4. IMPLEMENTASI SISTEM

4.1 Gambaran Umum Program

Aplikasi bagian Frontend akan dibuat menggunakan Flutter, bagian Backend akan dibuat menggunakan Go, Database akan menggunakan MariaDB dan Backend akan di jalankan di server Linux dengan versi Debian Bullseye (11). Untuk pengerjaan *Frontend* dan *Backend* akan menggunakan aplikasi *VS Code* dan *Goland*, untuk pemindahan atau menjalankan API pada server akan menggunakan *Bitvise*, dan untuk *repository online* akan menggunakan *Github*. Segmen Program yang disediakan adalah fungsi-fungsi penting untuk melakukan suatu aksi, untuk program secara utuh bisa dilihat pada halaman *Github* (Butuh akses); *Flutter* dan *Go*.

4.2 Relasi Gambar Wireframe dengan Segmen Program

Tabel 4.1 Relasi Gambar Wireframe dengan Segmen Program

Gambar Wireframe	Segmen Program
Gambar 3.18 Wireframe Halaman Splash Screen	-
Gambar 3.19 Wireframe Login Screen	Segmen Program 4.1., Segmen Program 4.2., Segmen Program 4.3.
Gambar 3.20 Wireframe Main Menu - Chat Rooms	Segmen Program 4.4., Segmen Program 4.5., Segmen Program 4.17., Segmen Program 4.18.
Gambar 3.21 Wireframe Main Menu - Reports Page	Segmen Program 4.15., Segmen Program 4.16.
Gambar 3.22 Wireframe Main Menu - Papan Informasi	Segmen Program 4.9., Segmen Program 4.10
Gambar 3.23 WIreframe Main Menu - Hierarchy List	Segmen Program 4.23., Segmen Program 2.24.

Gambar 3.24 Wireframe Private Chat Rooms Gambar 3.25 Wireframe Group Chat Room	Segmen Program 4.19., Segmen Program 4.20., Segmen Program 4.21., Segmen Program 4.22.
Gambar 3.26 Wireframe Group Details	Segmen Program 4.8.
Gambar 3.27 Wireframe Profile Page	-
Gambar 3.28 Wireframe Create Report Page	Segmen Program 4.13., Segmen Program 4.14.
Gambar 3.29 Wireframe Create Broadcast Page	Segmen Program 4.11., Segmen Program 4.12.
Gambar 3.30 Wireframe Create Group Page	Segmen Program 4.6., Segmen Program 4.7

4.3 Implementasi Program

Implementasi sistem sesuai dengan yang dibahas pada Bab 3 dapat dilihat pada Tabel 4.2

Tabel 4.2 Tabel Segmen Program

Nama Menu	Nama Submenu	Nama Form	Keterangan	Segmen Program
Login	Secara User – Password	/login	Autentikasi User apabila belum mendapatkan JWT	Segmen Program 4.3., Segmen Program 4.2.
	Secara JWT	/login	Autentikasi User apabila telah mendapatkan JWT	Segmen Program 4.1., Segmen Program 4.2.
Group / Kegiatan	Buat Group	/createNewHierarchy	Membuat Group baru	Segmen Program 4.6.,

				Segmen Program 4.7
	Get Informasi Group	/getGroupDetails	Mengambil informasi terkait group	Segmen Program 4.8., Segmen Program 4.9.
Papan Informasi	Buat Broadcast	/createBroadcast	Membuat *Broadcast* baru	Segmen Program 4.12., Segmen Program 4.13.
	Get Broadcast	/getBroadcasts	Mengambil *Broadcast* yang ditujukan pada user	Segmen Program 4.10., Segmen Program 4.11
Laporan	Buat Laporan	/createNewReport	Membuat laporan baru	Segmen Program 4.14., Segmen Program 4.15.
	Get Laporan Saya	/getMyReports	Mengambil laporan yang pernah *user* buat	Segmen Program 4.16.,

	Get Laporan Untuk Saya	/getReportsForMe	Mengambil laporan yang ditujukan kepada *user*	Segmen Program 4.17.
Chatting	Get Chat Rooms	/getChatRooms	Mengambil *chat rooms* yang dimiliki oleh user	Segmen Program 4.18., Segmen Program 4.19.
	Get Group Chat Rooms	/getGroupRooms	Mengambil *group rooms* yang dimiliki oleh user	Segmen Program 4.4., Segmen Program 4.5.
	Kirim Chat	*Di atur oleh *WebSocket*	Mengirim *chat* baik ke privat / grup	Segmen Program 4.20., Segmen Program 4.21.
	Get Chat Logs	/ getChats / getGroupChats	Mengambil *chat logs user*	Segmen Program 4.22., Segmen Program 4.23.
Struktur Organisasi	Get Hierarchy List	/getAllGroupShortsAdmin	Mengambil Struktur Organisasi yang	Segmen Program 4.24.,

			dimiliki atau terdaftar di dalam aplikasi	Segmen Program 4.25.

4.4 Detail Implementasi

Pada tahap ini akan menjelaskan mengenai Langkah – Langkah pengerjaan dari Segmen Program yang telah di berikan pada poin sebelumnya beserta dengan *Tools* yang digunakan untuk mengerjakan Segmen Program tersebut.

4.4.1 Persiapan Lingkungan

Pada tahap ini akan dijelaskan *tools* apa saja yang akan digunakan selama pengerjaan Skripsi ini.

1. *VS Code* digunakan untuk merancang dan membuat aplikasi menggunakan *Flutter*
2. *Go Land* digunakan untuk merancnag dan membuat API menggunakan *Golang*
3. *Postman* digunakan untuk *testing* API *Call* yang sudah dibuat untuk memastikan tidak ada *error* sebelum dimasukkan ke dalam aplikasi

Selain dari *tools* yang disebutkan, Adapun struktur file yang digunakan untuk mempermudah pengerjaan Skripsi ini.

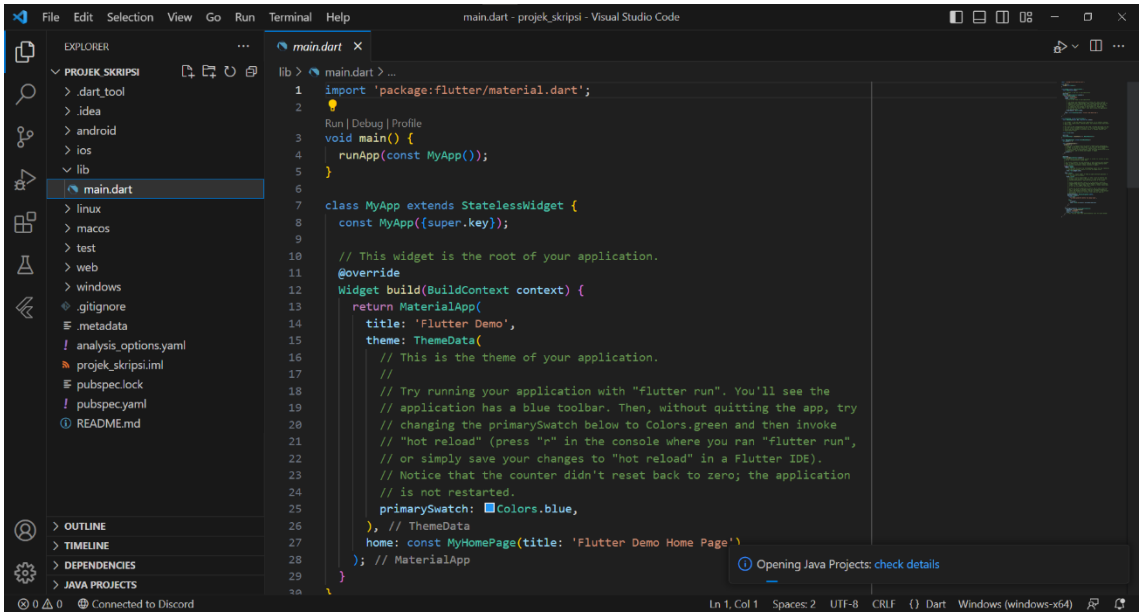
1. Struktur Folder *Flutter*

   Khusus untuk folder *Flutter* menyesuaikan dengan struktur yang sudah di *auto-generate* oleh *Flutter sendiri*. Untuk membuat projek baru, *user* bisa menggunakan *command* berikut

```
E:\Backup D\College\Semester 8\Skripsi\Main>flutter create projek_skripsi
Creating project projek_skripsi...
Running "flutter pub get" in projek_skripsi...                  2,328ms
Wrote 127 files.

All done!
In order to run your application, type:

  $ cd projek_skripsi
  $ flutter run

Your application code is in projek_skripsi\lib\main.dart.


E:\Backup D\College\Semester 8\Skripsi\Main>
```

Gambar 4.1 Buat Projek Flutter Baru

Lalu untuk mengedit / membuat aplikasi dapat dilakukan pada folder lib, dimana di dalamnya terdapat file main.dart



Gambar 4.2 Struktur Folder Projek Flutter

2. Struktur Folder *Golang*

Untuk struktur folder *Golang* akan di desain seperti demikian

```
|   main.go
|   server.log
+---cache
|       cache.go
+---config
|       config.go
|       config.json
+---controllers
|       broadcastManagement.controller.go
|       chatManagement.controller.go
|       files.controller.go
|       hierarchyManagement.controller.go
|       reportManagement.controller.go
|       userManagement.controller.go
|       websocket.controller.go
+---db
|       db.go
+---Files
|   +---Audio
|   +---Documents
|   +---Photos
|   \---Temps
+---models
|       broadcastManagement.model.go
|       cache.model.go
|       chatManagement.model.go
|       files.model.go
|       hierarchyManagement.model.go
|       reportManagement.model.go
|       response.go
|       userManagement.model.go
|       wartawarga.classes.go
\---routes
        routes.go
```

Gambar 4.3 Struktur Folder Projek Golang

Dimana folder *cache* akan diisi dengan file yang akan mengatur semua proses yang berhubungan dengan utilisasi fungsi *caching* pada program. Folder *config* yang akan diisi dengan *variable preset* yang sudah ditentukan dimana *variable* tersebut berisi *secret key* untuk fungsi AES dan untuk keperluan *sign* JWT, kredential akun yang digunakan untuk menghubungkan API dengan *database*, nama database yang digunakan, alamat *database*, dan port *database* yang digunakan. Untuk folder *db* akan diisikan dengan file yang dengan fungsi untuk mengatur koneksi antara API

dengan *database*. Folder Files akan digunakan untuk menampung *media* yang dikirim oleh pengguna aplikasi, dimana di kategorikan menjadi folder *Audio, Documents, Photos,* dan *Temps* yang berfungsi untuk menampung file yang tidak termasuk dari 3 kategori sebelumnya. Untuk Folder *routes, controllers,* dan *models* akan saling berhubungan, dimana pada folder *routes* akan diisi dengan file yang menampung semua API *Call* yang dimiliki dan dilayani oleh API yang nantinya akan diarahkan ke folder *controllers* yang diisikan dengan file-file yang memiliki fungsi sebagai *gatekeeper* untuk filter dan validasi parameter / variabel yang dikirimkan kepada API melalui API *Call*. Lalu untuk folder *models* berisikan dengan file-file yang memiliki fungsi utama untuk melakukan suatu aksi.

4.4.2 Login

1. Segmen Program 4.1

   Segmen Program ini berisi fungsi pada *flutter* yang digunakan untuk membuat sebuah *HTTP Request* dengan tujuan memverifikasi JWT yang dimiliki oleh pengguna. Tidak ada parameter yang dimasukan ke dalam *request* namun menambahkan *authorization header* yang berisikan JWT pada *request* itu sendiri yang dimulai dengan *string* "*Bearer* ", dimana *string* tersebut adalah ciri khas untuk autorisasi JWT. *Response* yang dikembalikan oleh API berupa konfirmasi apakah JWT masih berlaku atau tidak. Apabila tidak maka aplikasi akan membawa pengguna pada *login screen* untuk proses *login* ulang, namun apabila masih valid maka aplikasi akan membawa pengguna ke *main menu*.

2. Segmen Program 4.2

   Segmen Program ini berisikan fungsi pada *Golang* dimana terdapat logika untuk melakukan pengecekan JWT yang diterima dari *HTTP Request* yang dibuat pada Segmen Program 4.1. Beberapa hal yang di cek dalam JWT adalah *key* yang digunakan untuk memuat JWT tersebut, apakah sesuai dengan *key* yang digunakan oleh server, kalau tidak sesuai maka akan mengembalikan *response* "*unexpected signing method*". Apabila *key* valid maka selanjutnya algoritma akan melakukan pengecekan masa berlaku dari JWT tersebut, apabila sudah kadaluarsa maka akan mengambilkan *response* "*invalid token"*. Apabila JWT

masih valid dan belum kadaluarsa maka *response* yang dikembalikan berupa "*Status : OK*" yang nantinya dapat diproses pada bagian *flutter*.

Apabila *request* yang diterima tidak memiliki *authorization header* maka fungsi / Segmen Program akan melewati proses verifikasi JWT dan akan langsung melakukan *parsing* variable dari *request* yang diterima (*username* dan *password*). Setelah *parsing* maka fungsi tersebut akan memanggil fungsi utama *login* untuk melakukan logika *login*. Respons yang mungkin dikembalikan berupa gagal, error, dan sukses dimana pada saat gagal atau error tidak akan dibuat JWT baru sedangkan pada saat sukses akan dibuat JWT baru yang dapat disimpan oleh aplikasi pengguna.

3. Segmen Program 4.3

   Segmen Program ini berisikan fungsi pada *flutter* untuk membuat *HTTP Request* dengan tujuan melakukan proses *login* secara umum, dimana parameter *username* dan *password* user dari form yang dibuat pada aplikasi akan dimasukan. Pada Segmen Program 4.3, tidak ada *authorization header* dikarenakan asumsi bahwa aplikasi yang membuat atau memanggil fungsi ini adalah aplikasi yang belum memiliki JWT. API akan mengembalikan respons dengan "*Status*" berbeda, sukses apabila login berhasil, *failed* apabila terjadi kesalahan pada inputan data, dan *error* apabila terjadi masalah atau kesalahan pada sisi API.

4.4.3 Managemen Group dan Kegiatan

1. Segmen Program 4.4

   Pada Segmen Program ini berisi fungsi pada *flutter* yang digunakan untuk mengambil / menarik data list group yang dimiliki oleh pengguna. Parameter yang digunakan dalam *HTTP Request* yang dikirim berupa id dan group id dari pengguna, beserta dengan JWT. Terdapat 2 tipe respons dari API, gagal disertai dengan pesan kenapa terjadi kegagalan terjadi (bisa karena kesalahan logika ataupun masalah pada server), dan sukses dimana data berhasil diambil. Pada saat sukses, maka fungsi akan melakukan *parsing* dari *JSON* yang diterima dari respons tersebut

2. Segmen Program 4.5

Pada Segmen Program ini berisi fungsi pada *Golang* untuk mengambil data list group pengguna. Sebelum masuk ke fungsi utama, maka akan dilakukan *parsing* parameter dari request (pada fungsi dengan nama *Controller*), setelah *parsing* maka API akan memanggil fungsi utama dimana dilakukan logika pengambilan list group pengguna. Apabila terjadi kegagalan atau *error* dalam proses pengambilan maka fungsi akan mengembalikan *error* beserta dengan informasi terkait kenapa terjadi kegagalan tersebut dan akan dikembalikan ke aplikasi untuk diolah lebih lanjut. Apabila berhasil maka hasil akan disimpan dalam *cache* untuk mempercepat pengambilan data pada masa yang akan datang setelah itu akan dikembalikan ke pengguna dalam bentuk *JSON*

3. Segmen Program 4.6

   Pada Segmen Program ini berisikan fungsi *Golang* untuk membuat group baru. Sebelum masuk ke fungsi utama, akan dilakukan *parsing* parameter yang diterima dari request. Setelah itu akan memanggil fungsi utama dimana disini akan dilakukan logika untuk membuat group baru. Apabila terjadi kegagalan maka fungsi akan mengembalikan *error* dan informasi terkait kenapa terjadi error tersebut kepada pengguna untuk diolah lebih lanjut. Apabila berhasil maka akan mengambilkan "*Status: OK*" kepada pengguna untuk menandakan group baru telah berhasil terbentuk.

4. Segmen Program 4.7

   Pada Segmen Program ini berisikan fungsi pada *flutter* untuk membuat *HTTP Request* yang ditujukan untuk membuat group baru. Parameter yang diperlukan adalah kepala / penanggung jawab dari group baru, pembuat dari group tersebut (secara *default* adalah user id pengguna sekarang), status group (secara *default* adalah aktif), tipe group (apakah berupa kegiatan atau group biasa), group id yang dimiliki oleh pembuat group dan JWT. Terdapat 2 tipe respons dimana apabila gagal maka akan mengembalikan pesan *error* dan informasi kenapa *error* terjadi yang diterima dari API. Apabila sukses maka akan mengembalikan nilai boolean true yang menandakan bahwa group baru berhasil dibuat.

5. Segmen Program 4.8

Pada Segmen Program ini berisikan fungsi *flutter* untuk membuat *HTTP Request* yang ditujukan untuk mengambil detail dari group yang bersangkutan. Parameter yang diperlukan berupa id group dan JWT. Ada 2 macam tipe respons, apabila gagal maka akan mengembalikan object dari *GroupDetails* dengan informasi yang kosong (semacam *template*) yang menandakan terjadi *error* dalam proses pengambilan data pada API. Apabila berhasil maka akan mengembalikan object *GroupDetails* dengan detail mengenai group terkait.

6. Segmen Program 4.9

   Pada Segmen Program ini berisikan fungsi *Golang* untuk mengambil detail dari group yang bersangkutan. Sebelum masuk ke dalam fungsi utama, akan dilakukan *parsing* parameter dari request yang diterima. Setelah melakukan *parsing* maka akan memanggil fungsi utama dimana akan dilakukan logika pengambilan detail group. Apabila terjadi *error* maka akan mengembalikan status *error* beserta dengan informasi mengapa terjadi *error* kepada pengguna. Apabila berhasil maka akan mengembalikan status *ok* beserta dengan data detail group kepada pengguna.

4.4.4 Managemen Broadcast dan Papan Informasi

1. Segmen Program 4.10

   Pada Segmen Program ini berisikan fungsi pada flutter untuk membuat HTTP Request yang ditujukan untuk mengambil data Papan Informasi pada API. Parameter yang diperlukan berupa user id dari pengguna sekarang, group id dari pengguna sekarang, dan JWT. Terdapat 2 macam respons dari api, apabila gagal akan mengembalikan error dan pesan error namun pada sisi aplikasi hanya akan mengembalikan array kosongan, pesan tidak ditindaklanjuti. Apabila berhasil maka fungsi akan melakukan parsing dari respons JSON menjadi tipe data Broadcast dan akan mengembalikan array dari Broadcast tersebut

2. Segmen Program 4.11

   Pada Segmen Program ini berisikan fungsi pada Golang untuk mengambil data Papan Informasi. Sebelum masuk ke dalam fungsi utama akan dilakukan parsing parameter dari request yang diterima (fungsi controller). Setelah parsing maka akan memanggil fungsi utama di mana di sini akan dilakukan logika untuk mengambil data papan informasi dari pembuat request. Apabila gagal maka

akan mengembalikan error dan juga pesan error, apabila berhasil maka akan mengembalikan status ok dan data papan informasi pembuat request

3. Segmen Program 4.12

   Pada Segmen Program ini berisikan fungsi pada flutter untuk membuat broadcast baru. Parameter yang diperlukan adalah judul pengumuman, konten / isi dari pengumuman, tujuan pengumuman, pembuat pengumuman (secara default ada user id dari pengguna sekarang), prioritas pengumuman, sifat pengumuman, timestamp dari kapan dibuatnya pengumuman, dan opsi apakah ingin dikirim kedalam group bersangkutan (sesuai dengan tujuan pengumuman), dan JWT. Selain itu, pada fungsi ini juga ada bagian untuk memvalidasi file yang apabila diinginkan oleh pengguna untuk dikirim bersama dengan pembuatan broadcast. HTTP Request yang dibuat fungsi ini akan sedikit berbeda karena menggunakan tipe HTTP Multipart Form untuk mengakomodir pengiriman file. File sendiri akan diubah menjadi bytes stream sebelum ditambahkan kedalam HTTP Request. Ada 2 tipe respons yang akan dikirimkan oleh API, apabila terjadi error maka fungsi akan menangkap error beserta dengan pesan error tersebut untuk ditampilkan kepada aplikasi. Apabila sukses maka akan mengembalikan boolean true yang akan ditampilkan pada bagian aplikasi.

4. Segmen Program 4.13

   Pada Segmen Program ini berisikan fungsi Golang untuk membuat broadcast baru. Sebelum masuk ke dalam fungsi utama, akan dilakukan parsing terhadap parameter yang diterima dari request. File juga akan diparsing pada fungsi ini (controller). Setelah parsing akan memanggil fungsi utama dimana akan dilakukan logika untuk membuat pengumuman baru. Dalam fungsi utama selain logika pembuatan, ada juga fungsi untuk meng-handle file yang diterima dimana file akan di cek apakah ada file serupa yang disimpan pada sisi server atau tidak. Apabila terdeteksi file kembar maka file baru tersebut akan dibuang dan hanya akan menyimpan nama file asli saja, isi dari file akan tetap sama dengan file yang duplikat (sudah ada di server) sebelumnya. Apabila gagal maka akan mengembalikan status error dan pesan error kepada pembuat request. Apabila berhasil maka akan mengambilkan status ok, konfirmasi apakah file berhasil

disimpan (diupload) ke dalam server dan id baru dari pengumuman yang dibuat kepada pembuat request untuk dioleh lebih lanjut.

4.4.5 Managemen Laporan

1. Segmen Program 4.14

   Pada Segmen Program ini berisikan fungsi flutter untuk membuat laporan baru. Parameter yang diperlukan adalah judul laporan, konten / isi laporan, pembuat laporan (secara default user id pengguna sekarang), tujuan laporan, tipe laporan, urgensi laporan, dan timestamp dari laporan. Selain dari itu tipe HTTP Multipart Form juga digunakan yang sama dengan Segmen Program 4.12 untuk mengakomodir pengiriman file. Ada 2 tipe respons API, apabila gagal maka fungsi akan menerima error dan pesan error yang dapat diolah dan ditampilkan ke aplikasi. Apabila berhasil maka fungsi akan menerima status oke dan fungsi sendiri akan mengembalikan boolean oke yang dapat digunakan untuk menampilkan konfirmasi sukses membuat laporan

2. Segmen Program 4.15

   Pada Segmen Program ini berisikan fungsi Golang untuk membuat laporan baru. Sebelum masuk ke dalam fungsi utama akan dilakukan parsing parameter dari request yang diterima. Setelah parsing maka akan memanggil fungsi utama di mana akan dilakukan logika untuk membuat laporan baru dan meng-handle file yang dikirim oleh pembuat request. Apabila gagal maka fungsi akan mengembalikan error dan pesan error kepada pembuat request untuk diolah lebih lanjut, Apabila sukses maka fungsi akan memanggil fungi update cache untuk memberi tahu pengguna tujuan laporan bahwa ada laporan baru untuk beliau, selain dari itu akan mengembalikan status oke, id baru laporan yang dibuat, dan status berhasil atau tidak file di upload kepada pembuat request untuk diolah lebih lanjut.

3. Segmen Program 4.16

   Pada Segmen Program ini berisikan fungsi flutter untuk mengambil data laporan dari API. Terdapat 2 tipe fungsi di sini, yaitu mengambil laporan yang dibuat oleh pengguna dan mengambil laporan yang ditujukan untuk pengguna. Parameter yang diperlukan adalah user id dari pengguna sekarang dan JWT untuk kedua tipe fungsi tersebut. Terdapat 2 tipe respons yang akan dikirimkan oleh API,

apabila gagal maka fungsi akan menerima status error dan pesan error dari API, apabila sukses maka fungsi akan menerima status oke dan mengembalikan list MyReports atau ReportsForMe kepada aplikasi untuk diolah lebih lanjut

4. Segmen Program 4.17

   Pada Segmen Program ini berisikan dengan fungsi Golang untuk mengambil data laporan, laporan yang dibuat oleh pengguna dan laporan yang ditujukan untuk pengguna. Secara garis besar sebelum masuk ke dalam tiap fungsi utama, pertama pasti akan dilakukan parsing parameter pada fungsi controller. Setelah selesai parsing maka tiap fungsi akan memanggil fungsi utama masing-masing untuk melakukan logika aksi yang bersangkutan. Hal tersebut juga berlaku dalam kasus Segmen Program ini. Apabila terjadi error maka tiap fungsi akan mengembalikan status error dan pesan error kepada pembuat request. Apabila sukses maka akan mengambilkan status oke dan list data dalam bentuk JSON kepada pembuat request

4.4.6 Chatting

1. Segmen Program 4.18

   Pada Segmen Program ini berisikan fungsi flutter untuk mengambil list chat room yang dimiliki oleh pengguna. Parameter yang diperlukan adalah user id (secara default adalah user id dari pengguna sekarang) dan JWT. Terdapat 2 tipe respons dari API, apabila gagal maka API akan mengembalikan stauts error dan pesan error namun pada Segmen Program ini tidak ditindaklanjuti dan fungsi akan secara otomatis mengembalikan empty array kepada aplikasi. Apabila berhasil maka API akan mengembalikan reponse status oke dan data list chat rooms yang dimiliki oleh user dalam bentuk JSON. JSON tersebut akan di parse menjadi object ChatRooms yang akan dikembalikan ke aplikasi untuk ditampilkan / diolah lebih lanjut

2. Segmen Program 4.19

   Pada Segmen Program ini berisikan fungsi Golang untuk mengambil chat room yang dimiliki oleh pengguna. Pertama akan dilakukan parsing parameter, pada fungsi controller, lalu akan memanggil fungsi utama. Pada fungsi utama akan dilakukan logika pengambilan data chat room, apabila terjadi error maka fungsi akan mengembalikan status error dan pesan error kepada pembuat request

untuk ditindaklanjuti. Apabila berhasil maka fungsi akan menyimpan hash value dari total list chat room yang berhasil didapat untuk mempercepat pengambilan data pada masa yang akan datang dan mengembalikan respons status oke dan data list chat rooms dalam bentuk JSON kepada pembuat request

3. Segmen Program 4.20

   Pada Segmen Program ini berisikan potongan fungsi flutter untuk meng-connect kan aplikasi dengan WebSocket yang disediakan pada API. Dimana didalamnya hanya memerlukan tujuan / alamat dari API saja. Di dalam fungsi ini terdapat 2 tipe parsing, dimana parsing teks untuk menerima, menyimpan, dan menampilkan informasi teks dari lawan bicara. Tipe lainnya adalah biner untuk menerima, menyimpan, dan menampilkan informasi berupa file dari lawan bicara.

4. Segmen Program 4.21

   Pada Segmen Program ini berisi fungsi Golang untuk meng-handle WebSocket. Selama API masih berjalan maka fungsi ini akan menyimpan data (di RAM) tentang room apa saja yang membuka WebSocket, dimana di tiap room akan ditambahkan koneksi dari masing-masing pengguna. Ketika pengguna mengirimkan pesan maka fungsi ini akan menilai apakah pesan tersebut berupa teks / string saja ataukah berupa biner. Apabila biner (berarti berisi file) maka pesan tersebut akan dibuka (unmarshal) untuk mengambil bit file yang ada di dalamnya dan memasukan file tersebut ke dalam folder yang sesuai dengan tipe file dan menambahkan entry di tabel media. Setelah selesai maka akan memanggil fungsi kirim pesan kepada lawan bicara berisikan dengan nama file dan tipe file (awalnya berupa teks) kepada lawan bicara dimana lawan bicara nanti dapat mengunduh file ketika diperlukan. Apabila pesan berupa teks maka seperti dengan bagian file akan dilakukan ekstraksi pesan (unmarshal) untuk mendapatkan pesan yang bersangkutan. Setelah itu fungsi akan mengirimkan pesan tersebut ke room yang sesuai dengan detail dari pesan yang diterima dan pada saat yang sama disimpan ke dalam tabel chat log atau group chat log. Apabila pengguna keluar dari halaman chatting maka aplikasi akan mengirimkan pesan kepada WebSocket untuk menutup atau mengeluarkan koneksi user yang bersangkutan dari room, hal ini bertujuan untuk meringankan beban API

5. Segmen Program 4.22

   Pada Segmen Program ini berisikan fungsi flutter untuk mengambil data chat logs dari API. Parameter yang diperlukan adalah user id (secara default user id pengguna sekarang), id room, dan JWT. Terdapat 2 tipe respons yang dikirimkan dari API, apabila gagal maka API akan memberikan status error dan pesan error namun dalam hal ini tidak ditindaklanjuti melainkan mengembalikan array kosong kepada aplikasi untuk ditampilkan. Apabila berhasil maka fungsi akan melakukan parsing data chat logs dari JSON yang dikembalikan oleh API menjadi object ChatLogs untuk disimpan di lokal dan ditampilkan pada aplikasi

6. Segmen Program 4.23

   Pada Segmen Program ini berisikan fungsi Golang untuk mengambil chat log pengguna. Fungsi akan melakukan parsing parameter dari request yang diterima, setelah itu akan memanggil fungsi utama dimana akan dilakukan logika pengambilan chat logs. Apabila terjadi error maka fungsi akan mengembalikan status error dan pesan error kepada pembuat request, namun apabila sukses maka fungsi akan mengembalikan status oke dan data chat logs dalam bentuk JSON kepada pembuat request.

4.4.7 Struktur Organisasi

1. Segmen Program 4.24

   Pada Segmen Program ini berisikan fungsi flutter untuk mengambil data struktur organisasi yang ada di API. Parameter yang diperlukan berupa group id dari user bersangkutan dan JWT. Terdapat 2 tipe respons dari API, apabila gagal maka fungsi akan menerima status error dan pesan error namun hanya ditampilkan dalam debug view saja dan tidak ditindaklanjuti dan fungsi hanya mengembalikan array kosong untuk ditampilkan pada aplikasi. Apabila berhasil maka fungsi akan menerima status oke dan data struktur organisasi dalam bentuk JSON yang akan diparsing menjadi GroupLists yang akan dikembalikan ke aplikasi untuk diolah lebih lanjut

2. Segmen Program 4.25

   Pada Segmen Program ini berisikan fungsi Golang untuk mengambil struktur organisasi yang dimiliki oleh pengguna. Pertama akan dilakukan parsing parameter dari request yang diterima (pada fungsi controller), lalu akan

memanggil fungsi utama dimana disini akan dilakukan logika pengambilan struktur organisasi. Apabila terjadi error maka API akan mengembalika status error dan pesan error kepada pembuat request. Apabila sukses maka fungsi akan mengembalikan status oke dengan data struktur organisasi kepada pembuat organisasi dan pada saat yang sama akan melakukan perhitungan hash value untuk disimpan atau menggantikan hash value yang lama dari suatu struktur organisasi yang bertujuan untuk mempercepat pengambilan data pada masa yang akan datang.

4.4.8 Menjalankan API

Pada tahap ini akan menjelaskan Langkah – Langkah yang diperlukan untuk menjalankan API pada server publik milik PT. XYZ

4.5 Login

Setelah user melakukan autentikasi menggunakan PIN / Biometrik maka aplikasi akan melakukan pengecekan apakah user sudah *Login* kedalam aplikasi atau tidak. Apabila tidak maka *user* akan dibawa ke *Login Screen*. Hal ini dapat dilihat pada Segmen Program 4.1

Segmen Program 4.1 Fungsi Flutter untuk Memanggil Fungsi Verifikasi Token

```
Future<bool> VerifyToken() async {
  String urlString = global.wartawargaAPIServer;
  //Prepare Parameters
  var tokenString = global.tokenString;
  //Prepare HTTP Request
  var loginRequest = Uri.parse("$urlString/login");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString",
  };
  var response = await http.post(loginRequest, headers:
headers);

  if (response.statusCode == 200) {
    final respondData = jsonDecode(response.body);
    debugPrint("Debug Verify Token: ${respondData['Status']}");
    if (respondData["Status"] == "OK") {
      return true;
    } else {
      return false;
    }
  } else {
    return false;
  }
}
```

Ketika *API* menerima *HTTP Request* dengan *endpoint* "/login" maka *API* akan mencari fungsi yang dimaksud apabila ditemukan maka akan menjalankan fungsi tersebut namun apabila tidak ditemukan maka akan mengembalikan error 404. Dalam kasus ini *API* akan memanggil *controller* fungsi *login* dimana di dalam *controller* akan dilakukan validasi data sebelum dibawa ke tahap logika fungsi. Pada *controller* fungsi login akan ditempatkan logika untuk memverifikasi JWT apabila ditemukan *Bearer* JWT. Apabila tidak ditemukan maka akan dibawa ke dalam logika fungsi atau *model*. Hal tersebut dapat dilihat pada Segmen Program 4.2

Segmen Program 4.2 Fungsi Go untuk Verifikasi Token Login

```
func Login(c echo.Context) error {

   conf := config.GetConfig()

   authHeader := c.Request().Header.Get("Authorization")
   if authHeader != "" {
      tokenString := strings.TrimPrefix(authHeader, "Bearer ")
      token, err := jwt.Parse(tokenString, func(token *jwt.Token)
(interface{}, error) {
         if _, ok := token.Method.(*jwt.SigningMethodHMAC); !ok {
            return nil, fmt.Errorf("unexpected signing method: %v",
token.Header["alg"])
         }
         return []byte(conf.Secret), nil
      })
      if err == nil && token.Valid {
         // User has a valid token, check if it has expired
         claims := token.Claims.(jwt.MapClaims)
         exp := int64(claims["exp"].(float64))
         if time.Now().Unix() < exp {
            // Token is still valid, skip login flow and return the
existing token
            return c.JSON(http.StatusOK, map[string]string{"Token":
tokenString, "Status": "OK"})
         }
      }
   }

   //TODO If token is invalid then check if username and password are
empty or not, if empty then return it
   //TODO and give message / request to input username and password or
other login methods
   username := c.FormValue("username")
   password := c.FormValue("password")

   result, err := models.Login(username, password)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})   }


   return c.JSON(http.StatusOK, result)
}

func Login(username string, password string) (Response, error) {
   var res Response
   var hit int
```

Segmen Program 4.2 Fungsi Go untuk Verifikasi Token Login (Lanjutan)

```
   conf := config.GetConfig()


   con := db.CreateConnection()
   defer con.Close()

   checkSQLStatement := "SELECT count(username) FROM users WHERE
username = ? AND password = ?"

   stmt, err := con.Prepare(checkSQLStatement)
   if err != nil {
      return res, err
   }
   defer stmt.Close()

   row := stmt.QueryRow(username, password)
   if err := row.Scan(&hit); err != nil {
      return res, err
   }

   if hit == 1 {
      claims := JwtClaims{
         username,
         jwt.RegisteredClaims{
            ExpiresAt: jwt.NewNumericDate(time.Now().Add(time.Hour *
24)),
         },
      }

      token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
      tokenString, err := token.SignedString([]byte(conf.Secret))
      if err != nil {
         return res, err
      }
      getSQLStatement := "SELECT gm.user_id, gm.group_id,
u.user_displayname FROM group_member gm JOIN users u ON gm.user_id =
u.user_id join user_group ug on gm.group_id = ug.group_id WHERE
u.username = ? AND ug.group_type = 'Formal'"

      rows, err := con.Query(getSQLStatement, username)
      if err != nil {
         return res, err
      }
      defer rows.Close()

      var groupID string
      var user_id int
      var displayname string

      var clearanceLevel int

      for rows.Next() {
         if err := rows.Scan(&user_id, &groupID, &displayname); err !=
nil {
            return res, err
```

Segmen Program 4.2 Fungsi Go untuk Verifikasi Token Login (Lanjutan)

```
         }

      }

      //Encrypt the User ID before returning back to the requester
      encrypted, err := EncryptAES_CBC([]byte(strconv.Itoa(user_id)))
      if err != nil {
         return res, err
      }

      clearanceLevel = GetUserClearanceLevel(user_id)

      if clearanceLevel == -1 {
         res.Status = http.StatusInternalServerError
         res.Message = "Failed"
         res.Data = map[string]string{
            "Status": "Failed",
            "Data":   "Mohon menunggu beberapa saaat sebelum melakukan
proses login",
         }

         return res, nil
      }

      res.Status = http.StatusOK
      res.Message = "Login Success !"
      res.Data = map[string]interface{}{
         "Status":      "OK",
         "Token":       tokenString,
         "GroupID":     groupID,
         "UserID":      encrypted,
         "Displayname": displayname,
         "Clearance":   clearanceLevel,
      }

   } else {
      res.Status = http.StatusOK
      res.Message = "Login Failed !"
      res.Data = map[string]string{
         "Status":  "Failed",
         "Message": "Please check the credentials you've inputted",
      }
   }

   return res, nil
}
func GetUserClearanceLevel(requester_id int) int {
   clearanceLevel := -1

   con := db.CreateConnection()

   defer con.Close()

   temp, err := GetItemFromCache(strconv.Itoa(requester_id) + "-
```

Segmen Program 4.2 Fungsi Go untuk Verifikasi Token Login (Lanjutan)

```
ClearanceLevel")
   if err == nil {
      parsedTemp, _ := convertInterfacetoInt(temp)
      return parsedTemp
   }

   userSQLStatement := "SELECT COUNT(role) FROM group_member WHERE
role = 'Participant' AND user_id = ?"
   adminSQLStatement := "SELECT COUNT(role) FROM group_member WHERE
role = 'Admin' AND user_id = ?"
   rootSQLStatement := "SELECT COUNT(role) FROM group_member WHERE
role = 'Root' AND user_id = ?"

   var wg sync.WaitGroup
   wg.Add(3)

   results := make([]Result, 3)
   var mu sync.Mutex

   go executeSQLStatement(userSQLStatement, requester_id, &results[0],
&wg, &mu)
   go executeSQLStatement(adminSQLStatement, requester_id,
&results[1], &wg, &mu)
   go executeSQLStatement(rootSQLStatement, requester_id, &results[2],
&wg, &mu)

   wg.Wait()

   //Error checking
   for _, item := range results {
      if item.Error != nil {
         return -1
      }
   }

   if results[0].Count > 0 {
      clearanceLevel = 0
   }

   if results[1].Count > 0 {
      clearanceLevel = 1
   }

   if results[2].Count > 0 {
      clearanceLevel = 2
   }

   StoreItemToCache(strconv.Itoa(requester_id)+"-ClearanceLevel",
clearanceLevel)

   return clearanceLevel
}
```

Pada *Login Screen* user bisa mengisi *username* dan *password* yang diberikan oleh *admin* sebelumnya untuk *login* ke dalam aplikasi. Dari data yang sudah dimasukan oleh *user*, *Flutter* akan membuat *HTTP Request* untuk melakukan proses login yang akan dikirim kepada *API* di server yang sudah disediakan. Hal ini dapat dilihat pada Segmen Program 4.3

Segmen Program 4.3 Fungsi Flutter untuk memanggil Fungsi Login di API

```
Future<Map<bool, String>> LoginToAPI(String username, String
password) async {
  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var plainText = utf8.encode(password);
  var hashedPw = sha512.convert(plainText).toString();

  //Prepare HTTP Request
  var loginRequest = Uri.parse("$urlString/login");
  var body = {'username': username, 'password': hashedPw};
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName
  };

  var response = await http.post(loginRequest, body: body,
headers: headers);
  if (response.statusCode == 200) {
    final respondData = jsonDecode(response.body);
    if (respondData["Data"]["Status"] == "OK") {
      //Save to secured-local
      await secureStorage.write(
          key: "JWT", value: respondData["Data"]["Token"]);
      await secureStorage.write(key: "Username", value:
username);
      await secureStorage.write(
          key: "GroupID", value:
respondData["Data"]["GroupID"]);
      await secureStorage.write(
          key: "UserID", value: respondData["Data"]["UserID"]);
      await secureStorage.write(
          key: "Displayname", value:
respondData["Data"]["Displayname"]);
      await secureStorage.write(
          key: "ClearanceLevel",
          value: respondData["Data"]["Clearance"].toString());
```

Segmen Program 4.3 Fungsi Flutter untuk memanggil Fungsi Login di API (Lanjutan)

```
      //Set to global
      global.groupID = respondData["Data"]["GroupID"];
      global.currentID = respondData["Data"]["UserID"];
      global.tokenString = respondData["Data"]["Token"];
      global.displayname = respondData["Data"]["Displayname"];
      global.clearanceLevel = respondData["Data"]["Clearance"];

      HiveHelper.hiveLogin();

      return {true: ""};
    } else {
      return {false: respondData["Data"]["Data"]};
    }
  } else {
    return {
      false:
          "Terjadi kesalahan di sisi server, mohon menunggu
sejenak atau kontak admin"
    };
  }
}
```

4.6 Managemen Group dan Kegiatan

Setelah *user* berhasil *login*, memakai JWT atau *login* manual, *user* bisa mengakses fitur terkait group dan kegiatan. Namun *user* masih dibatasi dengan level akses, sehingga tidak semua fitur bisa digunakan oleh *user*. Secara umum fitur yang bisa digunakan oleh *user* dengan level akses 0 adalah mengambil daftar group yang dimana *user* menjadi anggota dari. Hal ini dapat dilihat pada Segmen Program 4.4 dan Segmen Program 4.5

Segmen Program 4.4 Fungsi Flutter untuk Memanggil Fungsi Get All Group Room di API

```
Future<List<GroupRooms>> GetAllGroupRooms() async {
  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var user_id = global.currentID;
  var requester_group = global.groupID;

  var tokenString = global.tokenString;

  //Prepare HTTP Request
  var getRequest = Uri.parse(
      "$urlString/getGroupRooms?requester_id=$user_id&group_id=$
requester_group");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };

  var request = await http.get(getRequest, headers: headers);

  if (request.statusCode == 200) {
    final requestData = jsonDecode(request.body);
    if (requestData["Data"]["Status"] == "OK") {
      final Box<GlobalParams> gpBox = HiveHelper.globalPrams;
      List<dynamic> temp = requestData["Data"]["Data"] ?? [];

      List<GroupRooms> arrObj =
          temp.map((data) =>
GroupRooms.fromJson(data)).toList();
```

Segmen Program 4.4 Fungsi Flutter untuk Memanggil Fungsi Get All Group Room di API (Lanjutan)

```
      gpBox.values.first.groupRoomsHashValue =
          requestData["Data"]["Hash"].toString();

      return arrObj;
    } else {
      debugPrint(
          "Debug(backendHandler.dart) | Failed to get your group
rooms, reason ${requestData.toString()}");
      return [];
    }
  } else {
    debugPrint(
        "Debug(backendHandler.dart) | Failed to get your group
rooms, reason : ${request.body.toString()}");
    return [];
  }
}
```

Segmen Program 4.5 Fungsi Go untuk Mengambil Group Rooms

```
func GetGroupRoomsController(c echo.Context) error {
   requester_id := c.FormValue("requester_id")
   group_id := c.FormValue("group_id")

   result, err := models.GetGroupRooms(requester_id, group_id)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }

   return c.JSON(http.StatusOK, result)
}

func GetGroupRooms(requester_id string, requester_group string)
(Response, error) {
   var res Response
   var obj GroupRooms
   var arrObj []GroupRooms
   var uniqueId []string

   var totalHash int

   con := db.CreateConnection()
   defer con.Close()

   requester_pt, err := DecryptAES_CBC(requester_id)
   if err != nil {
```

Segmen Program 4.5 Fungsi Go untuk Mengambil Group Rooms (Lanjutan)

```
      return res, err

   }

   if requester_pt == "" {
      res.Status = http.StatusOK
      res.Message = "Invalid User ID !"
      res.Data = map[string]interface{}{
         "Status": "Failed",
         "Data":   "Who are you ? Realy...",
      }

      return res, nil
   }
   user_id, _ := strconv.Atoi(requester_pt)

   hcy := strings.Split(requester_group, ",")

   getSQLStatement := "SELECT user_group.group_id, group_name,
group_type, gm.role, user_group.current_hash FROM user_group JOIN
group_member gm on user_group.group_id = gm.group_id WHERE
group_status = 'Aktif' AND gm.user_id = ?"

   result, err := con.Query(getSQLStatement, user_id)
   if err != nil {
      return res, err
   }

   for result.Next() {
      err = result.Scan(
         &obj.GroupID,
         &obj.GroupName,
         &obj.GroupType,
         &obj.UserRole,
         &obj.Hash,
      )

      if err != nil {
         return res, err
      }

      totalHash += obj.Hash

      arrObj = append(arrObj, obj)
      uniqueId = append(uniqueId, obj.GroupID)
      obj = GroupRooms{}
   }

   //Get The Parents
   var current string
   for i := 0; i < len(hcy)-1; i++ {
      current = strings.Join(hcy[:i+1], ",")
      parentSQLStatement := "SELECT user_group.group_id, group_name,
group_type, gm.role, current_hash FROM user_group JOIN group_member gm
on user_group.group_id = gm.group_id WHERE group_status = 'Aktif' AND
user_group.group_id = ?"
```

Segmen Program 4.5 Fungsi Go untuk Mengambil Group Rooms (Lanjutan)

```
      parentResult, err := con.Query(parentSQLStatement, current)
      if err != nil {
         return res, err
      }

      for parentResult.Next() {
         err = parentResult.Scan(
            &obj.GroupID,
            &obj.GroupName,
            &obj.GroupType,
            &obj.UserRole,
      &obj.Hash,

         )

         if err != nil {
            return res, err
         }

         totalHash += obj.Hash
         if detectObjectInArr(uniqueId, obj.GroupID) == false {
            arrObj = append(arrObj, obj)
            uniqueId = append(uniqueId, obj.GroupID)
         }
         obj = GroupRooms{}
      }

      current = ""
   }

   StoreItemToCache(requester_pt+"-GroupRooms_HV", totalHash)

   res.Status = http.StatusOK
   res.Message = "Success getting your group rooms !"
   res.Data = map[string]interface{}{
      "Status": "OK",
      "Hash":   totalHash,
      "Data":   arrObj,
   }

   return res, nil
}
```

User yang memiliki level akses 1 atau ke atas bisa membuat group baru. Hal tersebut dapat dilihat pada Segmen Program 4.6 dan Segmen Program 4.7

Segmen Program 4.6 Fungsi Go untuk Membuat Group Baru

```
func CreateNewHierarchyController(c echo.Context) error {

   //groupName string, groupDesc string, created_by int, parent_id
string, leader_id int
   groupName := c.FormValue("groupName")
   groupDesc := c.FormValue("groupDesc")
   parent_id := c.FormValue("parent_id")
   leader_id := c.FormValue("leader_id")
   created_by := c.FormValue("created_by")
   group_status := c.FormValue("group_status")
   group_type := c.FormValue("group_type")
   current_group := c.FormValue("current_group")

   result, err := models.CreateNewHierarchy(groupName, groupDesc,
created_by, parent_id, leader_id, group_status, group_type,
current_group)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }

   return c.JSON(http.StatusOK, result)
}

func CreateNewHierarchy(groupName string, groupDesc string, created_by
string, parent_id string, leader_id string, group_status string,
group_type string, current_group string) (Response, error) {
   var res Response

   user_pt, err := DecryptAES_CBC(created_by)
   if err != nil {
      return res, err
   }

   if user_pt == "" {
      res.Status = http.StatusOK
      res.Message = "Failed to create new group"
      res.Data = "Invalid User !"

      return res, nil
   }

   user_id, _ := strconv.Atoi(user_pt)

   created_on := time.Now().Format(timeFormat)

   var group_id string
   var nullValue sql.NullString
```

Segmen Program 4.6 Fungsi Go untuk Membuat Group Baru (Lanjutan)

```
   if CheckHierarchyPresence(groupName) == true {
      res.Status = http.StatusOK
      res.Message = "Failed to create new Hierarchy"
      res.Data = "Duplicate Name Detected !"

      return res, nil
   }

   con := db.CreateConnection()
   defer con.Close()

   createSQLStatement := "INSERT INTO user_group(group_name,
group_description, parent_id, leader_id, created_on, created_by,
group_id, group_status, group_type) VALUES (?,?,?,?,?,?,?,?,?)"

   stmt, err := con.Prepare(createSQLStatement)
   if err != nil {
      return res, err
   }

   if group_type == "Non-Formal" {
      parent_id = current_group
   }

   if len(parent_id) == 0 {
      group_id = strconv.Itoa(CheckRootsCount() + 1)
      nullValue.Valid = false
      res.Message = "Success Creating A Root Group"

      _, err = stmt.Exec(groupName, groupDesc, nullValue, leader_id,
created_on, user_id, group_id, group_status, group_type)
      if err != nil {
         return res, err
      }
   } else {
      group_id = parent_id + "," +
strconv.Itoa(CheckChildCount(parent_id)+1)
      res.Message = "Success Creating Child Group"
      _, err = stmt.Exec(groupName, groupDesc, parent_id, user_id,
created_on, user_id, group_id, group_status, group_type)
      if err != nil {
         return res, err
      }
   }
   AddGroupMember(user_id, group_id, user_id, "Admin")
   recalculateGroupRoomsHash(string(user_id))
   res.Status = http.StatusOK
   res.Data = map[string]string{
      "Status": "OK",
   }

   return res, nil
}
```

Segmen Program 4.7 Fungsi Flutter untuk Memanggil Fungsi Buat Group Baru di API

```
Future<bool> CreateGroup(String groupName, String groupDesc,
String parentID,
    String groupType) async {
  String urlString = global.wartawargaAPIServer;
  //Prepare Parameters
  var leaderID = global.currentID;
  var createdBy = global.currentID;
  var groupStatus = "Aktif";
  var group_type = groupType;
  var current_group = global.groupID;

  var tokenString = global.tokenString;

  //Prepare HTTP Request (POST)
  var postRequest = Uri.parse("$urlString/createNewHierarchy");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };
  var body = {
    'groupName': groupName,
    'groupDesc': groupDesc,
    'parent_id': parentID,
    'leader_id': leaderID,
    'created_by': createdBy,
    'group_status': groupStatus,
    'group_type': group_type,
    'current_group': current_group,
  };

  var request = await http.post(postRequest, headers: headers,
body: body);
  if (request.statusCode == 200) {
    return true;
  } else {
    debugPrint(
        "Debug(backendHandler.dart 698) | Failed to create new
group, reason : ${request.body.toString()}");
    return false;
  }
}
```

User bisa mengambil informasi terkait dengan group, tanpa Batasan level akses, yang dapat dilihat pada Segmen Program 4.8 dan Segmen Program 4.9

Segmen Program 4.8 Fungsi Flutter untuk Memanggil Fungsi Ambil Detail Group di API

```
Future<GroupDetails> GetGroupDetails(String groupID) async {
  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var tokenString = global.tokenString;

  //Prepare HTTP Request
  var getRequest =
Uri.parse("$urlString/getGroupDetails?group_id=$groupID");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };

  var request = await http.get(getRequest, headers: headers);
  if (request.statusCode == 200) {
    final requestData = jsonDecode(request.body);
    if (requestData["Data"]["Status"] == "OK") {
      GroupDetails obj =
GroupDetails.fromJson(requestData["Data"]["Data"]);
      List<dynamic> tempObj =
requestData["Data"]["Data"]["group_members"];
      obj.gMembers =
          tempObj.map((data) =>
GroupMembers.fromJson(data)).toList();
      return obj;
    } else {
      return GroupDetails(...
      );
    }
  } else {
    return GroupDetails(...
    );
  }}
```

Segmen Program 4.9 Fungsi Go untuk mengambil Detail Group

```
func GetGroupDetailsController(c echo.Context) error {
	group_id := c.FormValue("group_id")

	result, err := models.GetGroupDetails(group_id)

	if err != nil {
		return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
	}

	return c.JSON(http.StatusOK, result)
}
func GetGroupDetails(group_id string) (Response, error) {
	var res Response
	var obj GroupsDetails

	con := db.CreateConnection()
	defer con.Close()

	getSQLStatement := "SELECT group_id ,group_name,
group_description, group_status, user_group.created_by,
c.username, user_group.created_on, leader_id, l.username,
COALESCE(parent_id, '') FROM user_group JOIN users l ON
user_group.leader_id = l.user_id JOIN users c ON
user_group.created_by = c.user_id WHERE group_id = ?"

	result, err := con.Query(getSQLStatement, group_id)
	if err != nil {
		return res, err
	}
```

Segmen Program 4.9 Fungsi Go untuk mengambil Detail Group (Lanjutan)

```
	for result.Next() {
		err = result.Scan(
			&obj.GroupID,
			&obj.GroupName,
			&obj.GroupDesc,
			&obj.GroupStatus,
			&obj.CreatorID,
			&obj.CreatorName,
			&obj.CreatedTimestamp,
			&obj.LeaderID,
			&obj.LeaderName,
			&obj.ParentID,
		)
		if err != nil {
			return res, err
		}
		obj.CreatorID, _ =
EncryptAES_CBC([]byte(obj.CreatorID))
		obj.LeaderID, _ =
EncryptAES_CBC([]byte(obj.LeaderID))
	}
	obj = GetGroupParticipants(obj)
	obj.Files = getFilePaths(con, obj.GroupID)
	res.Status = http.StatusOK
	res.Message = "Success getting the group details !"
	res.Data = map[string]interface{}{
		"Status": "OK",
		"Data":   obj,
	}
	return res, nil
}
```

4.7 Managemen Broadcast dan Papan Informasi

Ketika *user* membuka halaman Papan Informasi, maka aplikasi pertama akan mencoba untuk mengambil *hash value* dari papan informasi dan group room yang dimana *user* menjadi anggota dari. Apabila *hash value* masih sama maka aplikasi akan memuat data dari *Hive* namun apabila berbeda maka aplikasi akan mengambil data dari *API* dan memasukan data yang baru ke dalam *Hive* untuk disimpan dan digunakan di waktu depan. Hal tersebut dapat dilihat pada Segmen Program 4.10 dan Segmen Program 4.11

Segmen Program 4.10 Fungsi Flutter untuk memanggil fungsi ambil data Papan Informasi di API

```
Future<List<Broadcast>> GetBroadcasts() async {
  String urlString = global.wartawargaAPIServer;
  var requesterUserID = global.currentID;
  var requesterGroupID = global.groupID;
  var tokenString = global.tokenString;
  //Prepare HTTP Request
  var getRequest = Uri.parse(
      "$urlString/getBroadcasts?requester_name=$requesterUserID&
&group_id=$requesterGroupID");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString",
  };
  var response = await http.get(getRequest, headers: headers);
  if (response.statusCode == 200) {
    final responseData = jsonDecode(response.body);
    if (responseData["Data"] != null &&
        responseData["Data"]["Status"] == "OK") {
      List<dynamic> arrObj = responseData["Data"]["Data"];
      List<Broadcast> finished =
          arrObj.map((data) =>
Broadcast.fromJson(data)).toList();
      return finished;
    } else {
      return [];
    }
  } else {
  return [];
  }
}
```

Segmen Program 4.11 Fungsi Go untuk Mengambil Data Broadcast

```
func GetBroadcastsController(c echo.Context) error {
   requester_name := c.FormValue("requester_name")
   requester_group_id := c.FormValue("group_id")
   result, err := models.GetBroadcasts(requester_name,
requester_group_id)
   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }
   return c.JSON(http.StatusOK, result)
}

func GetBroadcasts(requester_name string, requester_group_id string)
(Response, error) {
   var res Response
   var obj Broadcasts
   var arrObj []Broadcasts
   var tempTargets string
   con := db.CreateConnection()
   defer con.Close()
   getSQLStatement := "SELECT infoboard_id , infoboard_title,
infoboard_content, infoboard_target, information_board.created_by,
u.username, gm.group_id, gm.role, information_board.created_on,
COALESCE(information_board.updated_by, 0), COALESCE(e.username, ''),
COALESCE(information_board.updated_on, ''), infoboard_priority,
infoboard_type FROM information_board LEFT JOIN users u on
information_board.created_by = u.user_id LEFT JOIN users e on
information_board.updated_by = e.user_id LEFT JOIN group_member gm on
u.user_id = gm.user_id WHERE infoboard_target LIKE ? AND info_status =
0 ORDER BY infoboard_priority ASC"
   result, err := con.Query(getSQLStatement,
"%|"+requester_group_id+"|%")
   if err != nil {
      return res, err
   }
   var tempID, tempUpID int
   index := 0
   for result.Next() {
      err = result.Scan(
         &obj.BroadcastID,
         &obj.BroadcastTitle,
         &obj.BroadcastContent,
         &tempTargets,
         &tempID,
         &obj.CreatorName,
         &obj.GroupID,
         &obj.GroupRole,
         &obj.CreatedOn,
         &tempUpID,
         &obj.UpdatedName,
         &obj.UpdatedOn,
         &obj.BroadcastPriority,
         &obj.BroadcastType,)
```

Segmen Program 4.11 Fungsi Go untuk Mengambil Data Broadcast (Lanjutan)

```
        if err != nil {
            return res, err
        }
        //Beautify Targets
        obj.BroadcastTarget = stringToJSON(tempTargets, con)

        obj.CreatedBy, err = EncryptAES_CBC([]byte(string(tempID)))
        obj.UpdatedBy, err = EncryptAES_CBC([]byte(string(tempUpID)))

        //Get Files Associated with the Broadcast
        //This does not return the files, but just the filepaths
        obj.Files = getFilePaths(con, obj.BroadcastID)

        //Appends
        arrObj = append(arrObj, obj)

        //Clear Objs
        obj = Broadcasts{}
        tempTargets = ""

        index++
    }

    res.Status = http.StatusOK
    res.Message = "Success getting your broadcasts"
    res.Data = map[string]interface{}{
        "Status": "OK",
        "Data":   arrObj,
    }

    return res, nil
}
```

User dengan level akses 1 ke atas bisa membuat *broadcast* dimana pembuat diberikan opsi untuk mengirim ke papan informasi saja atau juga dikirim ke *group room* bersangkutan. Selain dari itu *user* juga bisa memilih tipe dan urgensi dari *broadcast* yang dibuat. Untuk tujuan *broadcast* bergantung pada jabatan yang dipegang oleh *user*, asumsi user memiliki level akses > 0, *user* hanya diberikan kebebasan untuk membuat *broadcast* kepada group yang dibawahi oleh *user* tersebut. Hal tersebut dapat dilihat pada Segmen Program 4.12 dan Segmen Program 4.13

Segmen Program 4.12 Fungsi Flutter untuk Memanggil Fungsi Buat Broadcast di API

```
Future<Map<String, dynamic>> CreateBroadcast(
    SingularBroadcast broadcast) async {
  Map<String, dynamic> funcResult = {};
  funcResult.addAll({"Result": false});

  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var info_title = broadcast.broadcastTitle;
  var info_content = broadcast.broadcastContent;
  var info_target = "|${broadcast.broadcastTargets.join('|')}|";
  var created_by = global.currentID;
  var info_priority = broadcast.broadcastUrgency;
  var info_sifat = broadcast.broadcastType;
  var info_timestamp = broadcast.broadcastTimestamp;
  var group_option = broadcast.sendToGroup;

  var tokenString = global.tokenString;

  //Prepare HTTP Request
  var createRequest = Uri.parse("$urlString/createBroadcast");
  var headers = {
    "Content-Type": "multipart/form-data",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };

  var request = http.MultipartRequest('POST', createRequest)
    ..headers.addAll(headers)
    ..fields['info_title'] = info_title
    ..fields['info_content'] = info_content
```

Segmen Program 4.13 Fungsi Flutter untuk Memanggil Fungsi Buat Broadcast di API (Lanjutan)

```
    ..fields['info_target'] = info_target
    ..fields['created_by'] = created_by
    ..fields['info_priority'] = info_priority
    ..fields['info_sifat'] = info_sifat
    ..fields['info_timestamp'] = info_timestamp
    ..fields['group_option'] = group_option.toString();

  List<String> fileErr = [];
  for (String filePath in broadcast.filePaths) {
    File file = File(filePath);

    if (file.existsSync()) {
      final file = File(filePath);
      final filename = filePath.split("/").last;
      final fileStream = http.ByteStream(file.openRead());

      final fileLength = await file.length();

      final multipartFile = http.MultipartFile('files',
fileStream, fileLength,
          filename: filename);

      request.files.add(multipartFile);
    } else {
      fileErr.add("Error accessing $filePath");
    }
  }

  funcResult.addAll({"File Errors": fileErr});

  var response = await request.send();
  try {
    final responseString = await
response.stream.transform(utf8.decoder).join();
    final rData = jsonDecode(responseString);
    if (response.statusCode == 200 && rData["Data"]["Status"] ==
"OK") {
      debugPrint("Request was successful");
      funcResult["Result"] = true;
    } else {
      debugPrint("Request encountered an error");
      funcResult["Result"] = false;
```

Segmen Program 4.14 Fungsi Flutter untuk Memanggil Fungsi Buat Broadcast di API (Lanjutan)

```
      funcResult.addAll({"Internal Server Error":
rData.toString()});
    }
  } catch (error) {
    debugPrint("Error Sending HTTP Request: $error");
    funcResult.addAll({"HTTP Error": error.toString()});
  }

  debugPrint("Result: ${funcResult['Result']}");
  return funcResult;
}
```

Segmen Program 4.15 Fungsi Go untuk Membuat Broadcast Baru

```
func CreateBroadcastController(c echo.Context) error {

   info_title := c.FormValue("info_title")
   info_content := c.FormValue("info_content")
   created_by := c.FormValue("created_by")
   info_target := c.FormValue("info_target")
   info_priority, _ :=
strconv.Atoi(c.FormValue("info_priority"))
   info_type := c.FormValue("info_sifat")
   group_option := c.FormValue("group_option")

   //Parse the file
   err := c.Request().ParseMultipartForm(32 << 20)
   if err != nil {
      return c.JSON(http.StatusBadRequest,
map[string]string{"Message": "Invalid form data"})
   }

   files := c.Request().MultipartForm.File["files"]

   result, err := models.CreateBroadcast(info_title,
info_content, info_target, created_by, info_priority, files,
info_type, group_option)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }

   return c.JSON(http.StatusOK, result)
}
```

Segmen Program 4.16 Fungsi Go untuk Membuat Broadcast Baru (Lanjutan)

```
func CreateBroadcast(info_title string, info_content string,
info_target string, created_by string, info_priority int, files
[]*multipart.FileHeader, info_type string, group_option string)
(Response, error) {
		var res Response
		var upload = false

		con := db.CreateConnection()
		defer con.Close()

		created_on := time.Now().Format(timeFormat)
		creator_pt, err := DecryptAES_CBC(created_by)
		if err != nil {
			return res, err
		}
		creator_id, _ := strconv.Atoi(creator_pt)

		insertSQLStatement := "INSERT INTO information_board
(infoboard_title, infoboard_content, infoboard_target,
infoboard_priority, created_by, created_on, infoboard_type)" +
			"VALUES (?,?,?,?,?,?,?)"

		stmt, err := con.Prepare(insertSQLStatement)
		if err != nil {
			return res, err
		}

		var newID string
		_, err = stmt.Exec(info_title, info_content,
info_target, info_priority, creator_id, created_on, info_type)
		if err != nil {
			return res, err
		}

		//Get the new id
		newIDSQLStatement := "SELECT infoboard_id FROM
information_board WHERE table_index = LAST_INSERT_ID();"
		err = con.QueryRow(newIDSQLStatement).Scan(&newID)
		if err != nil {
			return res, err
		}

		//Handle File
		handleFiles(newID, creator_id, upload, files, con)

		indivTargets := strings.Split(info_target, "|")

		for _, part := range indivTargets {
			if part != "" {
				updateBroadcastHashValue(part,
created_on)
			}
		}

		if group_option == "true" {
			var data NewGroupChat
```

Segmen Program 4.17 Fungsi Go untuk Membuat Broadcast Baru (Lanjutan)

```
                        // Remove the leading and trailing pipe
characters
                        info_target = strings.Trim(info_target, "|")

                        // Split the input string by the pipe character
                        substrings := strings.Split(info_target, "|")

                        // Sends to groups
                        for _, substring := range substrings {
                                data.GroupID = substring
                                data.Message = "--BROADCAST--\n" +
info_content
                                data.SenderID = created_by
                                data.MessageTimestamp =
time.Now().Format(timeFormat)
                                SocketSendGroupChat(data)
                                //Handle File
                                handleFiles(substring, creator_id,
upload, files, con)
                        }

                }

                res.Status = http.StatusOK
                res.Message = "Success creating Broadcast !"
                res.Data = map[string]interface{}{
                        "New ID":        newID,
                        "Uploaded File": upload,
                        "Status":        "OK",
                }

                return res, nil
}
```

4.8 Managemen Laporan

Di dalam aplikasi, *user* dapat membuat laporan yang di dalamnya terdapat beberapa variable yang harus diisi seperti judul laporan, isi laporan, lampiran, dan target / tujuan dari laporan tersebut. Secara umum tujuan laporan hanya kepada kepala group, seperti sebuah group, yang dimana *user* menjadi anggota dari. Hal tersebut dapat dilihat pada Segmen Program 4.14 dan Segmen Program 4.15

Segmen Program 4.18 Fungsi Flutter untuk Memanggil Fungsi Buat Laporan Baru di API

```
Future<Map<String, dynamic>> CreateReport(SingularReport report)
async {
  Map<String, dynamic> funcResult = {};
  funcResult.addAll({"Result": false});

  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var tokenString = global.tokenString;

  //Prepare HTTP Request
  var postRequest = Uri.parse("$urlString/createNewReport");
  var request = http.MultipartRequest('POST', postRequest);

  request.headers.addAll({
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  });

  debugPrint("Target ID: ${report.targetId}");

  request.fields['report_title'] = report.reportTitle;
  request.fields['report_content'] = report.reportContent;
  request.fields['created_id'] = report.userId;
  request.fields['report_target'] = report.targetId;
  request.fields['report_type'] = report.reportType;
  request.fields['report_urgency'] = report.reportUrgency;
  request.fields['report_timestamp'] = report.reportTimestamp;

  List<String> fileErr = [];
  for (String filePath in report.filePaths) {
    File file = File(filePath);
```

Segmen Program 4.19 Fungsi Flutter untuk Memanggil Fungsi Buat Laporan Baru di API (Lanjutan)

```
    if (file.existsSync()) {
      final file = File(filePath);
      final filename = filePath.split("/").last;
      final fileStream = http.ByteStream(file.openRead());

      final fileLength = await file.length();

      final multipartFile = http.MultipartFile('files',
fileStream, fileLength,
          filename: filename);

      request.files.add(multipartFile);
    } else {
      fileErr.add("Error accessing $filePath");
    }
  }

  funcResult.addAll({"File Errors": fileErr});

  var response = await request.send();

  try {
    final responseString = await
response.stream.transform(utf8.decoder).join();
    final rData = jsonDecode(responseString);

    if (response.statusCode == 200 && rData["Data"]["Status"] ==
"OK") {
      debugPrint("Request was successful");
      funcResult["Result"] = true;
    } else {
      debugPrint("Request encountered an error\n Reason:
${rData.toString()}");
      funcResult["Result"] = false;
      funcResult.addAll({"Internal Server Error":
rData.toString()});
    }
  } catch (error) {
    debugPrint("Error Sending HTTP Request: $error");
    funcResult.addAll({"HTTP Error": error.toString()});
  }
  return funcResult;
}
```

Segmen Program 4.20 Fungsi Go untuk Membuat Laporan Baru

```
func CreateReportController(c echo.Context) error {
   report_title := c.FormValue("report_title")
   report_content := c.FormValue("report_content")
   creator_id := c.FormValue("created_id")
   report_target := c.FormValue("report_target")

   //Parse the file
   err := c.Request().ParseMultipartForm(32 << 20)
   if err != nil {
      return c.JSON(http.StatusBadRequest,
map[string]string{"Message": "Invalid form data"})
   }

   files := c.Request().MultipartForm.File["files"]

   result, err := models.CreateReport(report_title,
report_content, creator_id, report_target, files)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,
         map[string]string{"Message": err.Error()},
      )
   }

   return c.JSON(http.StatusOK, result)
}

func CreateReport(report_title string, report_content string,
creator_id string, report_target string, files
[]*multipart.FileHeader) (Response, error) {
   var res Response
   var upload bool

   created_on := time.Now().Format(timeFormat)
   created_id, err := DecryptAES_CBC(creator_id)
   if err != nil {
      return res, err
   }

   if created_id == "" {
      res.Status = http.StatusOK
      res.Message = "Invalid ID !"
      res.Data = map[string]interface{}{
         "Status": "Failed",
         "Data":   "Who are you ? Really..",
      }

      return res, nil
   }
```

Segmen Program 4.15 Fungsi Go untuk Membuat Laporan Baru (Lanjutan)

```
   created_by, _ := strconv.Atoi(created_id)

   report_target_pt, err := DecryptAES_CBC(report_target)
   if err != nil {
      return res, err
   }

   if report_target_pt == "" {
      res.Status = http.StatusOK
      res.Message = "Invalid ID !"
      res.Data = map[string]interface{}{
         "Status": "Failed",
         "Data":   "To Who are you tying to report ? Really..",
      }

      return res, nil
   }
   target_id, _ := strconv.Atoi(report_target_pt)

   con := db.CreateConnection()
   defer con.Close()

   createSQLStatement := "INSERT INTO reports (report_title,
report_content, created_by, created_on, report_status,
report_target) VALUES(?,?,?,?,?,?);"

   stmt, err := con.Prepare(createSQLStatement)
   if err != nil {
      return res, err
   }

   _, err = stmt.Exec(report_title, report_content, created_by,
created_on, "Waiting for Review", target_id)
   if err != nil {
      return res, err
   }

   var newID string
   getNewIDSQLStatement := "SELECT report_id FROM reports WHERE
id = LAST_INSERT_ID();"

   err = con.QueryRow(getNewIDSQLStatement).Scan(&newID)
   if err != nil {
      return res, err
   }

   //Handle File
   handleFiles(newID, created_by, upload, files, con)
```

Segmen Program 4.15 Fungsi Go untuk Membuat Laporan Baru (Lanjutan)

```
    res.Status = http.StatusOK

    res.Message = "Success creating new Report !"
    res.Data = map[string]interface{}{
       "Status":          "OK",
       "New Report ID: ": newID,
       "Upload File(s)":  upload,
    }

    calculateMyReportsHashValue(created_by)
    calculateReportsForMeHashValue(target_id)

    return res, nil
}
```

Ketika *user* membuka halaman Laporan, maka aplikasi akan pertama membandingkan *hash value* dari apa yang disimpan di lokal dan di server, apabila berbeda maka aplikasi akan meminta data ke API namun apabila sama maka aplikasi akan menampilkan data yang sudah ada di lokal. Hal tersebut dapat dilihat pada Segmen Program 4.16 dan Segmen Program 4.17

Segmen Program 4.21 Fungsi Flutter untuk Memanggil Fungsi Ambil Data dari Laporan Pengguna dan Laporan Untuk Pengguna di API

```
Future<Map<String, dynamic>> GetMyReports() async {
  Map<String, dynamic> funcRes = {};
  funcRes.addAll({"Result": false});

  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var requester_name = global.currentID;
  var tokenString = global.tokenString;

  //Prepare HTTP Request
  var getRequest =
Uri.parse("$urlString/getMyReports?requester_name=$requester_nam
e");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };
  var response = await http.get(getRequest, headers: headers);
```

Segmen Program 4.16 Fungsi Flutter untuk Memanggil Fungsi Ambil Data dari Laporan Pengguna dan Laporan Untuk Pengguna di API (Lanjutan)

```
  if (response.statusCode == 200) {
    final responseData = jsonDecode(response.body);
    if (responseData["Data"]["Status"] == "OK") {
      List<dynamic> arrObj = responseData["Data"]["Data"] ?? [];

      List<MyReports> myReports =
          arrObj.map((data) =>
MyReports.fromJson(data)).toList();
      funcRes["Result"] = true;
      funcRes.addAll({"MyReports": myReports});
      return funcRes;
    } else {
      funcRes["Result"] = false;
      funcRes.addAll({
        "Error": "Terjadi kesalahan di server, mohon mencoba
beberapa saat lagi"
      });
      return funcRes;
    }
  } else {
    funcRes["Result"] = false;
    funcRes.addAll({
      "Error":
          "Terjadi kesalahan di server, mohon mencoba beberapa
saat lagi\nAlasan: ${response.body.toString()}"
    });
    return funcRes;
  }
}
Future<Map<String, dynamic>> GetReportsForMe() async {
  Map<String, dynamic> funcRes = {};
  funcRes.addAll({"Result": false});

  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var requester_name = global.currentID;

  var tokenString = global.tokenString;

  //Prepare HTTP Request
  var getRequest =
```

Segmen Program 4.16 Fungsi Flutter untuk Memanggil Fungsi Ambil Data dari Laporan Pengguna dan Laporan Untuk Pengguna di API (Lanjutan)

```
Uri.parse("$urlString/getReportsForMe?report_target=$requester_n
ame");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };

  var response = await http.get(getRequest, headers: headers);

  if (response.statusCode == 200) {
    final responseData = jsonDecode(response.body);
    if (responseData["Data"]["Status"] == "OK") {
      List<dynamic> arrObj = responseData["Data"]["Data"] ?? [];

      List<ReportsForMe> reportsForMe =
          arrObj.map((data) =>
ReportsForMe.fromJson(data)).toList();

      funcRes["Result"] = true;
      funcRes.addAll({"ReportsForMe": reportsForMe});
      return funcRes;
    } else {
      funcRes["Result"] = false;
      funcRes.addAll({
        "Error": "Terjadi kesalahan di server, mohon mencoba
beberapa saat lagi"
      });
      return funcRes;
    }
  } else {
    funcRes["Result"] = false;
    funcRes.addAll({
      "Error":
          "Terjadi kesalahan di server, mohon mencoba beberapa
saat lagi\nAlasan: ${response.body.toString()}"
    });

    return funcRes;
  }
}
```

Segmen Program 4.22 Fungsi Go untuk Mengambil Data Laporan Pengguna dan Laporan Untuk Pengguna

```
func GetMyReportsController(c echo.Context) error {

   //user_id int
   requester_name := c.FormValue("requester_name")

   result, err := models.GetMyReports(requester_name)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,

map[string]string{"Message": err.Error()})

   }

   return c.JSON(http.StatusOK, result)
}
func GetMyReports(requester_id string) (Response, error) {
   var res Response
   var obj Reports
   var arraObj []Reports

   con := db.CreateConnection()
   defer con.Close()

   var user_pt string
   user_pt, err := DecryptAES_CBC(requester_id)
   if err != nil {
      return res, err
   }

   user_id, _ := strconv.Atoi(user_pt)

   if user_id == -1 {
      res.Status = http.StatusOK
      res.Message = "Failed to get your reports !"
      res.Data = map[string]interface{}{
         "Status": "Failed",
         "Data":   "No user found",
      }

      return res, nil
   }

   getSQLStatement := "SELECT report_id, report_title,
report_content, report_status, report_target,
u.user_displayname, gm.role, reports.created_on,
COALESCE(report_comment, '') FROM reports LEFT JOIN users u on
u.user_id = reports.report_target LEFT JOIN group_member gm on
u.user_id = gm.user_id WHERE reports.created_by = ? "
```

Segmen Program 4.17 Fungsi Go untuk Mengambil Data Laporan Pengguna dan Laporan Untuk Pengguna (Lanjutan)

```
   result, err := con.Query(getSQLStatement, user_id)
   if err != nil {
      return res, err
   }
   defer result.Close()

   for result.Next() {
      err = result.Scan(
         &obj.Report_ID,
         &obj.Report_title,

         &obj.Report_content,
         &obj.Report_status,
         &obj.Report_target_id,
         &obj.Report_target_name,
         &obj.Report_target_role,
         &obj.Created_on,
         &obj.Report_Comment,
      )
      if err != nil {
         return res, err
      }

      //Encrypt Target ID Before returning for security reasons
      obj.Report_target_id, _ =
EncryptAES_CBC([]byte(obj.Report_target_id))

      obj.Created_by = requester_id

      //Get Files
      obj.Files = getFilePaths(con, obj.Report_ID)

      arraObj = append(arraObj, obj)
      obj = Reports{}
   }

   res.Status = http.StatusOK
   res.Message = "Success getting your Reports !"
   res.Data = map[string]interface{}{
      "Status": "OK",
      "Data":   arraObj,
   }

   return res, nil
}
```

Segmen Program 4.17 Fungsi Go untuk Mengambil Data Laporan Pengguna dan Laporan Untuk Pengguna (Lanjutan)

```
func GetReportsForMeController(c echo.Context) error {
   report_target := c.FormValue("report_target")
   result, err := models.GetReportsForMe(report_target)
   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }
   return c.JSON(http.StatusOK, result)
}
func GetReportsForMe(report_target string) (Response, error) {
   var res Response
   var obj ReportsForMe

   var arrObj []ReportsForMe


   target_id_pt, err := DecryptAES_CBC(report_target)
   if err != nil {
      return res, err
   }

   target_id, _ := strconv.Atoi(target_id_pt)
   if target_id == -1 {
      res.Status = http.StatusOK
      res.Message = "Failed to get reports for you !"
      res.Data = map[string]string{
         "Status": "Failed",
         "Data":   "User does not exist !",
      }
      return res, nil
   }
   con := db.CreateConnection()
   defer con.Close()
   getSQLStatement := "SELECT report_id, report_title,
report_content, report_status, COALESCE(report_comment, ''),
reports.created_by, u.username, gm.role, reports.created_on,
COALESCE(reports.update_reference_id, '') FROM reports LEFT JOIN
users u on u.user_id = reports.created_by LEFT JOIN group_member
gm on reports.created_by = gm.user_id WHERE
reports.report_target = ? AND report_status = 'Waiting for
Review'"
   result, err := con.Query(getSQLStatement, target_id)
   if err != nil {
      return res, err
   }
   defer result.Close()
```

Segmen Program 4.17 Fungsi Go untuk Mengambil Data Laporan Pengguna dan Laporan Untuk Pengguna (Lanjutan)

```
   for result.Next() {
      err = result.Scan(
         &obj.Report_ID,
         &obj.Report_Title,
         &obj.Report_Content,
         &obj.Report_Status,
         &obj.Report_Comment,
         &obj.Report_Creator_Id,
         &obj.Report_Creator_Name,
         &obj.Report_Creator_Role,
         &obj.Report_Timestamp,
         &obj.UpdateReferenceID,
      )
      if err != nil {
         return res, err
      }

      obj.Report_Creator_Id, err =
EncryptAES_CBC([]byte(obj.Report_Creator_Id))
      if err != nil {
         return res, err
      }

      //Get Files
      obj.Files = getFilePaths(con, obj.Report_ID)

      arrObj = append(arrObj, obj)
      obj = ReportsForMe{}
   }

   res.Status = http.StatusOK
   res.Message = "Success getting Reports For You !"
   res.Data = map[string]interface{}{
      "Status": "OK",
      "Data":   arrObj,
   }

   return res, nil
}
```

4.9 Chatting

Ketika *user* membuka *chatting page*, aplikasi akan mengambil *hash value* dari *API* mengenai *chat rooms* yang dimiliki oleh *user*, apabila berbeda maka aplikasi akan mengambil data dari *API,* apabila sama maka aplikasi akan memuat data dari *local*. Hal tersebut dapat dilihat pada Segmen Program 4.18 dan Segmen Program 4.19.

Segmen Program 4.23 Fungsi Flutter untuk Memanggil Fungsi Ambil Data Chat Room di API

```
Future<List<ChatRooms>> GetChatRooms() async {
  String urlString = global.wartawargaAPIServer;
  //Prepare Parameters
  var user_id = global.currentID;
  var tokenString = global.tokenString;
  //Prepare HTTP Request
  var getRequest =
Uri.parse("$urlString/getChatRooms?username=$user_id");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString",
  };

  var response = await http.get(getRequest, headers: headers);

  final responseData = jsonDecode(response.body);
  if (response.statusCode == 200) {
    if (responseData["Data"]["Status"] == "OK") {
      final Box<GlobalParams> gpBox = HiveHelper.globalPrams;
      List<dynamic> temp = responseData["Data"]["Data"];

      List<ChatRooms> arrObj =
          temp.map((data) => ChatRooms.fromJson(data)).toList();
      gpBox.values.first.privateChatsHashValue =
          responseData["Data"]["Hash"].toString();
      return arrObj;
    } else {
      return [];
    }
  } else {
    debugPrint("Failed to get chat rooms, reason:
${responseData.toString()}");
    return [];
  }
}
```

Segmen Program 4.24 Fungsi Go untuk Mengambil Data Chat Room

```
func GetChatRoomsController(c echo.Context) error {
   username := c.FormValue("username")

   result, err := models.GetChatRooms(username)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }

   return c.JSON(http.StatusOK, result)
}
func GetChatRooms(username string) (Response, error) {
   var res Response
   var obj ChatRooms
   var arrObj []ChatRooms

   var user_id int
   con := db.CreateConnection()
   defer con.Close()

   user_pt, err := DecryptAES_CBC(username)
   if err != nil {
      return res, err
   }

   user_id, _ = strconv.Atoi(user_pt)

   if user_id == 0 {
      res.Status = http.StatusOK
      res.Message = "Success getting your chat rooms"
      res.Data = map[string]string{
         "Status": "OK",
         "Data":   "",
      }
   }

   getSQLStatement := "" +
      "SELECT chat_room_id, room_name, participant_1,
u1.user_displayname, participant_2, u2.user_displayname,
current_hash, COALESCE(pub_key_1, ''), COALESCE(pub_key_2, ''),
u1.username, u2.username " +
      "FROM chat_rooms " +
      "JOIN users u1 ON chat_rooms.participant_1 = u1.user_id "
+
      "JOIN users u2 ON chat_rooms.participant_2 = u2.user_id "
+
      "WHERE participant_1 = ? OR participant_2 = ?"
```

Segmen Program 4.19 Fungsi Go untuk Mengambil Data Chat Room (Lanjutan)

```
result, err := con.Query(getSQLStatement, user_id, user_id)
   if err != nil {
      return res, err
   }
   defer result.Close()
   index := 0
   var d1, d2, pub1, pub2, u1, u2 string
   var p1, p2 int
   for result.Next() {
      err = result.Scan(
         &obj.Room_id,

         &obj.Roomname,
         &p1,
         &d1,
         &p2,
         &d2,
         &obj.CurrentHash,
         &pub1,
         &pub2,
         &u1,
         &u2,
      )
      if err != nil {
         fmt.Println("Error in Get Chat Rooms: ", err)
      }
      if user_id == p1 {
         obj.TargetDisplayname = d2
         obj.MyID = p1
         obj.TargetID = p2
         obj.MyPubKey = pub1
         obj.TargetPubKey = pub2
         obj.MyUsername = u1
         obj.TargetUsername = u2
         obj.MyDisplayname = d1
         obj = GetNote(obj.MyID, 1, obj)

         if len(obj.Roomname) == 0 {
            obj.Roomname = d2
         }
      } else {
         obj.TargetDisplayname = d1
         obj.MyID = p2
         obj.TargetID = p1
         obj.MyPubKey = pub2
         obj.TargetPubKey = pub1
         obj.MyUsername = u2
         obj.TargetUsername = u1
```

Segmen Program 4.19 Fungsi Go untuk Mengambil Data Chat Room (Lanjutan)

```
        obj.MyDisplayname = d2
          obj = GetNote(obj.MyID, 2, obj)

          if len(obj.Roomname) == 0 {
             obj.Roomname = d1
          }
       }

       arrObj = append(arrObj, obj)
       obj = ChatRooms{}

       index++
    }

    byteVal, err := json.Marshal(arrObj)
    if err != nil {
       panic(err)
    }
    hash_value := murmur3.Sum32(byteVal)

    res.Status = http.StatusOK
    res.Message = "Success getting your chat rooms"
    res.Data = map[string]interface{}{
       "Status": "OK",
       "Data":   arrObj,
       "Hash":   hash_value,
    }

    StoreItemToCache(user_pt+"-CR_Value", hash_value)

    return res, nil
}
```

Untuk pengiriman *chat*, aplikasi pertama akan menyambungkan diri (aplikasi) dengan *websocket* yang disediakan pada *API* agar bisa mengirimkan *chat*. Namun apabila *user* tidak tersambung dengan internet atau tidak bisa berkomunikasi dengan *API*, maka *user* tidak dapat mengirimkan *chat* baru kepada lawan bicara. Data yang dapat diterima oleh aplikasi adalah Foto(.jpg, .png, .jpeg), Dokumen(.pdf, .pptx, .docx), dan Audio(.mp3, .wav). Hal tersebut dapat dilihat pada Segmen Program 4.20 dan Segmen Program 4.21

Segmen Program 4.25 Fungsi Flutter untuk Connect dengan WebSocket

```
...
wsChannel = IOWebSocketChannel.connect(
        "ws://${globe.wartawargaAPIWebSocket}/chattingWS?room_id
=${widget.roomID}");

    wsChannel.stream.listen((data) async {
      if (data is String) {
        //Text Message
        //Parse incoming message
        var result = jsonDecode(data);
        ChatLogs message = ChatLogs.fromJson(result['Payload']);

        //Add to cached chat logs
        setState(() {
          tempChatLogs.add(message);
        });

        //Add to hive
        safeData(message);
      } else {
        String jsonString = utf8.decode(data);
        Map<String, dynamic> result = jsonDecode(jsonString);

        if (result.containsKey('FileData') &&
result.containsKey('FileInfo')) {
          String fileData = result["FileData"];
          Map<String, dynamic> fileInfo = result['FileInfo'];

          Files newFile = Files(
            filePath: fileInfo["file_path"],
            fileName: fileInfo['file_name'],
            fileExt: fileInfo['file_extension'],
            fileSize: fileInfo['file_size'],
            localFilePath: "",
          );
```

Segmen Program 4.20 Fungsi Flutter untuk Connect dengan WebSocket (Lanjutan)

```
          var checkRes = await checkFilePresence(newFile);
          if (checkRes["Exist"]) {
            newFile.existLocal = true;
            newFile.localFilePath = checkRes["LocalPath"];
          } else {
            newFile.existLocal = false;
          }
          ChatLogsHive newData = ChatLogsHive(
            roomId: result["RoomId"],
            senderId: result["SenderId"],
            senderUsername: globe.username,
            encryptedMessage: fileInfo["file_name"],
            encryptionKey: "key here",
            timeStamp: DateFormat('yyyy-MM-dd
HH:mm:ss').format(DateTime.now()),
            isFile: true,
            isRead: true,
            isVisible: true,
          )..file = FilesHive.fromLocal(newFile);

          setState(() {
            tempChatLogs.add(ChatLogs.fromHive(newData));
          });

          await chatLogsBox.add(newData);
        }
      }
    });
...
```

Segmen Program 4.26 Fungsi Go untuk Handle Koneksi WebSocket

```
func ChattingWebSocketController(c echo.Context) error {
   ws, err := chattingUpgrader.Upgrade(c.Response(),
c.Request(), nil)
   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }
   defer ws.Close()

   roomID := c.FormValue("room_id")
   groupID := c.FormValue("group_id")

   if roomID != "" {
      AddPrivChatConnection(roomID, ws)
   }

   if groupID != "" {
      AddGroupSocketParticipant(groupID, ws)
   }

   for {
      messageType, message, err := ws.ReadMessage()
      if err != nil {
         if websocket.IsCloseError(err,
websocket.CloseGoingAway) {
            log.Println("Websocket closed by client")
         }
         if websocket.IsCloseError(err,
websocket.CloseNoStatusReceived) {
            log.Println("WebSocket closed abruptly.")
         } else {
            log.Println("Error reading message:", err)
         }
         roomID := GetPrivChatRoomID(ws)
         groupID := GetGroupIDv2(ws)

         if roomID != "" {

            RemovePrivChatConnection(roomID, ws)
            break
         }

         if groupID != "" {
            RemoveGroupSocketParticipant(groupID, ws)
            break
         }

         break
      }
```

Segmen Program 4.21 Fungsi Go untuk Handle Koneksi WebSocket (Lanjutan)

```
      //File Handling
      if messageType == websocket.BinaryMessage {
         // Process the file data and other information
         var messageData map[string]interface{}
         if err := json.Unmarshal(message, &messageData); err !=
nil {
            fmt.Println(err)
            continue
         }
         // Retrieve the file data and other information from
the JSON payload
         destTable, _ := messageData["header"].(string)
         senderId, _ := messageData["senderId"].(string)
fileData, _ := messageData["fileData"].(string)
         fileName, _ := messageData["filename"].(string)
         fileType, _ := messageData["fileType"].(string)
         hashValueStr, _ := messageData["hashValue"].(string)

         // Decode the base64-encoded file data
         decodedFileData, err :=
base64.StdEncoding.DecodeString(fileData)
         if err != nil {
            fmt.Println("Error decoding file data:", err)
            continue
         }
         // Calculate the hash value of the file data
         hashValue := md5.New()
         _, err = hashValue.Write(decodedFileData)
         if err != nil {
            fmt.Println("Error computing hash value:", err)
            continue
         }
         // Verify the hash value
         computedHashValue := fmt.Sprintf("%x",
hashValue.Sum(nil))
         if strings.ToLower(computedHashValue) !=
strings.ToLower(hashValueStr) {
            fmt.Println("Hash values do not match")
            continue
         }
         //Save to database
         obj, err :=
models.HandleFilesFromWebSocket(decodedFileData, fileName,
destTable, senderId, fileType)
         if err != nil {
            fmt.Println("Error saving file:", err)
            continue
         }
```

Segmen Program 4.21 Fungsi Go untuk Handle Koneksi WebSocket (Lanjutan)

```
         //Send to the websockets
         if strings.Contains(destTable, "ChatRoom") {
            SendFileToRoom(destTable, []byte(fileData), obj,
senderId)
         } else {
            SendFileToGroups(destTable, []byte(fileData), obj,
senderId)
         }
      } else if messageType == websocket.TextMessage { //Decode
the data received into an interface for sorting
         fmt.Println("Got Text Message...")
         var data map[string]interface{}
         if err := json.Unmarshal(message, &data); err != nil {
            fmt.Println(err)
            break
         }
         //Retrieve the header
         dataHeader, ok := data["header"].(string)
         if !ok {
            break
         }
         if dataHeader == "Personal" {
            jsonStr, err := json.Marshal(data["payload"])
            if err != nil {
               panic(err)
               break
            }
            var chatMessage models.NewChat
            if err := json.Unmarshal(jsonStr, &chatMessage); err
!= nil {
               // handle error
            }
            //Save to Database
            err = models.WebSocketSendChat(chatMessage)
            if err != nil {
               panic(err)
               continue
            }
            SendToRoom(chatMessage.Room_ID, chatMessage)
         } else if dataHeader == "Group" {
            jsonStr, err := json.Marshal(data["payload"])
            if err != nil {
               panic(err)
               break
            }
            var chatMessage models.NewGroupChat
            if err := json.Unmarshal(jsonStr, &chatMessage); err
!= nil {
            }
```

Segmen Program 4.21 Fungsi Go untuk Handle Koneksi WebSocket (Lanjutan)

```
            err = models.SocketSendGroupChat(chatMessage)
            if err != nil {
               panic(err)
               continue
            }
            SendToGroups(chatMessage.GroupID, chatMessage)
         }

      } else {
         fmt.Println("Un-Supported File Received")
      }
   }

   return nil
}
```

Ketika *user* membuka halaman *chatting*, baik privat ataupun group, maka aplikasi akan mengambil *hash value* dari *API* untuk dibandingkan dengan *hash value lokal,* apabila berbeda maka aplikasi akan mengambil data baru dari *API* untuk dimasukan ke dalam penyimpanan *local*, apabila sama maka aplikasi akan memuat data dari *lokal*. Hal tersebut dapat dilihat pada Segmen Program 4.22 dan Segmen Program 4.23

Segmen Program 4.27 Fungsi Flutter untuk Memanggil Fungsi Ambil Data Chat Logs di API

```
Future<List<ChatLogs>> GetChatLogs(String room_id) async {
  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var requested_id = global.currentID;

  var tokenString = global.tokenString;

  //Preapare HTTP Request

  var getRequest = Uri.parse(
      "$urlString/getChats?room_id=$room_id&&requester_id=$reque
sted_id");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };
  var request = await http.get(getRequest, headers: headers);
  if (request.statusCode == 200) {
    final requestData = jsonDecode(request.body);
    if (requestData["Data"]["Status"] == "OK") {
      List<dynamic> temp = requestData["Data"]["Data"] ?? [];
      //Cast to ChatLogs type
      List<ChatLogs> arrObj =
          temp.map((data) => ChatLogs.fromJson(data)).toList();
      //Verify File Existence
      for (var item in arrObj) {
        if (item.isFile) {
          var funcResult = await checkFilePresence(item.file);
```

Segmen Program 4.22 Fungsi Flutter untuk Memanggil Fungsi Ambil Data Chat Logs di API (Lanjutan)

```
          if (funcResult["Exist"]) {
            item.file.existLocal = true;
            item.file.localFilePath = funcResult["LocalPath"];
          } else {
            item.file.existLocal = false;
          }
        }
      }

      return arrObj;
    } else {
      return [];
    }
  } else {
    debugPrint(
        "Debug(backendHandler.dart) | Failed to get chat rooms,
reason : ${request.body.toString()}");
    return [];
  }
}
```

Segmen Program 4.28 Fungsi Go untuk Mengambil Data Chat Logs

```
func GetChatsController(c echo.Context) error {
       //room_id int, requester_id int
       room_id := c.FormValue("room_id")
       requester_id := c.FormValue("requester_id")
       current_index, _ :=
strconv.Atoi(c.FormValue("current_index"))

       result, err := models.GetChats(room_id, requester_id,
current_index)

       if err != nil {
              return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
       }

       return c.JSON(http.StatusOK, result)
}
func GetChats(room_id string, requester_id string, current_index
int) (Response, error) {
       var res Response
       var obj ChatLogs
       var arrObj []ChatLogs
```

Segmen Program 4.23 Fungsi Go untuk Mengambil Data Chat Logs (Lanjutan)

```
	requester_pt, err := DecryptAES_CBC(requester_id)
	if err != nil {
		return res, err
	}

	if requester_pt == "-1" {
		res.Status = http.StatusOK
		res.Message = "Failed to get your chat logs !"
		res.Data = "User does not exist !"

		return res, nil
	}

	actor_id, _ := strconv.Atoi(requester_pt)
	if current_index < 0 {
		current_index = 0
	}

	con := db.CreateConnection()
	defer con.Close()

	getSQLStatement := "SELECT room_id, sender_id,
u.username, chat_content, encrypted_key, chat_timestamp,
is_read, is_visible, is_file FROM chat_logs JOIN users u ON
chat_logs.sender_id = u.user_id WHERE room_id = ? AND
chat_logs.chat_logs_index > ? ORDER BY chat_timestamp ASC"
	result, err := con.Query(getSQLStatement, room_id,
current_index)
	if err != nil {
		return res, err
	}
	defer result.Close()

	var tempSenderID, index int
	var fixedSenderID string
	index = 0

	for result.Next() {
		err = result.Scan(
			&obj.Room_id,
			&tempSenderID,
			&obj.Sender_Name,
			&obj.Message,
			&obj.Key,
			&obj.Timestamp,
			&obj.Read,
			&obj.Visibility,
			&obj.File,
		)
```

Segmen Program 4.23 Fungsi Go untuk Mengambil Data Chat Logs (Lanjutan)

```
		//Limit to only encrypt once to save up time and
resource (precaution for large chat logs)
		if actor_id == tempSenderID {
			obj.Sender_id = requester_id

		} else {
			if index == 0 {
				fixedSenderID, _ =
EncryptAES_CBC([]byte(strconv.Itoa(tempSenderID)))
			}
			obj.Sender_id = fixedSenderID
			index++
		}
		if obj.File == true {
			obj.FileInfo = getFileInfo(con,
obj.Room_id, obj.Message)
			obj.Message = obj.FileInfo.FileName
		}

		arrObj = append(arrObj, obj)
		obj = ChatLogs{}
	}

	res.Status = http.StatusOK
	res.Message = "Success getting chat logs !"
	res.Data = map[string]interface{}{
		"Status": "OK",
		"Data":   arrObj,
	}

	if actor_id != tempSenderID {
		updateSQLStatement := "UPDATE chat_logs SET
is_read = true WHERE room_id = ? AND recipient_id = ? AND
is_read = false"

		_, err = con.Exec(updateSQLStatement, room_id,
actor_id)
	}

	return res, nil
}
```

4.10 Struktur Organisasi

User bisa melihat struktur organisasi yang telah didaftarkan dan sedang berlaku dalam lingkup aplikasi. Hal tersebut dapat dilihat pada Segmen Program 4.24 dan Segmen Program 4.25

Segmen Program 4.29 Fungsi Flutter untuk Memanggil Fungsi Get Struktur List di API

```
Future<List<GroupLists>> GetGroupAllGroupListsShort() async {
  String urlString = global.wartawargaAPIServer;

  //Prepare Parameters
  var tokenString = global.tokenString;

  //Prepare HTTP Request
  var getRequest =
Uri.parse("$urlString/getAllGroupShortsAdmin?group_id=1");
  var headers = {
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": global.deviceName,
    "Authorization": "Bearer $tokenString"
  };

  var request = await http.get(getRequest, headers: headers);

  if (request.statusCode == 200) {
    final requestData = jsonDecode(request.body);
    if (requestData["Data"]["Status"] == "OK") {
      List<dynamic> temp = requestData["Data"]["Data"] ?? [];

      List<GroupLists> arrObj =
          temp.map((data) =>
GroupLists.fromJson(data)).toList();

      return arrObj;
    } else {
      debugPrint(
          "Debug(backendHandler.dart) | Failed to get all group
short lists, reason: ${requestData.toString()}");
      return [];
    }
  } else {
    debugPrint(
        "Debug(backendHandler.dart) | Failed to get group short
lists !");
    return [];
  }
}
```

Segmen Program 4.30 Fungsi Go untuk Mengambil Struktur Organisasi

```
func GetAllGroupShortsAdminController(c echo.Context) error {
   group_id := c.FormValue("group_id")

   result, err := models.GetAllGroupShortsAdmin(group_id)

   if err != nil {
      return c.JSON(http.StatusInternalServerError,
map[string]string{"Message": err.Error()})
   }


   return c.JSON(http.StatusOK, result)
}
func GetAllGroupShortsAdmin(group_id string) (Response, error) {
   var res Response
   var obj GroupShorts
   var arrObj []GroupShorts

   con := db.CreateConnection()
   defer con.Close()

   rootHierarchy := string(group_id[0])

   getSQLStatement := "SELECT group_id, group_name FROM
user_group WHERE group_id LIKE ? AND group_type = 'Formal'"

   result, err := con.Query(getSQLStatement, rootHierarchy+"%")
   if err != nil {
      return res, err
   }
   defer result.Close()
   for result.Next() {
      err = result.Scan(
         &obj.GroupID,
         &obj.GroupName,
      )
      if err != nil {
         return res, err
      }
      temp := obj.GroupID
      obj.Depth = len(strings.ReplaceAll(temp, ",", ""))
      arrObj = append(arrObj, obj)
      obj = GroupShorts{}
   }
```

Segmen Program 4.25 Fungsi Go untuk Mengambil Struktur Organisasi (Lanjutan)

```
    res.Status = http.StatusOK
    res.Message = "Success getting hierarchical group"
    res.Data = map[string]interface{}{
       "Status": "OK",
       "Data":   arrObj,
    }
    //Save the hash value to cache for fast-get requests
    byteVal, err := json.Marshal(arrObj)
    if err != nil {
       return res, err
    }
    hash_value := murmur3.Sum32([]byte(byteVal))
    StoreItemToCache(rootHierarchy+"-Hierarchy_HV", hash_value)

    return res, nil
}
```

4.11 Menjalankan API

Pada skripsi ini, API akan di *host* / dijalankan pada server publik milik PT. XYZ, spesifiknya pada node Proxmox dengan spesifikasi sebagai berikut:

1. Memory : 1 Gb
2. Processors : 1 Socket(s), 1 Core(s)
3. OS : Debian Bullseye (11)
4. Storage : 10 Gb

Server / Node, kedepannya akan dirujuk sebagai Node, pada Skripsi ini telah diberikan / di-*assign* IP Publik oleh pihak IT PT. XYZ sehingga memungkinkan untuk diakses dari jaringan internet eksternal, tanpa harus menggunakan VPN ataupun harus berada di jaringan lokal. Selain dari itu, Node juga sudah disiapkan kredential untuk koneksi SSH dari jaringan eksternal. Berikut adalah langkah – langkah yang diperlukan untuk menjalankan API yang telah dibuat pada poin – poin sebelumnya :

1. Buka koneksi SSH ke Node yang bersangkutan

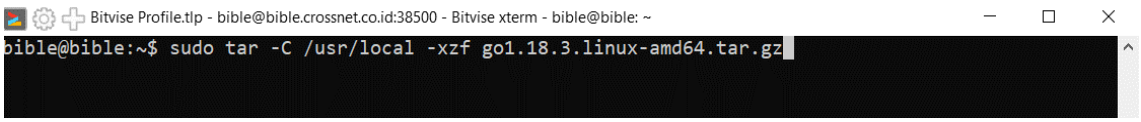
Memory	1.00 GiB
Processors	1 (1 sockets, 1 cores)
BIOS	Default (SeaBIOS)
Display	Default
Machine	Default (i440fx)
SCSI Controller	VirtIO SCSI
CD/DVD Drive (ide2)	local:iso/debian-11.0.0-amd64-netinst.iso,media=cdrom
Hard Disk (scsi0)	HDD-1T:vm-115-disk-0,size=10G
Network Device (net0)	e1000=FA:C8:05:73:20:8D,bridge=vmbr0,firewall=1

Gambar 4.4 Spesifikasi Sistem

2. Menginstall Golang untuk Linux (versi menyesuaikan)
   a. Download Installer Go untuk Linux

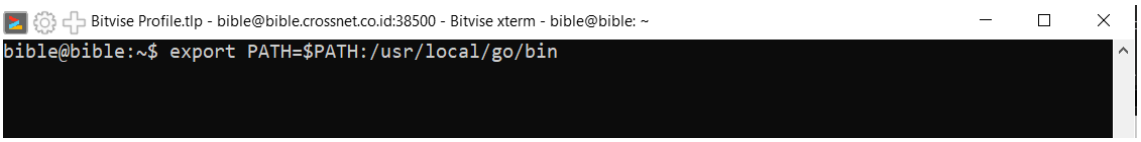
Gambar 4.5 Download Installer Go

   b. Extract dan install Go



Gambar 4.6 Extract dan Install Go

c. Tambahkan PATH Go ke Environment Variable Linux



Gambar 4.7 Tambah Path Go ke Environment Variable Linux

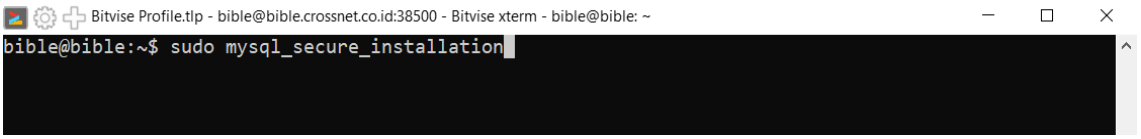
d. Verifikasi Instalasi Go

Gambar 4.8 Verifikasi Instalasi Go

3. Menginstall MariaDB untuk Linux (versi menyesuaikan)
   a. Download Instalasi MariaDB Server

Gambar 4.9 Download Instalasi MariaDB

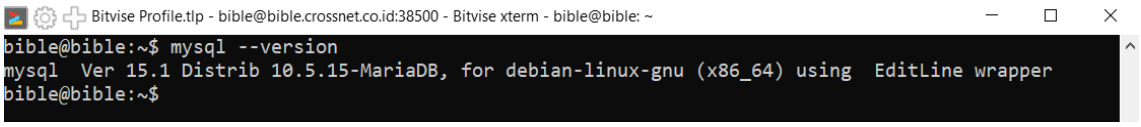
b. Jalankan / Install MariaDB Server



Gambar 4.10 Jalankan Instalasi MariaDB

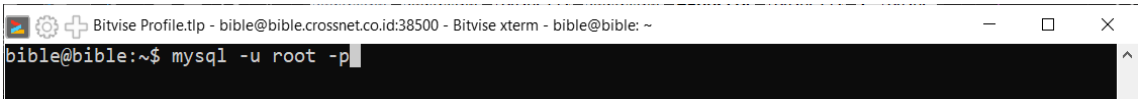
Pada tahap ini akan diberikan beberapa pertanyaan bersangkutan dengan password untuk akun root dan konfirmasi menggunakan kredential akun root di Linux sebagai kredential akun root di MariaDB

c. Verifikasi Instalasi MariaDB



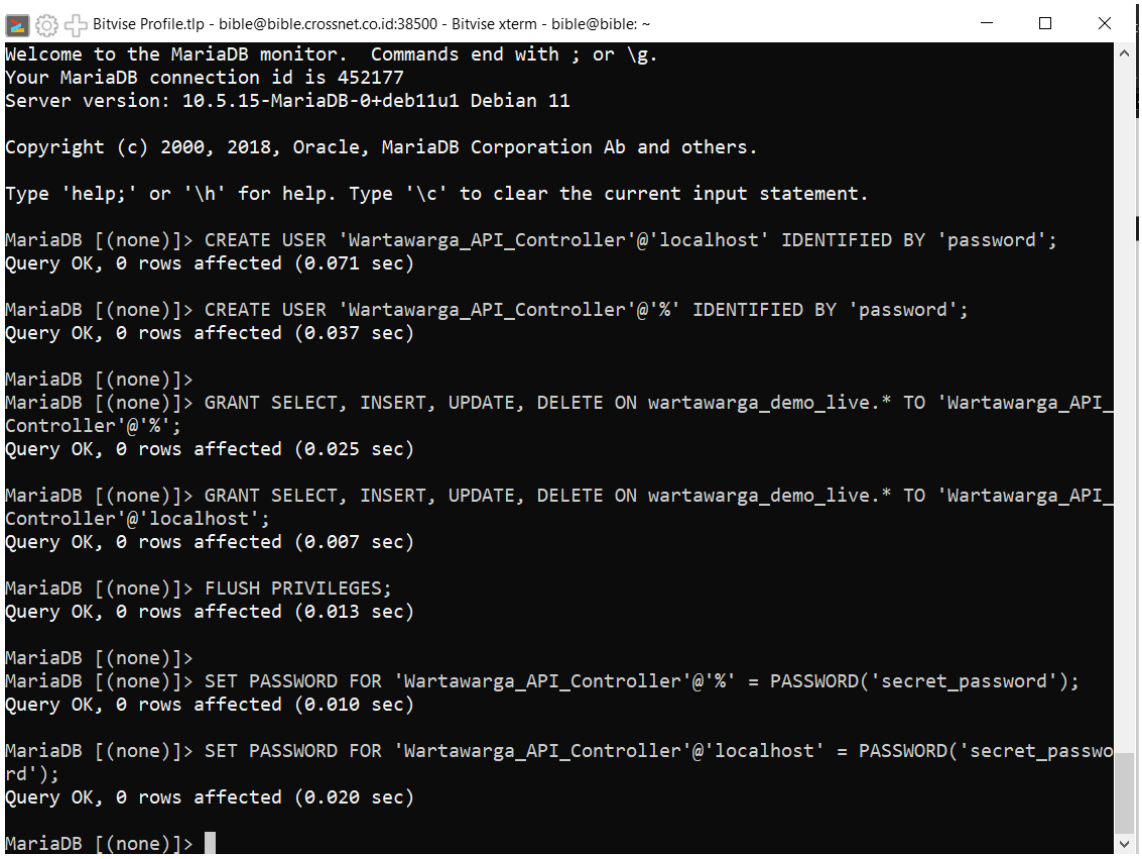
Gambar 4.11 Verifikasi Instalasi MariaDB

4. Membuat Akun untuk MariaDB dan mengatur Hak Akses untuk akun yang dibuat
    a. Masuk / Login ke dalam akun MariaDB



Gambar 4.12 Login ke dalam Mysql / MariaDB

    b. Buat Akun dan Berikan / Atur hak akses



Gambar 4.13 Buat Akun Baru dan Set Hak Akses

c. Verifikasi Akun Terbuat

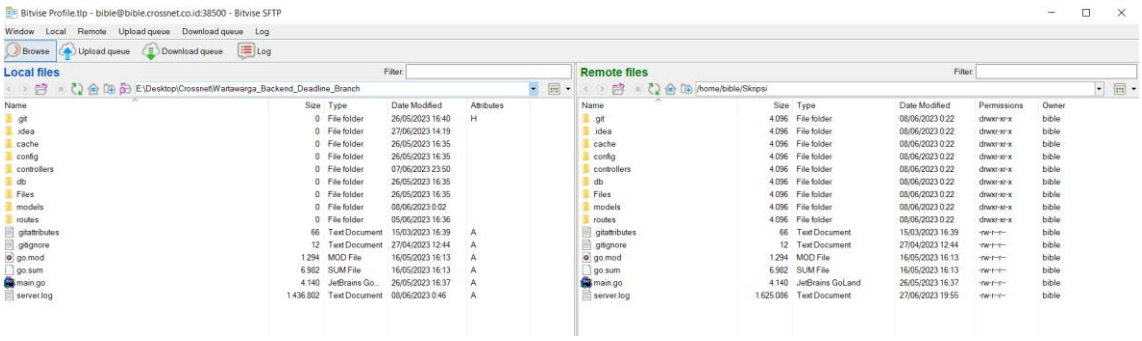
```
MariaDB [(none)]> SELECT user FROM mysql.user WHERE user = 'Wartawarga_API_Controller';
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id:    452178
Current database: *** NONE ***

+---------------------------+
| User                      |
+---------------------------+
| Wartawarga_API_Controller |
| Wartawarga_API_Controller |
+---------------------------+
2 rows in set (0.025 sec)

MariaDB [(none)]>
```

Gambar 4.14 Verifikasi Terbuatkan Akun

```
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
| Grants for Wartawarga_API_Controller@%
                       |
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
| GRANT USAGE ON *.* TO `Wartawarga_API_Controller`@`%` IDENTIFIED BY PASSWORD '*E0A6057BDBD3D6A85C3
B232B9EE04DBBBAD5E9BD' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `wartawarga_demo_live`.* TO `Wartawarga_API_Controller`@`%
`                      |
+-----------------------------------------------------------------------------------------------------------------------
----------------------+
2 rows in set (0.028 sec)

MariaDB [(none)]>
```

Gambar 4.15 Verifikasi Hak Akses Akun

5. Memasukan file API yang telah dibuat



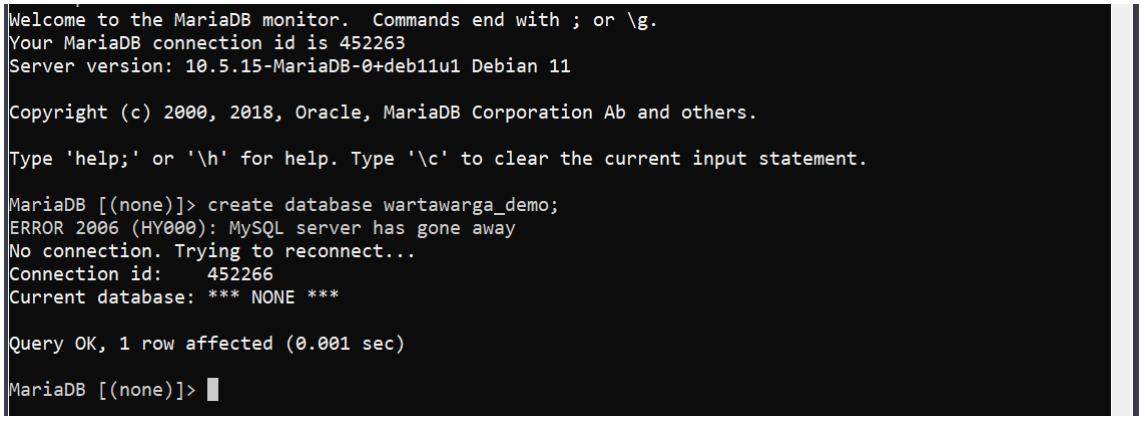
Gambar 4.16 Memasukan File Lokal ke Server dengan menggunakan SFTP

Pada tahap ini akan menggunakan protokol *SFTP* untuk memindahkan / *copy file* API ke dalam server. *Bitvise* menyediakan fitur untuk *SFTP*, mirip dengan *WinSCP*, yang perlu dilakukan adalah menavigasi ke folder yang bersangkutan dan melakukan *drag-and-drop* dari sisi lokal ke sisi *remote*. Tunggu beberapa saat untuk menyelesaikan proses transfer, apabila sudah selesai maka di bagian *remote* akan terlihat bahwa folder sudah sukses di *copy*.

6. Import Struktur Database yang telah dibuat ke dalam MariaDB Server
   Dengan metode yang sama seperti pada poin 5, pindahkan file DDL ke Server

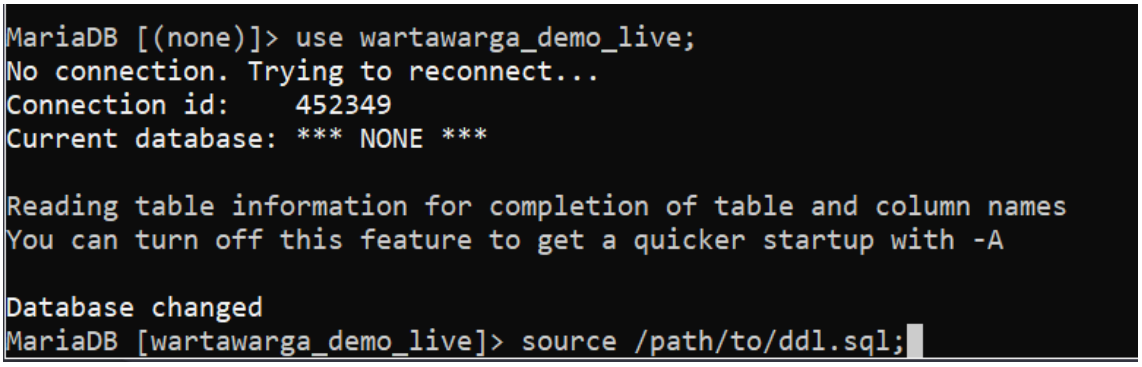
Gambar 4.17 Hasil setelah memindahkan DDL ke Server

Buat Database terlebih dahulu pada sisi Server dan pastikan nama Database **SESUAI** (*case sensitive*) dengan nama Database yang ada pada DDL



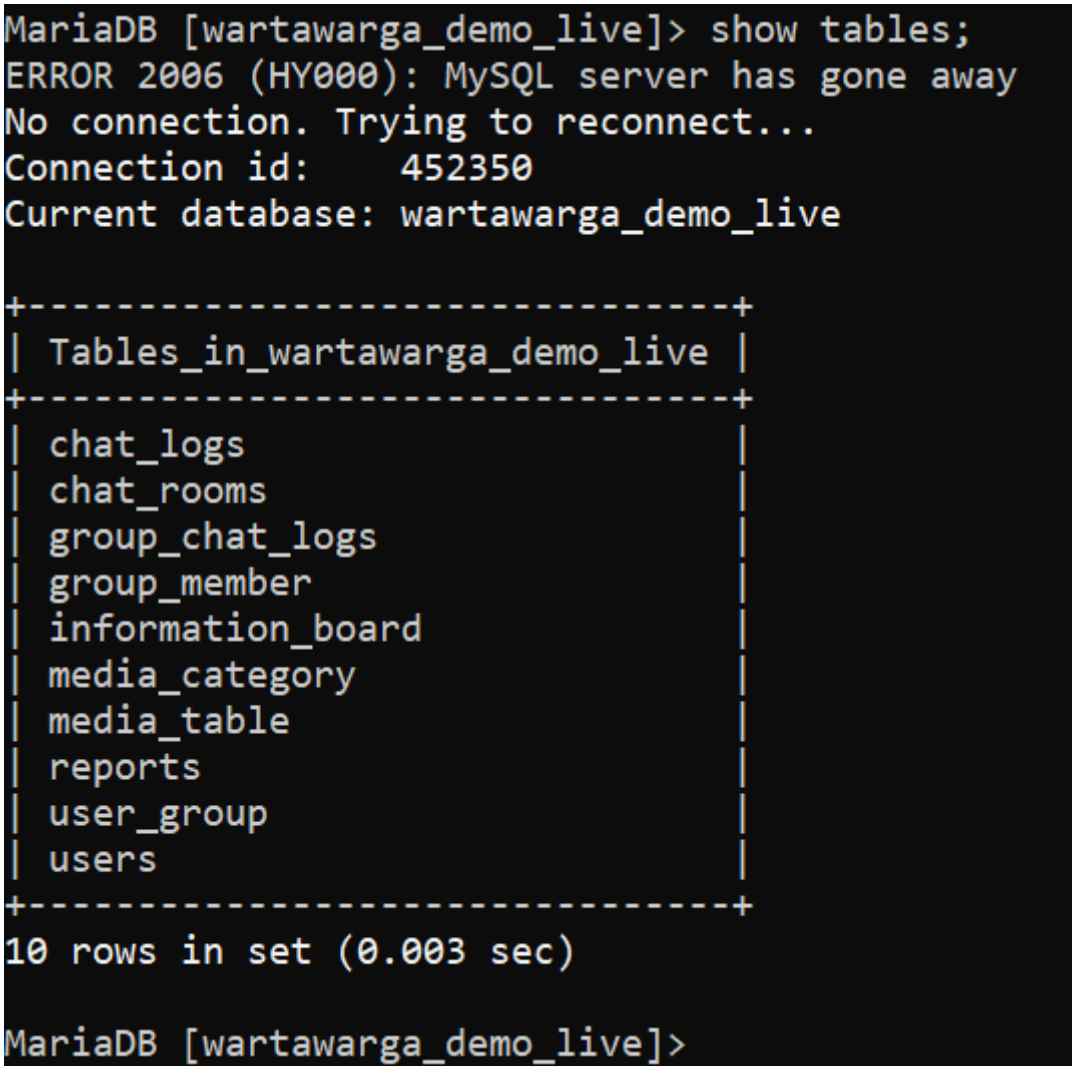
Gambar 4.18 Membuat Database sesuai dengan Nama Database pada DDL

Setelah berhasil dibuat maka langkah selanjutnya adalah memilih database dan melakukan proses *import*



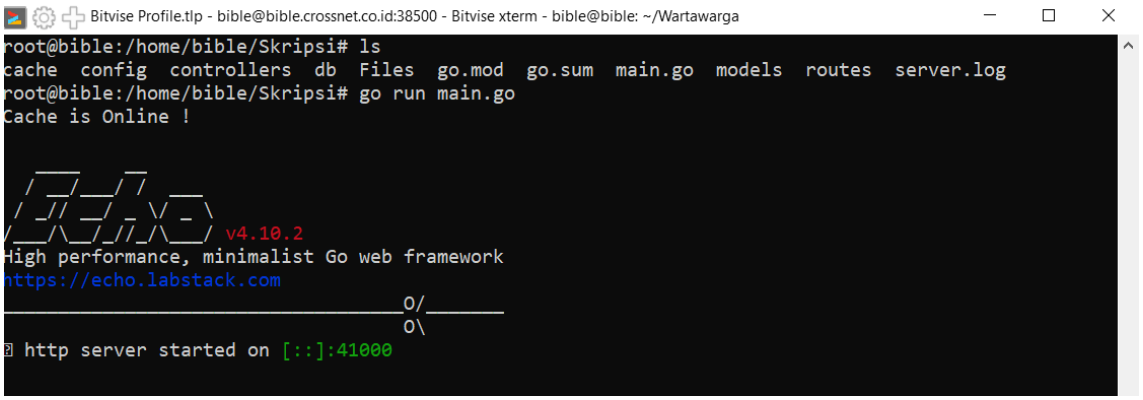
Gambar 4.19 Import DDL ke dalam Mysql / MariaDB

Pastikan DDL sudah ter-*import* dengan sesuai

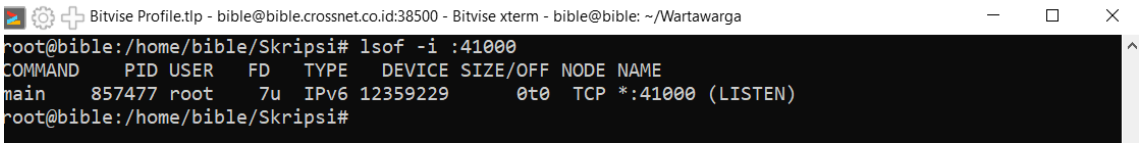


Gambar 4.20 Verifikasi hasil import DDL

7. Menjalankan API
   Untuk tahap ini belum berupa .bin (executable) file karena API masih akan diupdate seiring dengan berjalannya waktu dan memastikan bahwa API sudah berjalan

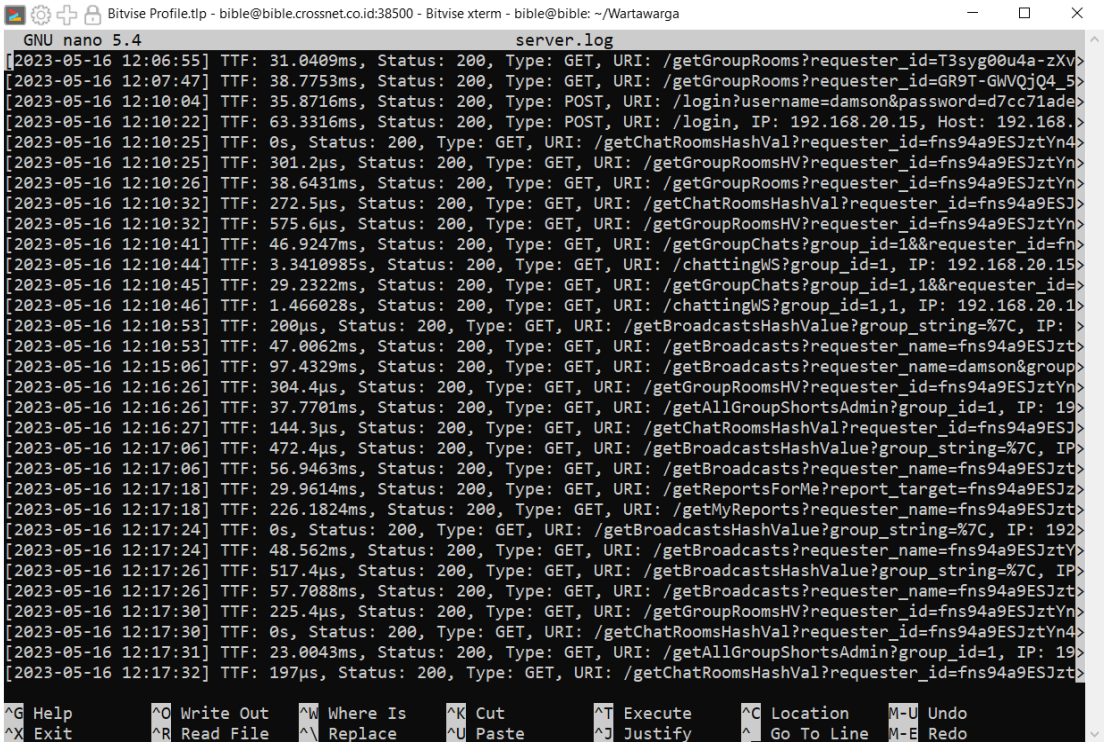


Gambar 4.21 Menjalankan API



Gambar 4.22 Verifikasi jalannya API

8. Memantau Log File yang di auto-generate oleh API
   Semua *http request* yang diterima oleh API akan secara otomatis dicatat ke dalam log file. Secara *default* akan disimpan pada folder yang sama dengan nama server.log



Gambar 4.23 Sample tampilan Log File