BAB 3

PELAKSANAAN KERJA MAGANG

3.1. Kedudukan dan Koordinasi

Pelaksanaan kerja magang di PT. Gihon Telekomunikasi Indonesia dijalani dengan mengemban posisi *Full Stack Web Developer* dalam divisi IT dengan status *intern*, di mana divisi IT bertugas merancang aplikasi untuk internal perusahaan. Di bawah naungan ibu Erlina Prasetyowati sebagai pembimbing lapangan selama kerja magang, di mana Ibu Erlina memberikan *requirement* yang dibutuhkan dari perusahaan dalam membangun aplikasi *messaging* ini, Kemudian saling bertukar pikiran dengan tim yang lain dalam memahami *requirement* serta mencari solusi yang tepat dalam membangun aplikasi ini, serta memberikan saran berupa *fitur-fitur* tambahan yang diperlukan.

3.2. Tugas yang dilakukan

Tugas selama magang yang dilakukan yaitu merancang dan membangun aplikasi *messaging* yang akan digunakan oleh pihak internal perusahaan. Sebelum dimulai, diperlukan identifikasi spesifikasi dari projek yang akan dikerjakan, dimulai dari penggunaan bahasa pemrograman yang akan dipakai. Projek ini dibagi menjadi dua yaitu pembuatan *BackEnd* dan *FrontEnd*. Kemudian menentukan *Framework* apa yang akan dipakai dari setiap *BackEnd* maupun *FrontEnd*. Selain itu juga mempelajari konsep dari teknologi yang akan digunakan dalam projek ini.

Selama pengerjaan projek, untuk menyimpan perubahan yang dilakukan menggunakan *Version Control System (VCS)* untuk mempermudah dalam proses *development* dari aplikasi. Penggunaan *VCS* sangat membantu selain dalam proses *development* juga bisa berkolaborasi dengan orang lain ketika ingin bekerja dalam tim.

3.3. Uraian Pelaksanaan

3.3.1. Proses Pelaksanaan

Proses pengembangan aplikasi *messaging* pada PT. Gihon Telekomunikasi Indonesia membutuhkan beberapa bantuan *software. Software* yang digunakan memiliki fungsi-fungsi yang berbeda. Berikut *software* yang digunakan dalam mengembangkan aplikasi *messaging*:

1. Windows Terminal
2. Visual Studio code versi1.51.1
3. Laragon 4.0.16
4. Git version 2.19.2.windows.1
5. OS Windows 10 Home
6. Mysql versi 14.14
7. Anaconda versi 1.7.2
8. Python 3.8.3
9. NodeJs versi 12.18.3

Untuk menjalankan *software* yang telah disebutkan sebelumnya, dibutuhkan *Hardware* yang dapat menjalankan itu semua. Untuk spesifikasi *hardware* yang digunakan sebagai berikut:

1. Intel core i7-7500u CPU @ 2.70 GHz 2 core 4 thread
2. Acer aspire e5-475G-70XV
3. RAM: 8192MB DDR 4
4. Nvidia GeForce 940MX
5. Storage: 250 GB SSD, 1 TB HDD

Pada awal pembuatan aplikasi, dilakukan pencarian referensi dari beberapa aplikasi serupa untuk melihat gambaran tentang aplikasi *messaging* yang seutuhnya. Setelah itu mulai menyusun alur aplikasi dari awal sampai akhir. Dimulai dari *landing page,* kemudian masuk ke *Login*, setelah itu masuk ke *Home* yang dimana *user* dapat melakukan banyak aktivitas pada laman tersebut. *user* dapat menuju ke *Profile,* melakukan *chat* dengan *user* lain, dan membuat *room* untuk *Video Conference*.

3.3.2. Framework dan Alur kerja yang Digunakan

Projek aplikasi yang dikerjakan dibagi menjadi *BackEnd* dan *FrontEnd*. Pengerjaan *BackEnd* menggunakan *Framework* Django. Django adalah sebuah *High-Level Web Framework* yang dibangun dari bahasa *Python* yang membuat pengembangan *Web* menjadi sangat cepat dan mudah dengan desain yang *pragmatic* (Django Software Foundation, 2020). Selain itu, Django sangat menjaga keamanan dari *Web* dengan fungsi bawaannya sehingga *developer* tidak perlu khawatir terhadap ancaman berupa injeksi pada *Web* yang membahayakan. Alasan lain penggunaan Django sebagai *BackEnd* dikarenakan bersifat *open* source dan Framework ini telah menyediakan fitur bawaan yaitu django admin untuk kebutuhan *administrative* yang bisa langsung dipakai, sehingga bisa langsung fokus dalam pengembangan fitur lainnya.

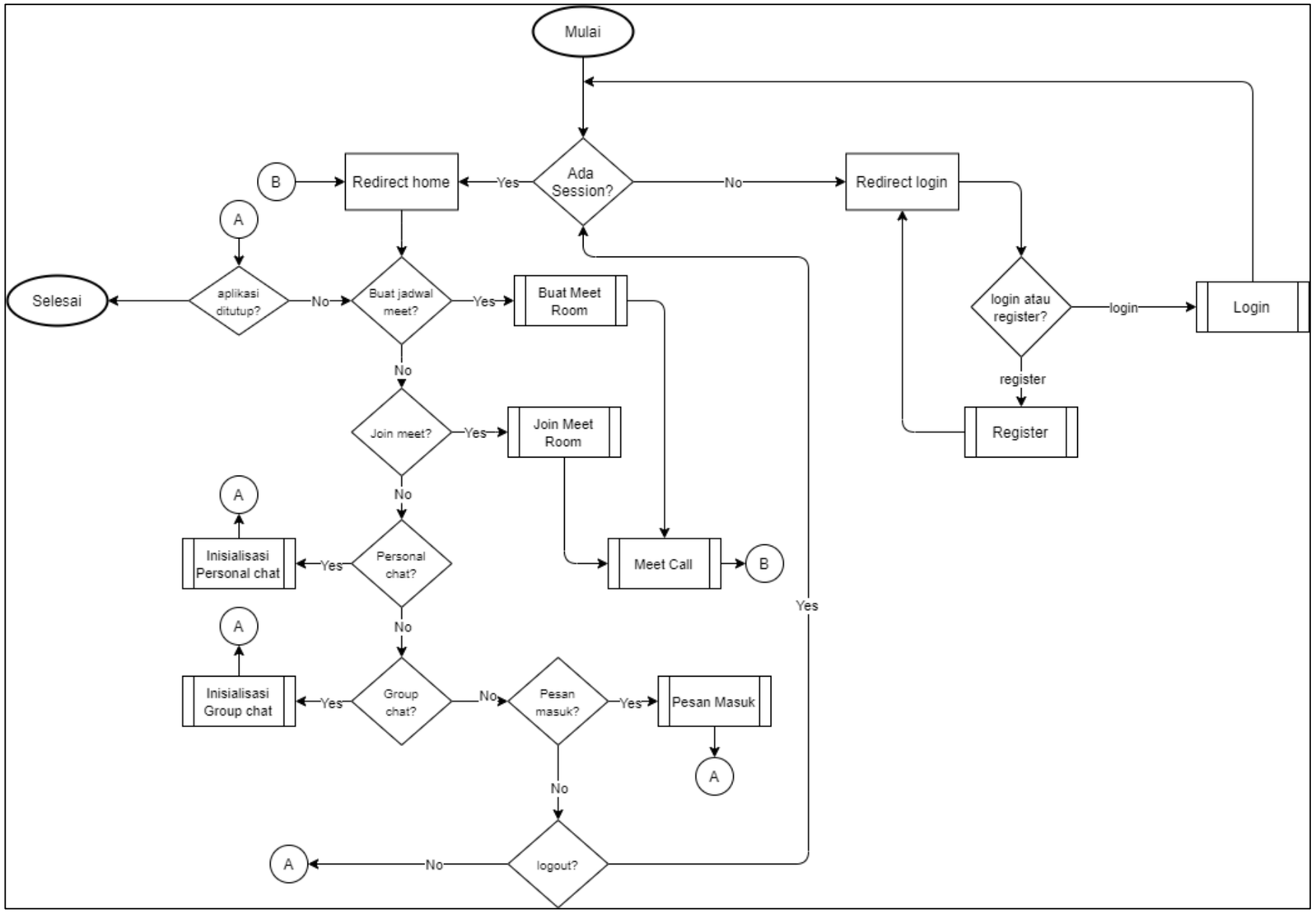
Database yang menjadi sumber utama penyimpanan data menggunakan *mysql*. Penggunaan *mysql* dinilai lebih mudah karena sintaks yang familiar dan telah dipelajari sebelumnya. Database ini yang akan tersambung dengan *framework* Django menggunakan fungsi bawaannya, hanya perlu mengisi ketentuan dari dokumentasi yang diberikan. Untuk Pengerjaan *FrontEnd* menggunakan *framework* jquery untuk javascript dan juga Bootstrap untuk css. Jquery digunakan karena telah familiar dengan sintaksnya dan juga telah dipelajari sebelumnya. Jquery sangat membantu dalam penulisan sintaks dikarenakan jquery mempersingkat sintaks vanilla javascript DOM yang sangat panjang menjadi pendek dan mudah diingat. Penggunaan *framework* css Bootstrap dikarenakan dalam pengembangan *user interface*, hal yang ingin dicapai yaitu tampilan yang simpel dan cepat untuk digunakan. Bootstrap telah menyediakan beberapa komponen siap pakai sehingga tidak perlu dibuat dari awal yang memakan waktu cukup lama.

3.3.3. Perancangan Sistem

A. Flowchart

A.1. Flowchart utama

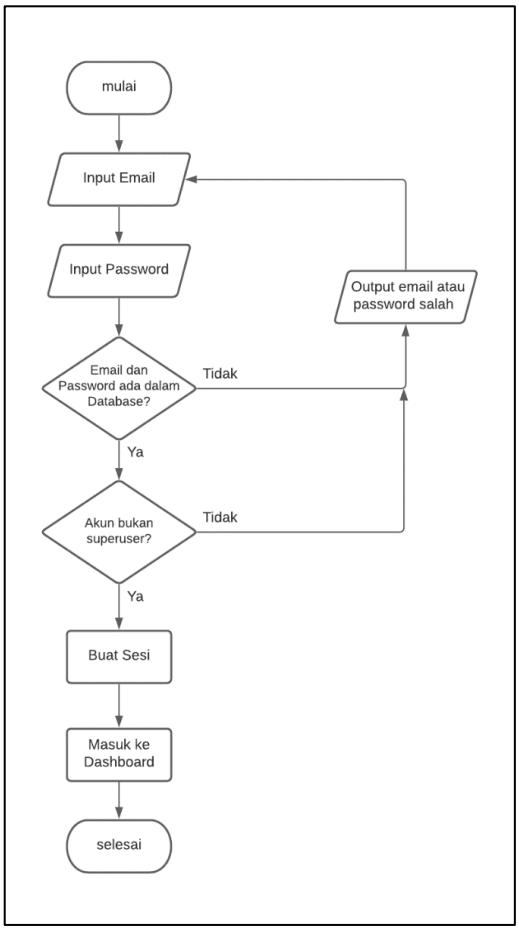
Melalui aplikasi ini, *user* dapat melakukan *chat* dan *video call*. Berikut ini adalah proses yang terjadi pada aplikasi secara umum yang akan digambarkan pada gambar 3.1.



Gambar 3.1 *Flowchart* utama aplikasi

A.2. Login

Berikut adalah contoh *flowchart* sistem *Login* pada gambar 3.2

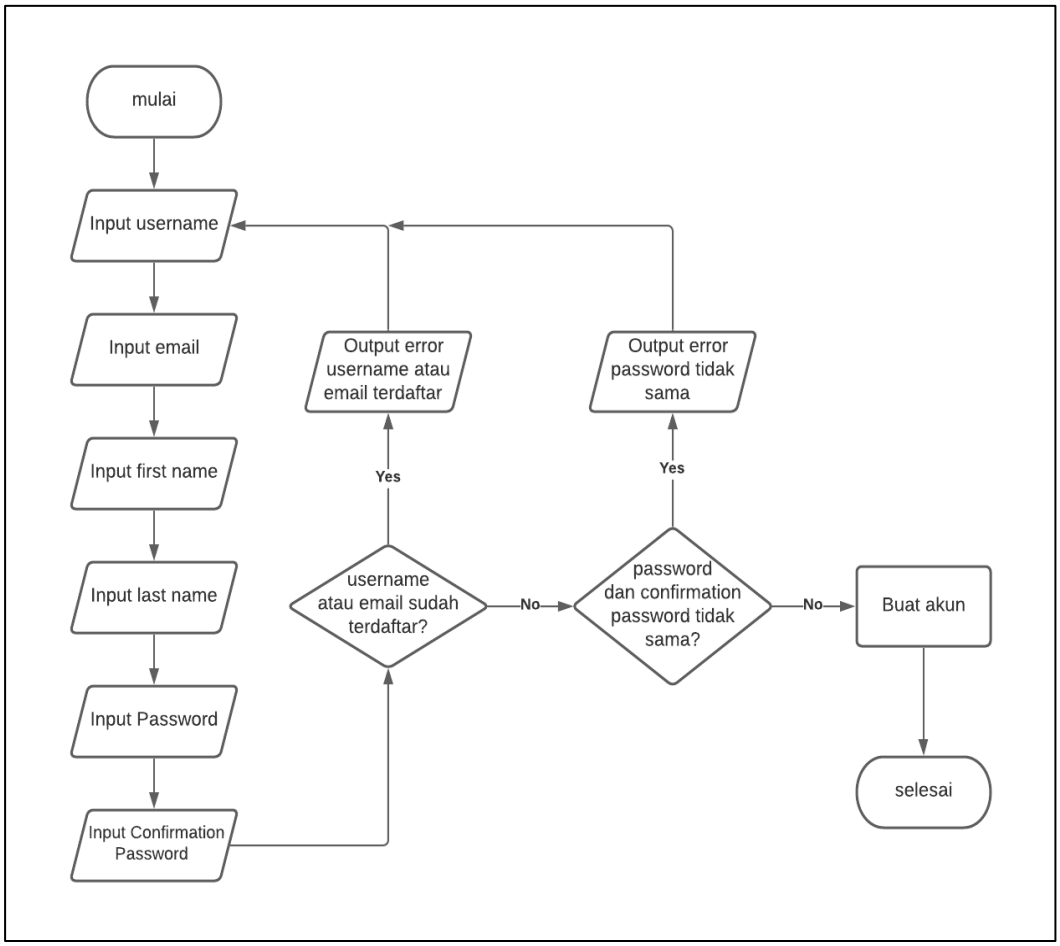


Gambar 3.2 *Flowchart* sistem *login*

Login akan menjadi fitur pertama yang ditemui pada aplikasi ini, yang berguna untuk memastikan *credential user* melalui *email* dan *password* yang diberikan. Jika berhasil *login* maka *user* tersebut bisa masuk ke dalam aplikasi untuk menggunakan fitur lain di dalamnya. Jika tidak, maka *user* tersebut tidak bisa masuk kedalam aplikasinya. Data dari *user* berupa *email* dan *password* akan dikirimkan dan dicek dalam *database* untuk memastikan apakah ada akun dengan *email* dan *password* tersebut.

A.3. Register

Selain melakukan *login*, *user* bisa melakukan registrasi. Berikut adalah proses dari *register* pada Gambar 3.3



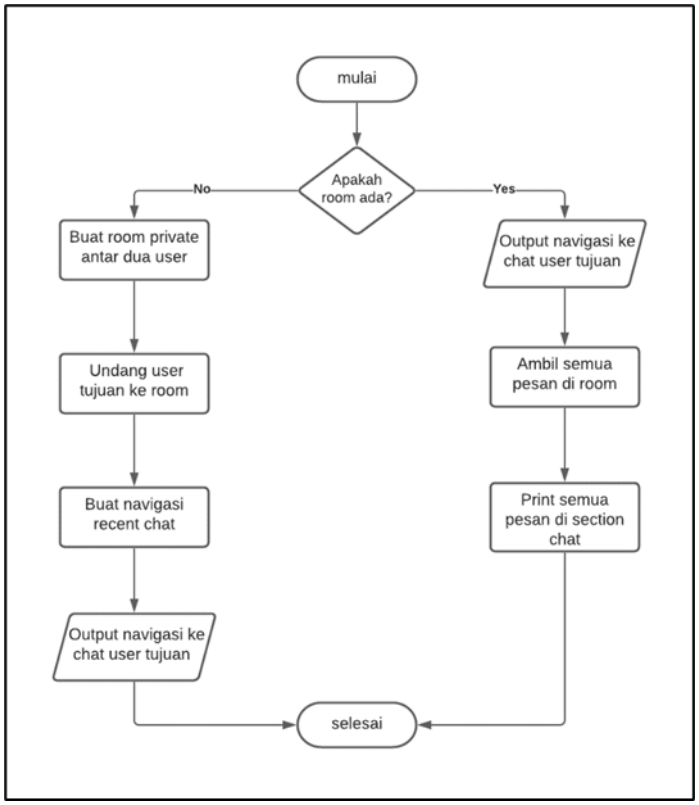
Gambar 3.3 *Flowchart register account*

Pada laman ini, *user* akan diberikan sebuah *form* yang memiliki beberapa *input* yaitu *username*, *email*, *first name*, *last name*, *password*, dan *confirmation password*. Dalam membuat akun baru, hal utama yang akan diperiksa yaitu *email* dan *username* dikarenakan *email* dan *username* bersifat *unique*, tidak boleh ada yang sama dengan data yang sudah ada. Selain itu juga *password* dan *confirmation password* harus sama untuk memastikan *password* yang dimasukkan sudah benar. setelah semua *input* dimasukkan, data tadi akan dicek apakah *username* atau *email*

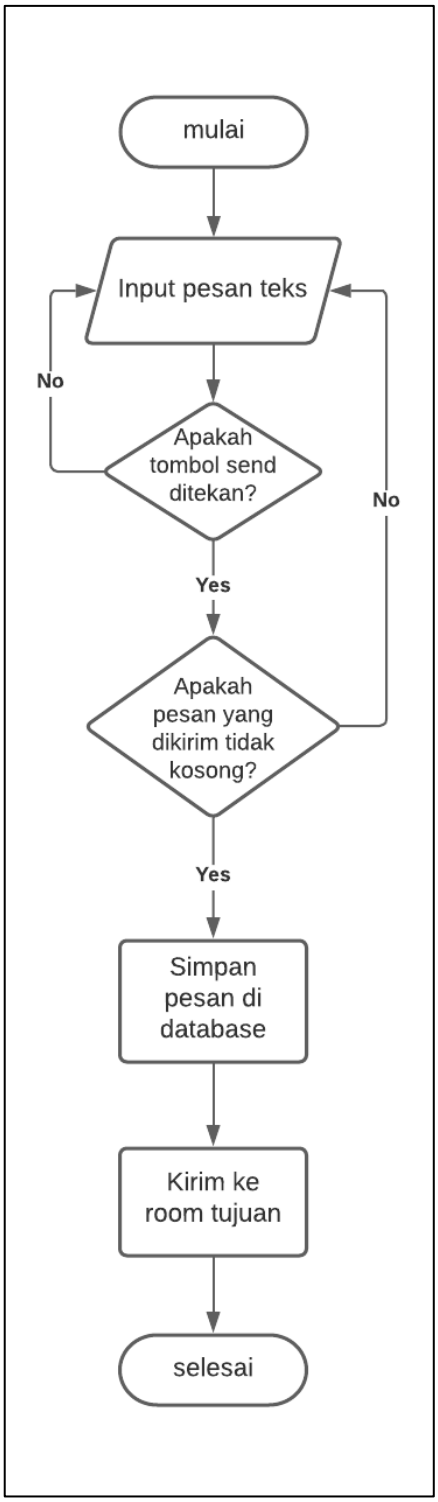
yang dimasukkan ada di *database.* jika ada, maka muncul *output* pesan *error* “username atau email telah terdaftar”. Jika tidak, dilanjutkan dengan mengecek *password* dan *confirmation password*. jika tidak sama, maka akan muncul pesan *error* “confirmation password tidak sama”. Jika sama, akan dilanjutkan dengan proses pembuatan akun dan *user* akan diarahkan ke laman *login*.

A.4. Personal Chat

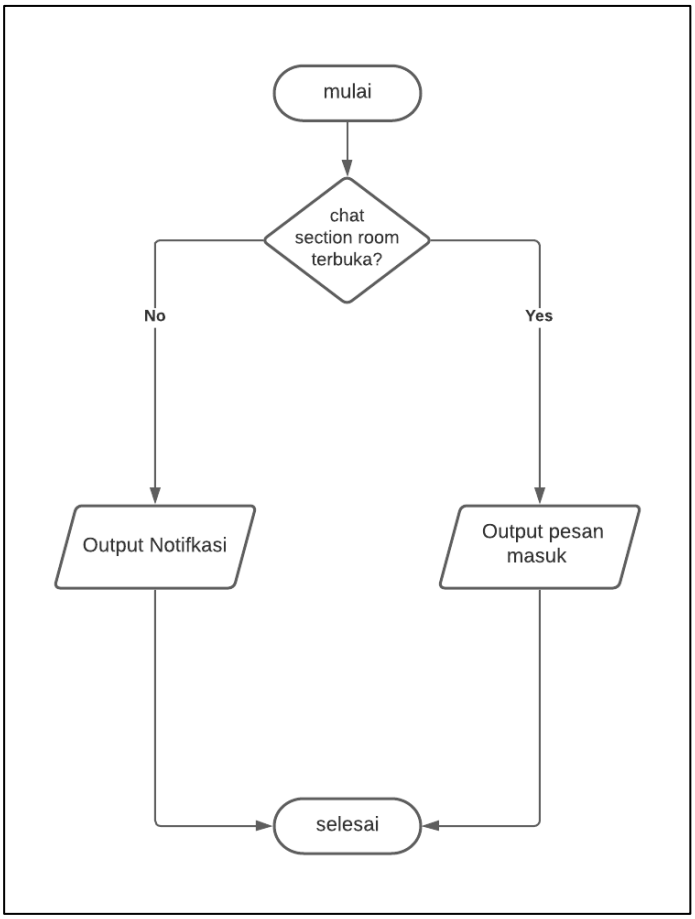
Fitur *personal chat* akan dibagi menjadi beberapa proses yang memiliki fungsi yang berbeda-beda. Berikut adalah penjelasan dalam bentuk flowchart dari proses *personal chat*.



Gambar 3.4 *Flowchart* inisialisasi *personal chat*



Gambar 3.5 *Flowchart* kirim pesan ke *room personal chat*



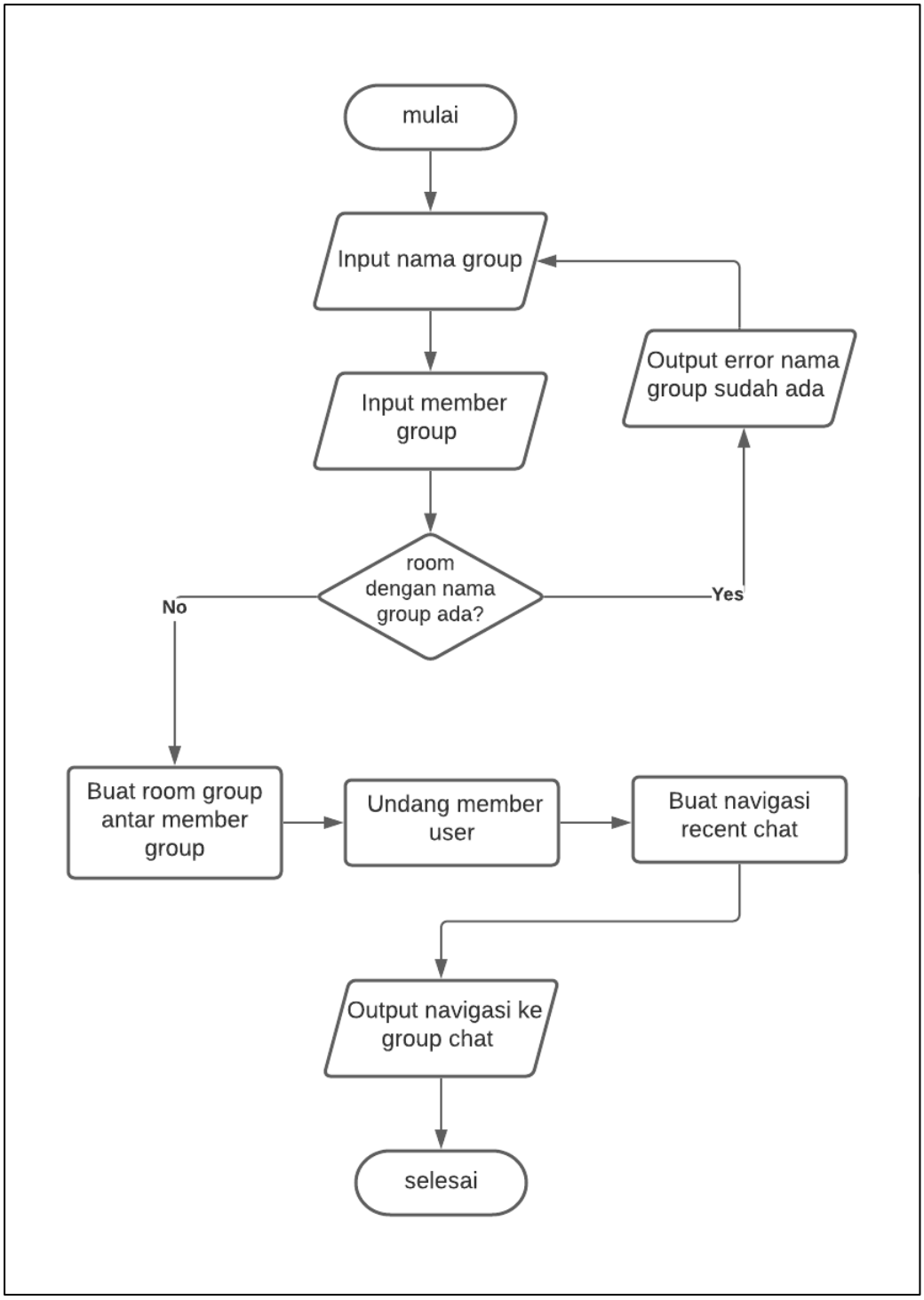
Gambar 3.6 *Flowchart* pesan masuk

Fitur *personal chat* ini memungkinkan setiap *user* untuk mengirimkan pesan kepada satu orang atau secara *personal*. Proses dimulai dari *user* menekan tombol *chat* kontak dari *user* yang ingin di-*chat*. Ketika tombol *chat* ditekan, maka *request* akan dikirimkan ke *server* untuk mengecek apakah *room* dengan partisipan antara *user* yang *chat* dengan *user* yang ingin di-*chat* ada dalam *database.* Jika ada, maka *server* akan merespons dengan nama *room* yang sudah ada tersebut, dan diarahkan ke *section chat* dari *room* tersebut. setelah pindah ke *section chat room*, *request* baru akan dikirimkan lagi ke *server* untuk mengambil semua pesan dari *room* tersebut dari *database* dan dibalikkan lagi oleh *server* kemudian pesan tadi dicetak pada *section chat room*. Jika tidak ada, maka akan dibuat *private room* dengan mengisikan partisipan dari dua *user* tersebut, kemudian *server* akan merespons

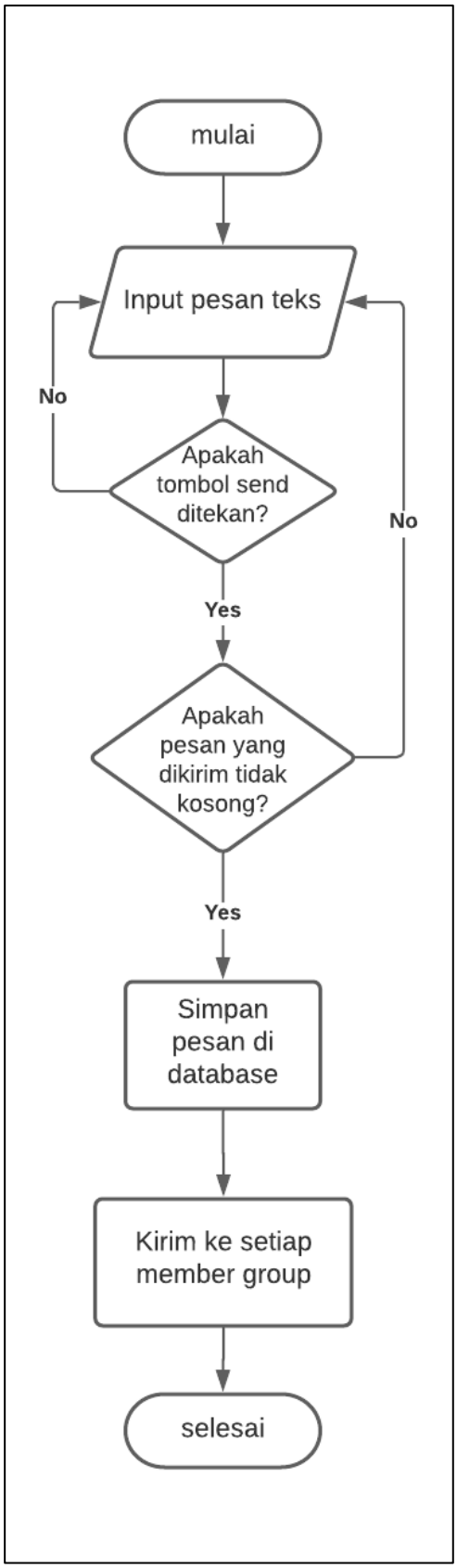
dengan mengembalikan nama *room* yang sudah dibuat sebelumnya. Setelah itu, *user* yang melakukan *chat* akan masuk ke dalam *room* yang sudah dibuat tadi dan juga akan mengundang *user* yang akan di-*chat* untuk masuk ke *room* yang sama. Setelah berhasil gabung, maka *user* bisa mulai mengirimkan pesan ke *user* tujuan. Ketika pesan dikirimkan, pesan akan dicek apakah pesan yang dikirmkan kosong atau tidak. Jika tidak, pesan akan dikirimkan ke *server* untuk disimpan ke dalam *database*, kemudian pesan tadi akan dikirimkan ke *room* tersebut. pesan yang dikirim ke *room* tadi, akan diterima oleh *user* tujuan yang tergabung dalam *room* tersebut. *user* tujuan akan mengecek apakah *section chat* sedang terbuka pada *room* tersebut. Jika terbuka, maka pesan yang masuk tadi akan langsung dicetak pada *section chat*. Jika tidak, maka *output*-nya berupa notifikasi pesan masuk.

A.5. Group Chat

Fitur *group chat* akan dibagi menjadi beberapa proses yang memiliki fungsi yang berbeda-beda. Berikut adalah bentuk *flowchart* dari proses *group chat***.**



Gambar 3.7 *Flowchart create group chat*

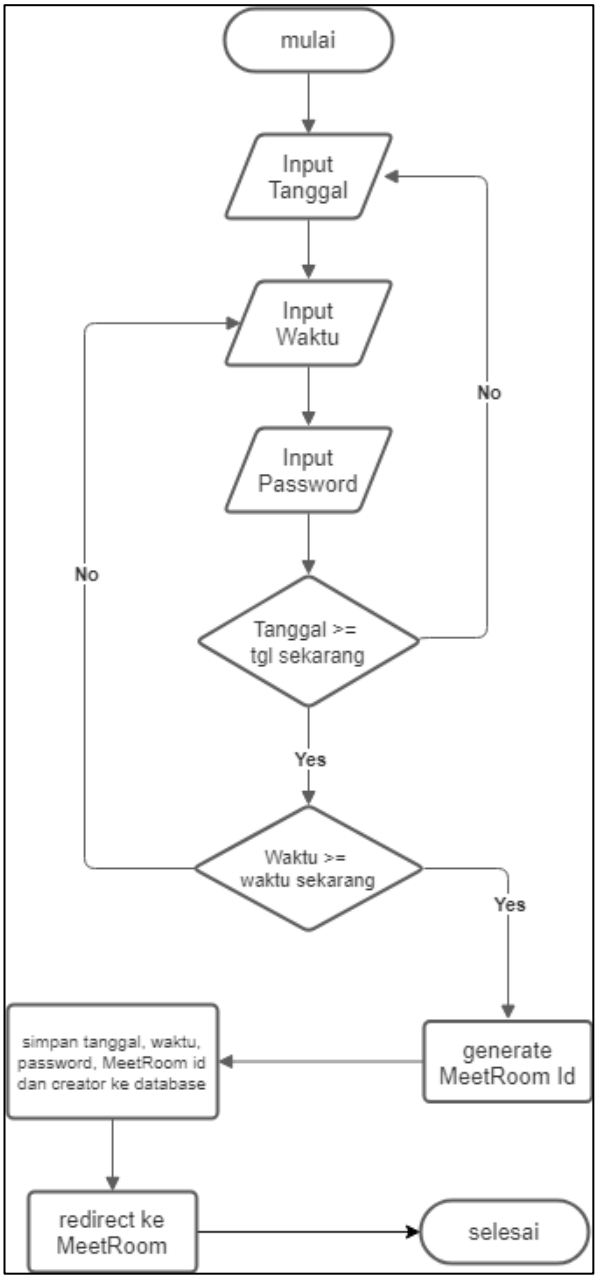


Gambar 3.8 *Flowchart* kirim pesan ke *room group chat*

Fitur *group chat* adalah fitur pengembangan setelah fitur *personal chat*. Dimana *user* bisa mengirimkan pesan ke lebih dari satu orang dengan cara membuat *group* yang beranggotakan *user* yang ingin tergabung. Pada fitur ini, *user* harus membuat sebuah *group* untuk bisa mengirimkan pesan. Proses dimulai saat *user* menekan tombol "create group". *User* akan mengisi *form* yang isinya nama group dan anggota group. Anggota group bersifat *multiple* yang artinya bisa dipilih lebih dari satu. Setelah semau *input* selesai dimasukkan, *server* akan mengecek apakah nama dari group sudah ada di *database*. jika ada, maka *group* tersebut tidak bisa dibuat dan akan mengembalikan pesan *error*. Jika tidak ada, maka *room group* tersebut akan dibuat dengan partisipan sesuai *user* yang dimasukkan sebelumnya dan *user* yang tergabung akan diundang. setelah itu *user* akan dipindahkan ke *section chat*. *User-user* yang diundang tadi akan masuk ke dalam room yang sama dengan pembuat *group*. Setelah berhasil bergabung, *user* bisa langsung mengirimkan pesan dan pesan yang dikirim akan disimpan di *database* kemudian disebarkan pada room tersebut sehingga bisa diterima oleh *user* lain yang tergabung.

A.6. Create Meet Room

Proses ini merupakan bagian dari fitur *Meet Call* dan langkah awal ketika ingin melakukan *Meet Call*. Berikut adalah *flowchart* dari *create meet room*.

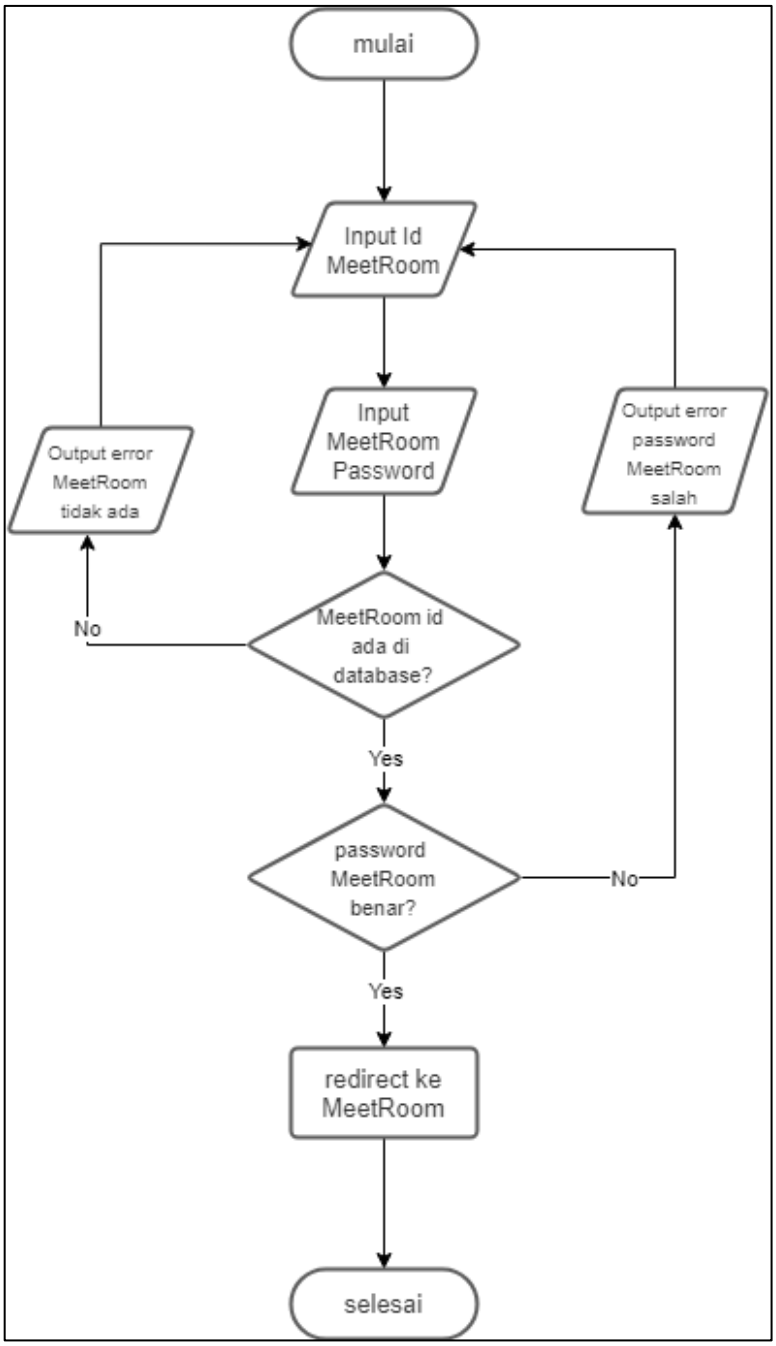


Gambar 3.9 *Flowchart create meet room*

Proses ini akan menampilkan sebuah *form* yang berisikan tanggal, waktu, dan *password*. ketiga *input* tersebut bersifat wajib dan tidak bisa dikosongkan. *Input* waktu bersifat *real-time* artinya range waktu yang bisa dimasukkan minimal waktu sekarang atau waktu yang akan datang. Begitu juga dengan tanggal yang dimasukkan, batas tanggal yang bisa dimasukkan mulai dari tanggal sekarang atau tanggal yang akan datang. *Password* berguna untuk memvalidasi setiap *user* dari luar yang akan masuk ke dalam *meet room.* Jika waktu dan tanggal telah sesuai, *room id* akan dibuat menggunakan format UUID kemudian disimpan dalam *database* beserta waktu, tanggal, *password*, dan *creator*. *Creator* diambil dari *username* pada *session* dari *user* yang sedang *login.* Setelah disimpan, *user* akan diarahkan ke dalam *meet room* dengan *path* yang ditambahkan sesuai dengan id yang di-*generate* sebelumnya.

A.7. Join Meet Room

Join meet room merupakan fitur yang memfasilitasi setiap *user* untuk bisa masuk ke dalam *meet room* yang sudah ada. Berikut merupakan *flowchart* dari *join meet room*



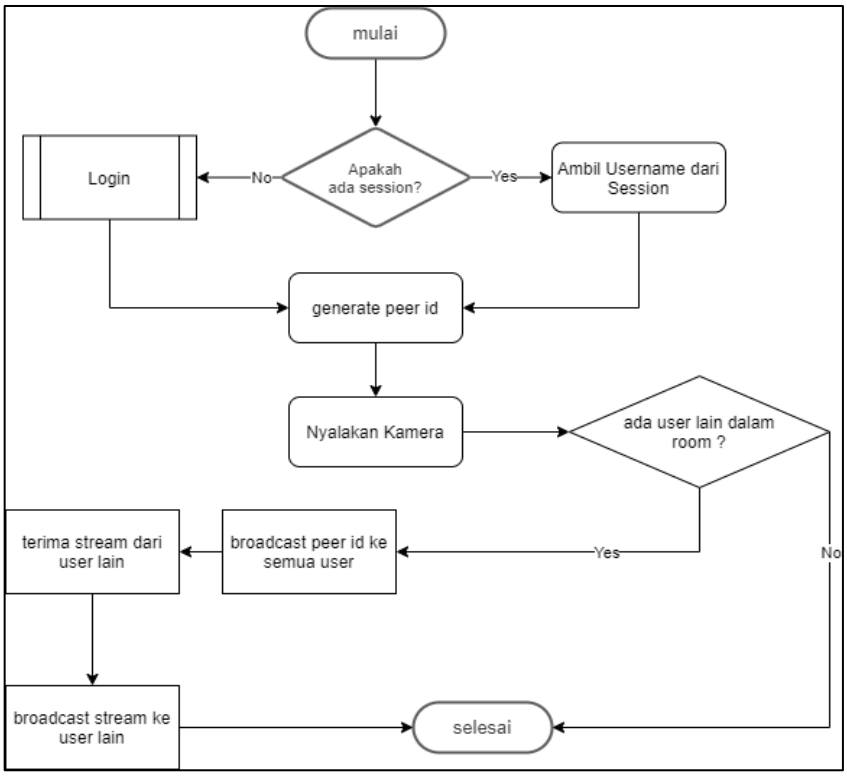
Gambar 3.10 *Flowchart join meet room*

Pada fitur ini akan ditampilkan sebuah *form* yang berisikan dua *input*. Yang pertama yaitu *room id* dan kedua adalah *password room*. *Room id* adalah *id* pada sebuah *room* yang bersifat *unique* sebagai pembeda sehingga tidak akan sama dengan *meet room* yang lain. *Unique id* ini dibuat pada proses *create meet room* sehingga jika ingin masuk pada *meet room* tersebut, maka pembuat *meet room* harus

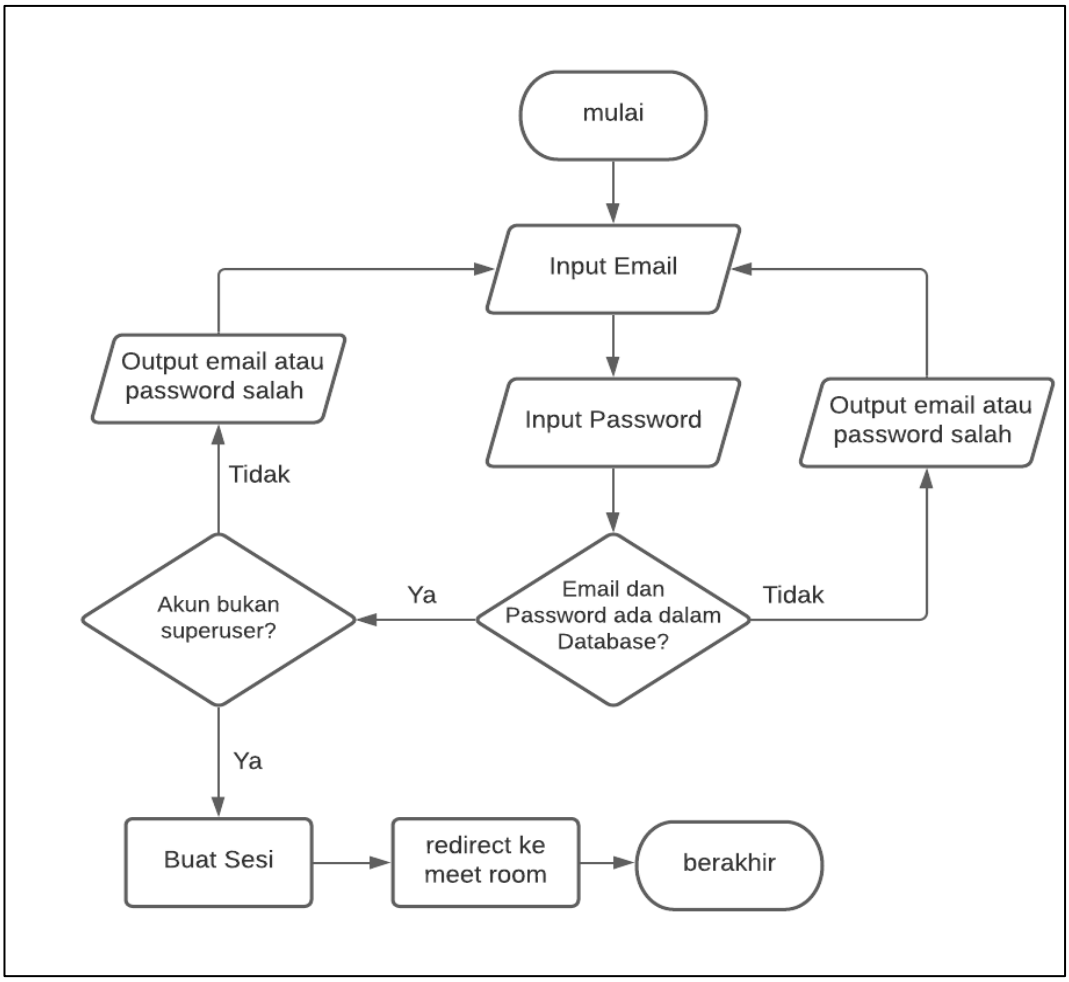
membagikannya kepada *user* yang ingin bergabung. *Password* berguna untuk memvalidasi *user* yang lain sehingga tidak sembarangan masuk ke *meet room*. ketika semua *input* dimasukkan, *meet room id* akan dicek di *database*, jika tidak maka akan muncul pesan *error* yang mengatakan *room* tidak ada. jika ada maka *password meet room* akan dicek. Jika tidak benar, maka akan muncul pesan *error* yang mengatakan *password room* salah. Jika benar, *user* akan diarahkan ke dalam *meet room* tersebut.

A.8. Meet Call

Fitur *meet call* merupakan fitur inti kedua selain fitur *chat* pada aplikasi ini. Fitur *meet call* memungkinkan panggilan secara virtual melalui video dan suara secara daring. Berikut merupakan contoh *flowchart* dari fitur *meet call*.



Gambar 3.11 *Flowchart meet call*



Gambar 3.12 *Flowchart login meet room*

Fitur ini merupakan lanjutan setelah *user* selesai membuat sebuah *meet room* dan *user* yang ingin masuk ke dalam *meet room* yang sudah ada. Untuk masuk ke dalam sebuah *meet room*, ada dua alternatif yang ditawarkan bagi *user*. Pertama menggunakan *join* pada *dashboard* seperti yang dijelaskan sebelumnya, Kedua dengan menggunakan *url*. *url* tersebut akan dibagikan oleh pembuat *meet room* yang bersangkutan. Setelah masuk ke dalam *meet room*, proses akan dimulai mengecek apakah *user* memiliki *session*. Hal ini berguna untuk memastikan *user* yang masuk adalah pihak internal perusahaan. Jika tidak memiliki *session*, maka *user* akan diarahkan ke laman *login* khusus untuk *meet room*. perbedaan dengan *login* yang dijelaskan di awal hanya terdapat pada bagian akhir dimana *login* di

awal laman *web*, setelah berhasil masuk maka akan diarahkan ke laman *dashboard*. Untuk *login* pada *meet room*, setelah berhasil masuk maka *user* tersebut akan diarahkan lagi ke *url* yang sama untuk masuk ke dalam *meet room* dan *session* akan dibuat. *session* ini beguna sebagai pengenal dari setiap video yang nantinya akan masuk dan keluar. Setelah *username* diambil, *peer id* akan langsung di-*generate* dan kamera *webcam* akan menyala. Kemudian sistem akan mengecek apakah di *meet room* yang dimasuki ini ada *user* yang lain yang sudah masuk. Jika tidak, *user* tersebut akan terus menunggu dan proses seleasi. Jika ada, *user* tersebut akan melakukan *broadcast peer id* ke semua *user* di dalamnya terkecuali *user* itu sendiri. Setelah itu, *user* akan menerima semua *stream* yang masuk dari *user* lain sebagai *feedback*, dan *user* tersebut akan membalas juga dengan *stream*-nya sendiri. Proses menyebar *stream* ini berguna agar semua *user* bisa saling menerima video *webcam* yang sedang menyala.

A.9. Screen Sharing

Untuk mengoptimalkan proses diskusi selama panggilan berjalan, fitur *screen sharing* dapat menunjangnya. *Screen sharing* memungkinkan setiap *user* untuk dapat melihat layar yang sedang dibagikan oleh *user* lain. Berikut merupakan *flowchart* dari fitur *screen sharing*.

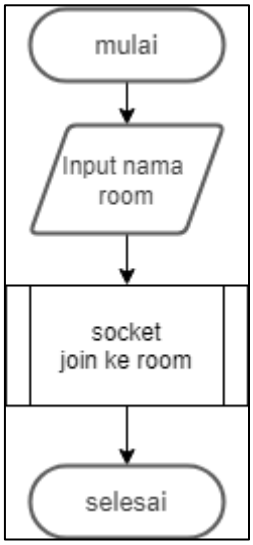
Gambar 3.13 *Flowchart screen share*

Fitur ini menggunakan *api* bawaan dari browser, maka dari itu tidak semua browser mendukung fitur ini. Untuk alurnya dimulai ketika *user* menekan tombol screen share. Ketika menekan tombol tersebut, akan muncul sebuah *alert* yang menampilkan beberapa tampilan pada layar yang sedang ditampilkan. Jenis layar akan terbagi menjadi tiga yaitu *entire screen, application window,* dan *browser tab. Entire screen* adalah seluruh layar *desktop,* jika memilih ini maka semua aktivitas pada layar tersebut akan ditampilkan termasuk membuka program dll. *Application window* adalah layar pada aplikasi yang sedang dibuka. Jika memilih ini, maka layar yang akan dibagikan hanya pada aplikasi itu saja. Ketika *user* membuka aplikasi selain yang dipilih tadi, maka layar yang sedang dibagikan terhadap aplikasi tersebut akan menjadi *blank* artinya layar tidak akan menampilkan apa-apa. *Browser tab* yaitu layar pada *tab browser* yang sedang dibuka. Perilakunya sama seperti *application window* jika *tab* yang dibuka sedang tidak digunakan maka layar akan *blank.* Setelah *user* memilih layar yang ingin dibagikan, *stream* dari layar tersebut akan dibagikan kepada *user* lain yang tergabung pada *meet room* yang

sama. *User* yang dibagikan *stream* tadi akan menerimanya dan menampilkan ke layar mereka masing-masing.

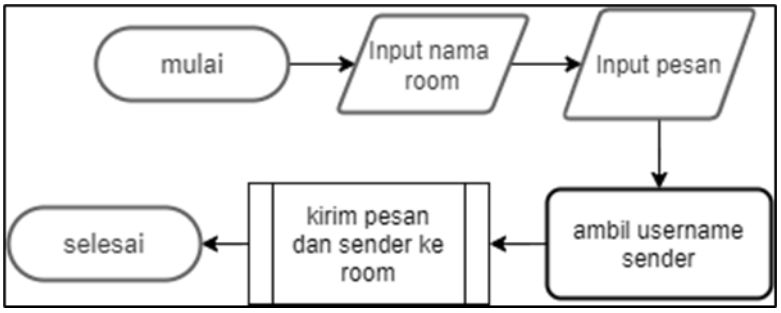
A.10. Messaging Server

Selain berjalan pada *web server*, aplikasi ini menggunakan *server* tambahan untuk menyambungkan setiap *user* yang masuk ke aplikasi ini yaitu *messaging server*. *Messaging server* ini akan dibagi menjadi beberapa proses dalam setiap penanganan *request*, berikut adalah *flowchart* dari *messaging server*.



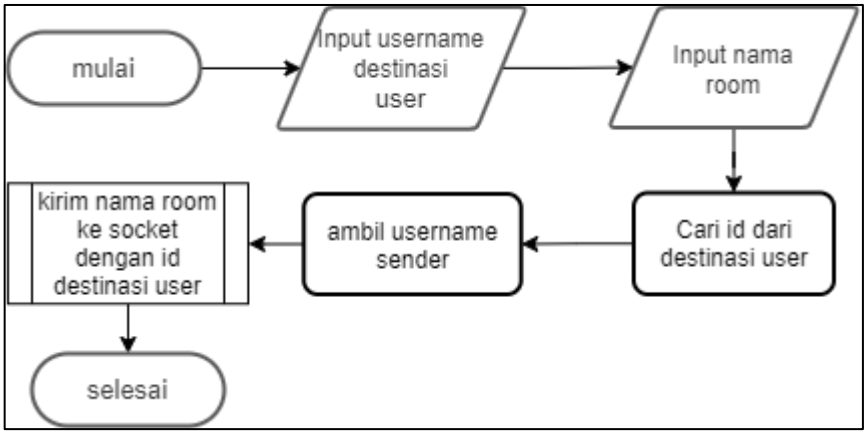
Gambar 3.14 *Flowchart join meet room*

Proses di atas menggambarkan bagaimana *user* saat pertama kali masuk ke laman *Home*. Sistem menggunakan *socket* untuk menandakan setiap *client* yang berhasil terhubung ke *messaging server*. ketika *user* berhasil tersambung dengan *messaging server*, *user* akan mengirimkan semua *room chat* yang pernah dimasuki. Setelah *room chat* diterima oleh *messaging server*, maka *socket* dari *user* tersebut akan bergabung ke dalam *room chat* yang diterima tadi.



Gambar 3.15 *Flowchart* transmisi pesan

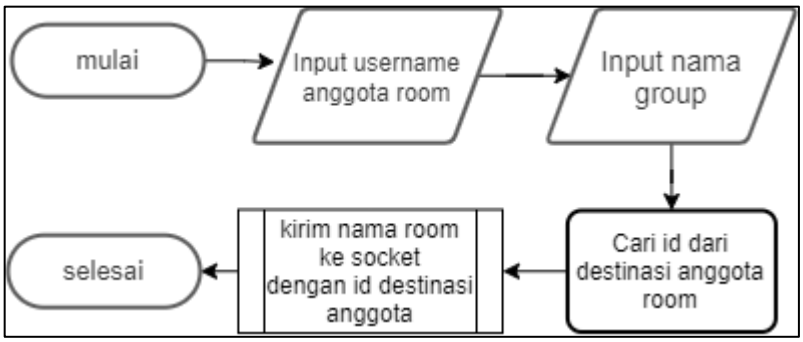
Dalam mengirimkan pesan, *user* akan mengirimkan informasi berupa nama *room chat* yang ingin dituju dan pesan yang dikirimkan yang nantinya akan diterima oleh *server* untuk diteruskan. sebelum *server* memproses pengiriman, *server* akan mengambil *username* pada *socket sender* dan dimasukkan ke dalam informasi tadi yang diterima. Setelah itu, *server* akan membroadcast informasi tadi ke dalam *room chat* sesuai informasi yang diterima sehingga bisa sampai pada *user* yang tergabung pada *room chat* yang sama.



Gambar 3.16 *Flowchart server* inisialisasi *personal chat*

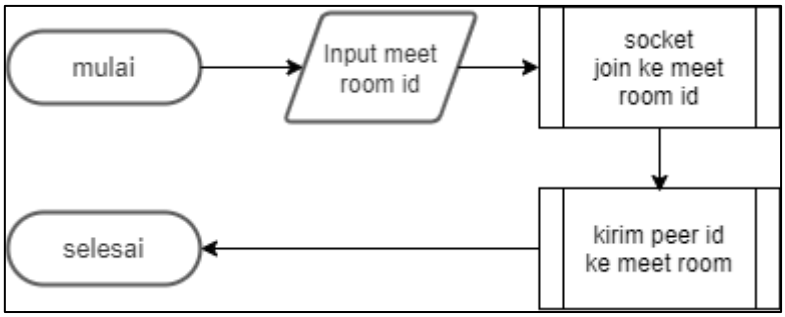
Jika sebelumnya *user* belum pernah melakukan *chat* dengan *user* lain, maka *user* harus menginisialisasi terlebih dahulu *room chat* yang berguna sebagai media penyebaran pesan bagi *user* pengirim dan *user* penerima. Proses inisialisasi dimulai dari *user* mengirimkan informasi berupa *username* dari *user* yang ingin dituju dan nama *room chat* yang ingin dituju. setelah informasi diterima oleh *server*, *server*

akan melihat *username* dari destinasi *user* kemudian mencari id dari pemilik *socket* dengan *username* tadi. setelah id ditemukan, selanjutnya *server* akan mengambil *username* dari *sender* untuk ditambahkan pada informasi sebelumnya dan mengirimkan informasi tadi ke *socket id* dari *user* tujuan. Pengiriman informasi ini tidak dalam suatu *room chat* melainkan dari satu ke satu orang saja. Setelah *user* yang dituju menerima informasi tadi, *user* tersebut akan mengirimkan informasi untuk melakukan *join* pada *room chat* seperti yang dijelaskan pada gambar 3.17.



Gambar 3.17 *Flowchart server* inisialisasi *group chat*

Proses ini hampir mirip seperti melakukan inisialisasi *personal chat*, *user* akan mengirimkan informasi ke *messaging server* berupa anggota-anggota dari *group* dan nama *group* yang akan dijadikan *room chat* bagi anggota *group* tersebut. setelah informasi diterima oleh *server*, dilanjutkan dengan mencari id dari setiap *socket* anggota grup. Setelah ditemukan, informasi nama *room chat* akan dikirimkan melalui *socket* dari setiap anggota group. Ketika informasi sampai ke tiap anggota, maka anggota tersebut akan mengirimkan informasi nama *group* tadi untuk melakukan *join room chat* seperti pada gambar 3.14.

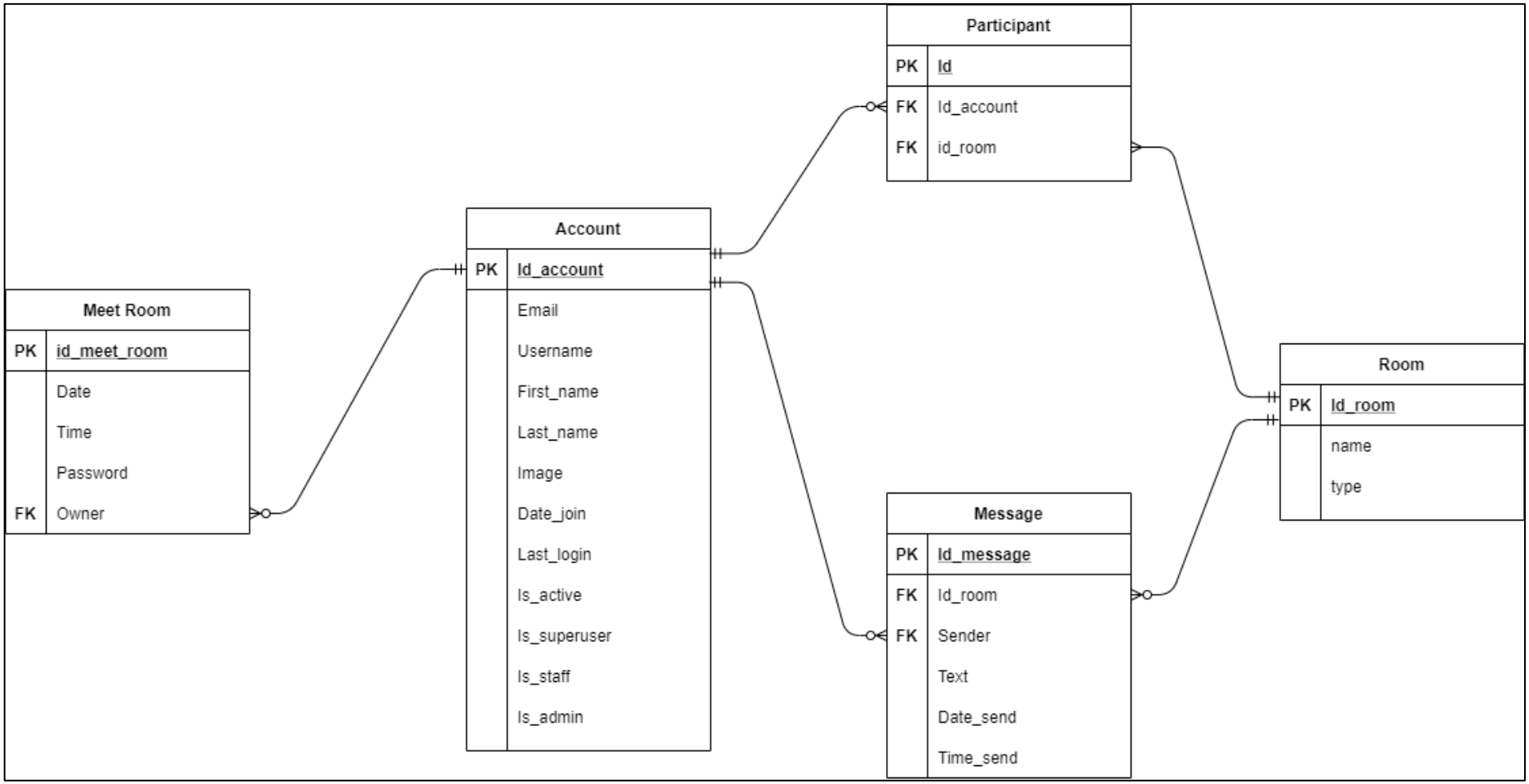


Gambar 3.18 *Flowchart join meet room call*

Dalam melakukan panggilan, setiap *user* wajib untuk terhubung dengan *messaging server*. Hal ini untuk memantau setiap *user* yang masuk dan keluar dari setiap *call* secara *real-time* sehingga bisa di-*handle* dengan tepat. Prosesnya dimulai ketika *user* masuk ke dalam laman *meet room*, ketika *user* terhubung dengan *server, user* akan mengirimkan *meet room id* dan *peer id* yang telah dibuat dan dijelaskan pada gambar 3.13. pada sisi *server,* ketika *meet room id* dan *peer id* telah diterima, maka *socket* dari *user* akan bergabung ke dalam *meet room* dengan nama sesuai dengan *meet room id* yang diberikan. Setelah bergabung, *peer id* tadi akan di-*broadcast* pada *meet room* sehingga bisa diterima oleh *user* yang lain dan bisa saling melakukan pertukaran video *stream*.

B. Database Schema

Database Schema digunakan untuk menggambarkan struktur dari *database* yang isinya berupa tabel dan *field-field* yang dipakai beserta relasi antar tabel yang disusun. Berikut adalah struktur dari *database schema* yang ditunjukkan pada gambar 3.21



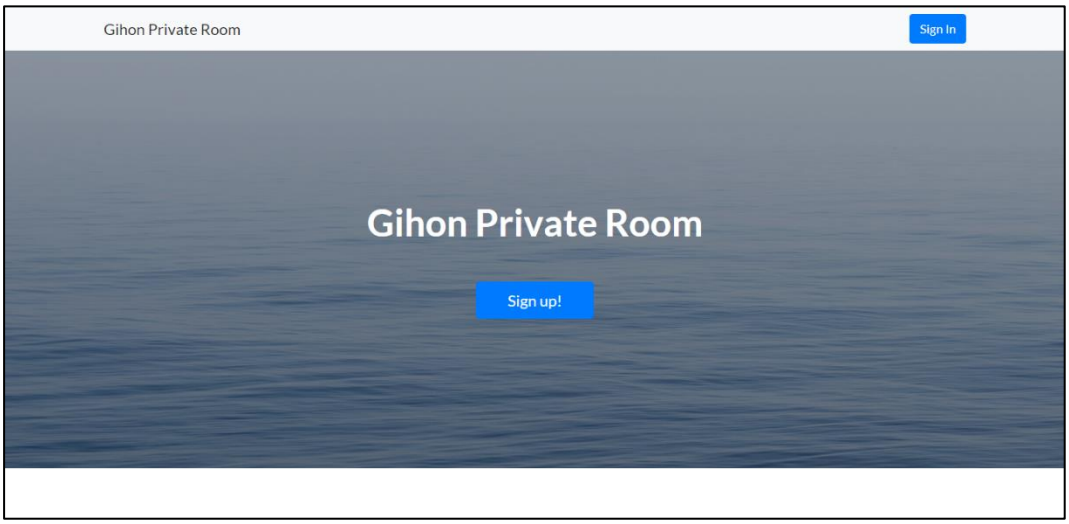
Gambar 3.19 *Database Schema*

Terdapat lima tabel yang digunakan untuk kebutuhan aplikasi. Setiap registrasi akun yang berhasil, data tersebut akan disimpan ke dalam tabel Account. Kemudian, jika user ingin memulai chat dengan orang lain, data tersebut akan disimpan di tabel Room untuk dikelompokkan user mana yang sudah pernah melakukan percakapan. Di setiap Room, pasti ada partisipan yang mengisi dan partisipan yang mengisinya bisa lebih dari dua orang, partisipan itu berasal dari Account, sehingga terjadi relasi antara Account dan Room secara many to many dan harus dibuatkan tabel yang baru yaitu Participant, untuk menampung semua user dari Room tertentu. Semua pesan pada Room tersebut akan disimpan pada tabel Message yang sudah jelas untuk menyimpan isi pesan yang sudah dikirimkan di antara user. Untuk melakukan Video *Conference Call*, maka sistem akan membuat suatu data yang menyimpan identitas dari room yang akan dibuat yang disimpan pada tabel *MeetRoom*.

3.3.4. Implementasi

A. Landing Page

User pertama kali akan masuk ke tampilan awal aplikasi yaitu *landing page*. Di sini *user* hanya dapat menuju ke halaman *Login*, tidak banyak aktivitas lain yang dilakukan. Tampilan *Landing Page* sangat simpel dan hanya terdiri dari sebuah gambar dan tombol untuk menavigasikan ke halaman *Login*.

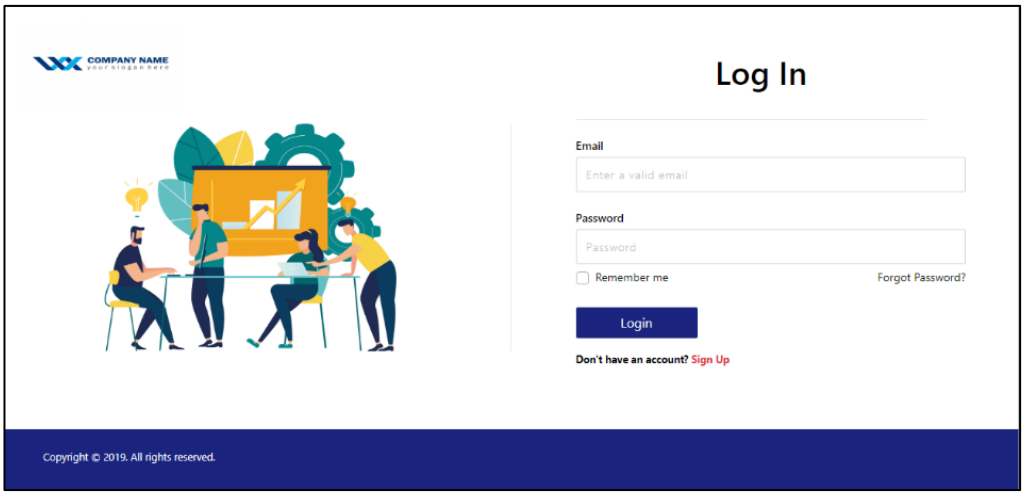


Gambar 3.20 Tampilan *landing page*

Pada halaman *landing page*, tidak ada yang dapat dibahas banyak karena hanya berupa tampilan awal saja.

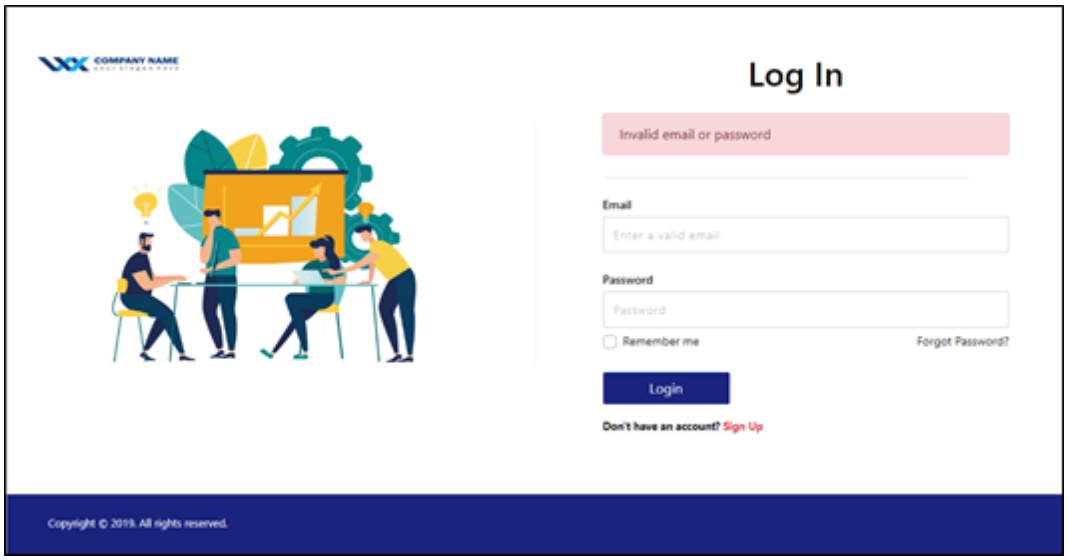
B. Login Page

Pada halaman ini, *user* diminta untuk memasukkan identitas berupa *email* dan *password* layaknya sistem *Login* pada umumnya. Berikut merupakan bentuk tampilan *Login* beserta cara kerjanya yang dapat dilihat pada gambar berikut:



Gambar 3.21 Tampilan *login page*

Pada gambar 3.21 terlihat sebuah *form* dengan *input email* dan *password*. *user* wajib mengisi kedua *input* tersebut sebagai validasi sebelum masuk ke laman dashboard. Jika authentikasi gagal, akan muncul sebuah pesan *error* dalam bentuk *component alert* berwarna merah.

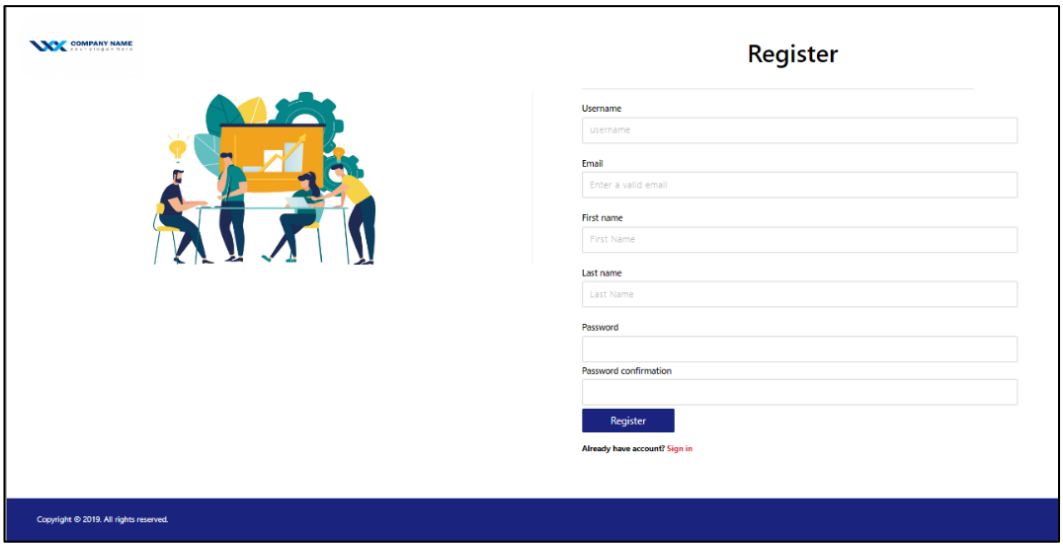


Gambar 3.22 Tampilan autentikasi gagal

setelah itu, *form* akan dikosongkan dan *user* harus mengisi lagi *form email* dan *password*. jika authentikasi berhasil, *user* akan diarahkan ke halaman *dashboard* dan bisa menggunakan semua fitur di dalamnya.

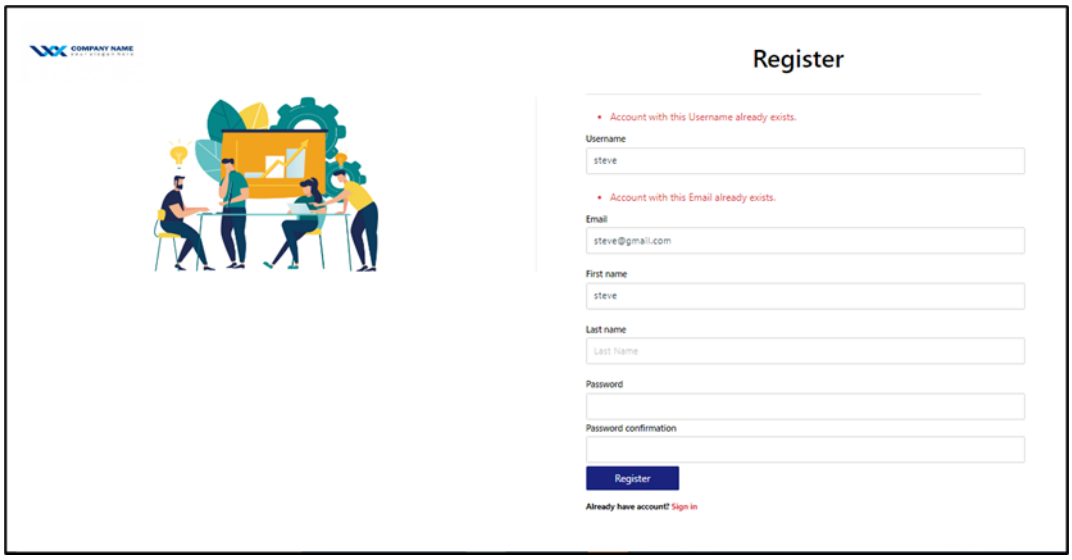
C. Register

Halaman ini bisa diakses pada sebuah link di bawah tombol *login* pada laman *login.* Berikut adalah tampilan laman *register.*



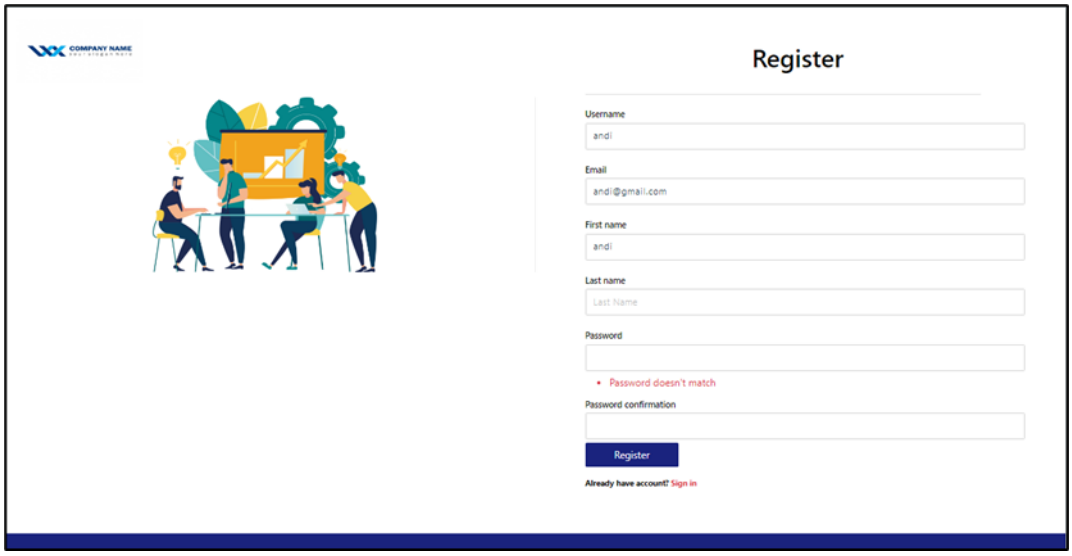
Gambar 3.23 Tampilan *register page*

Sekilas halaman *register* sangat mirip dengan halaman *login.* Perbedaannya di sini hanya pada *form input* saja, ada perubahan pada kolom *input* yang dibutuhkan dalam melakukan registrasi seperti *username, email, first name, last name, password,* dan *confirmation password.* Seperti yang telah dijelaskan sebelumnya pada proses *register,* kolom *username* dan *email* harus *unique* dengan data-data sebelumnya. Maka dari itu jika semua *form* dimasukkan, *email* dan *username* akan diperiksa terlebih dahulu apakah data tersebut telah ada sebelumnya. Jika ada maka akan muncul sebuah pesan *error* di setiap kolom *input* yang salah dan *user* harus mengisinya lagi.



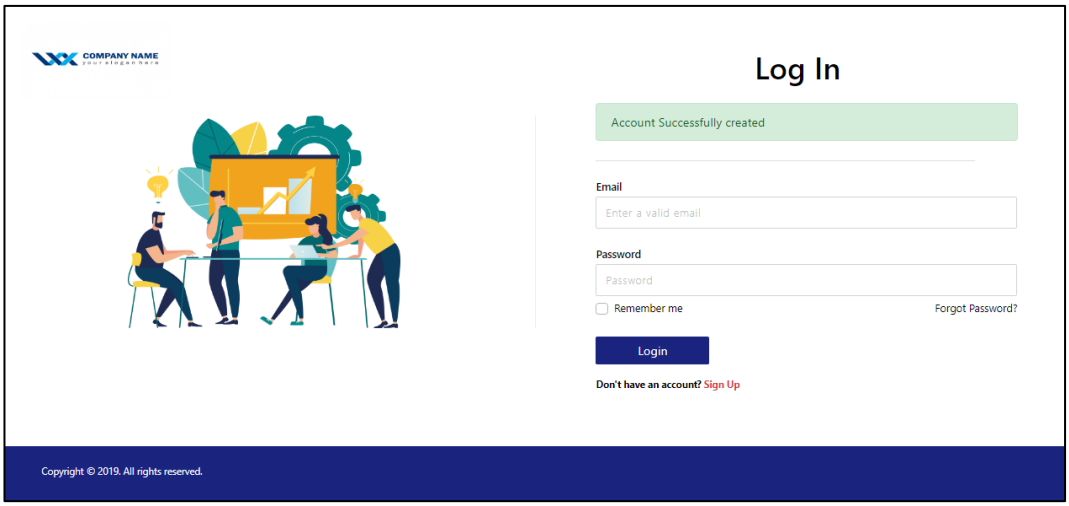
Gambar 3.24 Tampilan *error email* atau *username* yang sudah ada

Hal yang sama juga berlaku pada *input password* dan *confirmation password*. jika tidak sama, maka akan muncul pesan *error* di kolom tersebut.



Gambar 3.25 Tampilan *error password* dan *confirmation password*

Jika semua *input* telah berhasil divalidasi dan akun berhasil dibuat, *user* akan diarahkan ke halaman *login* beserta dengan sebuah *alert* hijau yang menandakan akun berhasil dibuat.

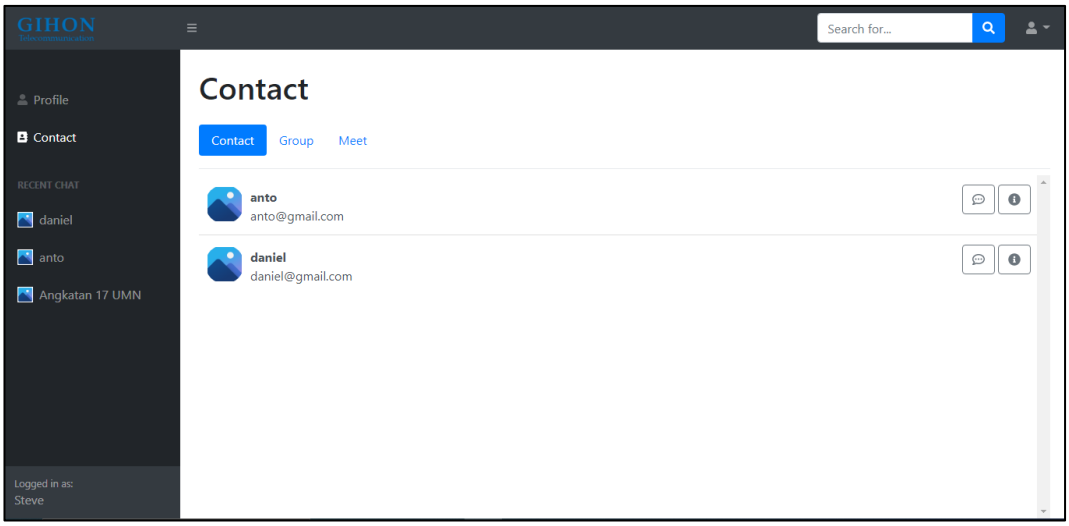


Gambar 3.26 Tampilan *alert* akun berhasil dibuat

etelah itu, *user* bisa langsung *login* menggunakan akun yang sudah dibuat tadi dan masuk ke halaman *dashboard.* Pada halaman *register* juga terdapat sebuah *link* dibawah tombol *register* untuk bisa pindah ke halaman *login.*

D. Home Page

Halaman *Home* adalah halaman yang penting pada aplikasi ini. Setelah *user* melakukan *Login* dan berhasil, maka *user* akan diarahkan pada halaman *Home*. pada halaman ini, *user* bisa melakukan berbagai aktivitas di sini mulai dari memulai *chat* dengan *user* lainnya, mengganti *password* akun, dan lain-lain. Terdapat beberapa navigasi yang mengarahkan ke *section* yang berbeda. Berikut tampilan halaman *Home* yang dibuat:



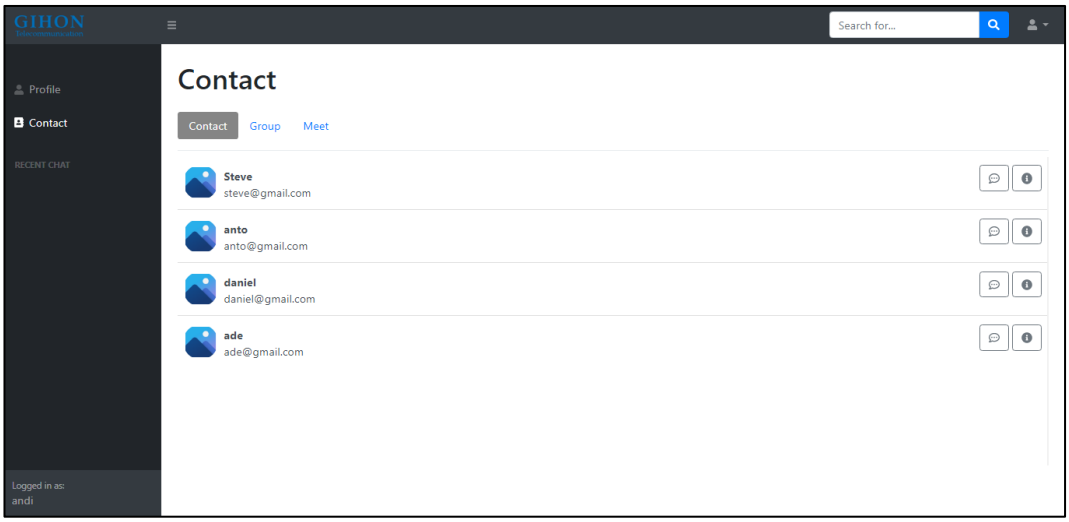
Gambar 3.27 Tampilan *home page*

Beberapa navigasi tersebut ada *Profile*, *Contact*, dan *Recent Chat.* Ketika *user* meng-klik *Profile*, maka *section Profile* akan terbuka. Dalam pengembangan fitur selanjutanya di *Profile*, *user* bisa melihat informasi akun berupa *email*, *username*, dan juga ada tombol untuk mengganti *password* akun. Pada *section Contact*, terdapat beberapa navigasi kecil di dalamnya seperti *Contact*, *Group*, dan *Meet.* Untuk navigasi *Recent Chat* hanya bisa di klik ketika *user* sudah pernah *chat* dengan *user* yang lain dan akan menampilkan daftar *user* yang di-*chat*.

E. Personal Chat

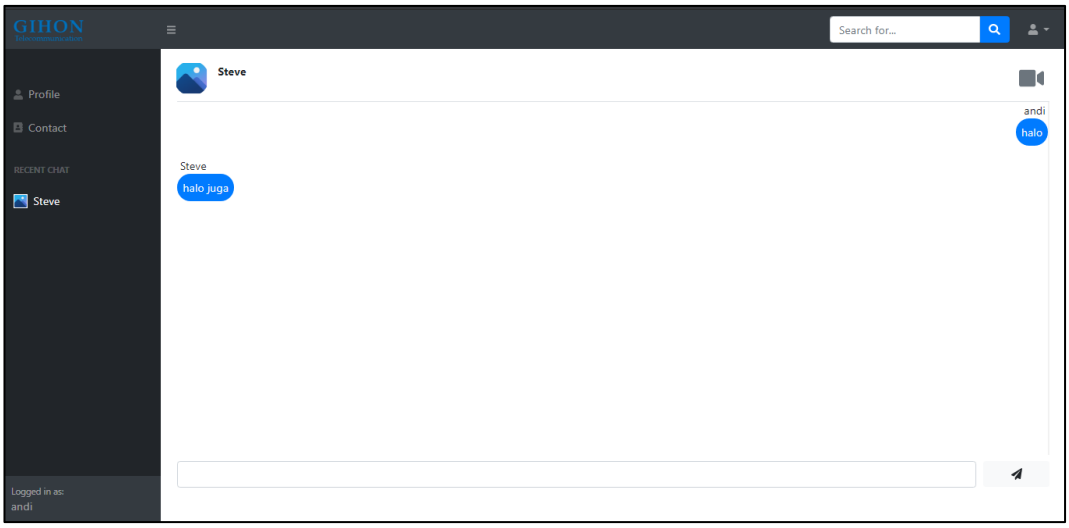
Fitur *Chat* merupakan salah satu *fitur* inti dari aplikasi ini. Seperti pada umumnya, fitur *Chat* berguna untuk melakukan komunikasi secara teks dengan *user* lain. Pada aplikasi ini, *user* bisa melakukan *Personal Chat* dengan *user* lainnya yang sudah memiliki akun. Fitur ini akan berhubungan dengan navigasi pada *Recent Chat* dimana ketika *user* melakukan C*hat* dengan *user* lain, maka akan muncul sebuah navigasi dibawah *Recent Chat* yang dinamakan sesuai dengan orang

yang di-*chat*. proses ini melibatkan bagian *BackEnd* untuk menyimpan informasi yang dikirimkan dari proses menginisialisasi *chat*:



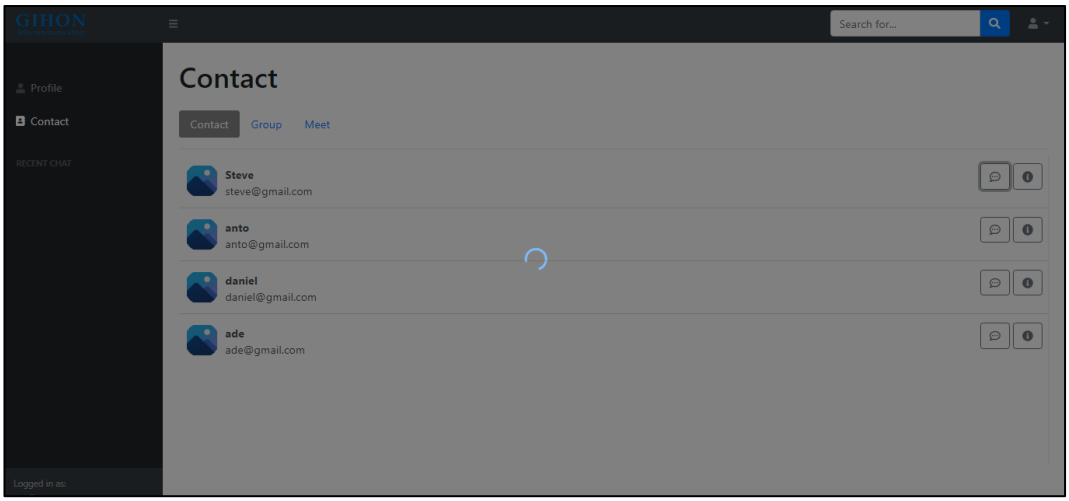
Gambar 3.28 Tampilan *tab contact* pada *section contact*

Pada *tab Contact* di *section Contact*, terdapat *list* dari *user* yang sudah mempunyai akun. Jika ingin memulai *chat* dengan orang lain, *user* hanya perlu menekan tombol di kontak sebelah *user* yang ingin di-*chat.* Kalau sebelumnya sudah pernah melakukan chat, maka navigasi *Recent Chat* dengan *user* tujuan akan langsung terbuka dan pesan yang sudah dikirimkan akan di ambil dari *database.*



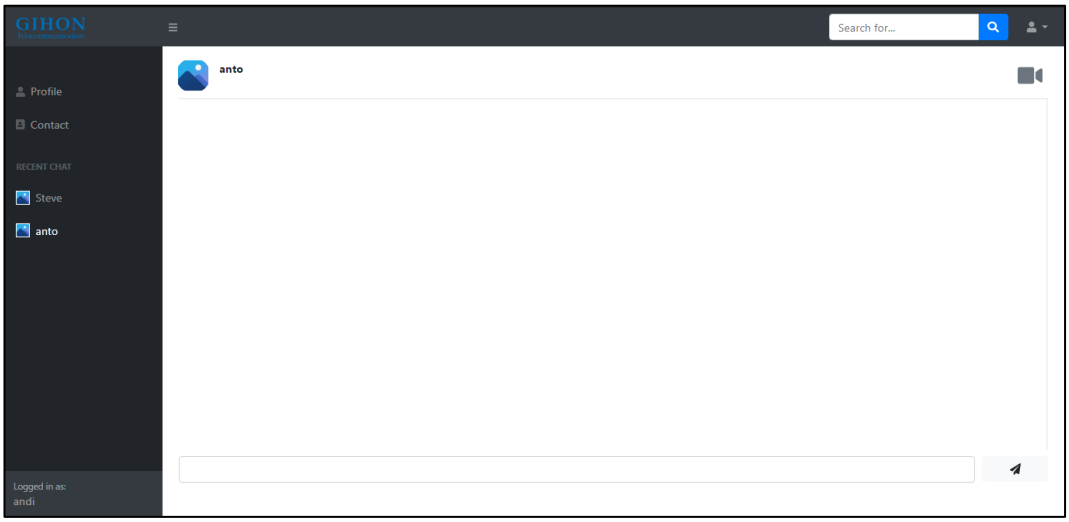
Gambar 3.29 Tampilan *personal chat* dengan *user* lain

Jika belum pernah melakukan *chat*, maka sistem akan memunculkan *loader*, *loader* tersebut dibarengi oleh *request* ke *server* untuk membuat sebuah *Room* dengan partisipan yang diisi oleh *user* pengirim dan *user* penerima.



Gambar 3.30 Tampilan *loading create chat*

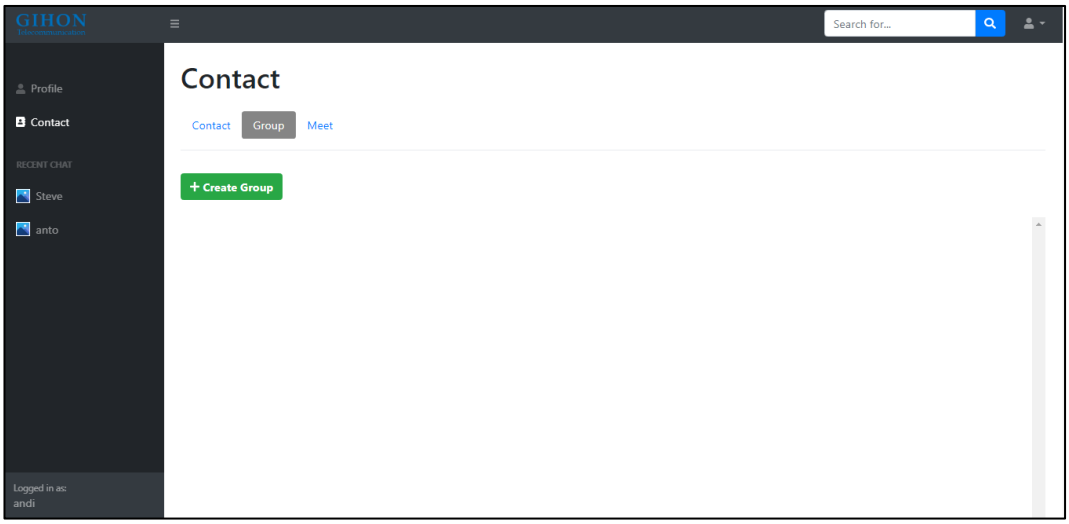
Kemudian, akan muncul *section chat* dengan *user* tujuan yang masih kosong.



Gambar 3.31 Tampilan *personal chat* kosong

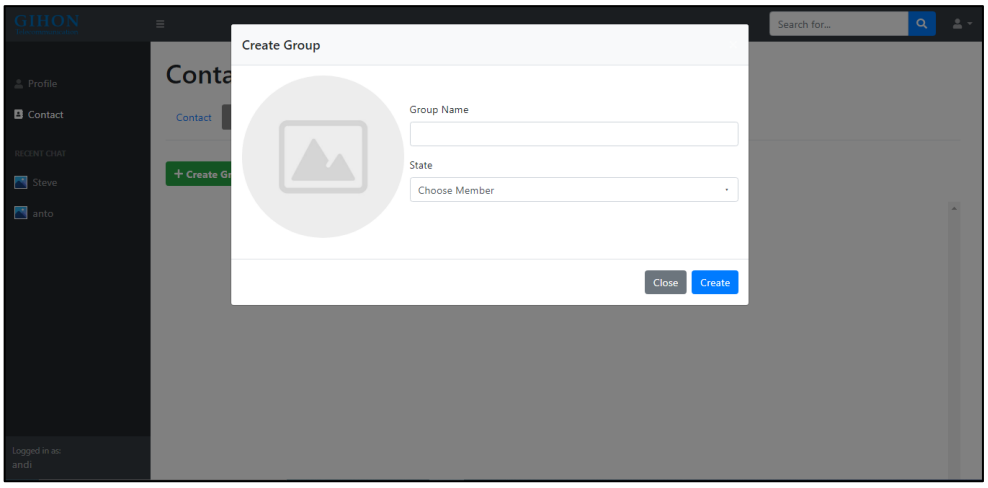
G. Group Chat

Selain melakukan *Personal Chat*, *user* juga bisa melakukan *chat* secara *group*. Fitur *Group Chat* berguna untuk melakukan percakapan secara teks dengan dua atau lebih *user*. proses *Group Chat* dimulai dari membuat *Group Chat* sampai melakukan *chat*. Berikut adalah tampilan dari *Group Chat* yang dibuat:



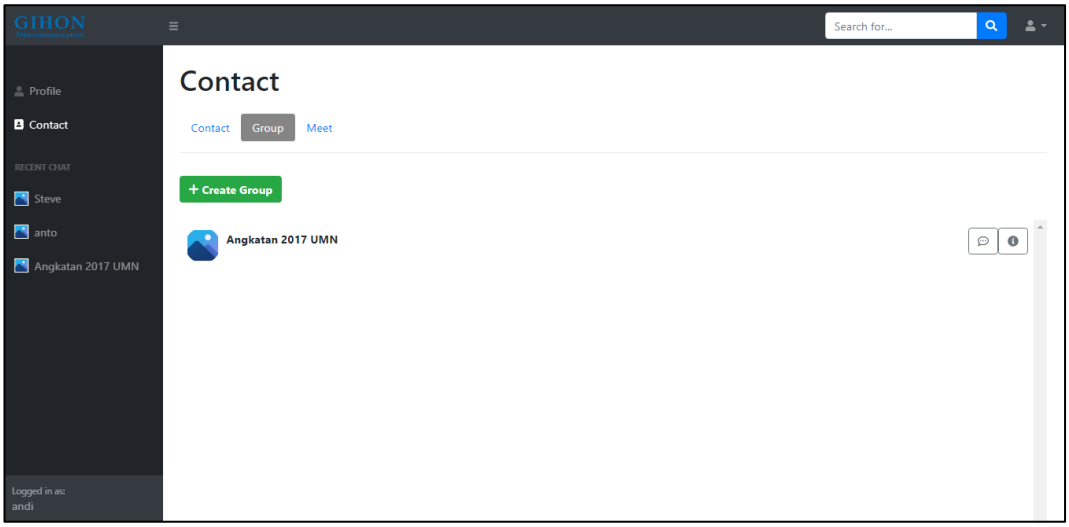
Gambar 3.32 Tampilan *section group chat*

Pada *tab Group* di *section Contact*, ada sebuah tombol untuk membuat *Group*. Ketika tombol ditekan, maka akan menampilkan sebuah *modal* yang berisi *form* untuk membuat *Group*, kolom inputan pada *form* tersebut ada nama *Group Name* dan *Member*.



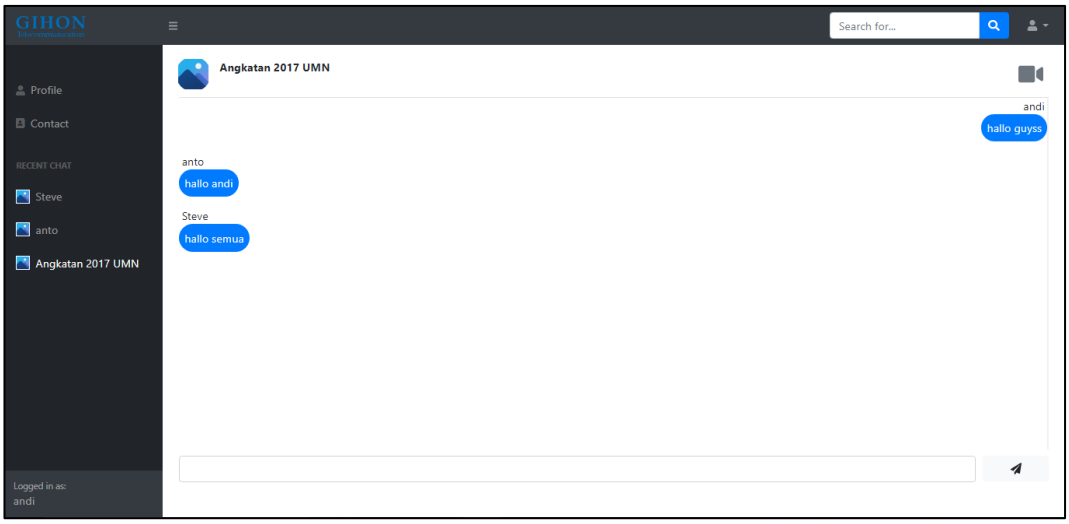
Gambar 3.33 Tampilan *modal form create group chat*

Pada kolom *input Member*, akan muncul sebuah *dropdown* yang menampilkan semua *user* yang mempunyai akun. Kolom *input Member* tipenya *multiselect* artinya bisa memilih lebih dari satu pilihan di *dropdown* yang tertera. Setelah mengisi semua kolom *input* dan menekan tombol *create*, maka semua data tadi dari nama *group* dan semua *member* akan dibawah ke *server*. *Server* akan mengecek apakah *room* dengan nama yang diberikan sudah ada atau belum. Jika belum, maka akan dibuat *room* dengan nama sesuai data yang diberikan dan semua member akan dicatat kemudian akan disimpan dalam *database.* Setelah itu, navigasi *Recent Chat* akan dibuat dan *user* akan diarahkan ke *section chat* pada *group* tersebut. *user* yang dimasukkan dalam *group* tersebut akan muncul sebuah navigasi pada *Recent chat* mereka dengan nama group tadi karena *user* pembuat *group* akan mengundang semua membernya.



Gambar 3.34 Tampilan *recent chat group* yang baru dibuat

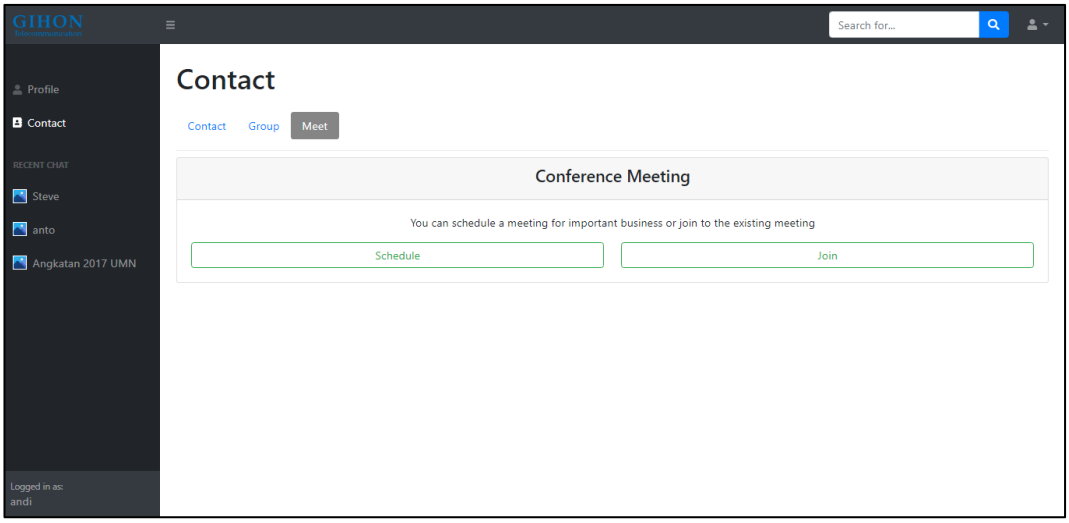
Setelah diundang, Semua *member* bisa langsung melakukan *chat* di *group* tersebut.



Gambar 3.35 Tampilan *group chat*

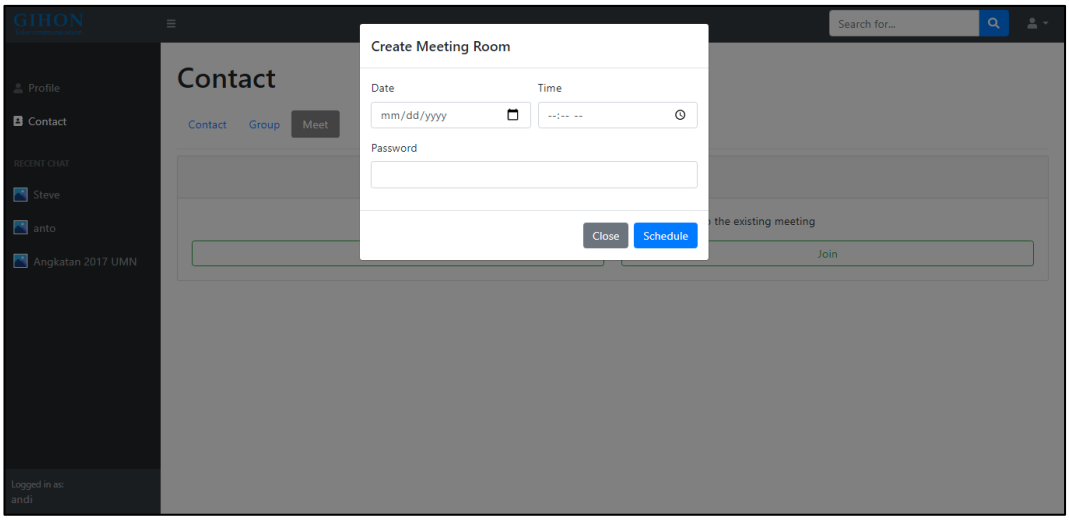
H. Create Meet Room

Fitur *Meet Call* adalah fitur inti selain *chat* yang sudah di jabarkan tadi, *Meet Call* memungkinkan panggilan secara daring melalui Video dan suara dalam jumlah yang banyak. Berikut adalah tampilan dari *Create Meet Room* yang dibuat:



Gambar 3.36 Tampilan *section tab meet*

Pada *tab Contact* di *section Contact*, ketika *user* menekan tombol "schedule", maka akan men-*trigger* sebuah *modal* yang menampilkan *form* untuk membuat *Meet Room.*

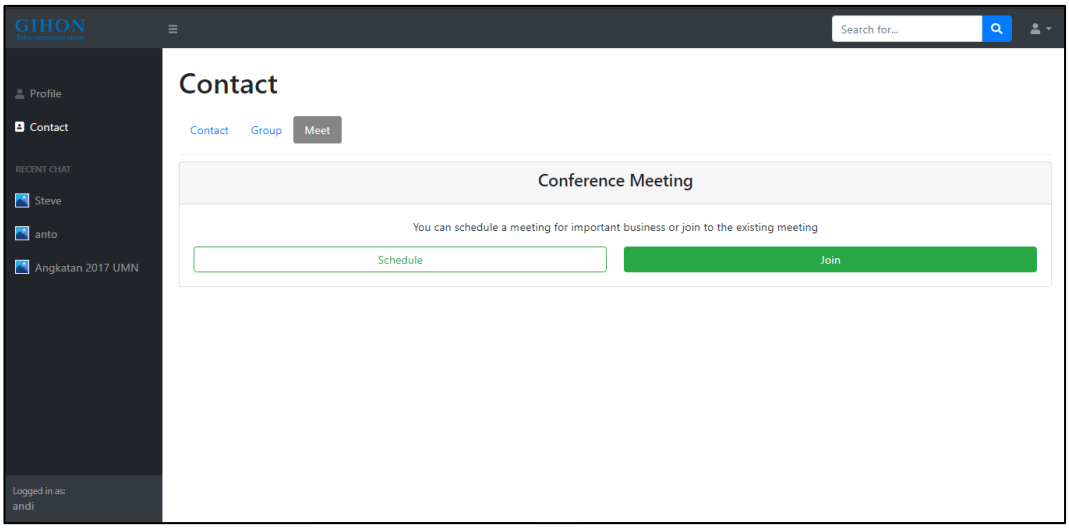


Gambar 3.37 Tampilan *modal form create meet room*

Form tersebut memiliki beberapa kolom *input* yaitu *date*, *time*, dan *password*. Untuk *input password*, berguna sebagai validasi ketika ada *user* yang lain ingin masuk ke *Meet Room* tersebut. Setelah berhasil membuat *Meet Room*, maka *user* akan diarahkan ke page *Meet Call*.

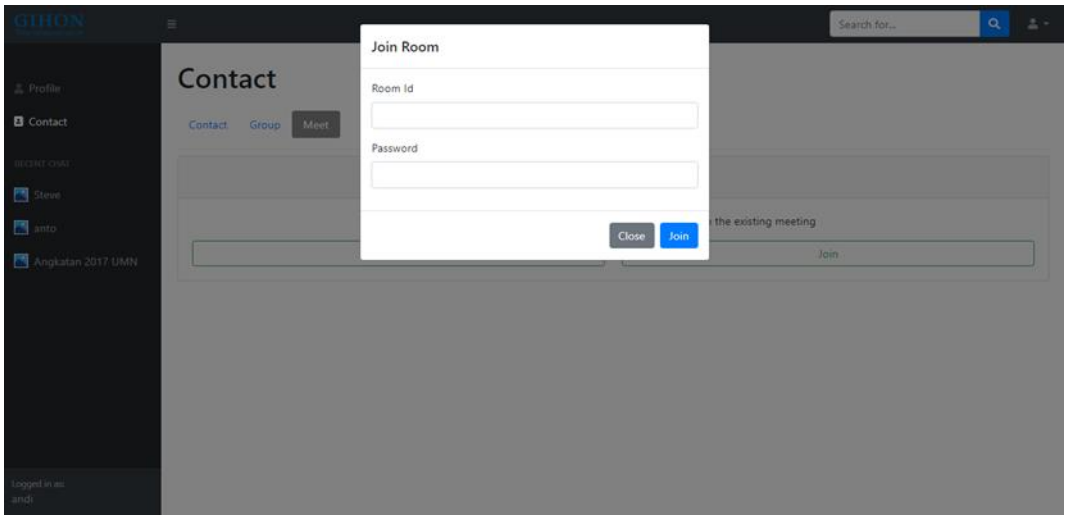
I. Join Meet Room

User yang ingin masuk ke dalam *MeetRoom* yang sudah ada bisa menggunakan dua cara, pertama dengan menyalin *link url* dari *Meet Room* yang sudah ada yang diberikan dari pembuat, kedua bisa menggunakan *join* yang terletak di *tab meet* pada *section Contact*. *Join* sendiri hanya bisa digunakan oleh *user* yang memiliki akun saja, karena dia hanya dapat diakses dari *Home page*. Sekaligus menjamin bahwa yang bisa masuk hanya karyawan dari perusahaan saja. tampilan dari *join*, dapat dilihat sebagai berikut:



Gambar 3.38 Tampilan *button join* pada *tab meet*

Tombol *join* terletak disebelah tombol *schedule* pada *section Contact* di *tab Meet*, ketika *user* menekan tombol tersebut, maka akan men-*trigger* sebuah *modal join room* dengan *dengan* sebuah *form* di dalamnya.



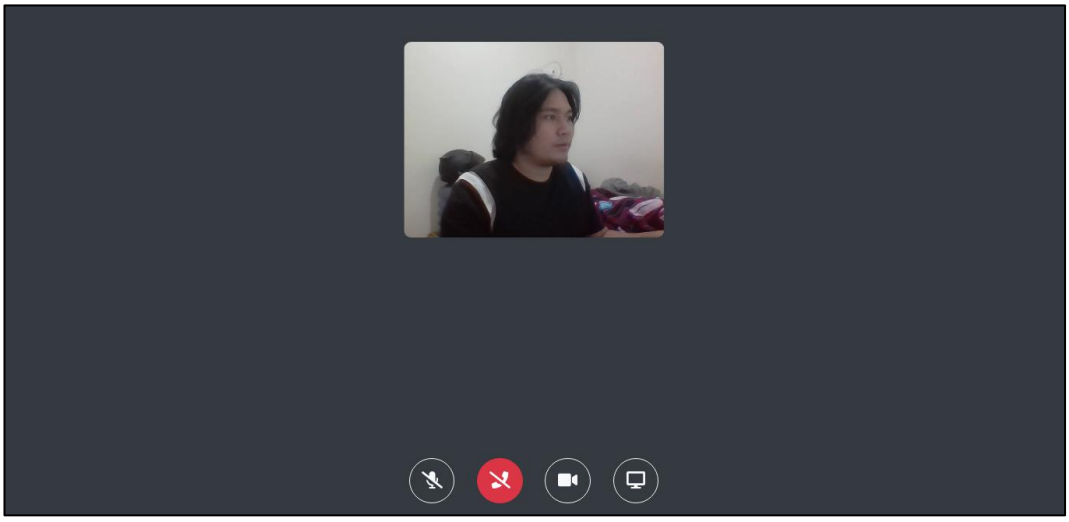
Gambar 3.39 Tampilan *modal form join room*

Terdapat beberapa kolom *input* untuk memasukkan *room id* yang ingin dimasuki dan *password* dari *meet room* tersebut. Setelah selesai mengisikan *room id* dan *password*, *input* tersebut akan dibawa ke *server* dan akan dicek apakah *room id* tersebut ada dalam *database*. Jika tidak ada, makan akan muncul *error* di bawah

room id dengan pesan “room doesn’t exist”. Jika ada, maka akan dilanjutkan dengan pesan “invalid password”.

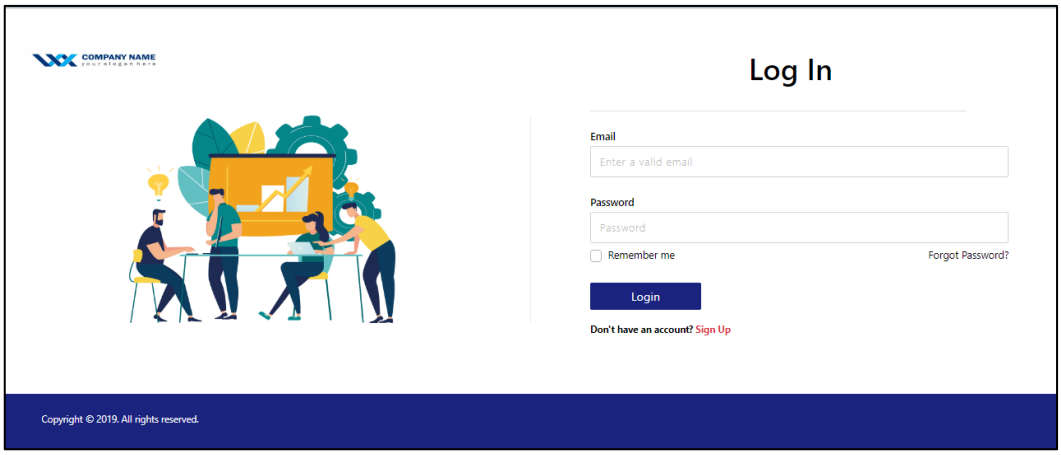
J. Meet Call

Ketika *user* berhasil *join* ke dalam suatu *Meet Room*, maka akan diarahkan ke *Meet Room Page*. *Meet Room Page* adalah sebuah halaman yang terpisah yang berguna untuk melakukan panggilan video. Semua video dari *user* lain yang bergabung akan muncul pada halaman ini. Proses dimulai ketika *user* berhasil masuk ke dalam *Meet Room* sampai berkomunikasi dengan *user* lain. tampilan dari *Meet Call* sebagai berikut:



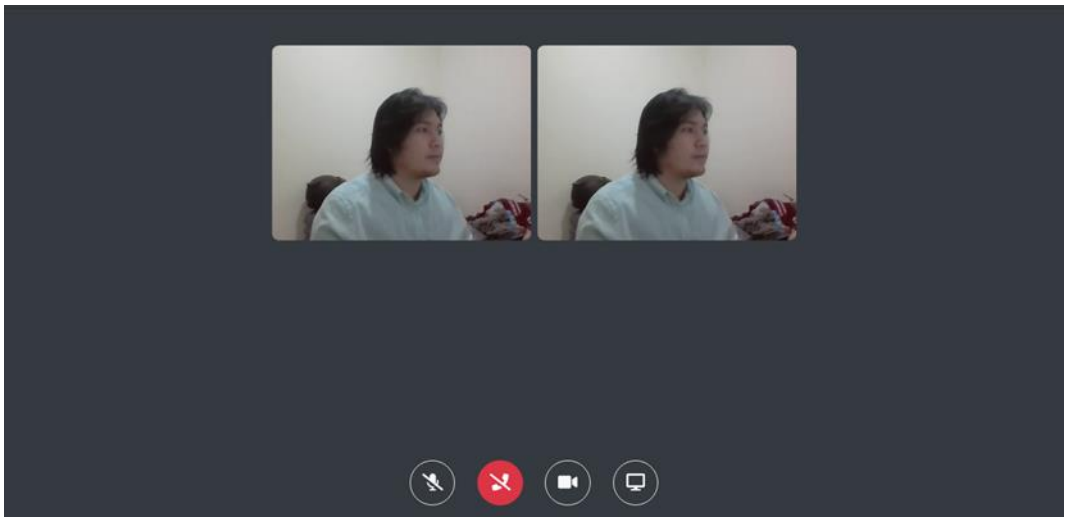
Gambar 3.40 Tampilan *meet call*

Setelah *user* masuk ke dalam *Meet Room page*, sistem akan mengecek apakah *user* memiliki *session*. Jika ada, maka sistem akan langsung membuat sebuah *peer id* yang akan berguna nantinya. Setelah itu, browser akan meminta izin untuk mengakses kamera *webcam* dan *microphone*, setelah diberikan akses untuk mengakses kamera dan mic, secara otomatis kamera akan menyala. Jika *session* tidak ada, maka *user* akan diarahkan ke halaman *login* untuk melakukan authentikasi.



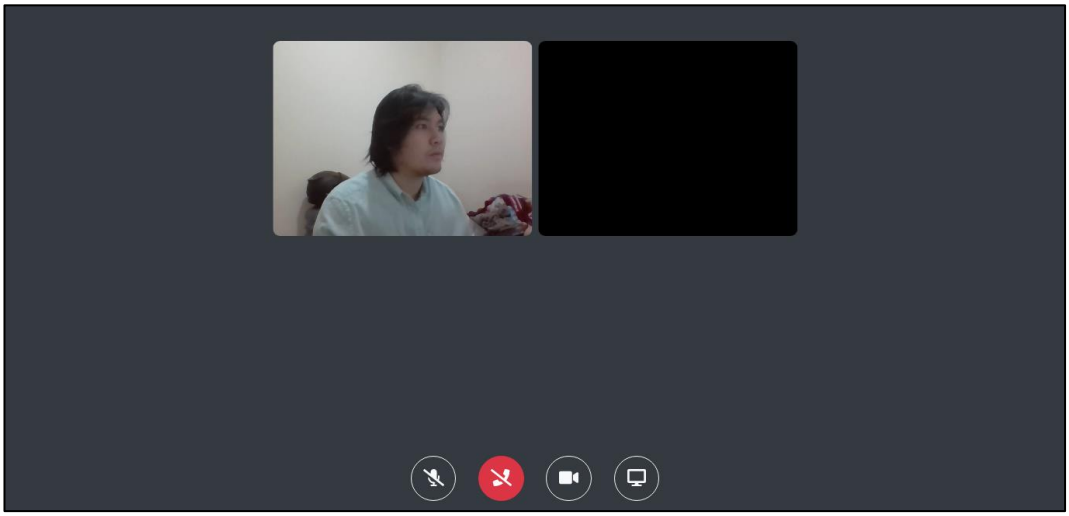
Gambar 3.41 Tampilan *meet room login*

seperti yang dijelaskan sebelumnya, sistem *login* ini berbeda dengan *login* pada awal ketika *user* membuka aplikasi dimana letak perbedaannya hanya pada kemana *user* akan diarahkan. Pada *login* ini, setelah *user* berhasil melakukan authentikasi, *user* akan diarahkan *url* yang sama, dan urutannya akan dimulai dari awal dia mengecek *session*. *Session* tadi yang berisi *username* berguna untuk ditempelkan pada *layout* video seperti pada gambar pertama beserta juga indikator kamera video dan mic yang menyala atau mati. Ketika sudah berhasil sampai pada tahap menyalakan kamera, *peer id* tadi yang dibuat akan digunakan untuk saling bertukar dengan *peer id* dari *user* lain yang tergabung dalam *Meet Call* yang sama.



Gambar 3.42 Tampilan *user* lain yang bergabung di *room* yang sama

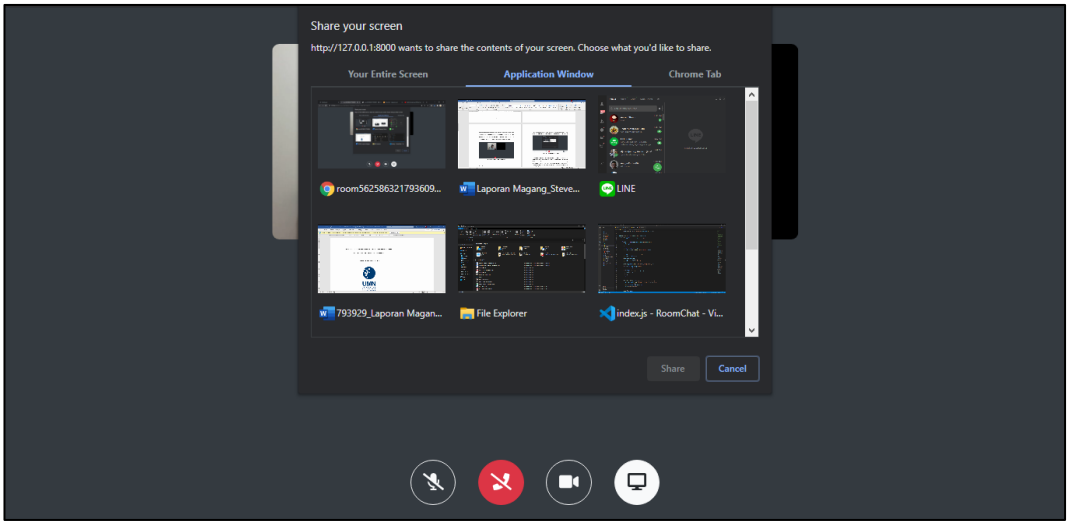
Metode untuk saling berbagi *peer id* disebut dengan *Signalling*. Dalam melakukan *Signalling,* dibutuhkan sebuah *server* lagi yang berfungsi sebagai perantara bagi *user* yang saling terhubung untuk mem-*broadcast peer id* dari *user* lain. Konsep yang sama juga telah diterapkan pada sistem *chatting* yang sudah dijabarkan sebelumnya. *Server* yang dimaksud adalah *messaging server*. Untuk *messaging server*, akan dijelaskan pada sub bab berikut. Sebelum masuk dalam pertukaran *peer id,* Sistem akan mengecek apakah ada *user* lain dalam *meet room* yang sama. Jika ada, *peer id* dari *user* yang baru masuk akan di broadcast ke semua *user* dalam *meet room* tersebut. *user* yang menerima *peer id* dari *user* yang baru masuk akan membalas dengan masing-masing video *stream*-nya mereka. *user* akan menerima setiap *stream* yang masuk dari *user* lain. Karena menggunakan kamera dan mic, *user* bisa mematikan kamera atau mic ketika melakukan panggilan. Ketika mematikan kamera atau mic, secara otomatis video atau suara *user* tersebut pada *user* lain akan mati dan begitu juga sebaliknya. Hal ini dikarenakan setiap *stream* pada *user* lain akan terus menerus di-*update*.



Gambar 3.43 Tampilan kamera *webcam* dimatikan

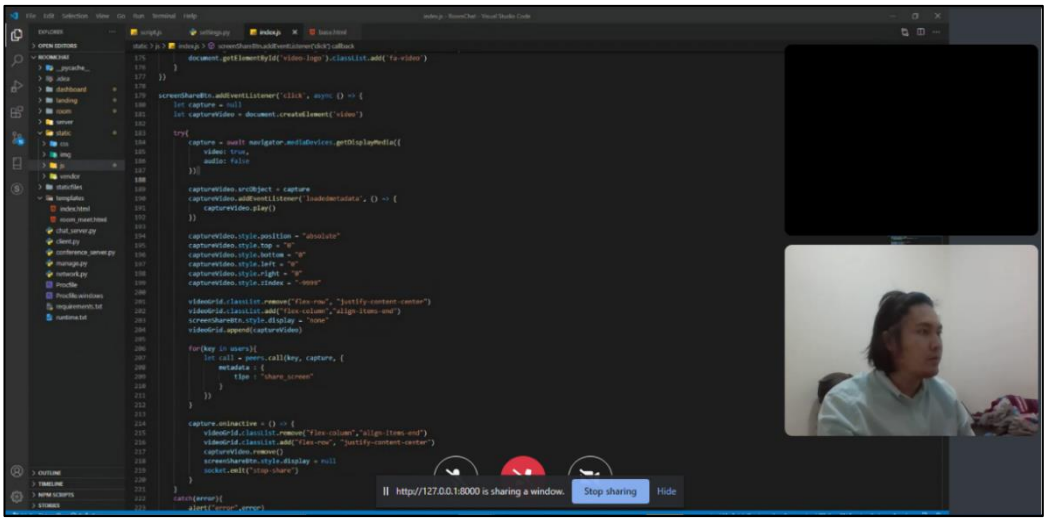
K. Share Screen

Selain terhubung secara video dengan *user* lain pada *Meet Call*, *user* juga bisa menggunakan fitur *Share Screen.* Fitur *Share Screen* adalah fitur yang bisa membagikan layar yang sedang ditampilkan oleh *user* secara langsung. Hal ini sangat membantu dalam presentasi atau menyelesaikan masalah karena *user* lain dalam *Meet Call* tersebut dapat melihat secara langsung apa yang ada di layar penge-*share*. Proses melakukan fitur *share screen* tidak terlalu rumit dan sangat simpel. Untuk mengetahui cara kerja dan alur dari fitur *share screen* dapat dilihat sebagai berikut:



Gambar 3.44 Tampilan layar yang ingin dipilih

Pada bagian bawah layout video, ada beberapa tombol yang tertera salah satunya tombol *share screen* dengan logo *monitor* yang terletak di samping kamera. Ketika *user* menekan tombol tersebut, maka akan muncul sebuah *alert* yang menampilkan semua layar yang tersedia. Layar yang bisa dipilih adalah layar yang sedang ditampilkan sekarang, aplikasi yang sedang dibuka, atau *tab* pada *browser* yang sedang dibuka.



Gambar 3.45 Tampilan *meet call share screen*

Setelah memilih layar yang akan dibagikan, layar tersebut akan tampil pada layout video dan semua video *user* akan berpindah di kanan layar. *Video stream* dari layar yang dibagikan akan di-*broadcast* ke semua *user* yang tergabung dalam *MeetRoom* yang sama. Ketika sedang melakukan *share screen*, *user* lain tidak bisa menggunakan fitur ini, secara otomatis tombol *share screen* pada *user* lain akan hilang dan hanya bisa dipakai jika tidak ada yang sedang melakukan *share screen.* Jika ingin berhenti *share screen*, *user* hanya perlu menekan tombol *stop* yang. berada pada layar yang dibagikan dan sistem akan memberitahukan bahwa *share screen* telah berhenti.

L. Messaging Server

Messaging Server adalah program perantara yang meng-*handle* pesan antara dua atau lebih aplikasi menggunakan *messaging* API (Rouse, 2005). *Messaging Server* berfungsi sebagai penyambung antar semua *user* yang aktif dan semua jalur komunikasi pada aplikasi akan melewati *server* ini. *Server* ini dibangun menggunakan *software* Node.Js serta menggunakan library javascript yaitu Socket.IO. Socket.IO memungkinkan komunikasi secara *real-time*, *bidirectional*,

dan mudah untuk diimplementasikan. *Server* ini dibangun dengan dua kapabilitas, yang pertama sebagai perantara dalam menerima dan mengirimkan pesan dari *user*. Kedua sebagai *signalling* yaitu perantara dalam pertukaran *peer id* pada *Meet Call*. Berikut adalah implementasi *messaging server* pada aplikasi ini.

```
D:\Gihon\Server>npm start

> server@1.0.0 start D:\Gihon\Server
> node server.js

server start at port 3000
[CONNECTION] Connect with ::ffff:127.0.0.1
create session Steve
Steve join Steve_Risa
Steve join Gruop Testing 10/12/2020
[CLOSE] Connection Close
{}
[CONNECTION] Connect with ::ffff:127.0.0.1
create session Steve
Steve join Steve_daniel
Steve join Steve_anto
Steve join Angkatan 17 UMN
Steve join andi_Steve
Steve join Angkatan 2017 UMN
```

Gambar 3.46 Tampilan *messaging server*

3.3.4. Kendala yang ditemukan

Terdapat beberapa kendala yang dijumpai selama melakukan praktek kerja magang. Kendala-kendala tersebut antara lain:

1. Aplikasi yang rumit.
2. *User requirement* yang sering berubah-ubah, sehingga harus merubah kode program.
3. Konsep teknologi yang dipakai masih sangat sulit untuk dipahami.
4. Terbatasnya *resource* sebagai referensi dalam mengerjakan aplikasi.

3.3.5. Solusi Atas Kendala yang Ditemukan

Dari kendala yang telah disebutkan selama proses kerja magang, tentu harus bisa ditangani agar tidak menghambat proses kerja magang. Ditemukan beberapa solusi yang kiranya dapat menangani kendala-kendala yang telah disebutkan. Solusinya berupa:

1. Berkonsultasi dengan anggota tim lain terkait kendala yang ditemukan yang berkaitan dengan *programming.*
2. Mencoba mencari referensi dari internet terkait perancangan aplikasi.
3. Mempelajari konsep dari teknologi yang digunakan.