

BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kedudukan selama durasi magang di PT Surya Semesta Internusa adalah sebagai *Fullstack Developer*. Posisi ini dijalankan di bawah bimbingan Bapak Eryan Fauzan, seorang *fullstack developer* senior, yang merupakan karyawan dari PT Suryacipta Swadaya, sebuah anak perusahaan milik PT Surya Semesta Internusa, yang juga berada dibawah naungan *IT Architecture and Development Manager*, dan *Chief Information Officer*. Tugas utama yang diberikan yaitu berperan sebagai *support* dalam pengembangan aplikasi *progress logger*, dalam hal perancangan, pembuatan kode, maupun *testing* produk.

Untuk proyek pengembangan aplikasi *form generator*, dijalankan dibawah bimbingan bapak Alvin Susanto, seorang *fullstack developer* senior dari PT Suryacipta Swadaya, dan tugas yang diberikan adalah menjadi *support* untuk Bapak Alvin Susanto dalam hal pembuatan kode dan *testing* produk sebelum dirilis untuk penggunaan *internal*.

Untuk proyek pengembangan aplikasi SSO (*Single Sign-On*) *Admin Dashboard*, dilaksanakan dibawah bimbingan Bapak Alvin Susanto. Tugas yang diberikan adalah membuat keseluruhan dari tampilan dan logika dan fungsionalitas dari aplikasi SSO *Admin Dashboard*.

3.2 Tugas Kerjaan Magang

Selama masa magang, ada beberapa tugas yang dikerjakan. Tugas-tugas tersebut adalah:

1. Merancang desain UI untuk aplikasi *progress logging* untuk proyek konstruksi.
2. Membantu *senior developer* untuk membuat hasil kodingan aplikasi *form generator*.
3. Merencanakan arsitektur dari *website* Admin Dashboard SSO.

3.3 Uraian Pekerjaan Magang

Pelaksanaan pekerjaan yang dilakukan dalam program magang dapat dilihat dari minggu-ke-minggu dalam Tabel 3.1.

Tabel 3.1 Pekerjaan berdasarkan minggu kerja

Minggu ke-	Uraian
1	Membuat prototype aplikasi <i>construction logger</i> di figma, lalu membuat demo simpel.
2	Melanjutkan prototyping web UI untuk aplikasi <i>construction logger</i> , menentukan basis data.
3	Memulai koding logika dasar aplikasi <i>progress logger</i> .
4	Melakukan programming <i>backend</i> untuk CRUD.
5	Membenarkan <i>bug</i> pada sistem navigasi aplikasi, dan fitur tambahan <i>user convenience</i> , seperti <i>import</i> data dari Microsoft Excel, pada aplikasi .
6	Membuat perbaikan terhadap UI dari aplikasi, dan melakukan perbaikan terhadap struktur kode.
7	Memulai membantu development dari aplikasi web <i>form generator</i> .
8	Menamambahkan seperasi dan abstraksi pada kode untuk struktur kode yang lebih rapi, dan juga fungsi CRUD.
9	Membuat input koordinat dengan peta dan mekanisme upload foto dan file.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A

Tabel 3.1 Pekerjaan berdasarkan minggu kerja (lanjutan)

Minggu ke-	Uraian
10	Melakukan <i>bug fixing</i> pada fitur-fitur UI dan CRUD aplikasi <i>form generator</i> .
11	Melakukan <i>bug fixing</i> pada fitur-fitur CRUD aplikasi <i>form generator</i> , dan juga melakukan <i>refactoring</i> logic agar lebih efisien.
12	Menambahkan fitur delete pada laman <i>data entry</i> + lebih banyak <i>bug fix</i> .
13	Melakukan pengembangan UI tampilan <i>Single Sign On (SSO) API Control Dashboard</i> .
14	Membuat fitur CRUD untuk <i>Dashboard Single Sign On</i> .
15	Membuat tampilan dan fitur CRUD lebih lanjut untuk tabel <i>mapping</i> .
16	Merevisi fitur CRUD dari <i>modal</i> menjadi <i>standalone page</i>

3.3.1 Metodologi dan Komunikasi

Bagian ini mendeskripsikan metodologi yang digunakan untuk membangun masing-masing aplikasi, serta metode komunikasi dan koordinasi untuk setiap proyek. Metodologi dan metode komunikasi untuk setiap proyek adalah sebagai berikut.

A) Aplikasi Form Generator

Aplikasi *form generator* dibuat menggunakan metodologi *project management Waterfall*. Ini dikarenakan *requirement* proyek memiliki sedikit sekali perubahan, dan juga melewati tahap perencanaan, pengembangan, dan pengujian

sebelum akhirnya berakhir di tahap *live deployment*, dengan perubahan yang akan dilakukan jika hanya ada tiket atau komplain yang serius dari *user*. Pembagian tugas diberikan melalui *chat* dan juga *voice call* di aplikasi Microsoft Teams dengan bapak Alvin Susanto, yang memberikan instruksi serta penjelasan untuk fitur-fitur yang akan dibuat secara sekuensial.

Proyek ini didesain menggunakan arsitektur *Model-View-Controller-Service*, atau kerap disingkat sebagai MVCS. *Model* merupakan representasi tabel dalam basis data, *view* merupakan tampilan, *service* merupakan abstraksi dimana *business logic* ditampung dan dijalankan, sementara *controller* merupakan jembatan antara *view* dan *service*, yang memproses *request* serta mengirimkan *response*. MVCS digunakan untuk menjunjung tinggi prinsip *Single Responsibility*, agar semua *file* bersifat *controller* hanya perlu fokus memproses respons, karena *business logic* yang kompleks dapat membuat *controller* menjadi berantakan.

B) Aplikasi SSO Admin Dashboard

Aplikasi *SSO Admin Dashboard* juga dibuat menggunakan metodologi *project management Waterfall*, dengan fase perencanaan, pengembangan, dan pengujian. Sama seperti aplikasi *form generator*, proyek ini jarang memiliki perubahan besar dalam segi desain maupun implementasi, karena *requirements* dari proyek sudah cukup jelas dan konkrit.

Instruksi dan pembagian tugas dilakukan oleh bapak Eryan Fauzan dan juga bapak Alvin Susanto melalui *chat* di Microsoft Teams. Selain itu, *code review* dan *feature review* dilakukan dengan Bapak Alvin Susanto melalui *voice call* di Microsoft Teams.

Proyek ini juga didesain menggunakan arsitektur MVCS. Ini dikarenakan aplikasi memiliki kompleksitas yang cukup tinggi dan repetitif, sehingga *business logic* sebaiknya dipisah.

C) Aplikasi Progress Logger

Aplikasi *progress logger* dibuat menggunakan sistem *prototyping* berbasis Agile. Sebuah *Minimum Viable Product* (MVP), yang merupakan produk paling simpel yang dapat digunakan dibuat. Produk tersebut terus-menerus diperbaiki berdasarkan umpan balik dari pengguna, dan juga diskusi antara anggota dari tim pengembangan.

Komunikasi untuk kelancaran proyek dilaksanakan melalui pesan secara verbal oleh *supervisor*, melalui *chat* di Microsoft Teams dengan *senior developer*, dan juga mengikuti *meeting* dengan tim IT di Microsoft Teams untuk melakukan *review* dari tugas-tugas yang sudah diberikan, serta mengikuti *meeting* dengan tim manajemen konstruksi untuk mendengar *concern* dan kebutuhan mereka.

Sama seperti *SSO admin dashboard* dan *form generator*, aplikasi ini juga dirancang dengan arsitektur MVCS. Kompleksitas aplikasi yang tinggi dan banyaknya fitur yang dapat berubah secara cepat membutuhkan modularisasi yang tinggi serta *Single Responsibility* untuk membuat kode tetap menjadi bersih.

3.3.2 Tools yang Dipakai

Bagian ini mendiskripsikan alat yang digunakan untuk membuat hasil dari setiap proyek. Alat yang digunakan dapat berupa bahasa pemrograman, *framework*, maupun sebuah aplikasi tersendiri.

A) Aplikasi Form Generator

Untuk bagian *frontend*, aplikasi *form generator* ini dibuat menggunakan HTML (dengan *ejs*) untuk menampilkan data, CSS (dengan *library Bootstrap*) untuk mempercantik tampilan, dan JavaScript untuk interaktivitas aplikasi. *Backend* dari aplikasi juga dibuat menggunakan JavaScript, untuk melakukan pengolahan data dan juga pemberian respons dari *server*.

B) Aplikasi SSO Admin Dashboard

SSO Admin Dashboard ini dibuat menggunakan React.js dengan TypeScript, dengan *library* TailwindCSS dan juga ShadCN untuk *frontend*. *Backend* dari proyek dibuat menggunakan Express.js.

C) Aplikasi Progress Logger

Aplikasi ini didesain menggunakan aplikasi Figma. Figma adalah sebuah aplikasi desain berbasis *open-source* di *web* yang dapat digunakan untuk membuat desain UI.

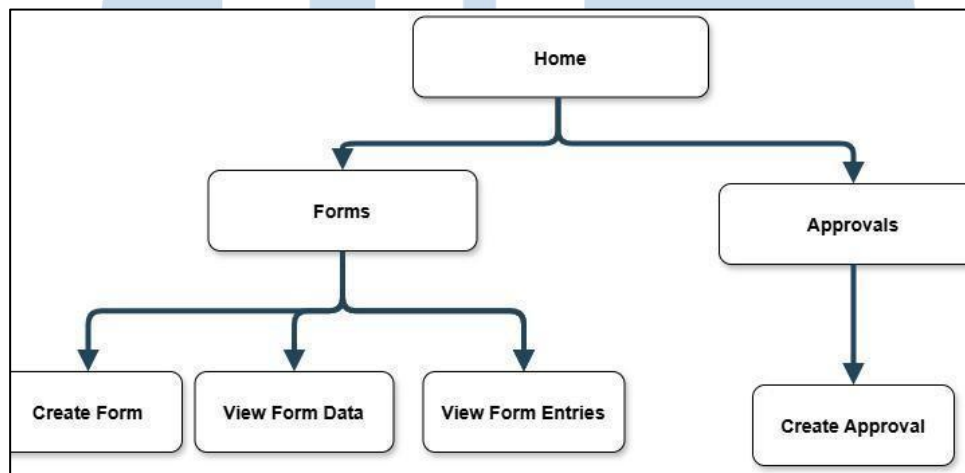
3.3.3 Hasil Magang

Proses pembuatan aplikasi dibagi menjadi dua bagian, yaitu proses perancangan atau *design* aplikasi dan proses pembuatan produk. *Design* adalah rancangan dari sebuah aplikasi. Proses ini meliputi perancangan tampilan atau *User Interface* dari aplikasi, alur laman aplikasi, dan juga proses yang digunakan oleh pengguna, dengan konsiderasi pada kenyamanan pengguna atau *User Experience*. Sementara itu, proses pembuatan meliputi pembuatan aplikasi baik secara fitur dan tampilan. Proses pembuatan aplikasi akan ditampilkan per-aplikasi, sebagai sub-babnya masing-masing.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

A) Aplikasi Form Generator

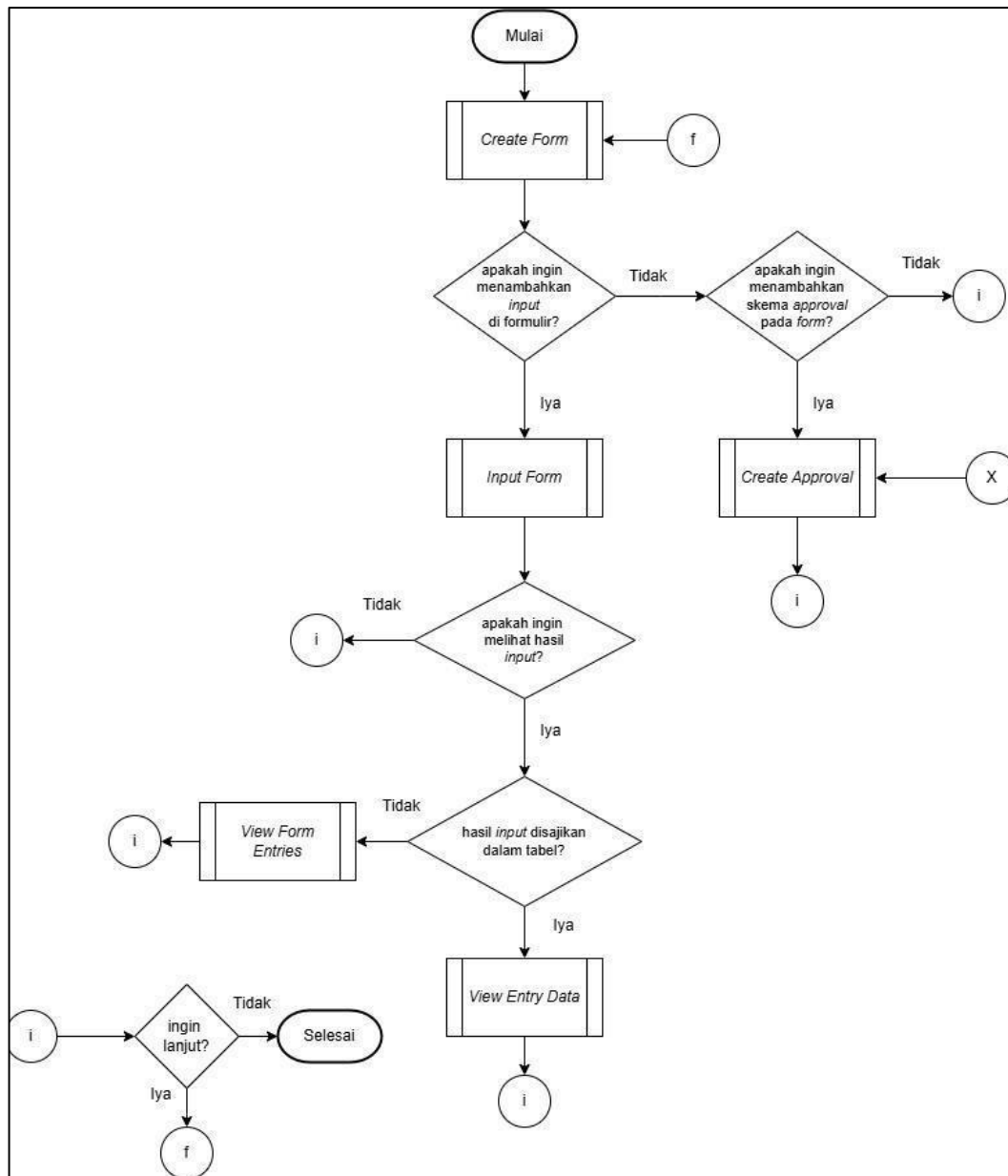
Aplikasi dirancang sesuai dengan alur navigasi seperti yang dapat dilihat di Gambar 3.1. Gambar tersebut menunjukkan relasi antar laman, mulai dari laman *login* yang bercabang ke laman utama, hingga laman *My Tasks*, *My Master Data*, serta laman-laman yang berada di dalam dua laman tersebut.



Gambar 3.1 Sitemap Aplikasi Form Generator

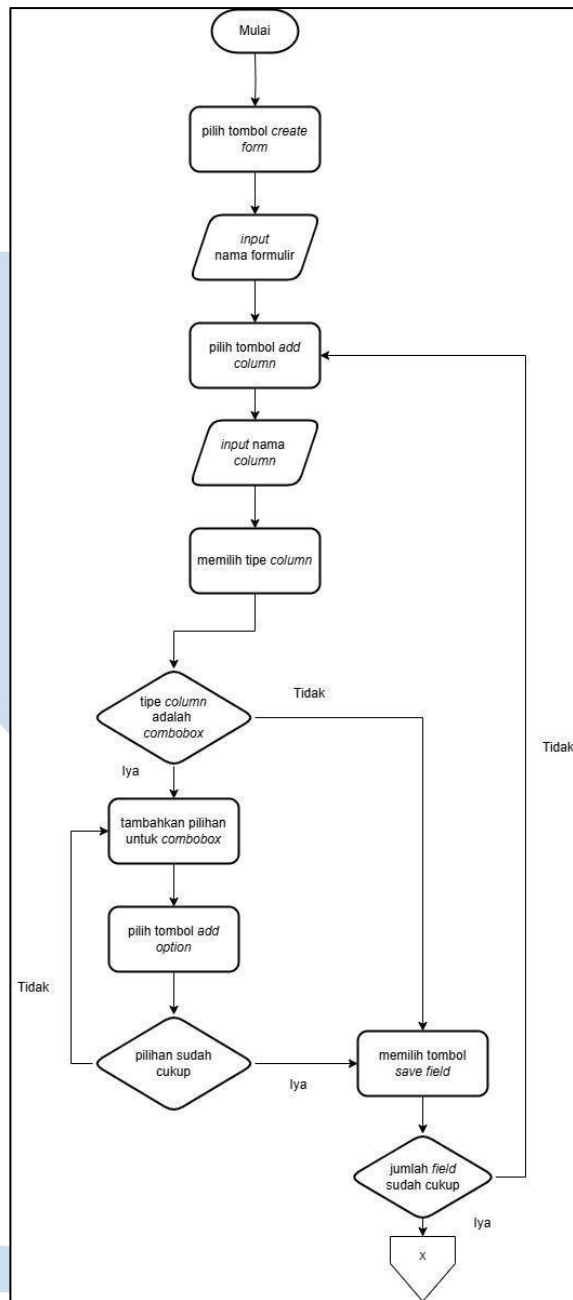
Setiap laman yang berada di *level* paling bawah terdiri dari proses-proses yang merupakan fungsi utama dari aplikasi. Proses-proses tersebut dapat direpresentasikan menggunakan *flowchart* untuk menunjukkan proses interaksi antara pengguna dan sistem. Aplikasi *form generator* memiliki beberapa proses utama yaitu *create form*, *input Form*, *view form data*, *view form entries*, dan *create approval*. Struktur dan alur dari proses-proses ini dapat dilihat dalam *master flowchart* untuk aplikasi *form generator*, yang dapat dilihat di Gambar 3.2.

UNIVERSITAS
MULTIMEDIA
NUSANTARA



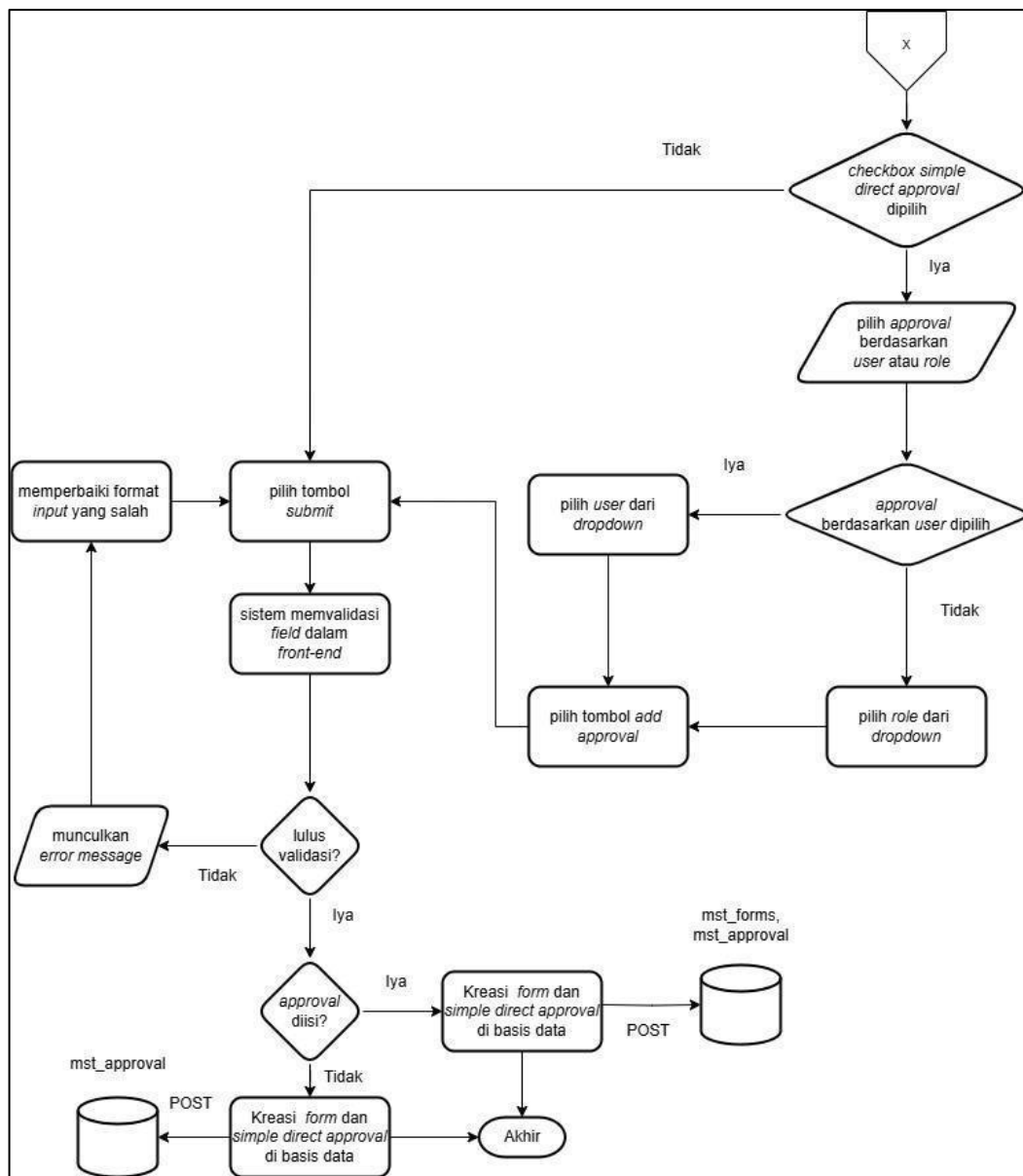
Gambar 3.2 Master Flowchart Aplikasi Form Generator

Flowchart ini menunjukkan urutan dari eksekusi setiap proses yang berada di aplikasi *form generator*. Proses pertama, yaitu proses *create form*, merupakan sebuah proses yang cukup kompleks, yang meliputi banyak pengambilan data dan komunikasi antara pengguna dan *server*. Proses ini dapat dilihat di Gambar 3.3.



Gambar 3.3 Flowchart Proses Create Form

UNIVERSITAS
MULTIMEDIA
NUSANTARA



Gambar 3.3 Flowchart Proses Create Form (lanjutan)

Proses *create form* dimulai dengan mengambil data dari *server* melalui *request GET* untuk semua pilihan *input* yang memungkinkan, agar pengguna dapat membuat pilihan yang relevan (seperti memilih *approver* yang tepat). Lalu, sistem melakukan pengecekan terhadap kolom-kolom yang telah diisi pengguna, untuk menentukan jika tipe *input* yang diberikan oleh pengguna sudah *valid* atau tidak. Jika tidak *valid*, maka proses pengumpulan *form* akan diberhentikan dan sebuah *popup* akan muncul dengan pesan *error*. Jika *input* yang diberikan oleh

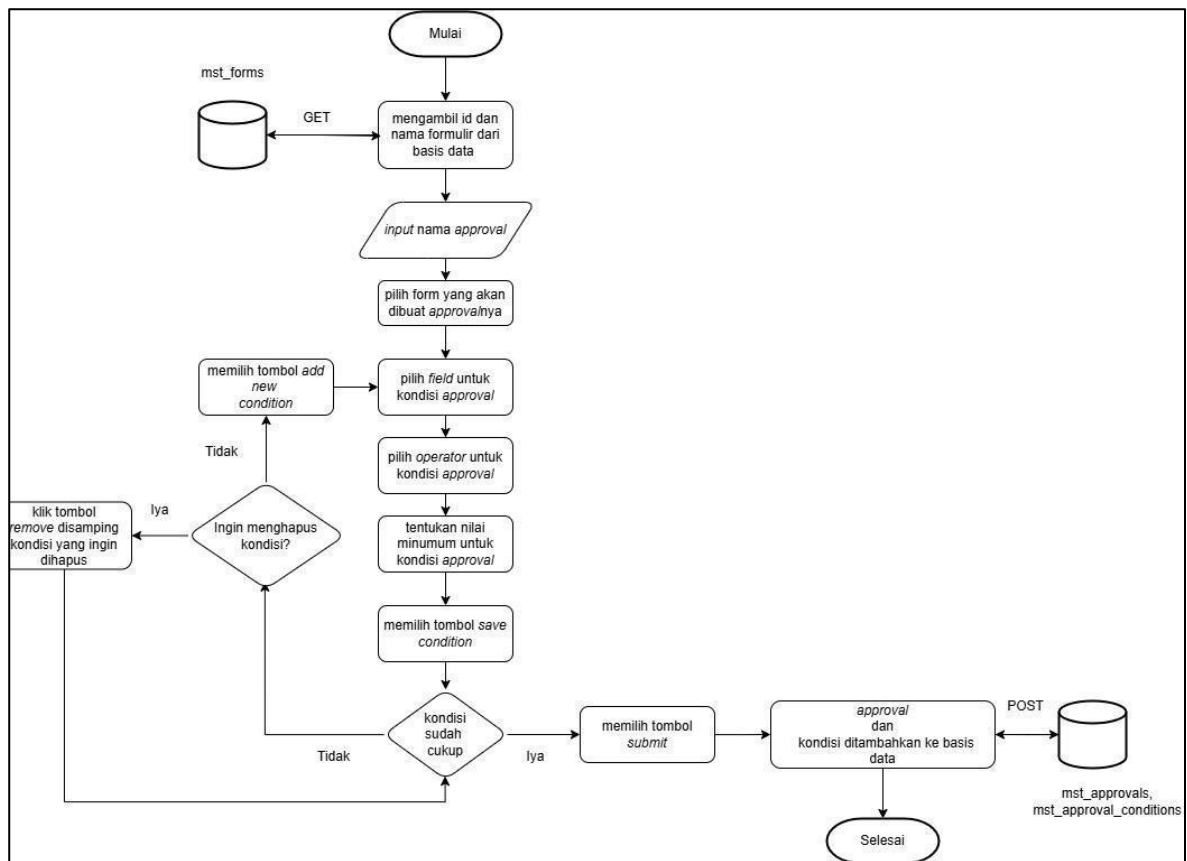
pengguna *valid*, maka aplikasi akan membuat formulir (bersama *simple approval* yang bersangkutan, jika dibuat) dan menyimpan data tersebut di basis data, melalui sebuah *request POST*.

Flowchart tersebut mendeskripsikan alur dari laman *Create Form*. Tampilan laman *Create Form* dapat dilihat di Gambar 3.4.

Gambar 3.4 Tampilan Laman *Create Form*

Laman tersebut memiliki *field* untuk pengisian nama formulir, serta tombol *Add Field* yang dapat menampilkan sebuah *combobox* yang dapat menampilkan pilihan tipe *field* dan nama *field* yang ingin ditambah untuk formulir. Laman juga memiliki *switch* yang dapat menampilkan *dropdown* pemilihan pengguna untuk *simple direct approval*, serta tombol *submit* untuk membuat formulir baru tersebut.

Proses *create approval* mencakupi pembuatan *master data approval*, yang mendefinisikan personel di organisasi yang dapat memberikan persetujuan, dan kondisi dari persetujuan tersebut. Dengan adanya *master data approval*, setiap masukan pada suatu formulir dapat dikirimkan ke *inbox* personel yang bertanggung jawab untuk memberikan *approval*, sesuai dengan kondisi yang telah ditentukan pada *master data*. Proses tersebut digambarkan pada Gambar 3.5.



Gambar 3.5 Flowchart Proses Create Approval

Pembuatan *approval* dimulai dari pengisian nama dari *approval* tersebut, sebagai bagian dari *metadata*. Lalu, pengguna dapat memilih minimal satu kondisi untuk persetujuan, atau *approval condition*, yang memiliki tiga komponen, yaitu *field* yang nilainya akan di cek, *logical operator* yang akan digunakan untuk mengevaluasi nilai, dan juga *threshold* atau batas nilai dari kondisi tersebut. Lalu, pengguna dapat memilih pengurus atau penanggung jawab yang akan ditugaskan untuk mengevaluasi persetujuan dari formulir yang bersangkutan.

Flowchart Create Approval menjelaskan alur dari laman *Create Approval*. Berikut adalah tampilan dari laman tersebut, yang dapat dilihat di Gambar 3.6.

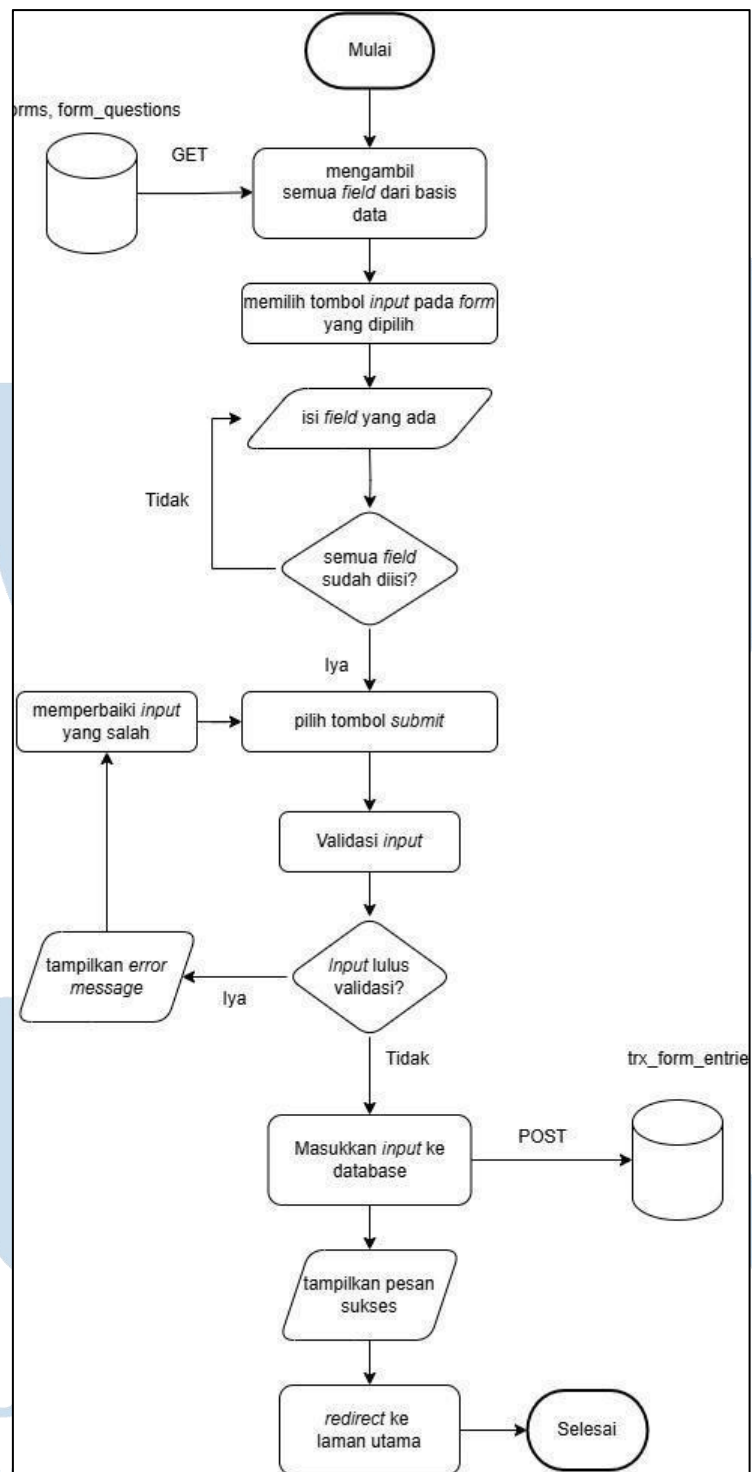
The screenshot shows the 'Create Approval' form. On the left is a sidebar with 'STAR Form' at the top, followed by 'Forms', 'Approval' (selected), and 'My Task'. The main form area has a top bar with a user profile icon and a 'Kembali' button. Below this are three main sections: 'Approval Name' with a text input, 'Form to Approve' with a dropdown menu, and 'Approval Conditions'. The 'Approval Conditions' section contains a table with four columns: 'Field', 'Operator', 'Threshold', and 'Approver'. The 'Field' column has a dropdown menu with 'Select a Field' and a blue 'Select a Field' button. The 'Operator' column has a dropdown menu with '>'. The 'Threshold' column has a text input. The 'Approver' column has a dropdown menu with 'Select Approver Type' and a red 'Remove' button. Below the table is a blue 'Create Approval' button. At the bottom left of the form, it says '2025 © Suryacipta'.

Gambar 3.6 Tampilan Laman *Create Approval*

Laman *Create Approval* memiliki *field* untuk pengisian nama *approval*, *dropdown* untuk memilih formulir yang ingin ditetapkan kondisinya, dan juga *section* pembuatan *approval condition* yang memiliki tiga *input*, yaitu *dropdown* untuk memilih *field* yang akan dievaluasi, *operator* untuk mengevaluasi kondisi, dan batas nilai untuk evaluasi. Pengguna juga dapat menghapus kondisi yang tidak diinginkan.

Untuk mengisi formulir, pengguna akan melewati proses *input form*. Proses ini mencakupi pengisian data formulir oleh pengguna, dan juga pengiriman data tersebut kepada basis data. Alur dari proses ini dapat dilihat secara lebih jelas di Gambar 3.7.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A



Gambar 3.7 Flowchart Proses Input Form

Ketika pengguna telah mengisi semua *field* dalam formulir, maka aplikasi akan melakukan validasi untuk mengecek apakah ada *field* yang kosong, atau memiliki jawaban dengan format yang sesuai dengan yang diminta oleh formulir tersebut. Jika lulus validasi, maka jawaban pengguna akan ditambahkan ke tabel “*trx_entries*” di basis data. Jika tidak, maka pengguna akan diharuskan untuk memperbaiki jawaban hingga lolos validasi.

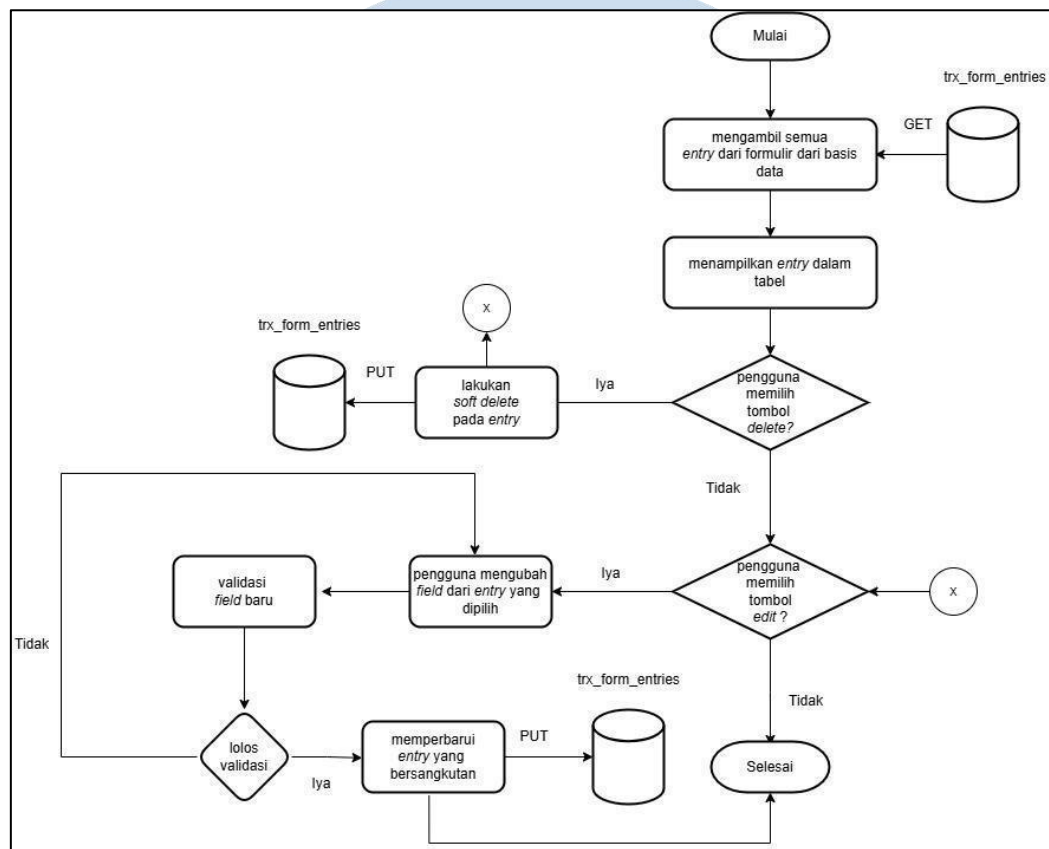
Flowchart Laman *Input Form* menjelaskan alur dari Laman *Input Form*. Hasil dari laman *Input Form* dapat dilihat di Gambar 3.8.

Gambar 3.8 Tampilan Laman *Input Form*

Untuk melihat setiap jawaban atau *entry* dari sebuah formulir, terdapat dua proses yaitu *view data* dan *view entry*. Perbedaan sentral dari kedua proses tersebut adalah *view data* menunjukkan data dalam format tabel, dan *view entry* menunjukkan *data* dalam format *popup*.

Dalam proses display *view data*, data dipersembahkan melalui *request* GET ke basis data. Data lalu disajikan dalam tabel yang dapat dilihat oleh pengguna yang berwenang.

Pengguna yang berwenang juga dapat melakukan *soft delete* atau melakukan modifikasi atau *edit* dari respon tertentu terhadap sebuah formulir, yang berguna jika terjadi kesalahan dalam pengisian formulir, yang dapat dilihat secara lebih detil di Gambar 3.9.



Gambar 3.9 Flowchart Laman View Entry Data

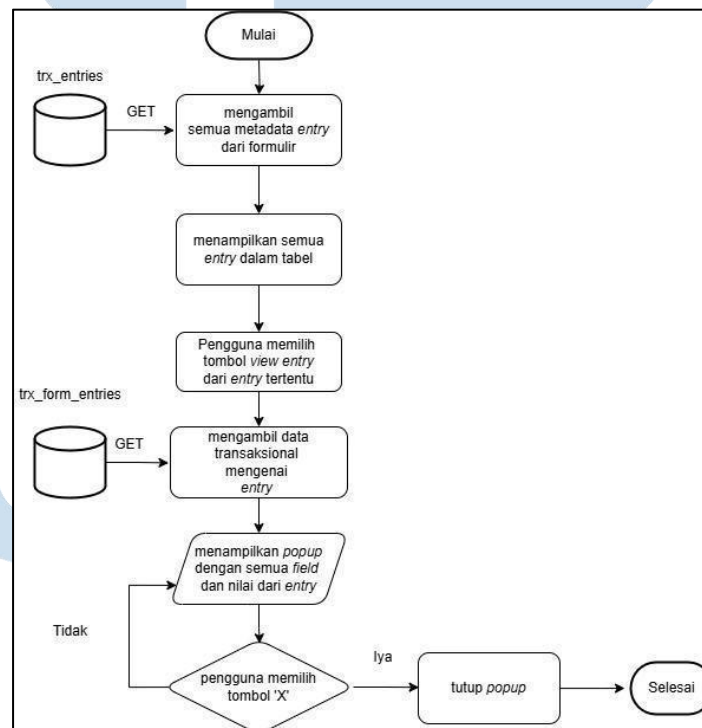
Selain melihat semua data *entry* dari formulir dalam format tabel, Pengguna yang berwenang dapat melakukan penghapusan *entry* yang dipilih, yang akan mengirim *request* POST kepada basis data, lalu melakukan *soft delete* kepada *entry* tersebut dengan mengubah nilai kolom *is_deleted* menjadi “true”. Jika laman dimuat ulang, maka *entry* tersebut akan menghilang dari pandangan pengguna, karena hanya *entry* yang memiliki kolom *is_deleted* dengan nilai “false” yang akan ditampilkan. Pengguna juga dapat melakukan *edit* pada setiap *entry*, dan jika semua *field* yang diubah lulus validasi, maka *entry* tersebut akan diperbarui di basis data.

Flowchart di Gambar 3.8 dan 3.9 menjelaskan alur dari Laman View Data Entry. Hasil dari laman View Data Entry dapat dilihat di Gambar 3.10.

No	Tanggal	Nomor Polisi	Lokasi Tujuan	Jenis Kendaraan	Foto Kendaraan	Foto Surat Jalan	Status	Actions
1	23/05/2025 16:43:00.000		Honda	Truk	Download Image	Download Image	Visitor	Delete
2	23/05/2025 16:41:00.000		Ford	Mobil	Download Image	Download Image	Visitor	Delete
3	23/05/2025 16:40:00.000		BYD	Truk	Download Image	Download Image	Visitor	Delete
4	23/05/2025 16:35:00.000		BYD	Truk	Download Image	Download Image	Visitor	Delete
5	23/05/2025 13:46:00.000		Honda	Mobil	Download Image	Download Image	Visitor	Delete

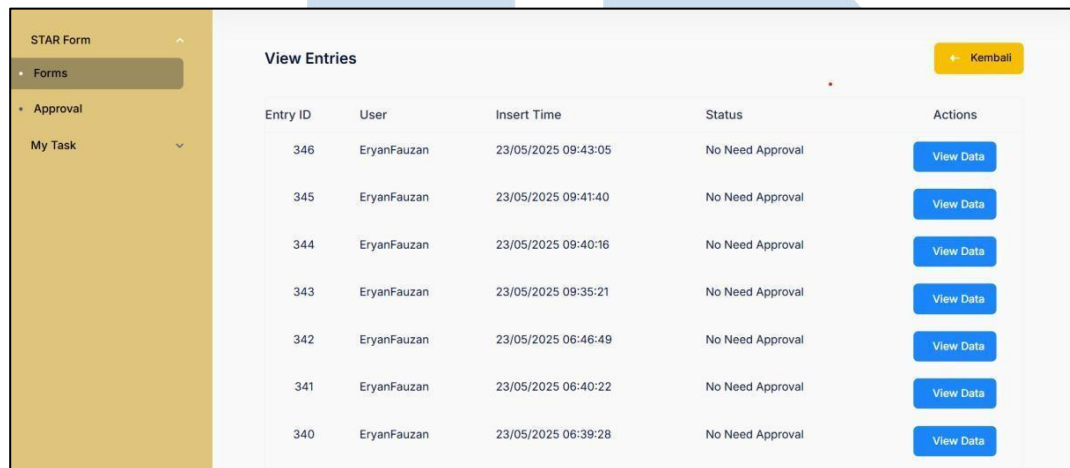
Gambar 3.10 Tampilan Laman *View Entry Data*

Selain proses *View Data Entry*, aplikasi *form generator* memiliki proses *View Entries*. Perbedaan dari kedua proses ini adalah *View Entries* menampilkan informasi mengenai *entry* di dalam sebuah *modal*, bukannya tabel. Ilustrasi dari proses ini dapat dilihat di Gambar 3.11.



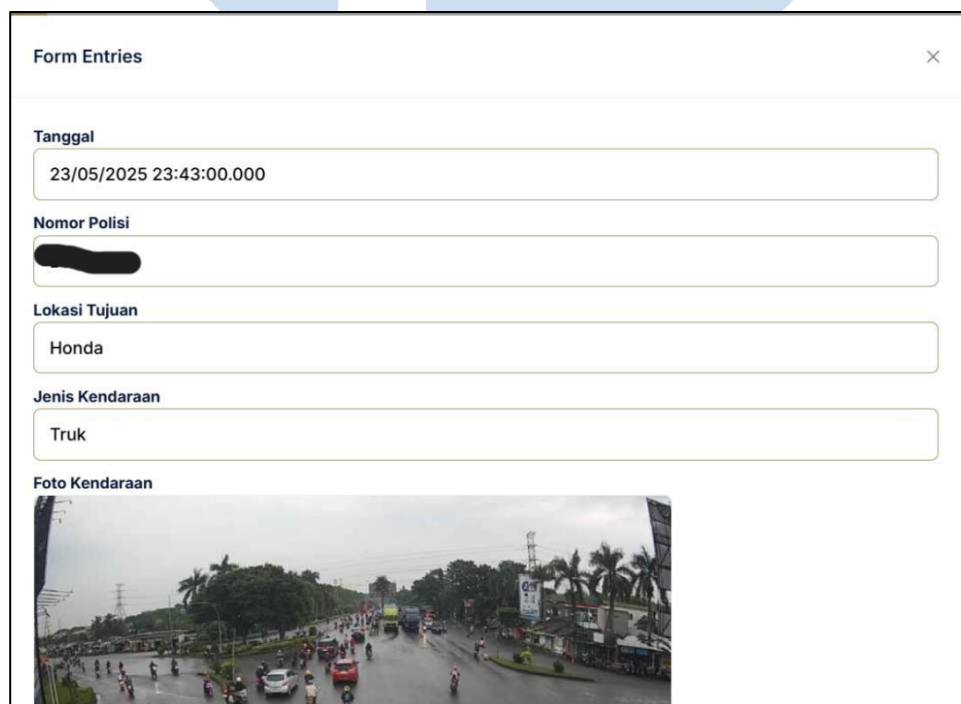
Gambar 3.11 Flowchart Laman *View Entries*

Laman *View Entries* adalah laman yang merepresentasikan proses *View Entries* di dalam aplikasi. Tampilan dari Laman *View Entries* dapat dilihat di Gambar 3.12 dan Gambar 3.13.



Entry ID	User	Insert Time	Status	Actions
346	EryanFauzan	23/05/2025 09:43:05	No Need Approval	View Data
345	EryanFauzan	23/05/2025 09:41:40	No Need Approval	View Data
344	EryanFauzan	23/05/2025 09:40:16	No Need Approval	View Data
343	EryanFauzan	23/05/2025 09:35:21	No Need Approval	View Data
342	EryanFauzan	23/05/2025 06:46:49	No Need Approval	View Data
341	EryanFauzan	23/05/2025 06:40:22	No Need Approval	View Data
340	EryanFauzan	23/05/2025 06:39:28	No Need Approval	View Data

Gambar 3.12 Tampilan Laman *View Entries*



Form Entries

Tanggal
23/05/2025 23:43:00.000

Nomor Polisi
[Redacted]

Lokasi Tujuan
Honda

Jenis Kendaraan
Truk

Foto Kendaraan


Gambar 3.13 Tampilan Modal di Laman *View Entries*

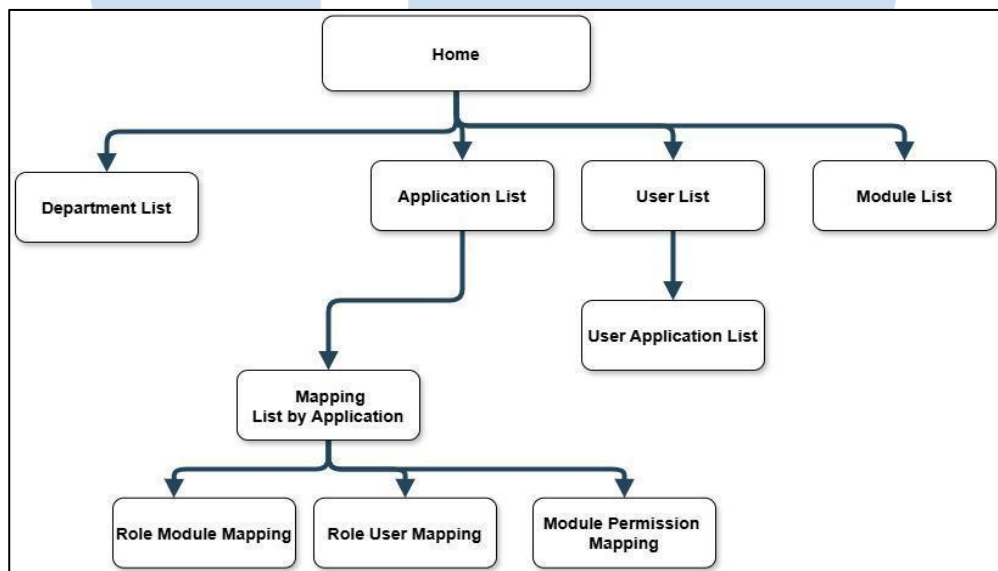
Laman *View Entries* memiliki sebuah tabel dimana *metadata* dan status *approval* dari masing-masing *entry* dapat dilihat. Jika pengguna memilih tombol

View Entries, maka *modal* berisi nilai-nilai untuk setiap *field* dalam *entry* akan muncul.

Dalam laman *outstanding approval*, pengguna yang berwenang dapat melihat nilai dari setiap *entry*, serta mengubah status *entry* tersebut dalam sebuah *popup*, dengan memberikan persetujuan atau ketidaksetujuan terhadap *entry* tersebut. Perubahan akan dikirimkan ke basis data melalui *request* POST.

B) Aplikasi SSO Admin Dashboard

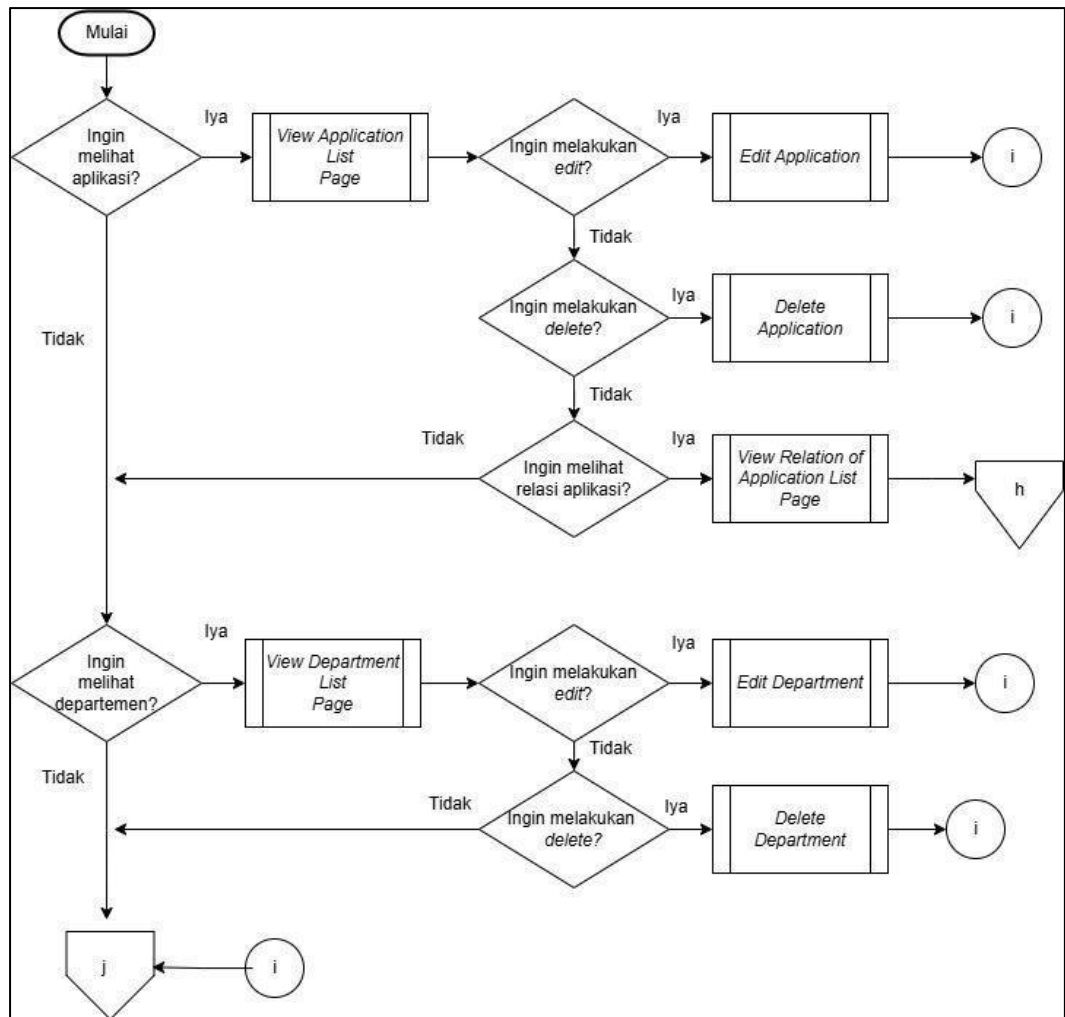
Proses pengembangan dari SSO *Admin Dashboard* meliputi pembuatan *sitemap* dari aplikasi, dan juga *flowchart*. *Sitemap* untuk *Admin Dashboard* dapat dilihat di Gambar 3.14.



Gambar 3.14 Sitemap SSO Admin Dashboard

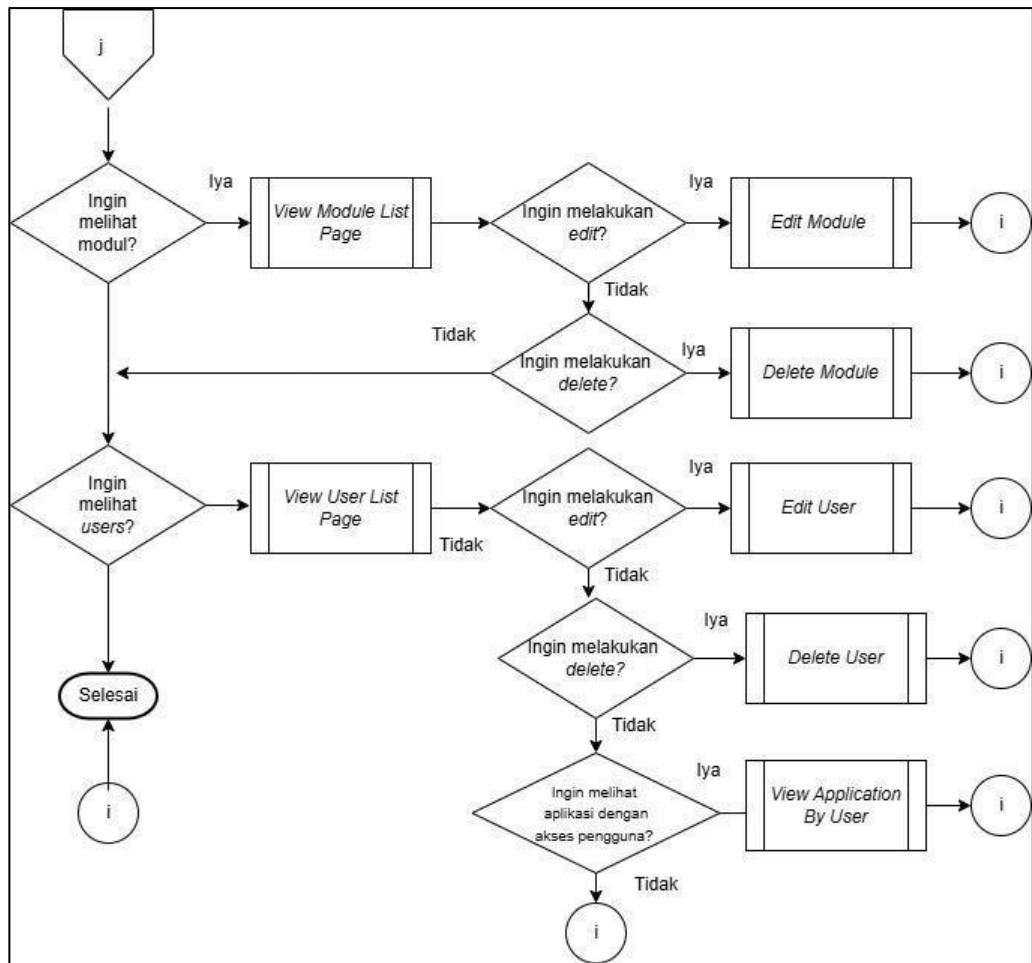
Master Flowchart dari Aplikasi SSO *Admin Dashboard* menjelaskan alur-alur dari setiap proses. *Master Flowchart* dapat dilihat di Gambar 3.15.

UNIVERSITAS
MULTIMEDIA
NUSANTARA

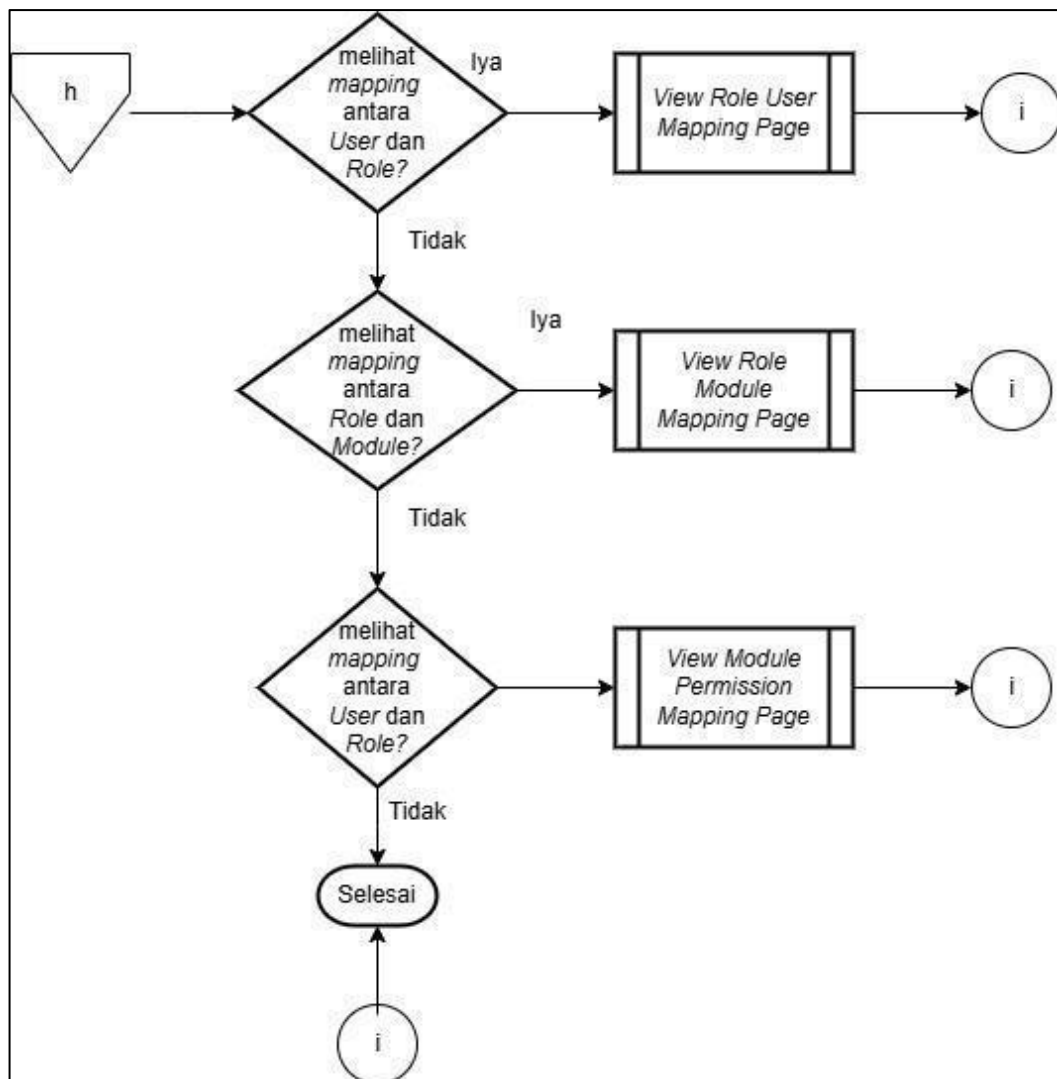


Gambar 3.15 Master Flowchart SSO API Admin Dashboard

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A

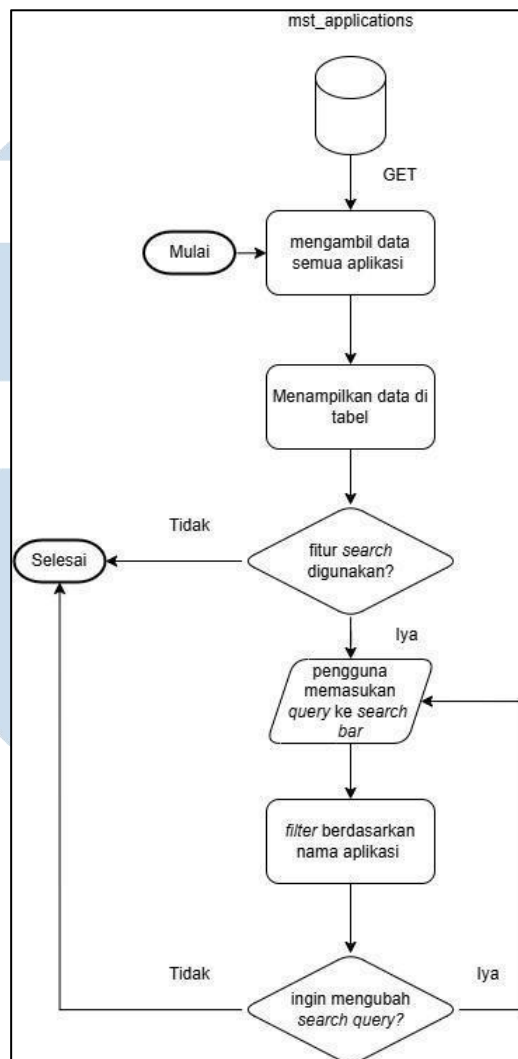


Gambar 3.15 Master Flowchart SSO API Admin Dashboard (lanjutan)



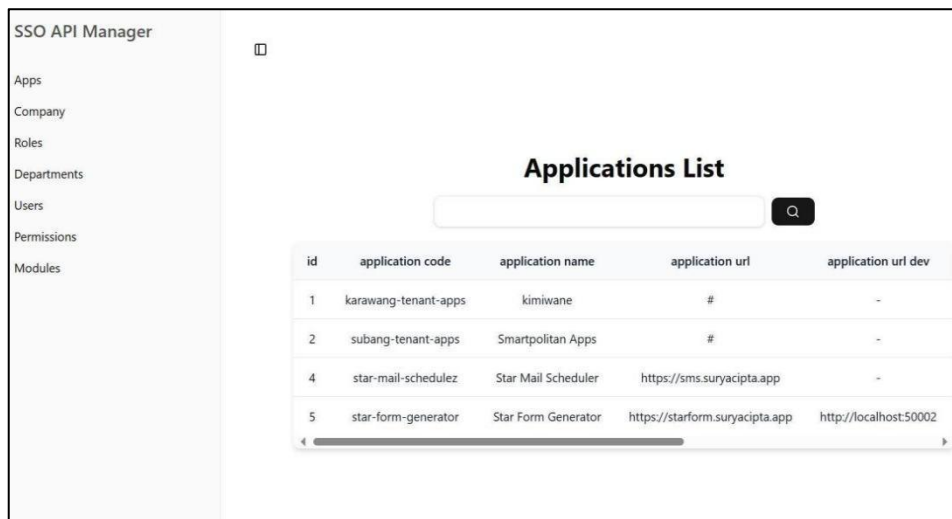
Gambar 3.15 Master Flowchart SSO API Admin Dashboard (lanjutan)

Laman pertama, *Application List* mengambil informasi tentang semua aplikasi dari basis data melalui *request* GET lalu menampilkan informasi tersebut ke dalam tabel. Alur dari proses ini dapat dilihat di Gambar 3.17, yang merupakan *flowchart* untuk laman aplikasi.

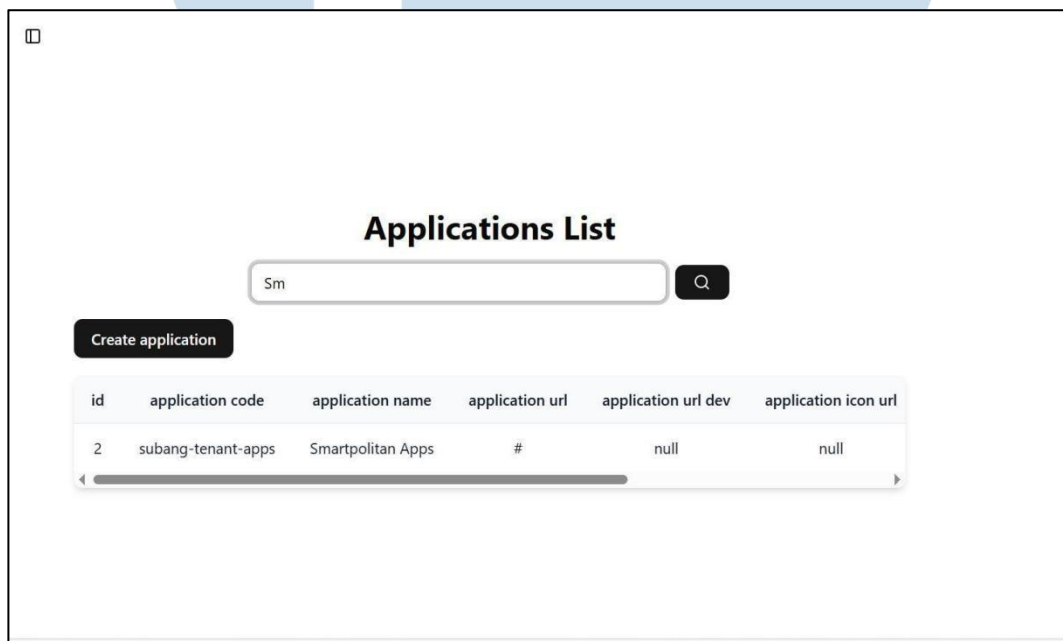


Gambar 3.17 Flowchart Laman *Application List*

Data ditampilkan pada tabel, dan pengguna dapat melakukan *filter* berdasarkan nama aplikasi. Selain itu, pengguna juga dapat membuat aplikasi baru dengan memasukkan nama aplikasi serta kode aplikasi, yang akan dikirimkan sebagai *request* POST ke basis data. Tampilan dari Laman *Application List* dapat dilihat di Gambar 3.18, dan Gambar 3.19.

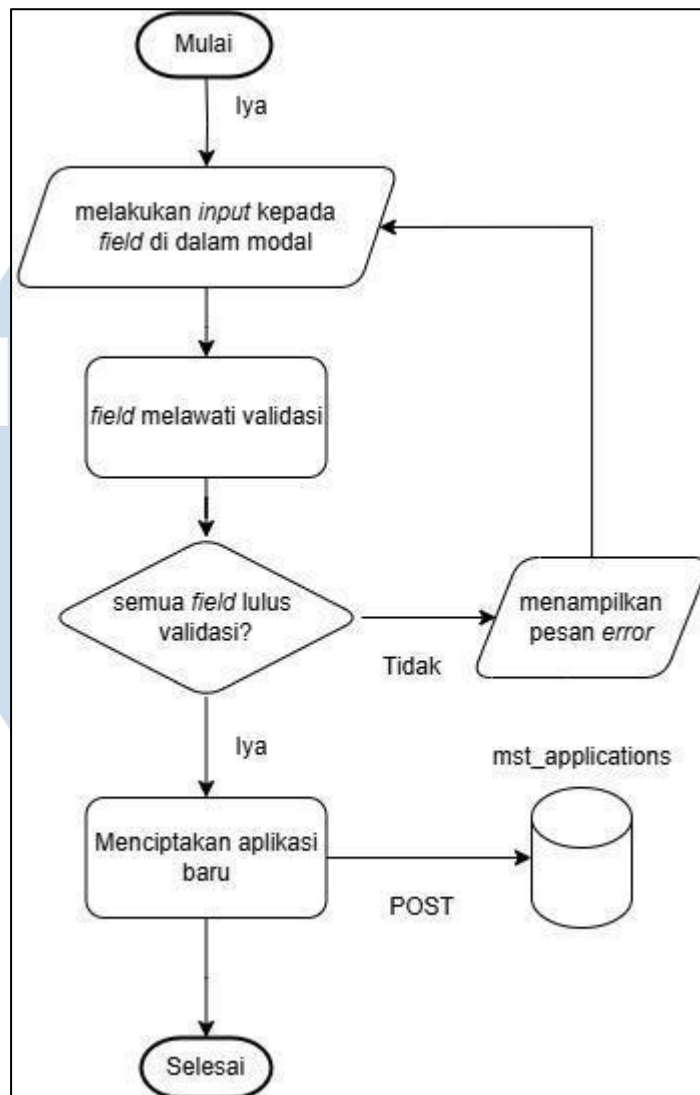


Gambar 3.18 Tampilan Laman *View Application List*



Gambar 3.19 Laman *View Application List* dalam Keadaan *Filtered*

Selain melakukan *search*, pengguna dapat membuat aplikasi baru lewat proses *Create Application*. Alur dari proses ini dapat dilihat di Gambar 3.20.



Gambar 3.20 Flowchart proses Create Application

Jika *input* pengguna lulus validasi, maka *entry* baru akan dibuat di basis data melalui *request* POST. *Modal* yang dipakai untuk proses ini dapat dilihat di Gambar 3.21, dan identik dengan desain *create modal* di laman-laman lain.

UNIVERSITAS
MULTIMEDIA
NUSANTARA

×

Create Entry

Application_name

application_name

Application_code

application_code

Application_url

application_url

Application_url_dev

application_url_dev

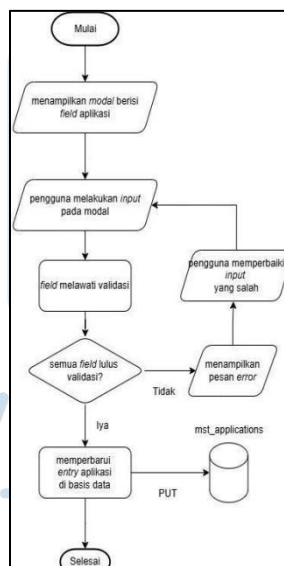
Application_icon_url

application_icon_url

Submit

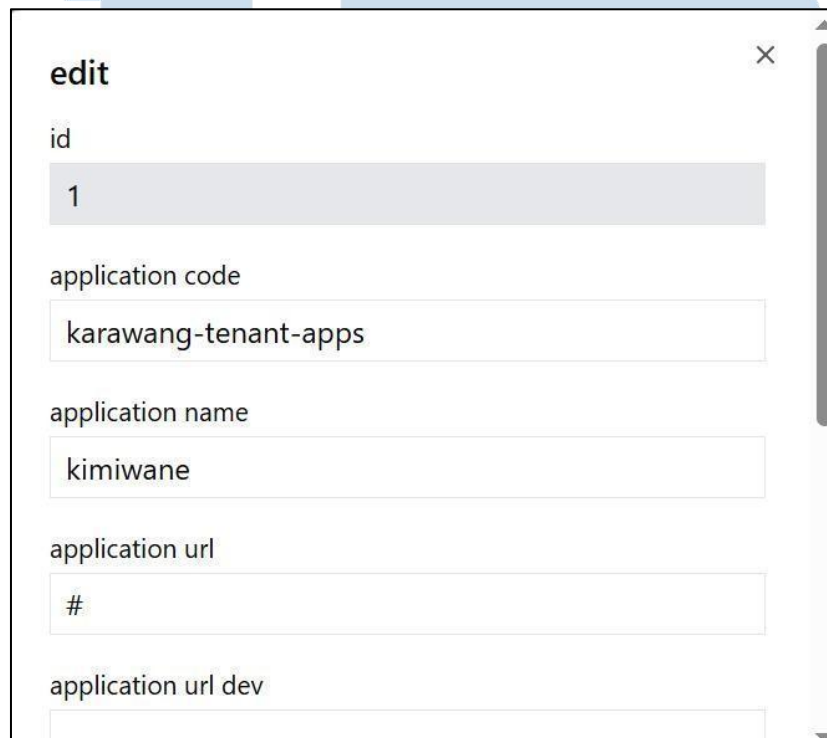
Gambar 3.21 *Modal Create Entry* pada Aplikasi

Pengguna juga dapat melakukan *edit* pada sebuah *entry* aplikasi. Proses ini diilustrasikan pada Gambar 3.22.



Gambar 3.22 *Flowchart proses edit application*

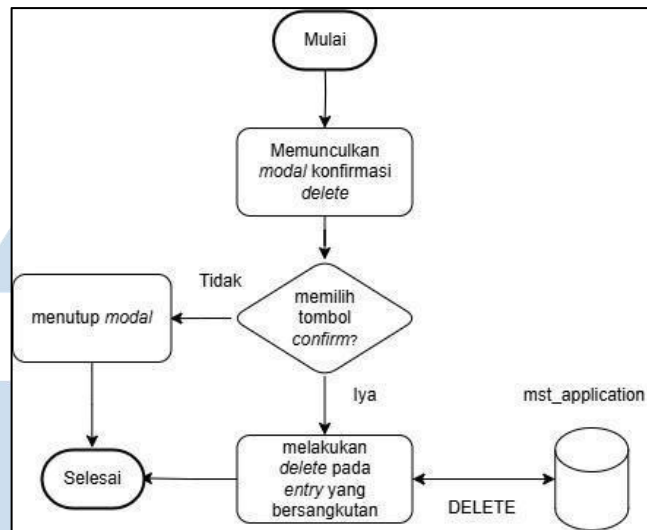
Proses *edit application* menampilkan sebuah *modal* dengan *field* yang sudah diisi dengan nilai *entry*. Pengguna dapat melakukan modifikasi pada nilai-nilai di dalam modal tersebut. Jika hasil akhir lulus validasi, maka *entry* aplikasi akan diperbarui di basis data melalui sebuah *request* PUT. Tampilan dari *modal edit application* dapat dilihat di Gambar 3.23.



Gambar 3.23 Tampilan Modal *Edit Application*

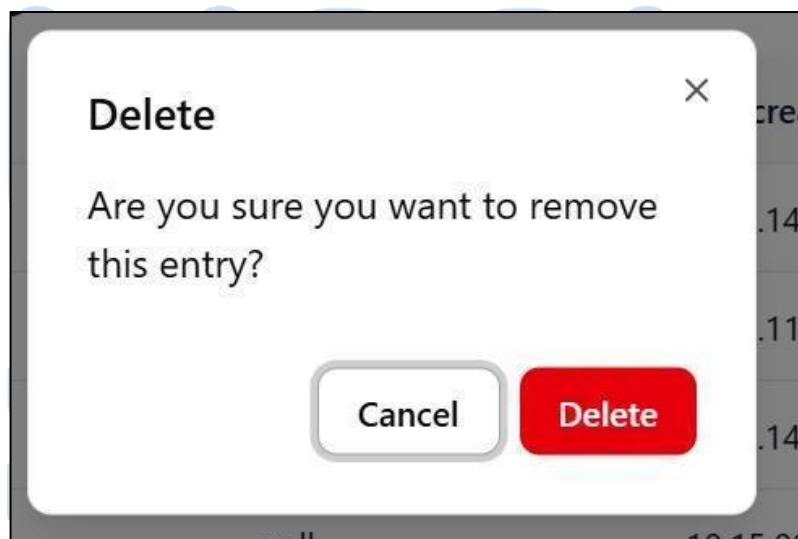
Pengguna juga dapat melakukan *delete* pada aplikasi. Alur dari proses *delete application* dapat dilihat di Gambar 3.23.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A



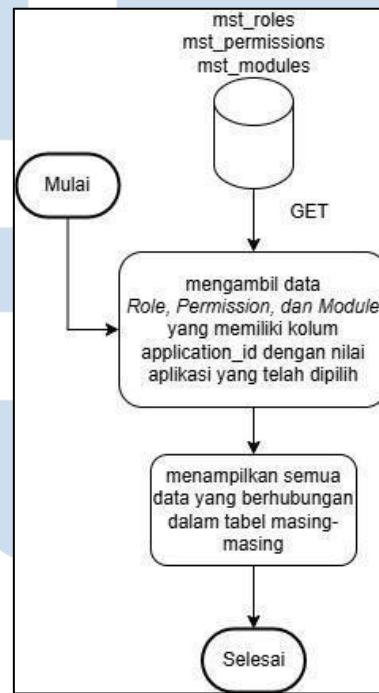
Gambar 3.23 Flowchart Proses Delete Application

Proses *Delete Application* akan menampilkan sebuah *modal* dengan dua tombol, yaitu tombol *Delete* dan tombol *Cancel*. Jika tombol *Cancel* dipilih, maka *modal* akan ditutup. Namun, jika tombol *Delete* dipilih, maka aplikasi akan mengirimkan request *DELETE* ke basis data dan menghapus *entry* yang bersangkutan. Tampilan dari *modal Delete Application* ini dapat dilihat di Gambar 3.24.



Gambar 3.24 Tampilan Modal Delete Application

Ketika salah satu *entry* dari tabel aplikasi dipilih, maka pengguna akan dikirimkan ke laman Relasi Aplikasi atau *Relations of Applications*. Flowchart untuk proses untuk penyajian data di laman *Relations of Application* dapat dilihat di Gambar 3.25.



Gambar 3.25 Flowchart *View Relations of Application*

View Relations of Application List merupakan proses dimana pengguna dapat melihat data dari tabel-tabel yang memiliki kolom “application_id” di basis data, yaitu *Module*, *Role*, dan juga *Permission*. Tampilan untuk laman *Relation of Application* dapat dilihat di Gambar 3.26.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A

Create Modules						
id	application id	module code	module name	createdAt	updatedAt	Actions
22	2	modtes	modul tes	17:19:20, 4/6/2025	17:19:20, 4/6/2025	

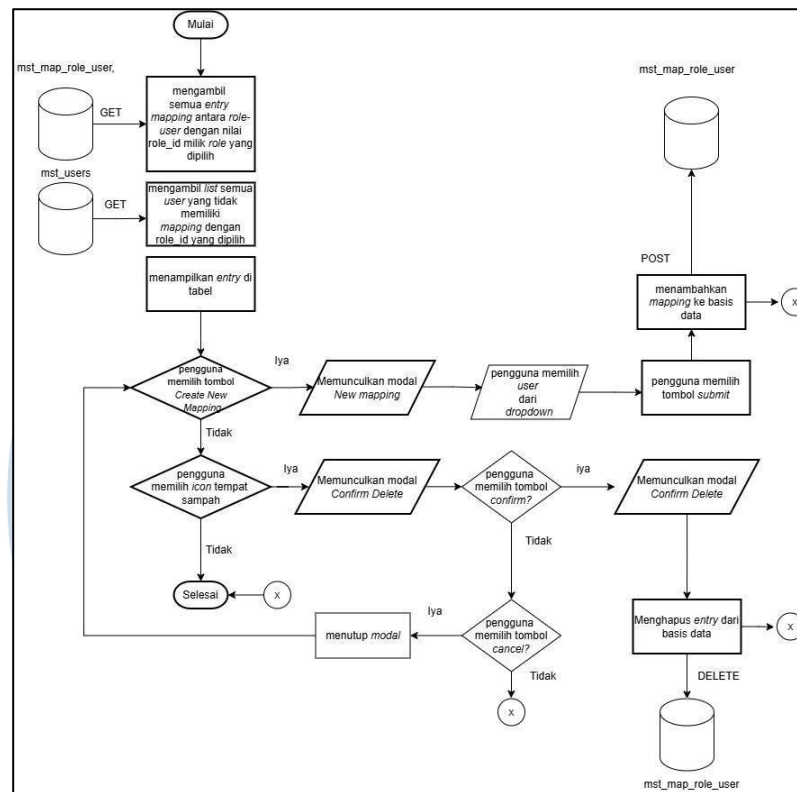
Create Roles				
id	application id	role code	role name	
2	2	SMARTPOLITAN_APPS_IT_MANAGER	Akses IT Manager untuk SMARTPOLITAN Apps	13
19	2	wkwkwkw	hey man	1
18	2	eefef	testroletz	21

Gambar 3.26 Tampilan Laman *Relation of Application List*

Jika pengguna memilih tombol *create* yang berada di atas masing-masing tabel, maka sebuah *modal* yang memiliki *field* nama dan kode akan muncul. Alur dari proses ini sama dengan alur yang dijelaskan di *flowchart Create Entry*.

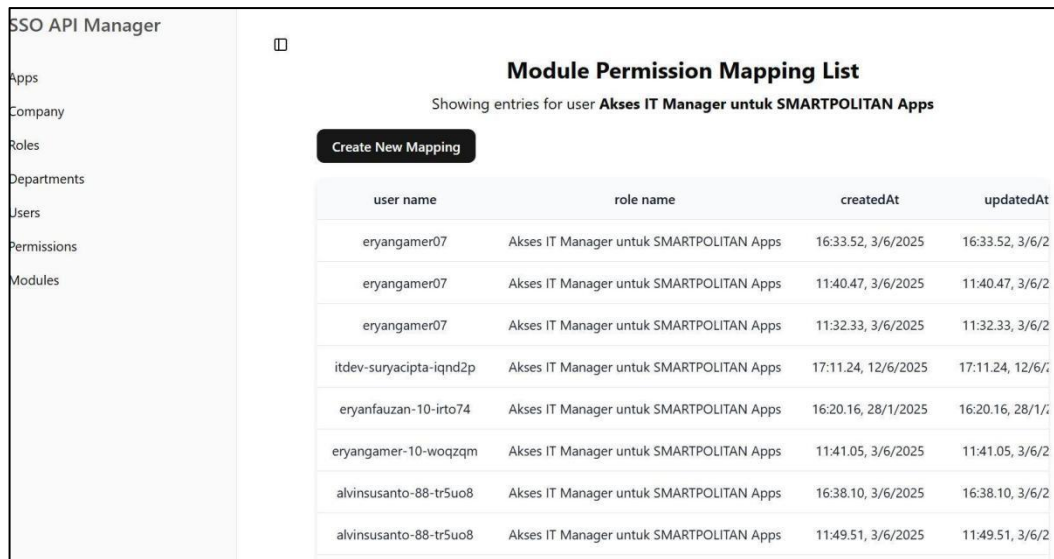
Jika pengguna memilih tombol *Add User* pada suatu *entry* di tabel *Roles*, maka proses *View Role User Mapping* akan mulai. Alur dari proses *View Role User Mapping* dapat dilihat di Gambar 3.27.





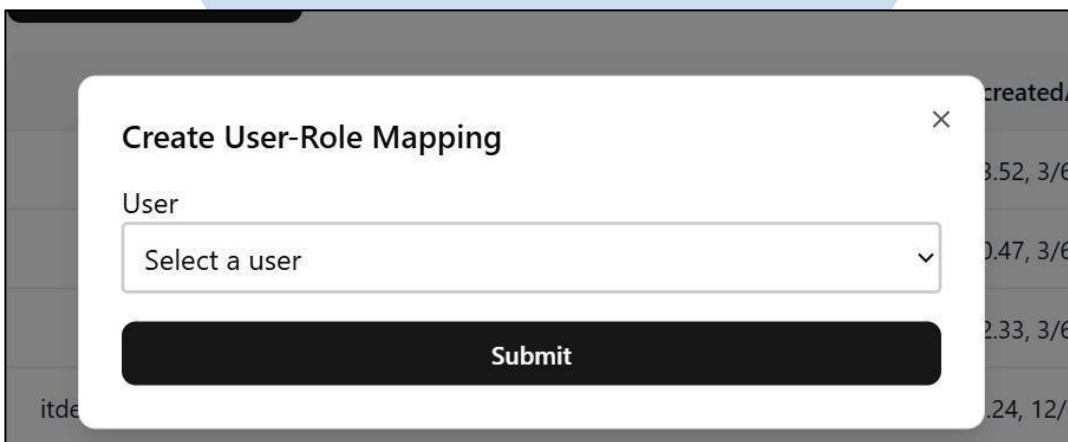
Gambar 3.27 Flowchart View Role User Mapping

Setelah aplikasi mengambil data dari tabel *mapping* dan pengguna sudah melihat data *role-user* mapping, pengguna dapat membuat *mapping* baru antara *role* (yang merupakan *entry* dimana pengguna memilih tombol *Add User*) dan *user*, yang dapat dipilih di dalam sebuah *dropdown* yang berisi semua *user* yang belum memiliki *mapping* dengan *role* yang telah ditentukan. Tampilan dari Laman *Role User Mapping* dapat dilihat di Gambar 3.28, dan *modal* dapat dilihat di Gambar 3.29.



user name	role name	createdAt	updatedAt
eryangamer07	Akses IT Manager untuk SMARTPOLITAN Apps	16:33.52, 3/6/2025	16:33.52, 3/6/2
eryangamer07	Akses IT Manager untuk SMARTPOLITAN Apps	11:40.47, 3/6/2025	11:40.47, 3/6/2
eryangamer07	Akses IT Manager untuk SMARTPOLITAN Apps	11:32.33, 3/6/2025	11:32.33, 3/6/2
itdev-suryacipta-iqnd2p	Akses IT Manager untuk SMARTPOLITAN Apps	17:11.24, 12/6/2025	17:11.24, 12/6/
eryanfauzan-10-irto74	Akses IT Manager untuk SMARTPOLITAN Apps	16:20.16, 28/1/2025	16:20.16, 28/1/
eryangamer-10-woqzqm	Akses IT Manager untuk SMARTPOLITAN Apps	11:41.05, 3/6/2025	11:41.05, 3/6/2
alvinsusanto-88-tr5uo8	Akses IT Manager untuk SMARTPOLITAN Apps	16:38.10, 3/6/2025	16:38.10, 3/6/2
alvinsusanto-88-tr5uo8	Akses IT Manager untuk SMARTPOLITAN Apps	11:49.51, 3/6/2025	11:49.51, 3/6/2

Gambar 3.29 Tampilan Laman *View Role User Mapping List*



Create User-Role Mapping

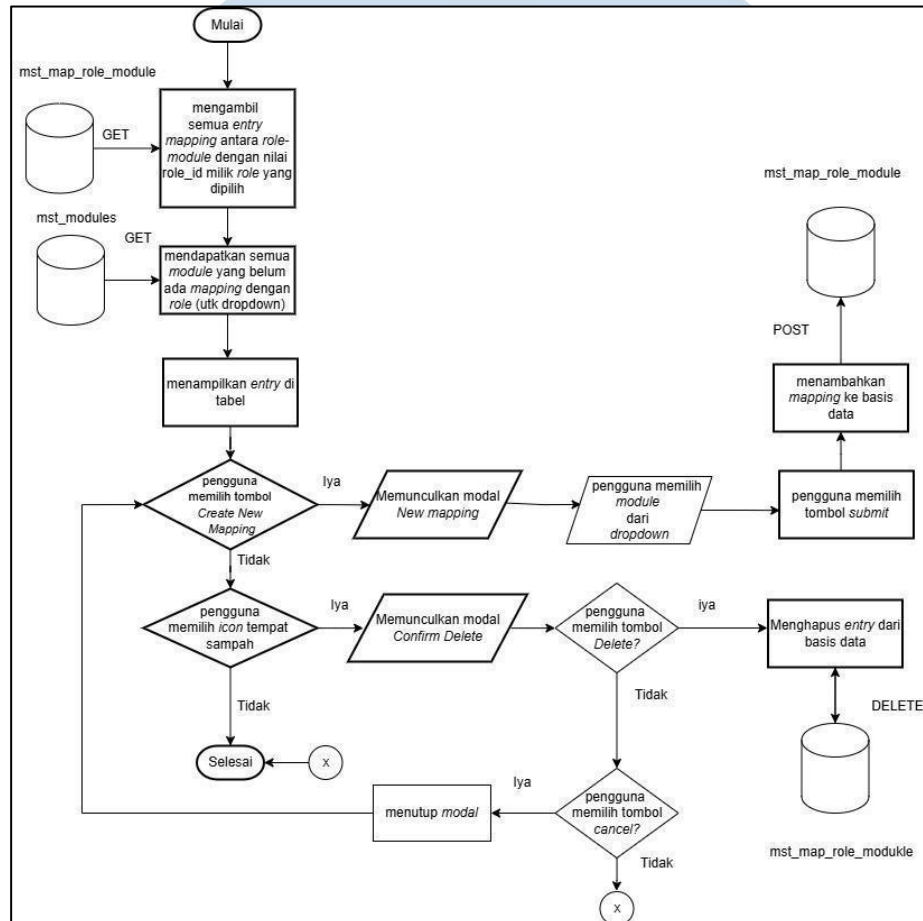
User

Select a user

Submit

Gambar 3.30 Tampilan *Modal Create New Mapping* di Laman *Role User Mapping List*

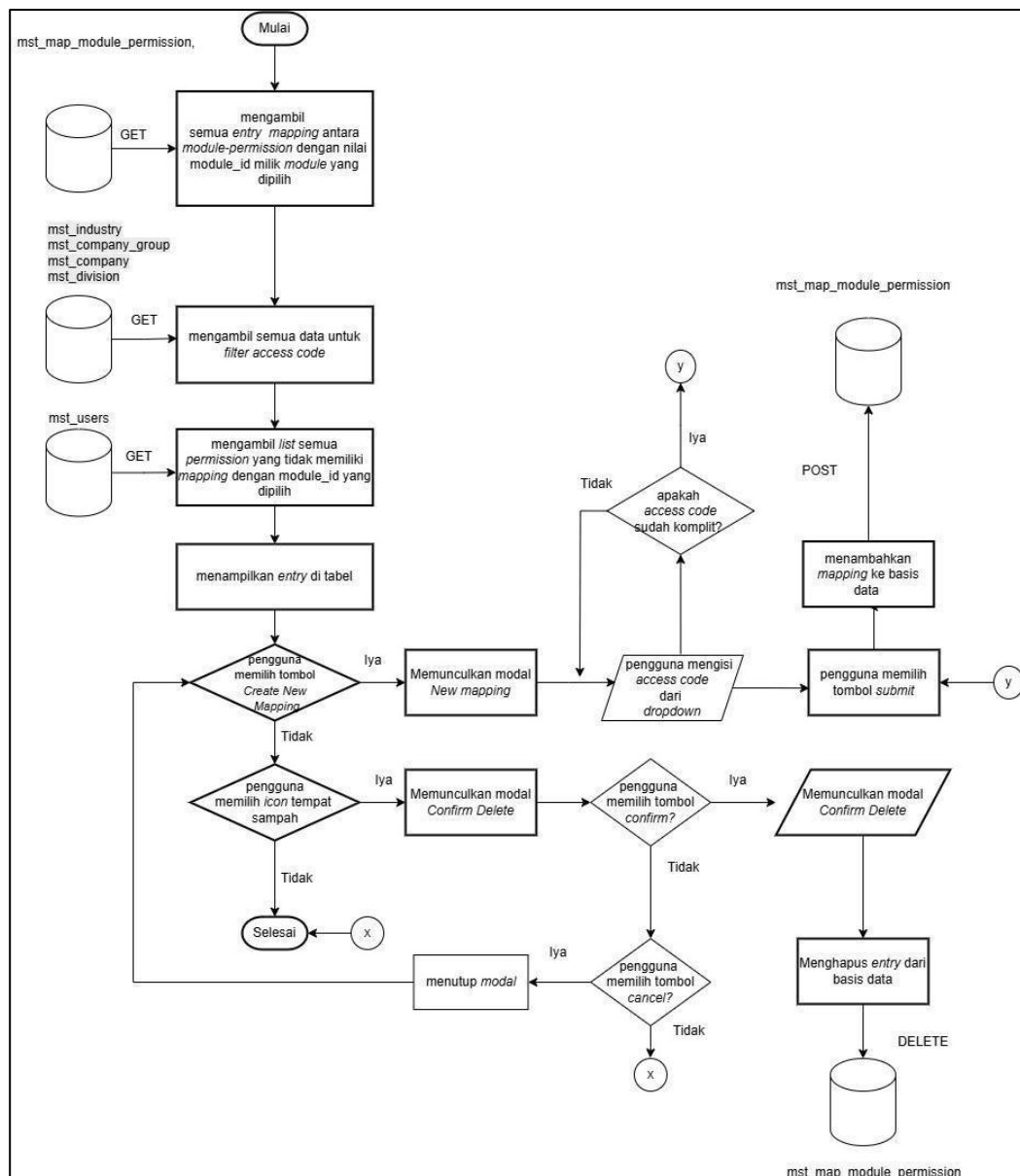
Selain pilihan *Add User*, pengguna juga dapat memilih *Add Module* untuk menambahkan *mapping* antara *Role* dan *Module*. Alur dari proses ini dideskripsikan di *Flowchart View Role Module Mapping*, yang dapat dilihat di Gambar 3.31.



Gambar 3.31 Flowchart View Role Module Mapping

Alur dari proses *View Role Module Mapping* kurang lebih sama dengan *View Role User Mapping*. Perbedaan terletak dalam data yang diambil, karena proses *View Role Module Mapping* mengambil data *Role* serta *Module*.

Terakhir, jika pengguna memilih tombol *Add Permission* di dalam tabel *Modules*, maka proses *View Module Permission Mapping* akan mulai dan pengguna akan di-redirect ke laman *Module Permission Mapping List*. Alur dari *View Module Permission Mapping* dapat dilihat di Gambar 3.32.

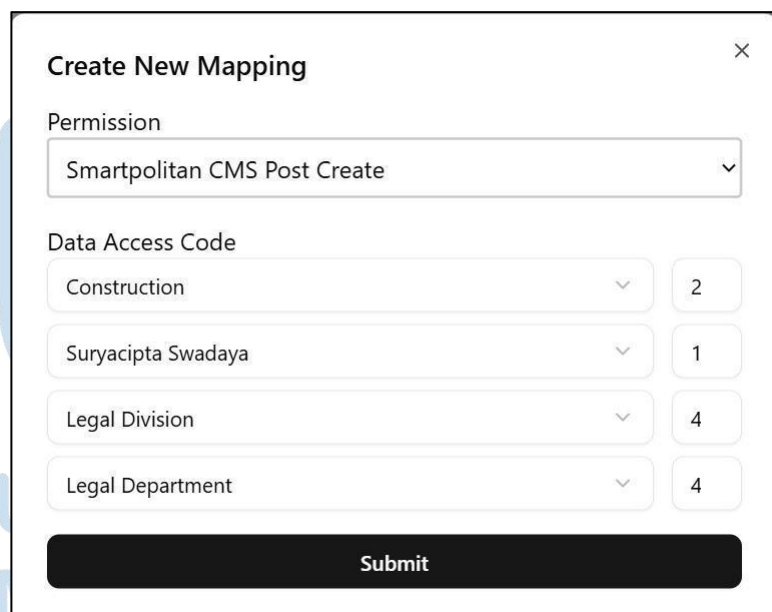


Gambar 3.32 Flowchart View Module Permission Mapping

Perbedaan sentral dari proses ini dibandingkan proses *mapping* lain adalah mekanisme pengisian *access code* pada sistem. *Access code* terdiri dari empat *digit*, yang masing-masing mewakili satu tingkatan dalam struktur organisasi. Setiap *digit* disimpan secara terpisah dalam kolom “*access_data_code*” di tabel terkait. Penjelasan tiap *digit* sebagai berikut:

1. *Digit* pertama: Menunjukkan tipe industri
 - a) Contoh : Perhotelan (kode 1), Konstruksi (kode 2)
2. *Digit* kedua : menunjukkan perusahaan yang berada di bawah industri yang telah dipilih
 - a) Contoh : Suryacipta Swadaya (kode 1), Surya Internusa (kode 2)
3. *Digit* ketiga : menunjukkan divisi yang berada di bawah perusahaan yang dipilih
 - a) Contoh: *IT Development* (kode 3), *Legal Division* (kode 4)
4. *Digit* keempat : menunjukkan departemen yang berada di bawah divisi yang dipilih
 - a) Contoh : *IT Department* (kode 4), *Legal Department* (kode 4)

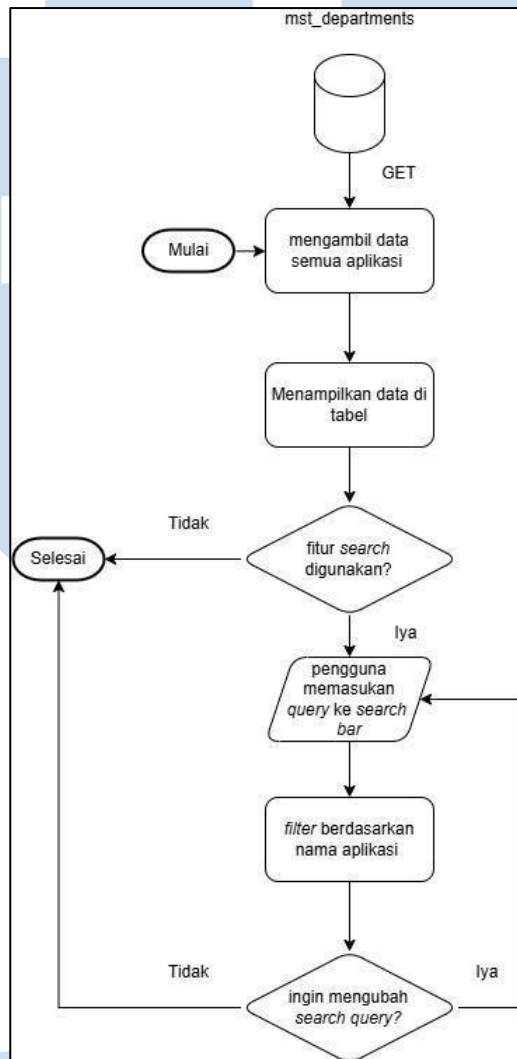
Karena *input access code* ini bersifat hierarkis dan masing-masing *digit* bergantung dengan *digit* sebelumnya, maka jika sebuah *digit* dihapus, maka semua *digit* yang datang setelahnya akan ikut terhapus juga. Tampilan dari *modal Create Module Permission Mapping* dapat dilihat di Gambar 3.33.



Data Access Code	
Construction	2
Suryacipta Swadaya	1
Legal Division	4
Legal Department	4

Gambar 3.33 *Modal Create New Module Permission Mapping*

Jika pengguna ingin melihat departemen yang ada, maka aplikasi akan melaksanakan proses *View Departments List*. *Flowchart* dari proses tersebut dapat dilihat di Gambar 3.34.



Gambar 3.34 *Flowchart View Departments List*

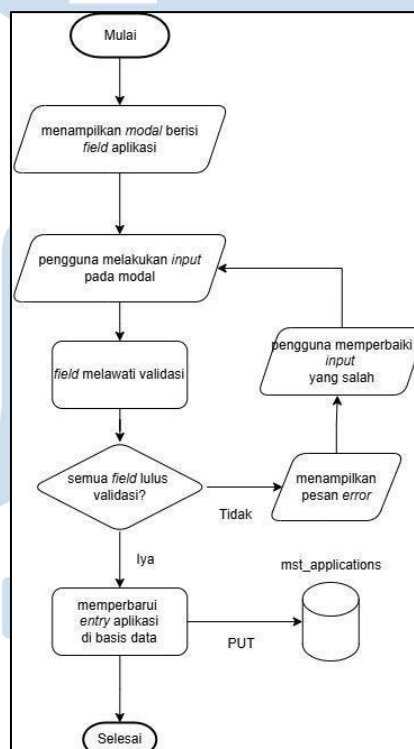
Alur dari proses ini cukup mirip dengan alur proses *View Application List*, namun dengan perbedaan bahwa data yang diambil adalah dari tabel *mst_departments*. Tampilan dari *Departments List* dapat dilihat di Gambar 3.35.

SSO API Manager	
Apps	
Company	
Roles	
Departments	
Users	
Permissions	
Modules	

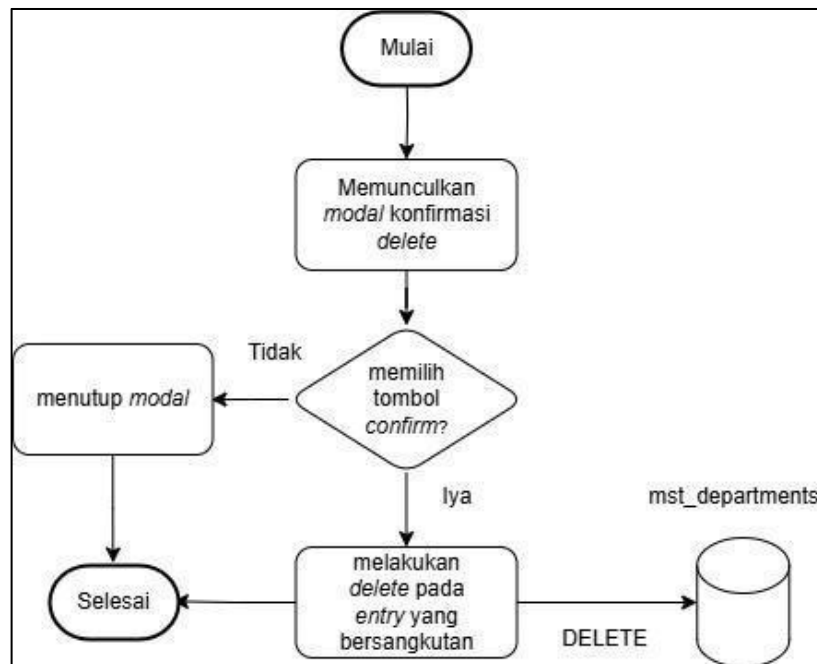
Departments List				
id	department name	createdAt	updatedAt	Actions
1	IT Development	09:38.01, 17/1/2025	09:38.01, 17/1/2025	⋮
3	Tenant Relation	09:38.01, 17/1/2025	09:38.01, 17/1/2025	⋮
4	Legal Department	09:38.01, 17/1/2025	09:38.01, 17/1/2025	⋮
5	Securitys	09:38.01, 17/1/2025	14:59.21, 2/6/2025	⋮
7	Project Management	09:47.00, 26/5/2025	09:47.00, 26/5/2025	⋮

Gambar 3.35 Tampilan Laman *Departments List*

Sama seperti di laman *View Application*, pengguna juga dapat melakukan *edit* dan *delete* pada *entry* di tabel. Alur untuk proses *Edit Department* dan *Delete Department* dapat dilihat di Gambar 3.36 dan Gambar 3.37.



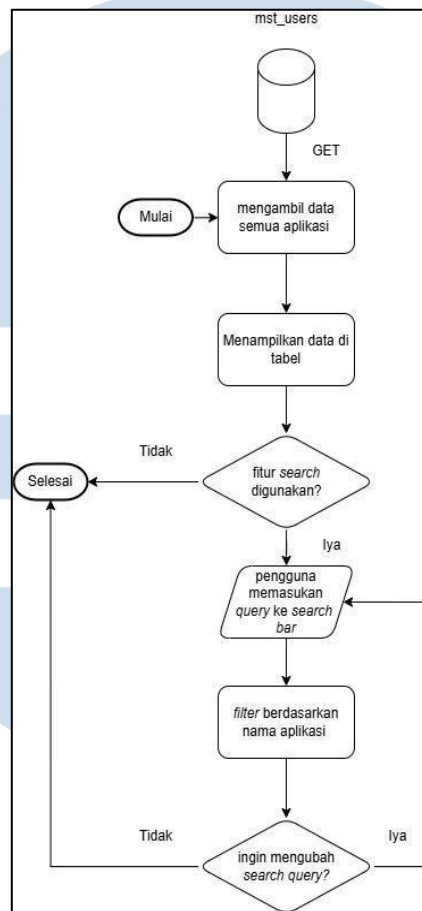
Gambar 3.36 Flowchart Proses *Edit Department*



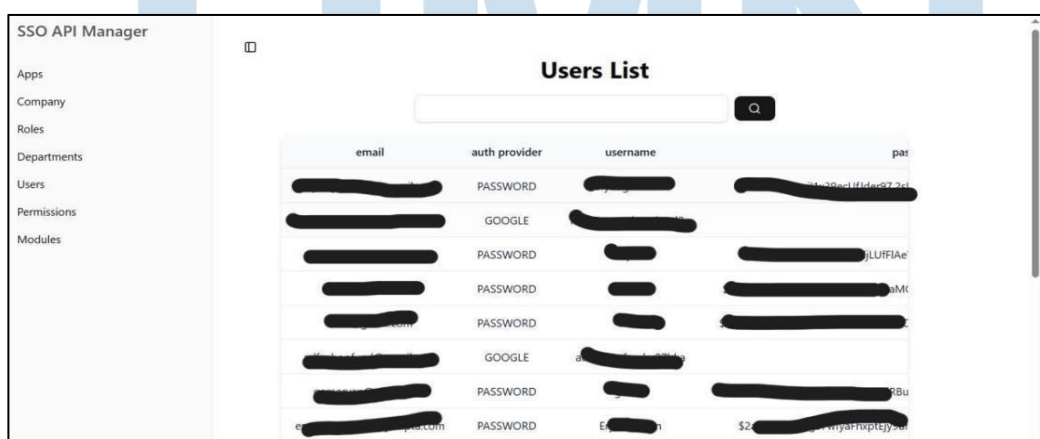
Gambar 3.37 Flowchart Proses Delete Department

Jika pengguna memilih tabel *Users*, maka aplikasi akan memulai proses *View User List*. Alur dari proses *View Users List* dapat dilihat di Gambar 3.38, dan tampilan dari laman *Users List* dapat dilihat di Gambar 3.39.



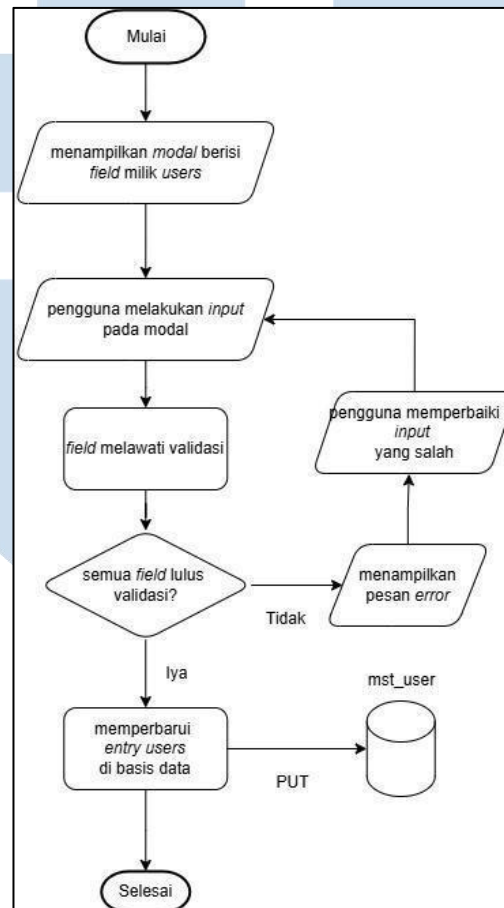


Gambar 3.38 Flowchart View Users List

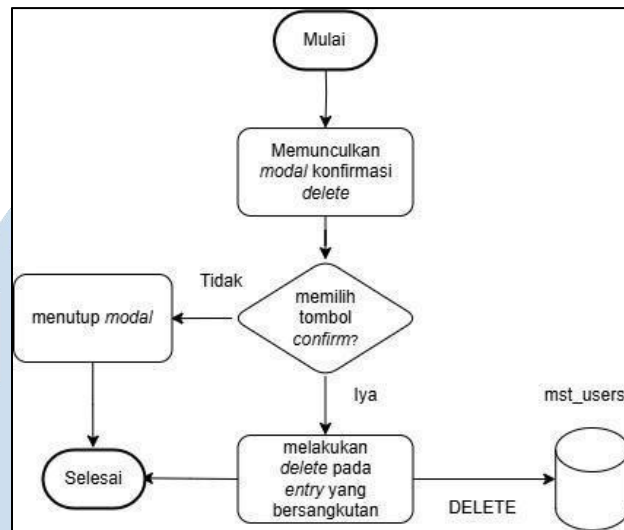


Gambar 3.39 Tampilan Laman Users List

Identik dengan proses *View Application List* dan *View Departments List*, pengguna juga dapat melakukan proses *Edit User* dan *Delete User* terhadap tabel *user*, yang dapat dilihat di Gambar 3.40 dan Gambar 3.41.



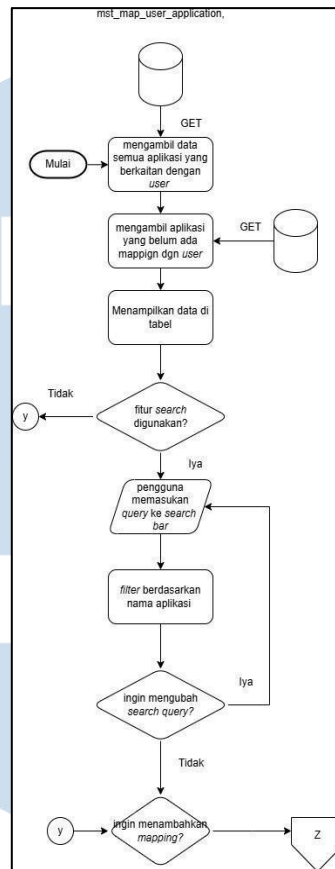
Gambar 3.40 Flowchart Edit User



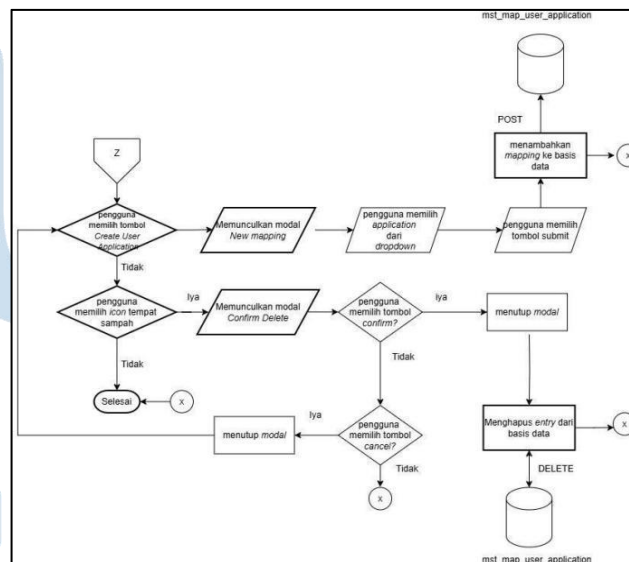
Gambar 3.41 Flowchart Delete User

Jika pengguna memilih sebuah *entry* dalam tabel *Users*, maka akan di-*redirect* ke laman *User Applications List*. Proses ini bernama *View Applications By User*. Alur dari proses ini dapat dilihat di Gambar 3.42.

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

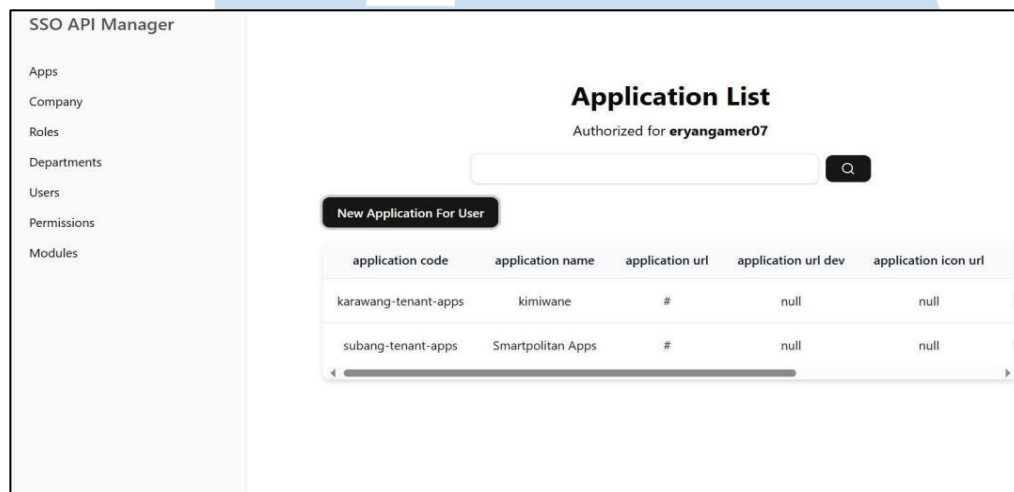


Gambar 3.42 Flowchart View Application By User

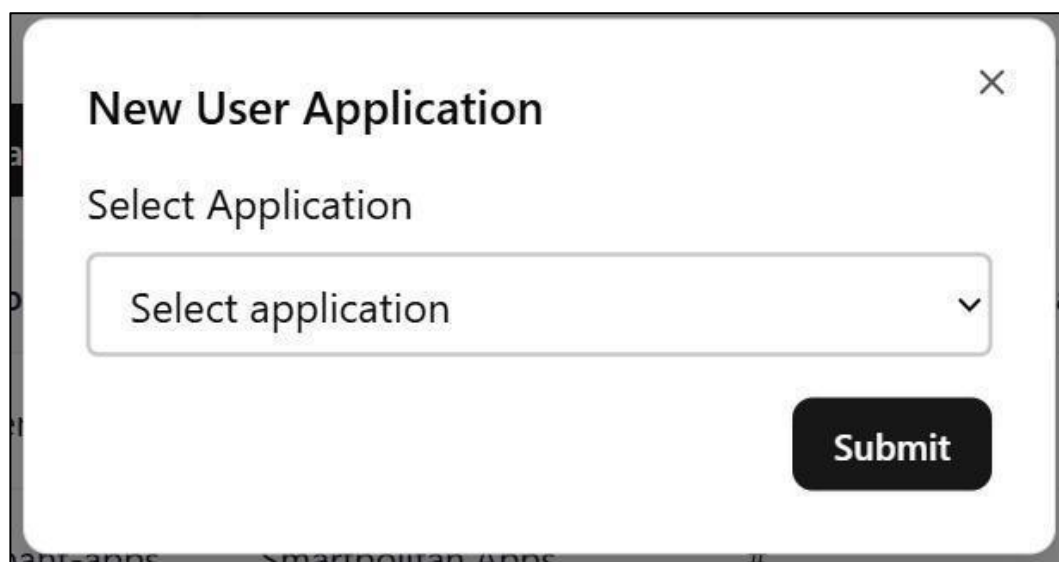


Gambar 3.42 Flowchart View Application By User (lanjutan)

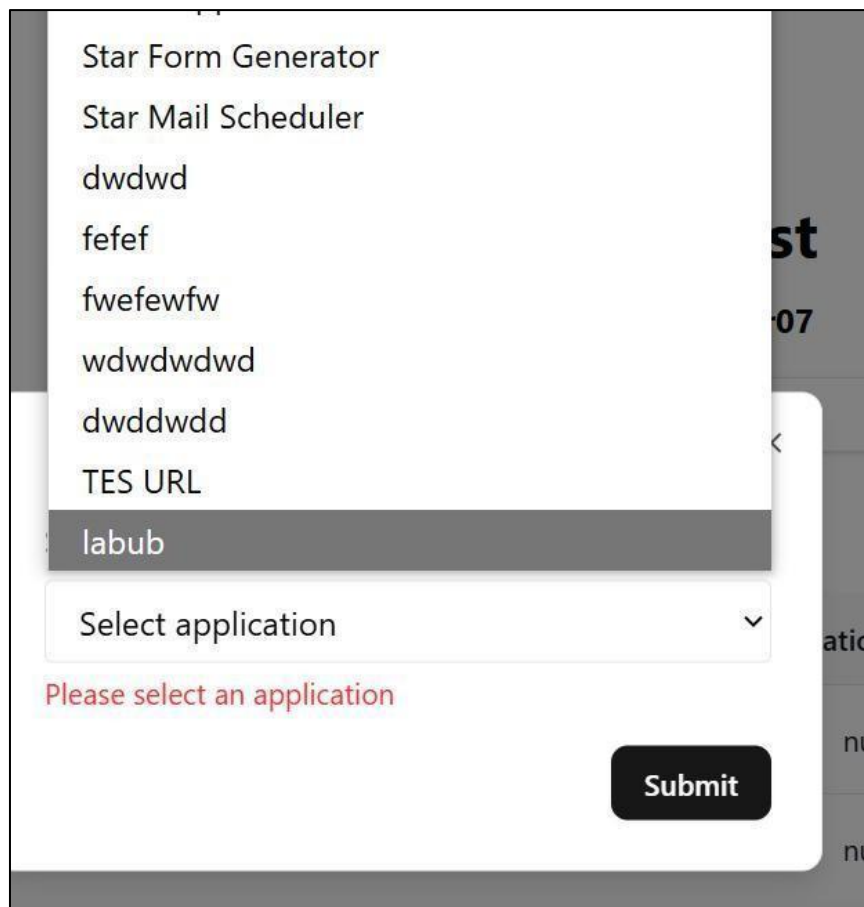
View User Application merupakan sebuah proses yang menyajikan *mapping* antara dua tabel, yaitu *User* dan *Application*. Namun, tidak seperti laman *mapping* lainnya, pengguna dapat melakukan *filter* pada nama dari aplikasi, untuk mendapatkan aplikasi spesifik sesuai dengan *query*. Tampilan dari laman dapat dilihat di Gambar 3.43 hingga 3.45.



Gambar 3.43 Tampilan Laman *User Application List* (lanjutan)



Gambar 3.44 Tampilan *Modal Create New User Application List*

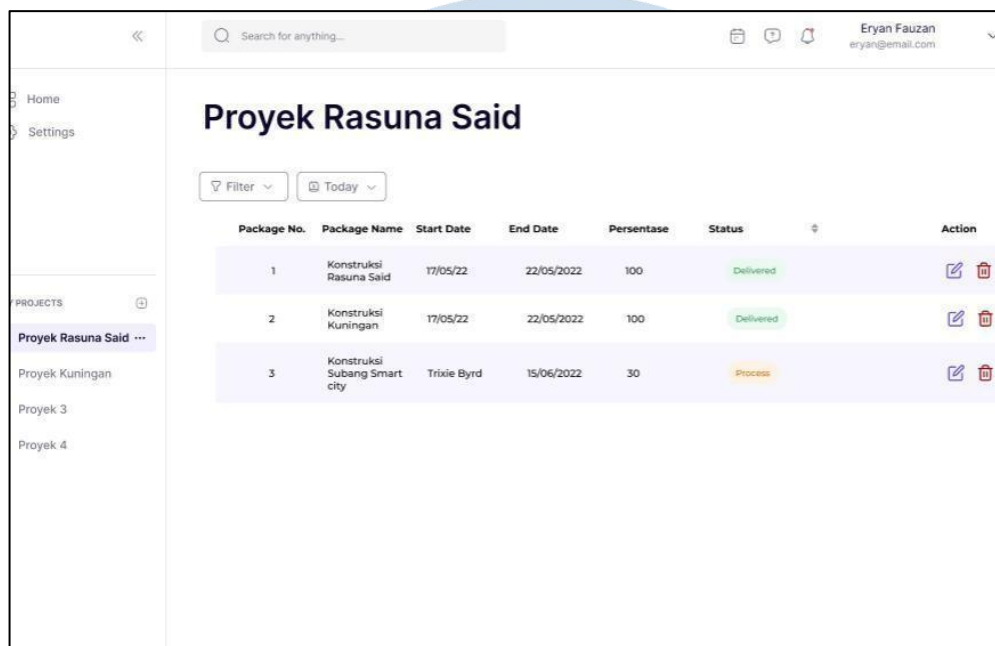


Gambar 3.45 Tampilan Modal Create New User Application List (focused)

C) Aplikasi Progress Logger

Aplikasi *progress logger* memiliki sebuah *draft* desain berupa mockup. Desain tersebut dibuat menggunakan aplikasi Figma, dan mencakupi desain kasar untuk laman *View Packages* dan *View Task*. Namun, karena pertimbangan berdasarkan efisiensi, waktu pengerjaan, dan antisipasi untuk kenaikan skala dalam proyek, maka desain tidak jadi dipakai, namun tampilan aplikasi dibuat menggunakan sebuah *template* berbasis HTML.

Laman *View Packages* adalah laman dimana paket pengerjaan proyek konstruksi dapat dilihat secara garis besar. *Mockup* untuk *View Packages* dapat dilihat di Gambar 3.46.

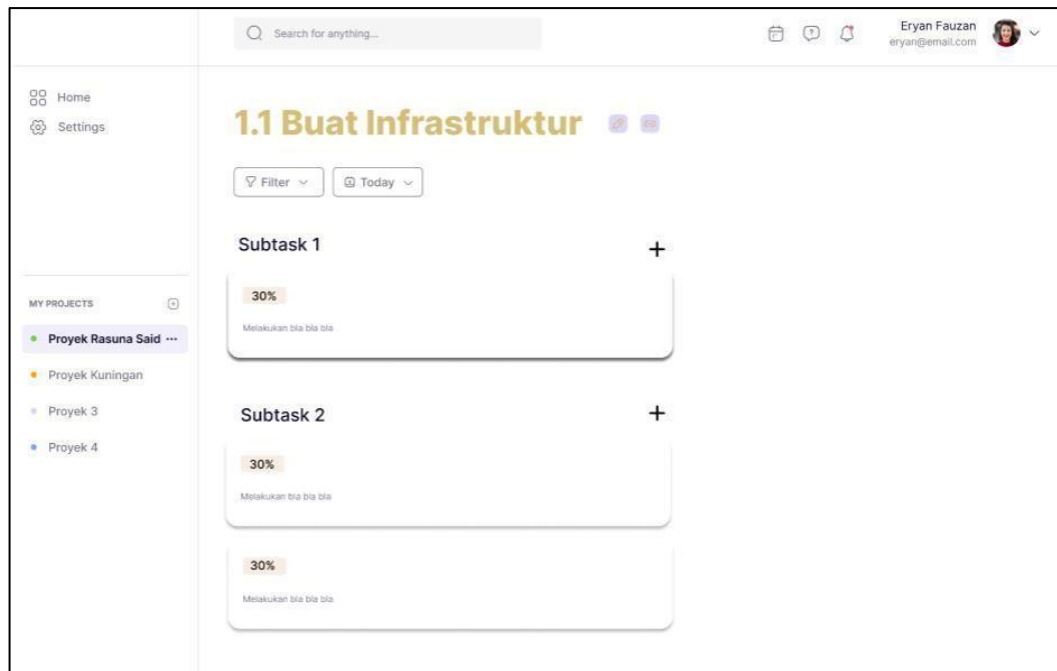


Gambar 3.46 Mockup Laman View Packages

Mockup untuk laman ini memiliki tabel di mana setiap paket pengerjaan ditampilkan, serta *dropdown* untuk menampilkan *filter* yang ada. Laman ini juga memiliki *sidebar* yang dapat digunakan untuk navigasi antar *menu*, serta *list* dari 4 proyek terakhir yang ada untuk mengganti proyek yang ditunjukkan secara cepat.

Laman *View Task* menampilkan setiap tugas serta sub-tugas yang berada di bawah tugas tersebut, untuk memberikan tampilan yang jelas, dan simpel kepada pengguna. *Mockup* untuk laman *View Task* dapat dilihat di Gambar 3.47.

U N I V E R S I T A S
M U L T I M E D I A
N U S A N T A R A



Gambar 3.47 Mockup Laman View Task

Dalam *mockup View Task*, sub-tugas dari tugas yang bersangkutan ditampilkan sebagai sebuah *note*. *Note* tersebut akan memiliki indikator *progress*, yang menunjukkan seberapa jauh sub-tugas tersebut sudah dijalani. Judul dari tugas yang dikerjakan ditampilkan dalam tampilan yang besar, dengan warna aksen dari desain (dengan nilai heksadesimal #D6BE76), yang dipilih dari skema warna resmi PT Surya Semesta Internusa Tbk.

3.4 Kendala dan Solusi

Dalam proses magang, terjadi beberapa kendala yang menghalangi proses kerja yang efisien. Kendala-kendala tersebut adalah:

1. Kurangnya pengertian terhadap perancangan arsitektur kode dan *best practice* penulisan kode dalam industri.
2. Kurangnya persiapan untuk menghadapi kompleksitas proyek yang tinggi
3. Banyaknya pekerjaan karena kurangnya personil dalam tim *developer*

Namun, untuk setiap masalah yang ada, telah ditemukan solusi. Solusi yang ditemukan adalah sebagai berikut:

1. Menanyakan konsep yang kurang dimengerti dan meminta arahan kepada *senior developer* yang berada di dalam tim, serta melakukan pembelajaran independen melalui sumber eksternal yang edukatif seperti video YouTube, diskusi di situs Stack Overflow, dan sebagainya.
2. Menggunakan pengetahuan yang didapatkan untuk membuat sistem yang lebih *scalable*, agar dapat mengurangi kode yang redundan, serta *overload* dalam sistem.
3. Menggunakan waktu se-efisien mungkin dan berkomunikasi dengan *senior developer* untuk menentukan pekerjaan yang perlu diselesaikan segera dan mana yang bisa diselesaikan nanti.

