

DPPL-003

DESKRIPSI PERANCANGAN PERANGKAT LUNAK

Aplikasi SocialLife

untuk:

-

Dipersiapkan oleh:

Kelompok 3

Diva Annisa Febecca - 1301204302

Johannes Raphael Nandaputra - 1301204243

Muhammad Naufal Abdillah - 1301201586


Muhammad Mufid Utomo - 1301204441

Ricardo Hamonangan - 1301204201

Program Studi S1 Informatika

Fakultas Informatika, Universitas Telkom

Jl. Telekomunikasi 1, Dayeuhkolot Bandung

	Prodi S1- Informatika Universitas Telkom	Nomor Dokumen		Halaman
		<i>DPPL-xx</i> <xx:no grp>		45
		Revisi	<nomor revisi>	Tgl: <isi tanggal>

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	
G	

INDEX TGL	-	A	B	C	D	E	F	G
Ditulis oleh								
Diperiksa oleh								
Disetujui oleh								

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi

Daftar Isi

Daftar Isi	3
Daftar Tabel	6
Daftar Gambar	7
1. Pendahuluan	7
1.1 Tujuan Penulisan Dokumen	8
1.2 Lingkup Masalah	8
1.3 Definisi dan Istilah	8
1.4 Referensi	9
2 Perancangan Global	10
2.1 Rencana Lingkungan Implementasi	10
2.2 Deskripsi Arsitektur Perangkat Lunak	10
3 Perancangan Rinci	12
3.1 Realisasi Use Case	12
3.1.1 Use Case #1 Register	12
3.1.1.1 Use Case Scenario #1 Register	12
3.1.1.2 UI Design dan Deskripsi Objek UI #1 Register	13
3.1.1.3 Identifikasi Object dan Tipe nya #1 Use Case Register	14
3.1.1.4 Class Diagram #1 Register	14
3.1.1.5 Sequence Diagram #1 Register	15
3.1.2 Use Case #2 Login	15
3.1.2.1 Use Case Scenario #2 Login	15
3.1.2.2 UI Design dan Deskripsi Objek UI #2 Login	16
3.1.2.3 Identifikasi Object Baru & Tipe Kelas #2 Login	17
3.1.2.5 Sequence Diagram #2 Login	18
3.1.3 Use Case #3 Log Out	18
3.1.3.1 Use Case Scenario #3 Log Out	18
3.1.3.2 UI Design dan Deskripsi Objek UI #3 Log Out	19
3.1.3.3 Identifikasi Object Baru & Tipe Kelas #3 Log Out	20
3.1.3.4 Class Diagram #3 Log Out	20
3.1.3.5 Sequence Diagram #3 Log Out	20
3.1.4 Use Case #4 Chat	21
3.1.4.1 Use Case Scenario #4 Chat	21
3.1.4.2 UI Design dan Deskripsi Objek UI #4 Chat	21
3.1.4.3 Identifikasi Object Baru & Tipe Kelas #4 Chat	22

3.1.4.4 Class Diagram #4 Chat	23
3.1.4.5 Sequence Diagram # 4 Chat	23
3.1.5 Use Case #5 Add Friends	23
3.1.5.1 Use Case Scenario #5 Add Friends	23
3.1.5.2 UI Design dan Deskripsi Objek UI #5 Add Friends	24
3.1.5.3 Identifikasi Object Baru & Tipe Kelas #5 Add Friends	25
3.1.5.4 Class Diagram #5 Add Friends	25
3.1.5.5 Sequence Diagram # 5 Add Friends	26
3.1.6 Use Case #6 Send Alert	26
3.1.6.1 Use Case Scenario #6 Send Alert	26
3.1.6.2 UI Design dan Deskripsi Objek UI #6 Send Alert	27
3.1.6.3 Identifikasi Object Baru & Tipe Kelas #6 Send Alert	28
3.1.6.4 Class Diagram #6 Send Alert	28
3.1.6.5 Sequence Diagram # 6 Send Alert	29
3.1.7 Use Case #7 Remind to Contact	29
3.1.7.1 Use Case Scenario #7 Remind to Contact	29
3.1.7.2 UI Design dan Deskripsi Objek UI #7 Remind to Contact	30
3.1.7.3 Identifikasi Object Baru & Tipe Kelas #7 Remind to Contact	30
3.1.7.4 Class Diagram #7 Remind to Contact	31
3.1.7.5 Sequence Diagram#7 Remind to Contact	31
3.1.8 Use Case #8 Customization	32
3.1.8.1 Use Case Scenario #8 Customization	32
3.1.8.2 UI Design dan Deskripsi Objek UI #8 Customization	32
3.1.8.3 Identifikasi Object Baru & Tipe Kelas #8 Customization	34
3.1.8.4 Class Diagram #8 Customization	34
3.1.8.5 Sequence Diagram #8 Customization	35
3.2 Diagram Kelas Keseluruhan	36
3.3 Perancangan Data / Basis Data	38
3.4 Perancangan Algoritma dan/atau Query	38
4. Matriks Keruntutan (Requirement Traceability Matrix)	43

Daftar Tabel

Tabel 1.1 Definisi dan Istilah	8
Tabel 2.1 Deskripsi Arsitektur Perangkat Lunak	11
Tabel 3.1 Realisasi Use Case	12
Tabel 3.1.1.2 Deskripsi Objek UI	13
Tabel 3.1.1.3 Tabel Object Perancangan	14
Tabel 3.2 Deskripsi Objek UI	16
Tabel 3.1.2.3 Tabel Object Perancangan	17
Tabel 3.3 Deskripsi Objek UI	19
Tabel 3.1.3.3 Tabel Object Perancangan	20
Tabel 3.4 Deskripsi Objek UI	21
Tabel 3.1.4.3 Tabel Object Perancangan	22
Tabel 3.5 Deskripsi Objek UI	24
Tabel 3.1.5.3 Tabel Object Perancangan	25
Tabel 3.6 Deskripsi Objek UI	27
Table 3.1.6.3 Tabel Object Perancangan	28
Tabel 3.7 Deskripsi Objek UI	30
Tabel 3.1.7.3 Tabel Object Perancangan	30
Tabel 3.8 Deskripsi Objek UI	32
Tabel 3.1.8.3 Tabel Object Perancangan	34

Daftar Gambar

Gambar 2.2 Deskripsi Arsitektur Perangkat Lunak	10
Gambar 3.1.1 UI Design Page Register	13
Gambar 3.1.2 UI Design Page Login	16
Gambar 3.1.3 UI Design Page Log Out	19
Gambar 3.1.4 UI Design Page Chat	21
Gambar 3.1.5 UI Design Page Add Friends	24
Gambar 3.1.6 UI Design Page Send Alert	27
Gambar 3.1.7 UI Design Page Remind to Contact	30
Gambar 3.1.8 UI Design Page Customization	33
Gambar 3.2 Diagram Kelas Keseluruhan	36
Gambar 3.3 Perancangan Data / Basis Data	38

1. Pendahuluan

1.1 Tujuan Penulisan Dokumen

Deskripsi Perancangan Perangkat Lunak (DPPL) ini adalah dokumen yang berisikan penjelasan perancangan dan pengembangan pada sistem aplikasi SocialLife. Dokumen ini akan dijelaskan secara detail dari desain diagram dan sequence diagram yang akan diimplementasikan untuk membangun sistem. Tujuan pembuatan dokumen ini adalah sebagai alat bantu dalam mendeskripsikan teknik kerja perangkat lunak yang akan dibangun dan dokumen ini juga digunakan sebagai acuan dalam pembangunan perangkat lunak dan evaluasi akhir perangkat lunak.

1.2 Lingkup Masalah

Aplikasi SocialLife dibuat sebagai alternatif dari media sosial yang sudah ada di mana aplikasi ini tidak akan mengeksploitasi pengguna sebagaimana media sosial yang lain. Sehingga SocialLife dibuat sebagai produk baru karena aplikasi ini bukan menggantikan aplikasi yang sudah ada melainkan menjadi alternatif baru. Antarmuka aplikasi ini bisa diakses lewat website sehingga bisa diakses lewat device yang bisa membuka browser. Lalu, aplikasi akan terhubung dengan satu *database* yang menyimpan semua data pengguna dan data yang penting akan dienkripsi untuk melindunginya dari pihak yang tidak bertanggungjawab.

1.3 Definisi dan Istilah

Dalam dokumen DPPL ini, terdapat beberapa istilah yang digunakan sebagai nama dari bagian sistem. definisi dari istilah tersebut dijelaskan pada tabel berikut.

Tabel 1.1 Definisi dan Istilah

CAPTCHA	Jenis tindakan keamanan yang dikenal sebagai autentikasi tantangan-tanggapan.
Encryption	Proses mengamankan suatu informasi dengan membuat informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan khusus.
Log In	Sebuah proses untuk masuk ke dalam sebuah layanan <i>online</i> dengan memasukkan kode identitas pengguna (ID dan Password).
Log Out	Sebuah proses untuk mengakhiri koneksi.
OS	<i>Operating System</i> merupakan perangkat lunak yang berfungsi sebagai penghubung antara pengguna dan perangkat keras.

Password	Sebuah kata sandi atau kode rahasia yang digunakan untuk masuk ke dalam sebuah akun.
RAM	<i>Random Access Memory</i> merupakan sebuah perangkat penyimpanan data.
Register	Pendaftaran terhadap suatu program yang berfungsi untuk menghubungkan data pribadi dengan program tersebut.
User	Pemakai atau pengguna sebuah aplikasi.

1.4 Referensi

1. Spesifikasi Kebutuhan Perangkat Lunak(SKPL) Aplikasi SocialLife(2020).

2 Perancangan Global

2.1 Rencana Lingkungan Implementasi

Untuk User, aplikasi ini dapat dioperasikan pada berbagai browser, seperti mozilla firefox, google chrome, dan safari, dengan syarat bahwa hardware user terkoneksi dengan internet.

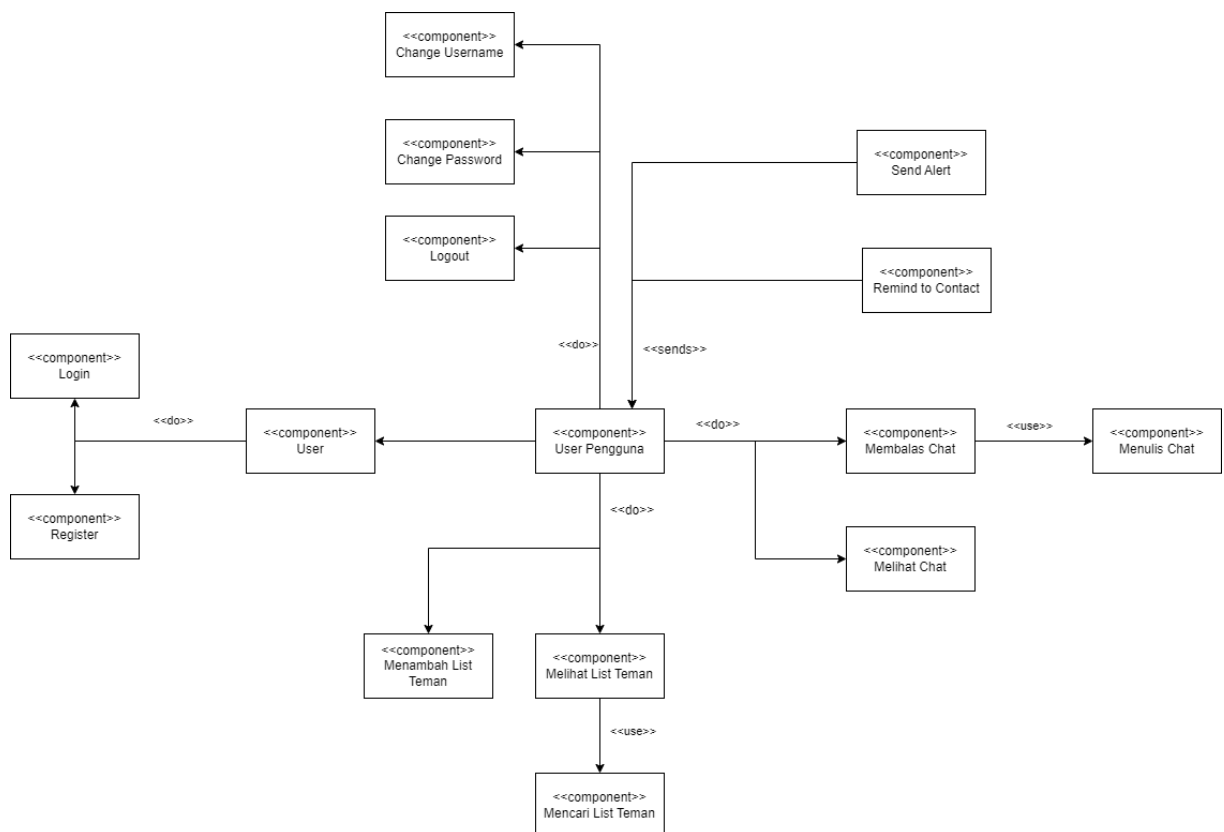
Aplikasi ini akan di develop pada sisi frontend dan backend, dengan tools:

- MERN (MongoDB, Express, React, NodeJS), dengan frontend menggunakan React, dan sisanya adalah tools untuk backend development.

2.2 Deskripsi Arsitektur Perangkat Lunak

Component Diagram :

Gambar 2.2 Deskripsi Arsitektur Perangkat Lunak



Tabel 2.1 Deskripsi Arsitektur Perangkat Lunak

No	Nama Komponen	Keterangan
1.	Login	Menu masuk aplikasi, agar bisa mengakses fitur lainnya
2.	Register	Menu untuk mendaftarkan diri ke database, agar bisa login ke aplikasi
3.	User	seseorang yang menggunakan aplikasi SociaLife
4.	Log Out	Menu untuk mengeluarkan akun dari aplikasi di device masing-masing
5.	Membalas Chat	Halaman untuk membalas pesan yang telah dikirim oleh user lainnya ke device seseorang
6.	Menulis Pesan	Kegiatan User untuk menulis suatu pesan yang akan dikirim ke user lainnya
7.	Melihat Chat	Halaman untuk melihat pesan yang pernah dilakukan oleh antar user
8.	Melihat List Teman	Halaman yang menampilkan list username yang ada di list friend user
9.	Menambah Teman	Halaman untuk menambahkan nama suatu username ke list friend user
10.	Mencari Teman	Halaman untuk mencari nama suatu username di dalam list friend user.
11.	Send Alert	Notifikasi yang menampilkan jika user sudah memakai aplikasi lebih dari 20 menit
12.	Remind To Contact	Notifikasi yang menampilkan jika user belum mengontak temannya selama lebih dari 14 hari
13.	Mengubah Password	Halaman yang memungkinkan user untuk mengubah password yang sudah tersimpan di database
14.	Mengubah Username	halaman yang memungkinkan user untuk mengubah username yang sudah tersimpan di database

3 Perancangan Rinci

3.1 Realisasi Use Case

Berisi TABEL USE CASE sebagai berikut :

Tabel 3.1 Realisasi Use Case

No	Nama Use Case	Deskripsi Use Case
#1	Register	Mendaftarkan akun <i>user</i> ke <i>database</i> .
#2	Login	Memasukan akun <i>user</i> .
#3	Logout	<i>User</i> mengeluarkan akun dari aplikasi.
#4	Chat	<i>User</i> dapat mengirim pesan ke teman yang sudah ditambahkan <i>user</i> .
#5	Add Friends	<i>User</i> dapat menambah teman melalui <i>username</i> .
#6	Send Alert	Memberikan peringatan terhadap <i>user</i> karena sudah menggunakan aplikasi selama 20 menit.
#7	Remind To Contact	Mengingatkan <i>user</i> untuk menghubungi teman nya yang belum di chat selama 14 hari atau lebih.
#8	Customization	Mengubah data yang sudah teregistrasi.

3.1.1 Use Case #1 Register

3.1.1.1 Use Case Scenario #1 Register

Skenario Use Case #1 :

I. Pre - Condition

Aktor *User* berada di Page *Register*.

II. Use Case Description

a. Primary Flow

1. *User* menginputkan data *username* dan *password* di *text field*.
2. Controller Registration mengambil data.
3. Controller Registration memvalidasi data ke *database user* menggunakan class validator.
4. Jika validator menganggap bahwa data sudah valid (tidak ada yang sama *username*nya di *database user*) maka akan *return* boolean True.
5. Controller Registration akan menyimpan data yang sudah diinputkan ke *database user*.
6. *Database user* mengirimkan sinyal apakah data sudah disimpan ke controller.
7. Controller Registration mengirimkan pesan ke boundary RegistrationUI untuk mengirim notifikasi ke aktor.
8. Aktor *user* menerima notifikasi bahwa data berhasil di simpan di *database user*.

b. Alternative Flow

Jika sistem mendeteksi bahwa *username* ada di dalam *database* :

1. Controller mengirim sinyal ke boundary untuk memberikan notifikasi bahwa *username* sudah terpakai.
2. Aktor *user* mendapatkan notifikasi bahwa *username* sudah terpakai.
3. Aktor menginputkan kembali *username* dan *password* untuk ke-2 atau n-kalinya.

III. Post - Condition

User berhasil terdaftar, akun tersimpan di *database*.

3.1.1.2 UI Design dan Deskripsi Objek UI #1 Register

Tabel 3.1.1.2 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
Page-001	Page Registrasi	Halaman untuk registrasi <i>user</i> .

Gambar 3.1.1 UI Design Page Register

SIGN UP

Username
text_field text_username_register

Password
text_field text_password_register

Confirm Password
text_field text_cpassword_register

Confirm
btn_confirm_register

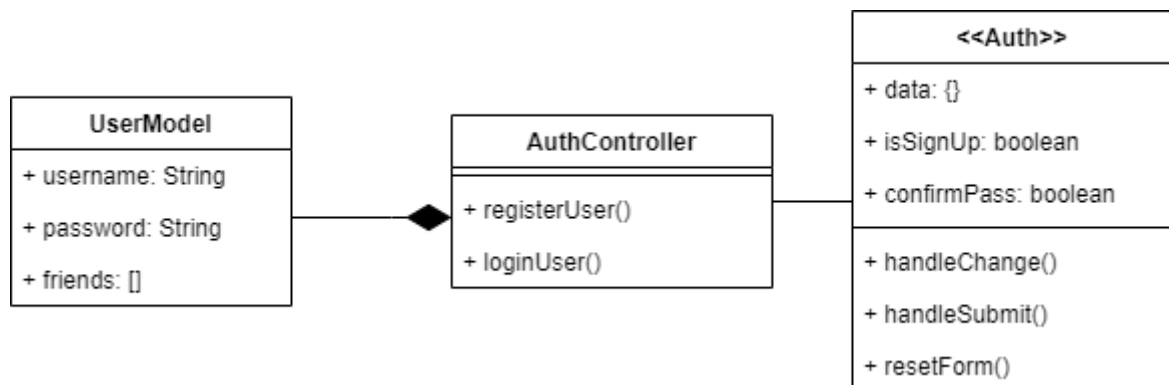
Id_Objek	JENIS	LABEL*	Keterangan**
<i>text_username_register</i>	<i>text_field</i>	<i>Username</i>	Text field yang digunakan untuk memasukkan <i>username</i> yang user inginkan.
<i>text_password_register</i>	<i>text_field</i>	<i>Password</i>	Text field yang digunakan untuk memasukkan <i>password</i> yang user inginkan.
<i>txt_confirm_button</i>	<i>button</i>	<i>Confirm</i>	Jika di tekan maka <i>text</i> yang sudah di isi diatas akan di simpan ke <i>database</i> apabila tidak ada <i>username</i> yang sudah sama di <i>system</i> .

3.1.1.3 Identifikasi Object dan Tipe nya #1 Use Case Register

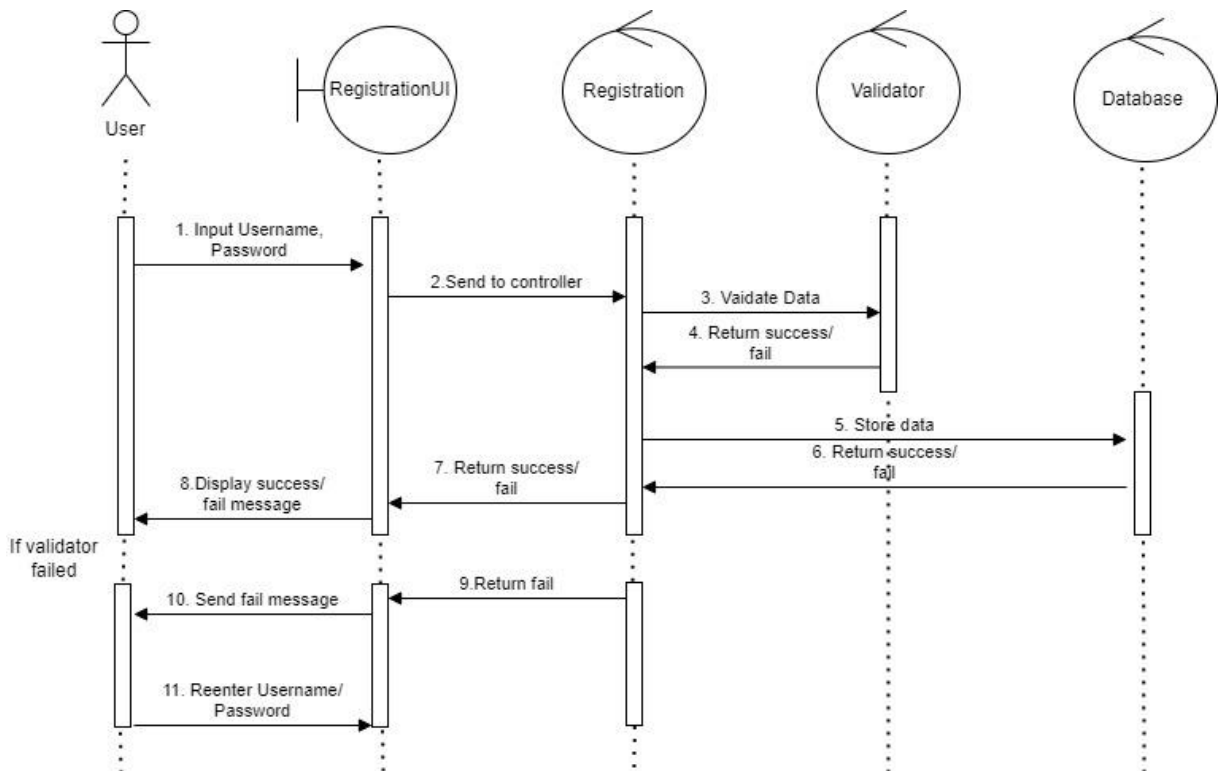
Tabel 3.1.1.3 Tabel Object Perancangan

No	Nama Object Baru	Jenis / Tipe Kelas
1	Auth	View
2	AuthController	Controller
3	UserModel	Database Model

3.1.1.4 Class Diagram #1 Register



3.1.1.5 Sequence Diagram #1 Register



3.1.2 Use Case #2 Login

3.1.2.1 Use Case Scenario #2 Login

Skenario Use Case #2 :

I. Pre - Condition

User sudah ter-registrasi dan berada di page Login.

II. Use Case Description

a. Primary Flow

1. User memasukkan *username* dan *password*.
2. User Controller mengirim data yang dimasukkan *user* ke User Validator.
3. User Validator mengautentikasikan data yang sesuai di *Database User*.
4. User Validator mengirim data bahwa *user* tervalidasi ke User Controller.
5. User Controller mengijinkan *user* masuk ke aplikasi.

b. Alternative Flow

1. User Validator mengirim data bahwa *user* tidak tervalidasi ke User Controller.
2. User Controller tidak mengijinkan *user* masuk dan memberikan peringatan bahwa data yang dimasukkan salah.

- III. Post - Condition
User berada di Page Chat

3.1.2.2 UI Design dan Deskripsi Objek UI #2 Login

Tabel 3.2 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
<i>Page-002</i>	<i>Page Login</i>	Halaman untuk <i>login user</i> .

Gambar 3.1.2 UI Design Page Login

The image shows a UI design for a login page on a grid background. At the top, the text "Log In" is displayed in a large, bold, black font. Below this, the label "Username" is followed by a text input field. The field has a light gray border and contains the placeholder text "text_field" in gray and "text_username_login" in red. Below the username field, the label "Password" is followed by another text input field. This field also has a light gray border and contains the placeholder text "text_field" in gray and "text_password_login" in red. At the bottom of the form, there is a rounded rectangular button with a black border and the text "Login" in black. Below the button, the label "btn_confirm_login" is written in red.

Page Login

Id_Objek	JENIS	LABEL*	Keterangan**
<i>text_username_login</i>	<i>text_field</i>	<i>Username</i>	Text field yang digunakan <i>user</i> untuk masuk dengan <i>username</i> yang teregistrasi.
<i>text_password_login</i>	<i>text_field</i>	<i>Password</i>	Text field yang digunakan <i>user</i> untuk masuk dengan <i>password</i> yang teregistrasi.

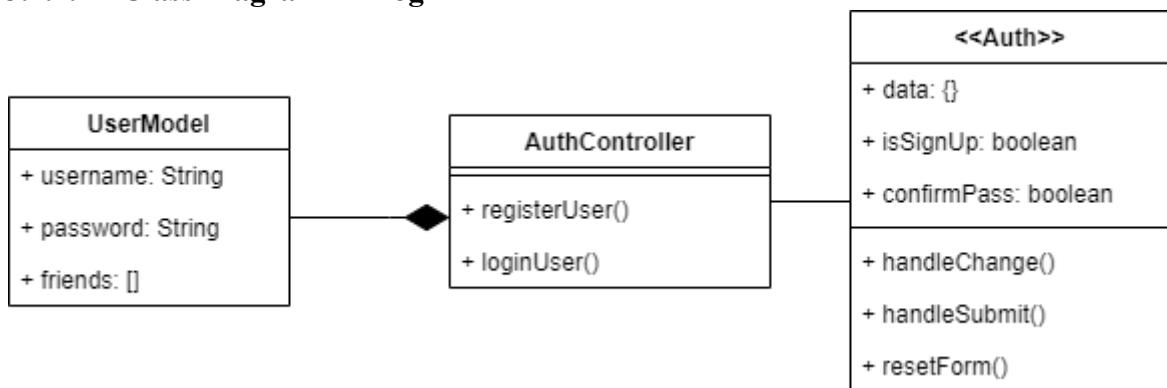
Id_Objek	JENIS	LABEL*	Keterangan**
<i>btn_confirm_login</i>	<i>button</i>	<i>Login</i>	Button yang berguna untuk validasi <i>username</i> dan <i>password user</i> . Lalu memasukkan <i>user</i> jika tervalidasi.

3.1.2.3 Identifikasi Object Baru & Tipe Kelas #2 Login

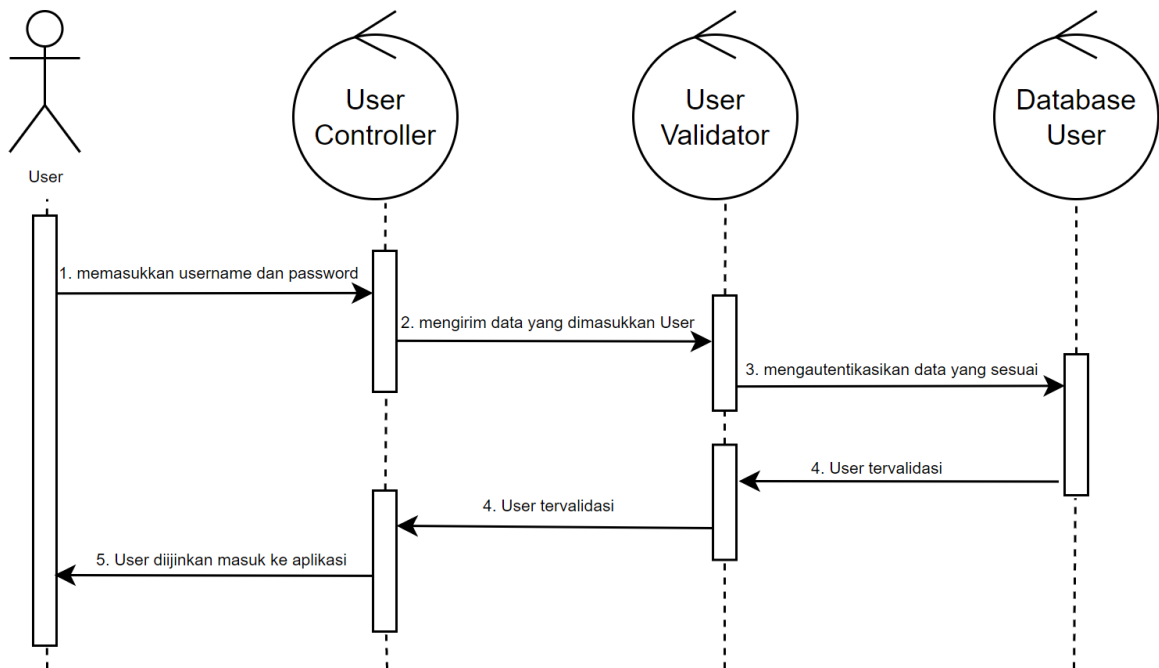
Tabel 3.1.2.3 Tabel Object Perancangan

No	Nama Object Baru	Jenis / Tipe Kelas
1	Auth	View
2	AuthController	Controller
3	UserModel	Database Model

3.1.2.4 Class Diagram #2 Login



3.1.2.5 Sequence Diagram #2 Login



3.1.3 Use Case #3 Log Out

3.1.3.1 Use Case Scenario #3 Log Out

Skenario Use Case #3 :

I. Pre - Condition

User berada di page logout

II. Use Case Description

a. Primary Flow

1. Aktor User menekan button logout yang ada di page
2. boundary LogoutUI mengirim notifikasi ke user bahwa apakah aktor ingin logout
3. boundary LogoutUI mengirim request ke controller logout bahwa user ingin logout
4. boundary LogoutUI menghapus session user di aplikasinya sendiri
5. controller mengirim sinyal ke boundary untuk meredirect aktor user ke homepage
6. user berada di homepage

b. Alternative Flow

1. Aktor menekan tombol cancel ketika konfirmasi logout
2. boundary akan meredirect aktor user ke home page

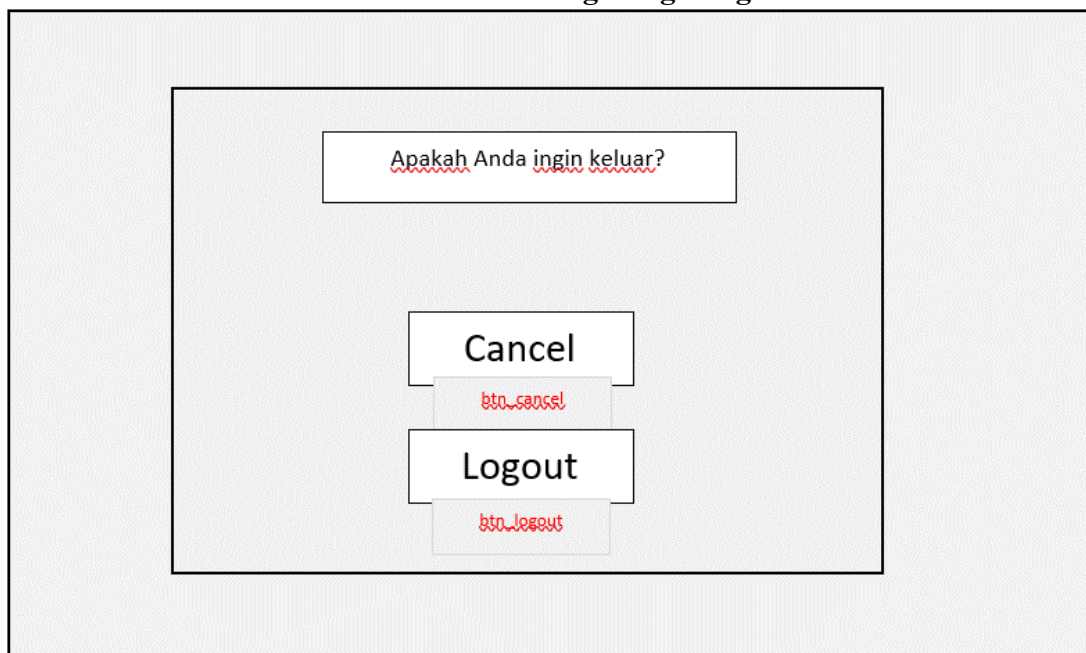
- III. Post - Condition
Akun user keluar dari aplikasi.

3.1.3.2 UI Design dan Deskripsi Objek UI #3 Log Out

Tabel 3.3 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
<i>Page-003</i>	<i>Page Log Out</i>	Halaman ketika user ingin keluar dari aplikasi.

Gambar 3.1.3 UI Design Page Log Out



Page Log Out

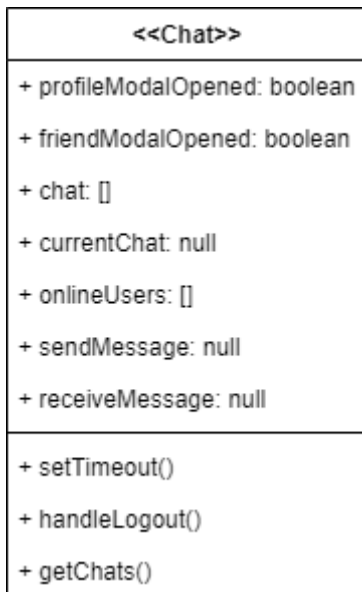
Id_Objek	JENIS	LABEL*	Keterangan**
<i>btn_cancel</i>	<i>button</i>	<i>Cancel</i>	Jika ditekan akan membatalkan proses logout.
<i>btn_logout</i>	<i>button</i>	<i>Logout</i>	Jika ditekan akan mengeluarkan akun user dari aplikasi.

3.1.3.3 Identifikasi Object Baru & Tipe Kelas #3 Log Out

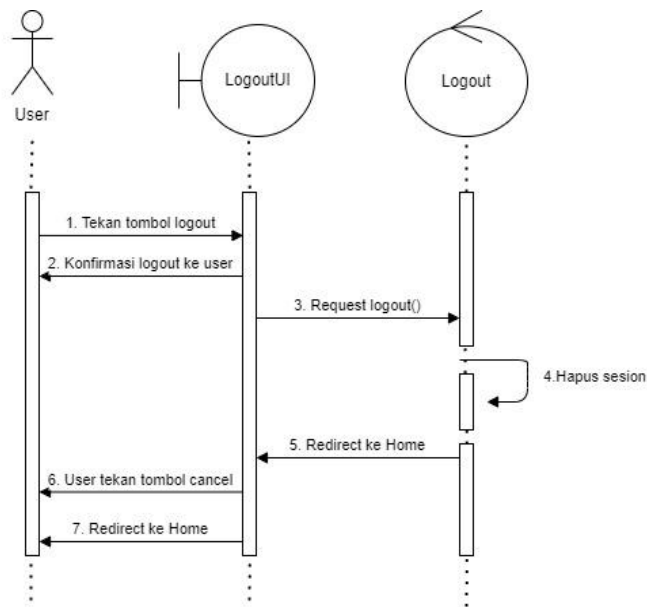
Tabel 3.1.3.3 Tabel Object Perancangan

No	Nama Object Baru	Jenis / Tipe Kelas
1	Chat	View

3.1.3.4 Class Diagram #3 Log Out



3.1.3.5 Sequence Diagram #3 Log Out



3.1.4 Use Case #4 Chat

3.1.4.1 Use Case Scenario #4 Chat

Skenario Use Case #4 :

I. Pre - Condition

User sudah menambah dan memiliki teman dalam list teman.

II. Use Case Description

a. Primary Flow

Langkah 1 : User pilih chat kontak yang ingin ditampilkan.

Langkah 2 : ChatUI mengambil chat history dengan kontak.

Langkah 3 : Chat menampilkan tampilan chat dengan kontak.

Langkah 4 : ChatUI menampilkan chat dengan kontak ke User.

b. Alternative Flow

-

III. Post - Condition

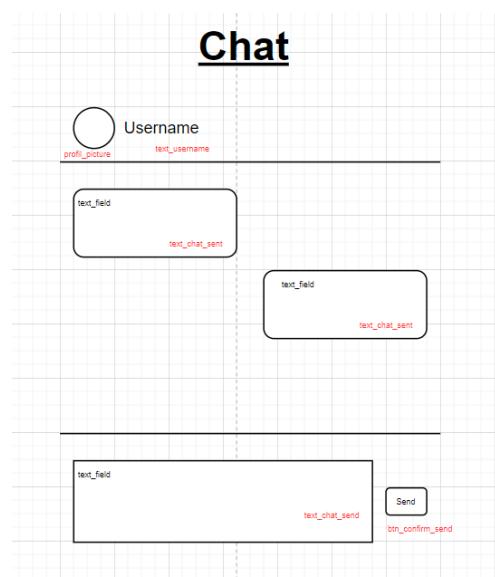
User berhasil mengirim text.

3.1.4.2 UI Design dan Deskripsi Objek UI #4 Chat

Tabel 3.4 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
Page-004	Page Chat	Halaman ketika user ingin chatting dengan user lain.

Gambar 3.1.4 UI Design Page Chat



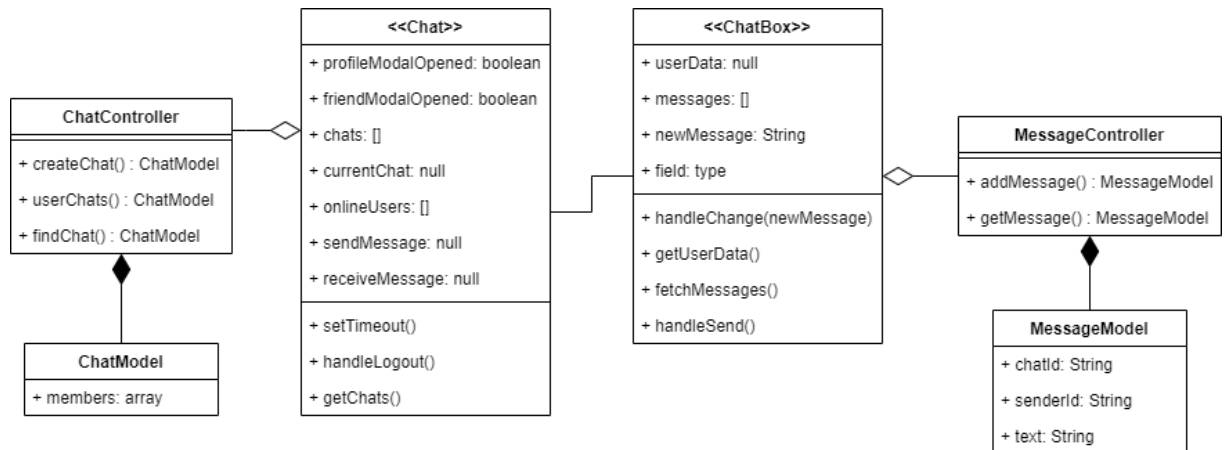
Id_Objek	JENIS	LABEL*	Keterangan**
<i>profil_picture</i>	<i>image</i>	<i>Profile Picture</i>	Gambar profil dari teman yang di chat user.
<i>text_username</i>	<i>text_field</i>	<i>Username</i>	Nama Username dari teman yang di chat user.
<i>text_chat_sent</i>	<i>text_field</i>	<i>Sent Chat</i>	Chat user yang sudah terkirim dan ter log dalam database chat.
<i>text_chat_send</i>	<i>text_editable</i>	<i>Enter message</i>	Kolom text dimana user bisa mengetik pesan yang ingin dikirim.
<i>text_confirm_send</i>	<i>button</i>	<i>Send</i>	jika di klik, maka text yang sudah diketik akan dikirim ke teman chat user.

3.1.4.3 Identifikasi Object Baru & Tipe Kelas #4 Chat

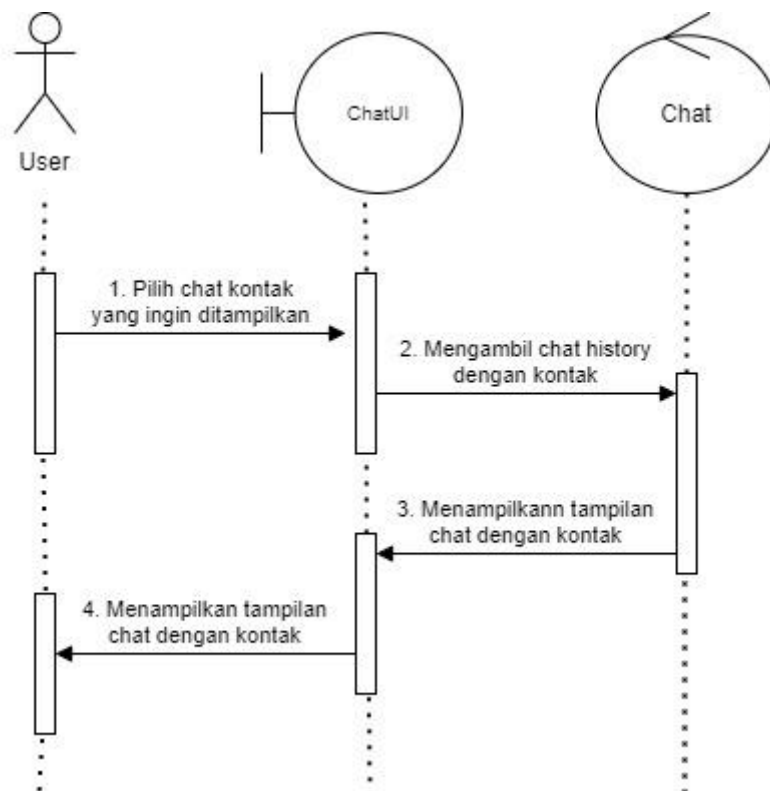
Tabel 3.1.4.3 Tabel Object Perancangan

No	Nama Object Baru	Jenis / Tipe Kelas
1	ChatModel	Database Model
2	ChatController	Controller
3	Chat	View Entity
4	ChatBox	View Entity
5	Message Controller	Controller
6	MessageModel	Database Model

3.1.4.4 Class Diagram #4 Chat



3.1.4.5 Sequence Diagram # 4 Chat



3.1.5 Use Case #5 Add Friends

3.1.5.1 Use Case Scenario #5 Add Friends

Skenario Use Case #5 :

I. Pre - Condition

User sudah login dan berada di add friends page

II. Use Case Description

a. Primary Flow

1. User menginput username teman yang ingin di add ke text field
2. Boundary AddFriendsUI mengirim sinyal ke database user untuk mengecek apakah username ada
3. database user mengirim sinyal bahwa teman ada, dan mengadd nya ke array teman aktor user, lalu mengirim sinyal bahwa process sukses
4. boundary AddFriendsUI mendisplay notifikasi bahwa process sukses

b. Alternative Flow

-

III. Post - Condition

List friends User bertambah satu teman

3.1.5.2 UI Design dan Deskripsi Objek UI #5 Add Friends

Tabel 3.5 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
Page-005	Page Add Friends	Halaman untuk menambahkan teman.

Gambar 3.1.5 UI Design Page Add Friends

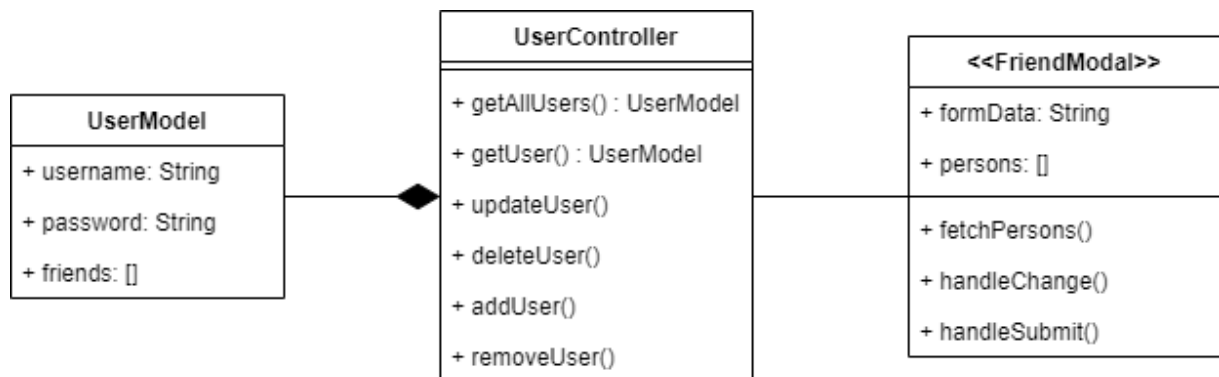
Id_Objek	JENIS	LABEL*	Keterangan**
<i>text_field_username</i>	<i>text_field</i>	<i>Username</i>	Text field yang digunakan untuk memasukkan username teman yang ingin ditambah.
<i>btn_confirm_add</i>	<i>button</i>	<i>Add</i>	Jika di klik, maka akan mencari lalu menambahkan teman ke dalam <i>list friends user</i> .
<i>btn_confirm_close</i>	<i>button</i>	<i>Close (X)</i>	Jika di klik, maka tampilan <i>add friends</i> akan hilang.

3.1.5.3 Identifikasi Object Baru & Tipe Kelas #5 Add Friends

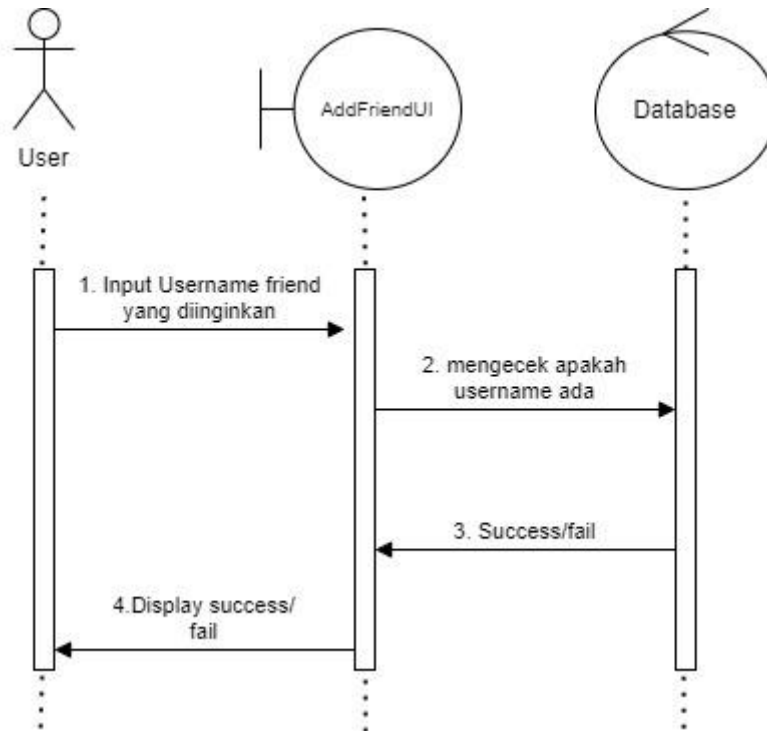
Tabel 3.1.5.3 Tabel Object Perancangan

No	Nama Object Baru	Jenis / Tipe Kelas
1	UserModel	Database Model
2	UserController	Controller
3	FriendModal	View Entity

3.1.5.4 Class Diagram #5 Add Friends



3.1.5.5 Sequence Diagram # 5 Add Friends



3.1.6 Use Case #6 Send Alert

3.1.6.1 Use Case Scenario #6 Send Alert

Skenario Use Case #6 :

I. Pre - Condition

User berada di page chat

II. Use Case Description

a. Primary Flow

1. Aktor User memasuki ruangan chat
2. Boundary ChatUI mengirim sinyal ke controller Timer bahwa aktor user memasuki ruangan chat
3. controller Timer memulai timer, dan mengecek terus menerus apakah user masih ada di ruangan chat
4. jika timer sudah lebih dari 20 menit, maka controller Timer mengirim sinyal ke boundary untuk mengirimkan alert
5. User mendapatkan alert bahwa aktor sudah memakai ruangan chat selama 20 menit

b. Alternative Flow

-

III. Post - Condition

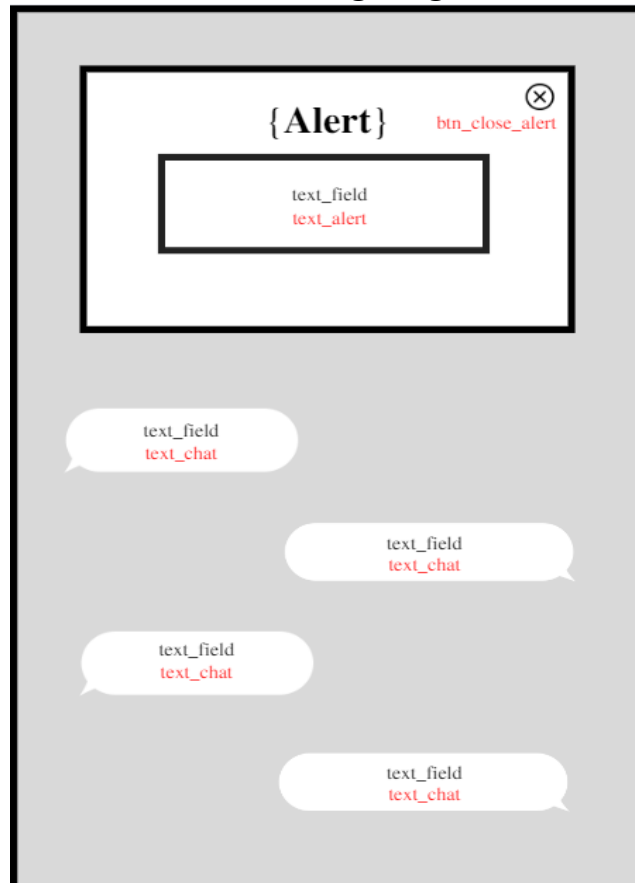
Sistem berhasil mengirim peringatan kepada user..

3.1.6.2 UI Design dan Deskripsi Objek UI #6 Send Alert

Tabel 3.6 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
Page-006	Page Send Alert	Halaman untuk send alert.

Gambar 3.1.6 UI Design Page Send Alert



Page Send Alert

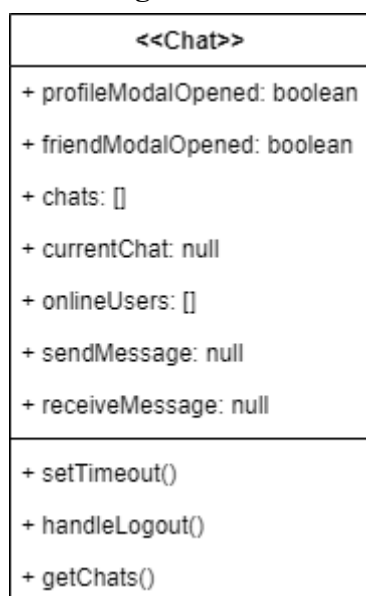
Id_Objek	JENIS	LABEL*	Keterangan**
Text_alert	text_field	Alert	Field text yang digunakan untuk pesan peringatan kepada user.
Text_chat	text_field	Chat	Field text yang digunakan untuk percakapan.
Btn_close_alert	button	Close (X)	Jika button diklik maka akan menutup pesan peringatan.

3.1.6.3 Identifikasi Object Baru & Tipe Kelas #6 Send Alert

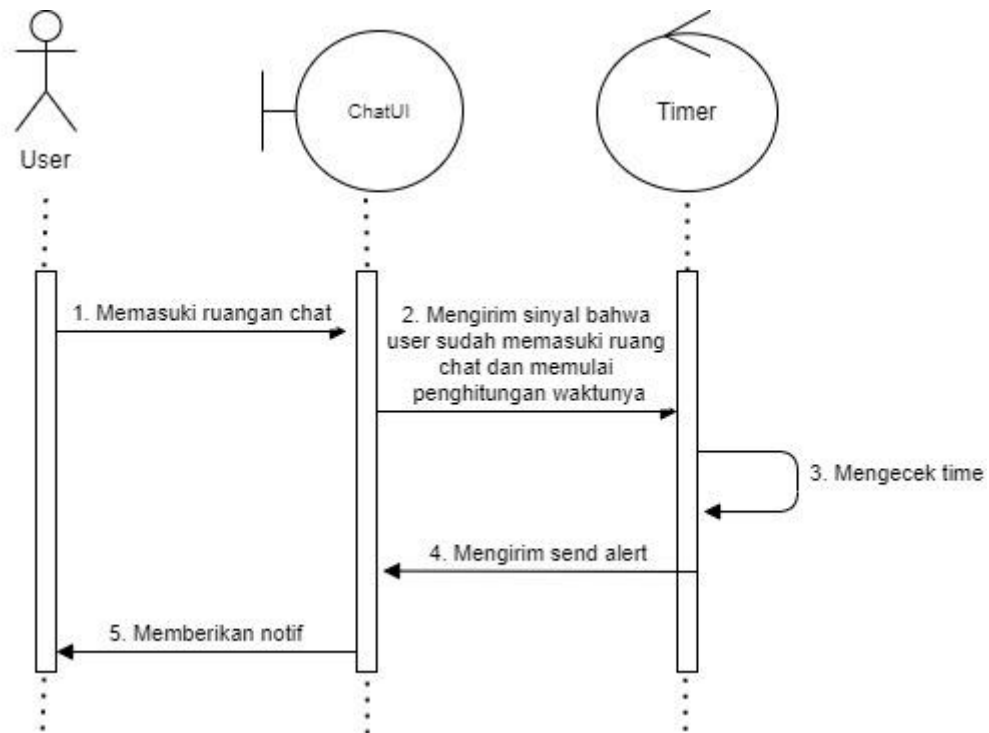
Tabel 3.1.6.3 Tabel Object Perancangan

No	Nama Object Baru	Jenis / Tipe Kelas
1	Chat	View Entity

3.1.6.4 Class Diagram #6 Send Alert



3.1.6.5 Sequence Diagram # 6 Send Alert



3.1.7 Use Case #7 Remind to Contact

3.1.7.1 Use Case Scenario #7 Remind to Contact

Skenario Use Case #7 :

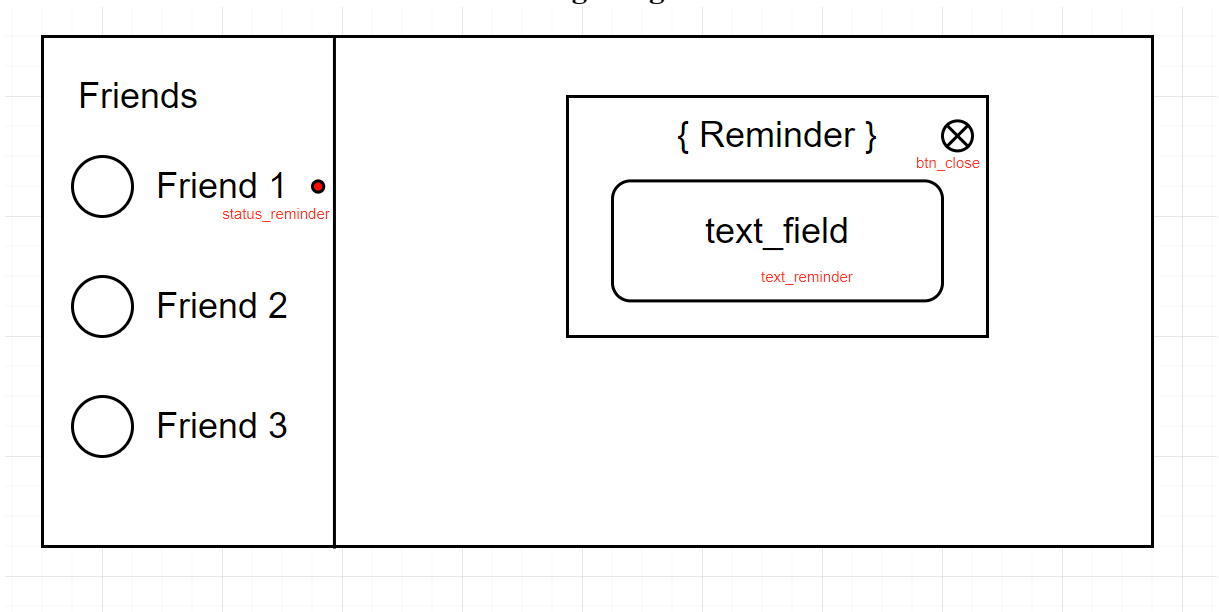
- I. Pre - Condition
User tidak berada di ruangan chat pada salah satu temannya
- II. Use Case Description
 - a. Primary Flow
 1. boundary ChatUI akan mengirimkan sinyal ke controller jika user tidak di dalam ruangan chat.
 2. controller Timer start timer, dan akan mengeceknya terus menerus apakah timer lebih dari 14 hari atau tidak
 3. jika sudah lebih dari 14 hari, controller Timer mengirim sinyal ke class alert bahwa timer sudah lebih dari 14 hari
 4. class Alert akan mengirim notifikasi ke Aktor user bahwa aktor user belum chat temannya lebih dari 14 hari
 - b. Alternative Flow
-

3.1.7.2 UI Design dan Deskripsi Objek UI #7 Remind to Contact

Tabel 3.7 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
Page-007	Page Remind to Contact	Halaman untuk memberi peringatan kepada user.

Gambar 3.1.7 UI Design Page Remind to Contact



Page Remind to Contact

Id_Objek	JENIS	LABEL*	Keterangan**
<i>status_reminder</i>	<i>Image</i>	<i>Status / Dot</i>	Keterangan untuk menunjukkan teman yang belum dikontak / melakukan percakapan
<i>btn_close</i>	<i>Button</i>	<i>Close (X)</i>	Jika button di klik, maka akan menutup pesan pengingat
<i>text_reminder</i>	<i>Text Field</i>	<i>Reminder</i>	Text field yang digunakan sebagai pengingat untuk user

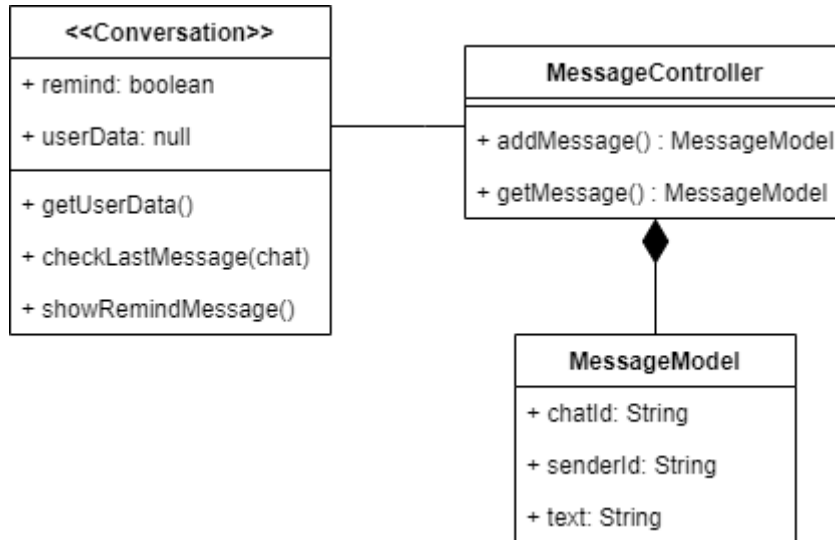
3.1.7.3 Identifikasi Object Baru & Tipe Kelas #7 Remind to Contact

Tabel 3.1.7.3 Tabel Object Perancangan

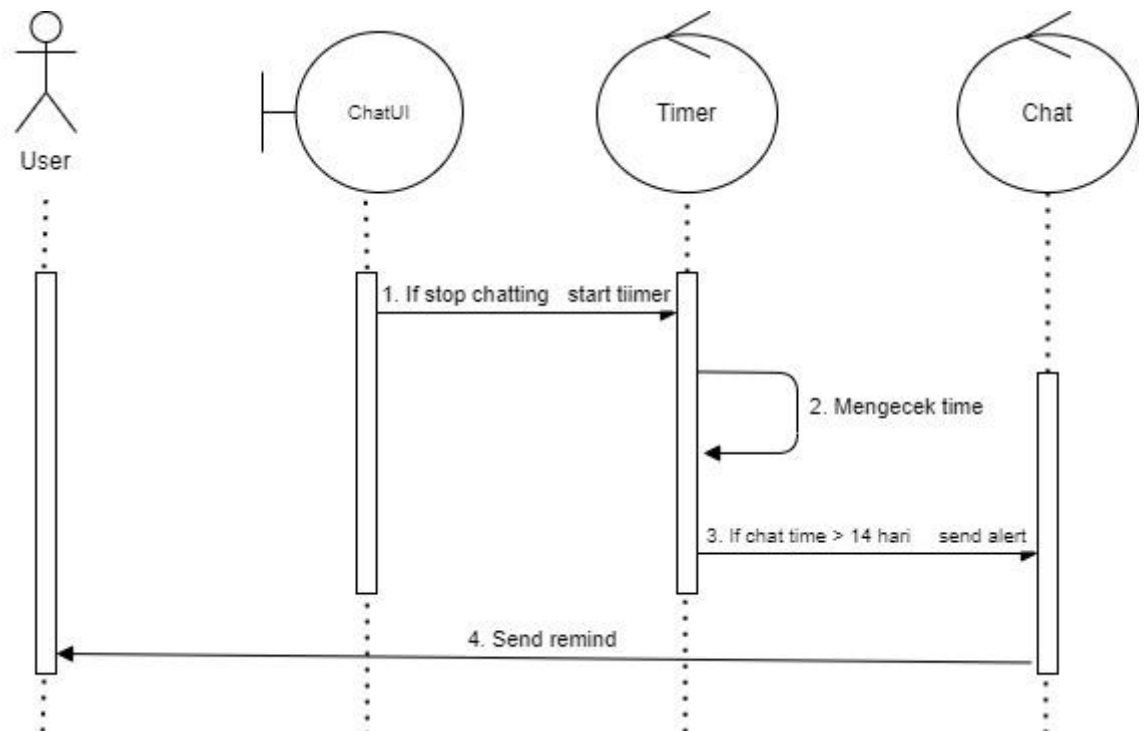
No	Nama Object Baru	Jenis / Tipe Kelas
1	MessageModel	Database Model

2	MessageController	Controller
3	Conversation	View Entity

3.1.7.4 Class Diagram #7 Remind to Contact



3.1.7.5 Sequence Diagram #7 Remind to Contact



3.1.8 Use Case #8 Customization

3.1.8.1 Use Case Scenario #8 Customization

Skenario Use Case #8 :

I. Pre - Condition

User berada di customization page

II. Use Case Description

a. Primary Flow

1. User memasukkan Username/New Username, dan/atau Password/New Password dan Confirm New Password
2. boundary CustomizationUI mengirim data ke controller Customization
3. controller Customization memakai class validator untuk mengecek apakah username dan/atau password sama seperti sebelumnya, atau username sudah ada di database user
4. class validator return hasil validation nya
5. jika valid, controller Customization mengirim data ke database user untuk mengganti data user tersebut
6. database user mereturn apakah process berhasil atau tidak
7. controller Customization mengirim sinyal ke boundary jika process berhasil
8. Aktor user mendapat notifikasi bahwa data berhasil diubah

b. Alternative Flow

1. Jika username sudah ada atau sama, controller mengirim sinyal ke boundary untuk mengirim notifikasi input ulang
2. boundary mengirim notifikasi input ulang ke user
3. User menginput ulang data yang ingin diganti

III. Post - Condition

Sistem berhasil mengubah username dan/atau password.

3.1.8.2 UI Design dan Deskripsi Objek UI #8 Customization

Tabel 3.8 Deskripsi Objek UI

ID. LAYAR	NAMA LAYAR	DESKRIPSI
Page-008	Page Customization	Halaman untuk Customization

Gambar 3.1.8 UI Design Page Customization

The image shows a UI design for a page customization form. It includes the following elements:

- Username**: A text field labeled `text_field` with the placeholder text `text_old_username`.
- New Username (optional)**: A text field labeled `text_field` with the placeholder text `text_new_username`.
- Confirm Username (optional)**: A text field labeled `text_field` with the placeholder text `text_confirm_username`.
- Password**: A text field labeled `text_field` with the placeholder text `text_old_password`.
- New Password (optional)**: A text field labeled `text_field` with the placeholder text `text_new_password`.
- Confirm Password (optional)**: A text field labeled `text_field` with the placeholder text `text_confirm_password`.
- Submit**: A button labeled `Submit` with the identifier `btn_submit`.

Page Customization

Id_Objek	JENIS	LABEL*	Keterangan**
<i>text_old_username</i>	<i>text_field</i>	<i>Username</i>	Text field yang digunakan untuk memasukkan username lama oleh user.
<i>text_new_username</i>	<i>text_field</i>	<i>Username</i>	Text field yang digunakan untuk memasukkan password yang user inginkan.
<i>text_confirm_username</i>	<i>text_field</i>	<i>Username</i>	Text field yang digunakan untuk memasukan ulang username baru.
<i>text_old_password</i>	<i>text_field</i>	<i>Password</i>	Text field yang digunakan untuk memasukan password yang lama oleh user.
<i>text_new_password</i>	<i>text_field</i>	<i>Password</i>	Text field yang digunakan untuk memasukan password yang baru sesuai yang diinginkan user.
<i>text_confirm_password</i>	<i>text_field</i>	<i>Password</i>	Text field yang digunakan untuk memasukan ulang password baru.

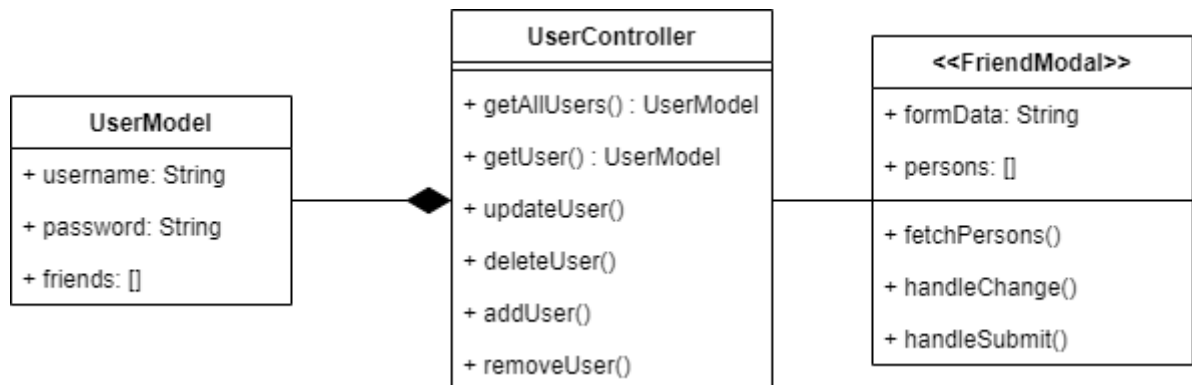
Id_Objek	JENIS	LABEL*	Keterangan**
<i>btn_sumbit</i>	<i>button</i>	<i>Confirm</i>	Jika di klik, akan memperbaharui username dan/atau password user ke database jika sudah sesuai peraturan.

3.1.8.3 Identifikasi Object Baru & Tipe Kelas #8 Customization

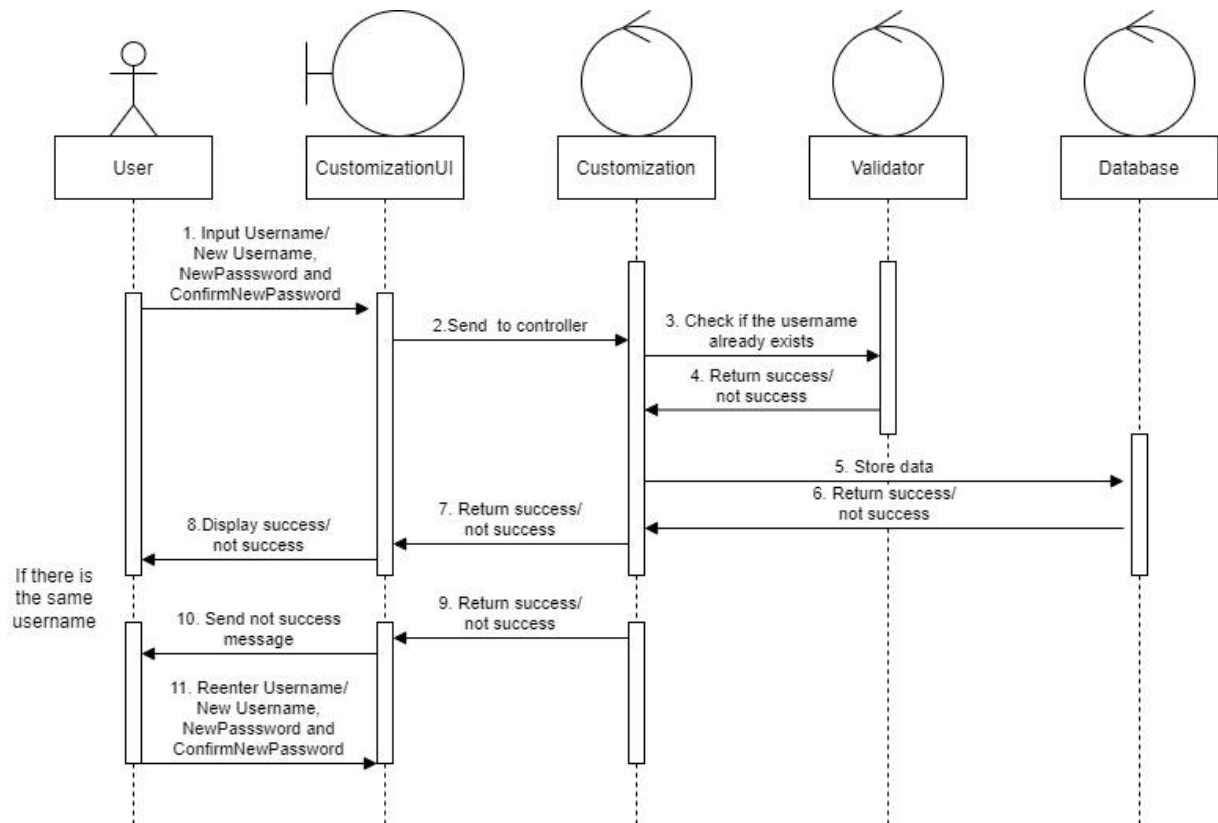
Table 3.1.8.3 Tabel Object Perancangan

No	Nama Object Baru	Jenis / Tipe Kelas
1	UserModel	Database Model
2	UserController	Controller
3	FriendModal	View Entity

3.1.8.4 Class Diagram #8 Customization



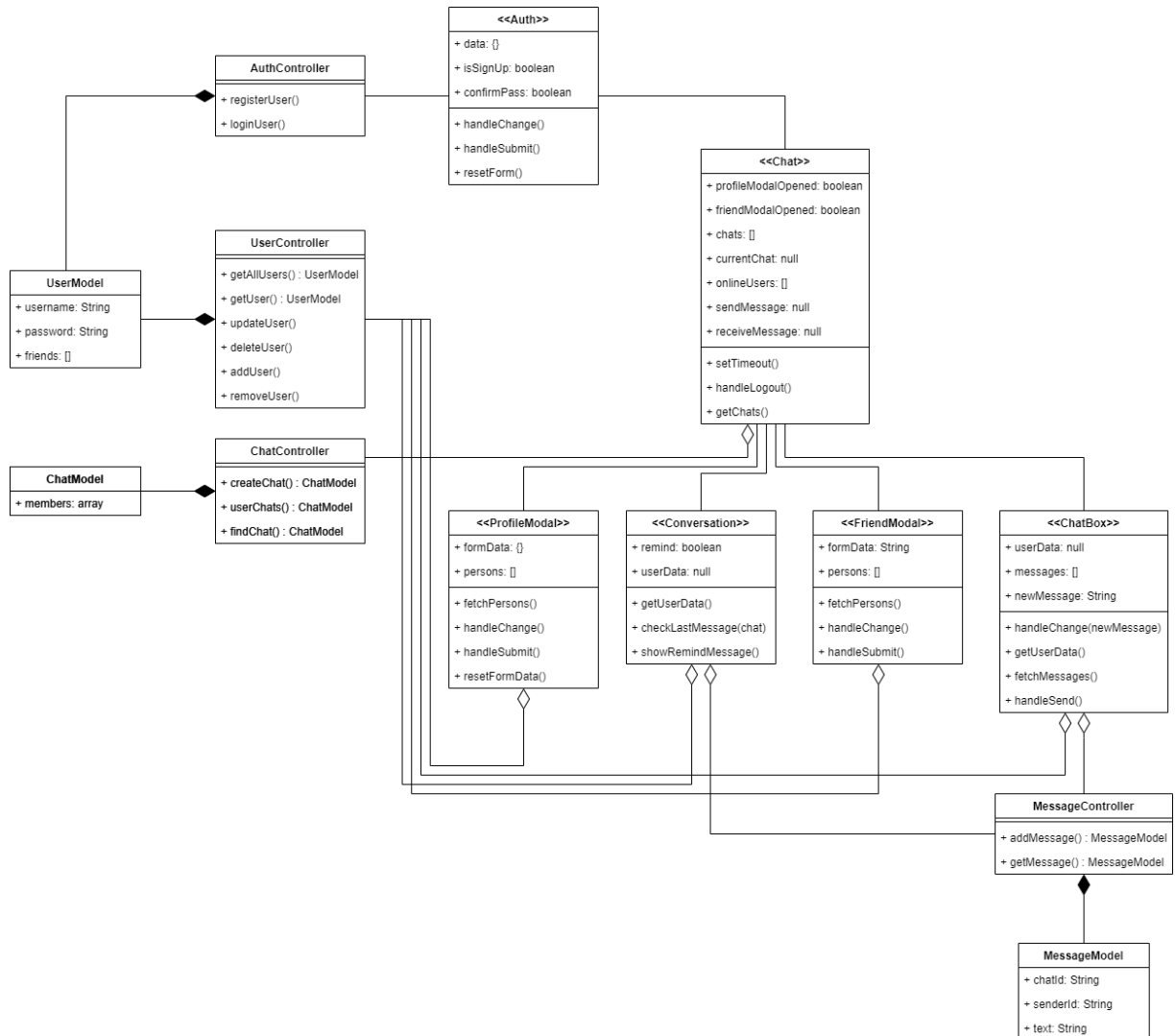
3.1.8.5 Sequence Diagram #8 Customization



3.2 Diagram Kelas Keseluruhan

TABEL KELAS :

Gambar 3.2 Diagram Kelas Keseluruhan



ID Kelas	Nama Kelas Perancangan	Atribute (visibility)	Method / Operation
class-001	UserModel	+ username: String + password: String + friends: []	-
class-002	ChatModel	+ members: []	-
class-003	MessageModel	+ chatId: String + senderId: String + text: String	-

class-004	UserController	-	+ getAllUsers(): UserModel + getUser(): UserModel + updateUser() + deleteUser() + addUser() + removeUser()
class-005	ChatController	-	+ createChat(): ChatModel + userChats(): ChatModel + findChat(): ChatModel
class-006	MessageController	-	+ addMessage(): MessageModel + getMessage(): MessageModel
class-007	AuthController	-	+ registerUser() + loginUser()
class-008	Auth	+ data: {} + isSignUp: boolean + confirmPass: boolean	+ handleChange() + handleSubmit() + resetForm()
class-009	Chat	+ profileModalOpened: boolean + friendModalOpened: boolean + chats: [] + currentChat: null + onlineUsers: [] + sendMessage: null + receiveMessage: null	+ setTimeout() + handleLogout() + getChats()
class-010	Conversation	+ remind: boolean + userData: null	+ getUserData() + checkLastMessage(chat) + showRemindMessage()
class-011	ChatBox	+ userData: null + messages: [] + newMessage: String	+ handleChange(new Message) + getUserData() + fetchMessage() + handleSend()
class-012	ProfileModal	+ formData: {} + persons: []	+ fetchPersons() + handleChange() + handleSubmit() + resetFormData()
class-013	FriendModal	+ formData: String	+ fetchPersons()


```

const newUser = new UserModel(req.body);
const { username } = req.body;
try {
  const oldUser = await UserModel.findOne( { username } );

  if (oldUser) {
    return res
      .status(400)
      .json( { message: "username is already registered" } );
  }
  const user = await newUser.save();

  const token = jwt.sign(
    {
      username: user.username,
      id: user._id,
    },
    process.env.JWT_KEY,
    { expiresIn: "1h" }
  );
  res.status(200).json( { user, token } );
} catch (error) {
  res.status(500).json( { message: error.message } );
}
};

```

3.4.2 Algoritma/Query Login

Nama Kelas : AuthController

Nama Operasi : loginUser()

Algoritma :

```

loginUser = async (req, res) => {
  const { username, password } = req.body;

  try {
    const user = await UserModel.findOne( { username: username } );

    if (user) {
      const validity = await bcrypt.compare(password, user.password);

      if (!validity) {
        res.status(400).json("Wrong password");
      } else {
        const token = jwt.sign(
          {
            username: user.username,

```

```

        id: user._id,
      },
      process.env.JWT_KEY,
      { expiresIn: "1h" }
    );
    res.status(200).json({ user, token });
  }
  } else {
    res.status(404).json("User does not exists");
  }
} catch (error) {
  res.status(500).json({ message: error.message });
}

```

3.4.3 Algoritma/Query Log out

Nama Kelas : Authentication

Nama Operasi : logoutUser()

Algoritma :

```

case 'LOG_OUT':
  localStorage.clear();
  return { ...state, authData: null, loading: false, error: false };
default:
  return state;

```

3.4.4 Algoritma/Query Chat

Nama Kelas : ChatController, MessageController

Nama Operasi : findChat(), addMessage(), getMessage()

Algoritma :

```

findChat = async(req,res) => {
  try {
    const chat = await ChatModel.findOne({
      members: {$all: [req.params.firstId, req.params.secondId]}
    })
    res.status(200).json(chat)
  } catch (error) {
    res.status(500).json(error)
  }
}

addMessage = async(req,res) => {
  const {chatId, senderId, text} = req.body
  const message = new MessageModel({
    chatId,

```



```

        senderId,
        text
    })
    try {
        const result = await message.save()
        res.status(200).json(result)
    } catch (error) {
        res.status(500).json(error)
    }
}

getMessages = async(req,res) => {
    const {chatId} = req.params

    try {
        const result = await MessageModel.find({chatId})
        res.status(200).json(result)
    } catch (error) {
        res.status(500).json(error)
    }
}

```

3.4.5 Algoritma/Query Add Friend

Nama Kelas : *UserController*

Nama Operasi : *addUser*

Algoritma :

```

addUser = async (req, res) => {
    const id = req.params.id;

    const { _id } = req.body;

    if (_id === id) {
        res.status(403).json("Action forbidden");
    } else {
        try {
            const addedUser = await UserModel.findById(id);
            const adderUser = await UserModel.findById(_id);
            if (!addedUser.friends.includes(_id)) {
                await addedUser.updateOne({ $push: { friends: _id } });
                await adderUser.updateOne({ $push: { friends: id } });
                res.status(200).json("User added!");
            } else {
                res.status(403).json("User is Already added by you");
            }
        } catch (error) {
            res.status(500).json(error);
        }
    }
}

```

```

    }
  }
};

```

3.4.6 Algoritma/Query Send Alert

Nama Kelas : Alert
Nama Operasi : send_alert()
Algoritma :

```

useEffect(() => {
  const timer = setTimeout(() => {
    alert("Kamu udah di app selama 15 menit! touch some grass pls.");
  }, 900000); //15 minute, change to test faster
  return () => clearTimeout(timer);
}, []);

```

3.4.7 Algoritma/Query Remind Chat

Nama Kelas : Alert
Nama Operasi : send_alert()
Algoritma :

```

useEffect(() => {
  const checkLastMessage = async(chat) => {
    try {
      const { data } = await getMessages(chat._id);
      const messagedTime = new Date(data[data.length - 1].createdAt).getTime();
      const currentTime = Date.now();

      const timeDifferenceInHours = ((currentTime - messagedTime) / 1000) / 3600;
      console.log(timeDifferenceInHours)
      if (timeDifferenceInHours > 24.0) {
        remind.current = true;
      }
    } catch (error) {
      console.log(error);
    }
  };
  checkLastMessage(chat);
}, [chat]);

```

3.4.8 Algoritma/Query Customization

Nama Kelas : UserController
Nama Operasi : updateUser()
Algoritma :

```

updateUser = async (req, res) => {
  const id = req.params.id;
  const { _id, password } = req.body;

  if (id === _id) {
    try {
      if (password) {
        const salt = await bcrypt.genSalt(10);
        req.body.password = await bcrypt.hash(password, salt);
      }

      const user = await UserModel.findByIdAndUpdate(id, req.body, {
        new: true,
      });
      const token = jwt.sign(
        { username: user.username, id: user._id },
        process.env.JWT_KEY,
        { expiresIn: "1h" }
      );
      res.status(200).json({ user, token });
    } catch (error) {
      res.status(500).json(error);
    }
  } else {
    res.status(403).json("Access Denied! you can only update your own profile");
  }
};

```

4. Matriks Keruntutan (Requirement Traceability Matrix)

Mapping Requirement dengan Use Case yang direalisasikan

Kode FR	Nama Functional Requirement	Nama Use Case
SKPL-P01	<i>Register</i>	Register
SKPL-P02	<i>Login</i>	Login
SKPL-P03	Chat	-Login -Add Friends -Chat
SKPL-P04	Friends	-Login -Add Friends
SKPL-P05	Send Alert	-Login -Add Friends

		-Chat -Send Alert
SKPL-P06	Remind User	-Login -Add Friends -Chat -Remind to Contact
SKPL-P07	Log Out	-Login -Logout
SKPL-P08	Customize	-Login -Customization