem informasi karena logic pada sistem akan dilakukan terpisah dengan logic pada user interface
2. Metode REST dapat diimplementasikan kedalam perancangan API untuk pengembangan sistem *chatting*

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan dapat disarankan beberapa hal sebagai berikut :

1. Pengembangan sistem berbasis API harus memperhatikan struktur penamaan URL, Kembalikan, dan Authentication
2. Untuk memanfaatkan back-end dari sistem chatting menggunakan API ini harus memiliki tampilan antarmuka. Untuk itu dibutuhkan pengembangan lebih lanjut terutama untuk antarmuka yang menjadi media interaksi antara pengguna dengan back-end server

DAFTAR PUSTAKA

Arslan, M. (2016, Oktober 19). *Memulai Pembuatan Aplikasi Web dengan Express.js (1): Instalasi dan Pengenalan*. Retrieved Mei 17, 2020, from Codepolitan: <https://www.codepolitan.com/memulai-pembuatan-aplikasi-web-dengan-express-js-1-instalasi-dan-pengenalan>

Azamuddin, M., & Mukhlisin, H. (2018). *Laravel : the PHP Framework for Web Artisans*. Jakarta.

Deswitansyah, I. (2017, 13 Juli). *Mengenal Apa itu Socket.io*. Retrieved Mei 17, 2020, from Dumetschool: <https://www.kursuswebsite.org/mengenal-apa-itu-socket-io/>

Doglio, F. (2018). *REST API Development with Node.js*. Canelones: Apress.

Jiri Hradil, Vilém Sklenak. (2017). Practical Implementation of 10 Rules for Writing REST APIs. *JOURNAL OF SYSTEMS INTEGRATION*, 45.

JSON.org. (n.d.). *Introduce JSON*. Retrieved Mei 17, 2020, from json.org: <https://json.org/json-id.html>

Lutfi, F. (2017, Januari 19). *Mengenal Node.js*. Retrieved Mei 17, 2020, from Codepolitan: <https://www.codepolitan.com/mengenal-nodejs-5880234fe9ae3>

Pranata, B. A. (2017). *SKRIPSI: Perancangan Application Programming Interface (API) Menggunakan Gaya arsitektur Representational State Transfer (REST) Untuk Pengembangan Sistem Informasi Administrasi Pasien Klinik Perawatan Kulit*. Lampung: Universitas Negeri Lampung.

Saputra, M. (2016, Oktober 24). *Mengenal MongoDB Database NoSql*. Retrieved Mei 17, 2020, from Erabelajar: <http://developer.erabelajar.com/mengenal-mongodb-database-nosql/>



UNIVERSITAS
Dinamika