BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Subbidang Pengembangan dan Pengelolaan Sistem Aplikasi merupakan bagian dari Bidang Pengembangan dan Penyelenggaraan Sistem Informasi yang bertanggung jawab atas perencanaan, pengembangan, implementasi, serta pemeliharaan aplikasi di lingkungan Kementerian Pertahanan (Kemhan). Subbidang ini memiliki peran krusial dalam memastikan bahwa sistem aplikasi yang digunakan dapat mendukung operasional Kemhan secara efisien, aman, dan sesuai dengan kebutuhan pengguna. Fokus utama dari subbidang ini adalah meningkatkan kinerja sistem aplikasi serta melakukan inovasi teknologi agar selaras dengan perkembangan teknologi informasi terbaru.

Gambar 3. 1 Logo Kementrian Pertahanan RI (Sumber: Kemhan.go.id)

Gambar 3.1 merupakan Logo Kementrian RI yang mencakup semua satker yang ada dilingkungan kementrian pertahanan. Dalam tahap pengembangan sistem aplikasi, subbidang ini melakukan analisis kebutuhan pengguna dengan berkoordinasi dengan berbagai unit kerja di Kemhan. Proses ini mencakup identifikasi kebutuhan bisnis, analisis fungsionalitas, serta pembuatan desain sistem yang mencakup aspek teknis dan keamanan. Setelah itu, tim pengembang dalam

subbidang ini melakukan perancangan arsitektur sistem dan pemrograman untuk membangun aplikasi yang mampu memenuhi kebutuhan organisasi. Setiap sistem yang dikembangkan harus melalui tahapan pengujian ketat guna memastikan keandalan, performa, dan keamanannya sebelum diimplementasikan secara penuh.

Selain pengembangan, subbidang ini juga memiliki tanggung jawab dalam pengelolaan dan pemeliharaan sistem aplikasi yang telah diterapkan. Hal ini mencakup monitoring kinerja aplikasi, perbaikan bug atau kesalahan sistem, serta peningkatan fitur agar sistem tetap relevan dengan perubahan kebutuhan organisasi. Pengelolaan ini juga mencakup pengamanan sistem aplikasi dari ancaman siber seperti serangan malware, peretasan, dan kebocoran data. Oleh karena itu, subbidang ini bekerja sama dengan tim keamanan informasi untuk menerapkan standar keamanan yang ketat, termasuk enkripsi data, firewall, dan mekanisme otentikasi pengguna yang kuat.

Subbidang ini juga berperan dalam integrasi sistem aplikasi, baik di lingkungan internal Kemhan maupun dengan sistem eksternal yang digunakan oleh instansi lain. Integrasi ini bertujuan untuk memastikan pertukaran data yang aman dan efisien antar sistem, sehingga meningkatkan efektivitas dalam pengambilan keputusan berbasis data. Dalam hal ini, subbidang bekerja dengan standar interoperabilitas dan API (Application Programming Interface) agar sistem yang berbeda dapat saling terhubung dengan lancar. Selain itu, pelatihan bagi pengguna juga menjadi bagian dari tanggung jawab subbidang ini agar aplikasi yang dikembangkan dapat dimanfaatkan secara optimal oleh pegawai Kemhan.

Seiring dengan perkembangan teknologi, subbidang ini juga berfokus pada riset dan inovasi dalam pengembangan sistem aplikasi. Tren seperti kecerdasan buatan (AI), big data analytics, dan cloud computing mulai diterapkan untuk meningkatkan efisiensi dan keamanan sistem yang digunakan. Dengan terus melakukan evaluasi dan pengembangan, subbidang ini memastikan bahwa aplikasi yang dikembangkan

selalu memenuhi standar teknologi terkini dan mendukung Kemhan dalam menjalankan tugas dan fungsinya dengan lebih baik.

3.2 Tugas dan Uraian Kerja Magang

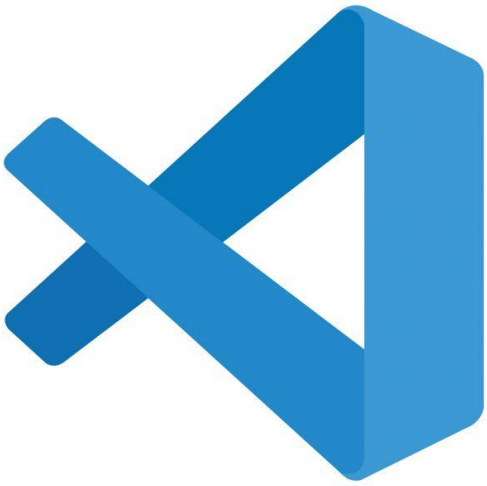
Bagian Tujuan berisi hal-hal yang dilakukan selama periode kerja magang.

Tabel 3.1 *Uraian kerja Magang*

No.	Pekerjaan	Minggu Ke-	Periode
1.	Pengenalan lingkungan dan penjelasan tugas. *Planning 1st Project*, dan pengumpulan data. Mempelajari terkait dengan Website dan Database yang akan digarap	1-2	3 – 14 Februari 2025
2.	Melanjutkan *project* yaitu Implementasi sistem autentikasi untuk 3 role (admin, operator, public), Membuat API endpoints untuk login, logout, dan manajemen session. Lalu melakukan Unit testing untuk sistem autentikasi dan Perancangan Terkait usecase dan activity diagram	2-3	10 Februari – 21 Februari 2025
3.	Implementasi CRUD untuk data_upload, *Sistem verifikasi dataset (data_verifikasi)* , Upload file dengan validasi format dan ukuran, Implementasi metadata dataset. Dan Perancangan sequence diagram	3-4	21 – 28 Februari 2025
4.	Implementasi metadata dataset, Pengembangan API untuk mengakses dataset publik, dan Unit testing untuk manajemen dataset	4	23-27 Februari 2025
5.	CRUD Infografis_upload, Verifikasi serta pengelolaan file infografis serta preview untuk admin dan publik	5	3-7 Maret 2025
6.	Kategorisasi Infografis, Sistem verifikasi infografis API untuk akses infografis public dan unit testing untuk manajemen infografis	6	10-14 Maret 2025
7.	Implementasi Tabel Contact_massages, Sistem tracking untuk Pengaduan, Dashboard Monitoring, Unit testing	7-8	17-28 Maret 2025

8.	Implementasi activity logging untuk admin/operator, Statistik penggunaan sistem, dashboard untuk admin, implementasi statistic satker	9-10	8-18 April 2025
9.	Integration Testing, Implementasi Master Data, pembuatan fitur update data dan kategori topik	11-12	21- 2 Mei 2025
10.	Testing Performance seluruh Komponen, PHP Mailer, Paparan proyek, Laporan dan Revisi terkait Project	13	4-9 Mei 2025

3.2.1 Minggu ke 1 (Pengenalan Lingkungan dan Kebutuhan)

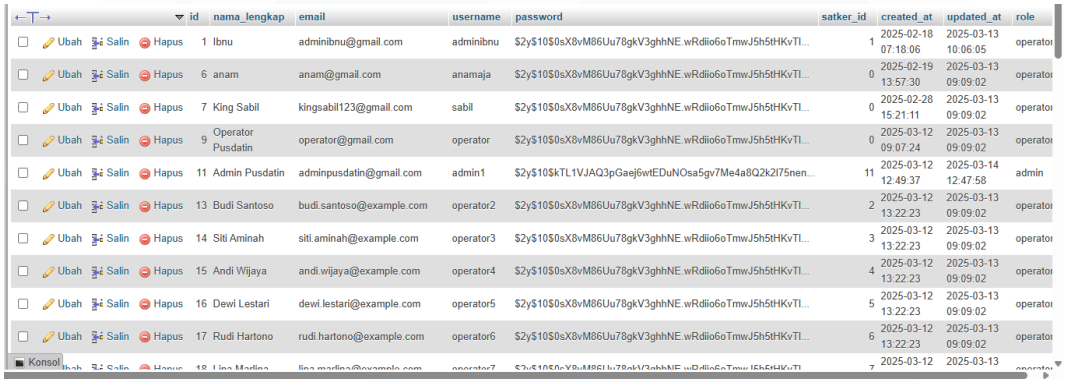


Gambar 3. 2 Logo Visual studio code

Gambar 3.2 merupakan Visual Studio Code sebagai lingkungan pengembangan dan menggunakan PHP serta MySQL untuk pengelolaan back-end dan database. Langkah pertama yang dilakukan adalah menganalisis struktur database yang ada, termasuk identifikasi tabel-tabel yang diperlukan, relasi antar tabel, serta jenis data yang akan disimpan. Struktur database yang baik akan mendukung efisiensi dalam pengelolaan data dan meningkatkan performa aplikasi, dengan pembuatan nama database “layanan_data”

Beberapa tabel yang direncanakan dalam database antara lain adalah tabel Users untuk menyimpan informasi pengguna, tabel Roles untuk mengelola peran pengguna (admin, operator, public), tabel Data_Records untuk menyimpan data

yang ditampilkan di website, serta tabel Logs untuk mencatat aktivitas pengguna. Setelah analisis, dilakukan optimasi terhadap struktur database yang ada untuk memastikan tidak ada redundansi data dan semua relasi antar tabel terdefinisi dengan baik, sehingga sistem dapat berjalan dengan efisien.



	id	nama_lengkap	email	username	password	satker_id	created_at	updated_at	role
Ubah Salin Hapus	1	Ibnu	adminibnu@gmail.com	adminibnu	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	1	2025-02-18 07:18:06	2025-03-13 10:06:05	operator
Ubah Salin Hapus	6	anam	anam@gmail.com	anamaja	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	0	2025-02-19 13:57:30	2025-03-13 09:09:02	operator
Ubah Salin Hapus	7	King Sabil	kingsabil123@gmail.com	sabil	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	0	2025-02-28 15:21:11	2025-03-13 09:09:02	operator
Ubah Salin Hapus	9	Operator Pusdatin	operator@gmail.com	operator	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	0	2025-03-12 09:07:24	2025-03-13 09:09:02	operator
Ubah Salin Hapus	11	Admin Pusdatin	adminpusdatin@gmail.com	admin1	$2y$10$kTL1VJAQ3pGaej6wtEDuNOsa5gv7Me4a8Q2k2l75nen...	11	2025-03-12 12:49:37	2025-03-14 12:47:58	admin
Ubah Salin Hapus	13	Budi Santoso	budi.santoso@example.com	operator2	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	2	2025-03-12 13:22:23	2025-03-13 09:09:02	operator
Ubah Salin Hapus	14	Siti Aminah	siti.aminah@example.com	operator3	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	3	2025-03-12 13:22:23	2025-03-13 09:09:02	operator
Ubah Salin Hapus	15	Andi Wijaya	andi.wijaya@example.com	operator4	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	4	2025-03-12 13:22:23	2025-03-13 09:09:02	operator
Ubah Salin Hapus	16	Dewi Lestari	dewi.lestari@example.com	operator5	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	5	2025-03-12 13:22:23	2025-03-13 09:09:02	operator
Ubah Salin Hapus	17	Rudi Hartono	rudi.hartono@example.com	operator6	$2y$10$0sX8vM86Uu78gkV3ghhNE.wRdlio6oTmwJ5h5tHKvTl...	6	2025-03-12 13:22:23	2025-03-13 09:09:02	operator
Konsol									

Gambar 3. 1 Tabel Users

Gambar 3.3 tersebut menampilkan tabel daftar pengguna sebuah sistem yang terdiri dari beberapa kolom informasi seperti ID, nama lengkap, email, username, password (dalam bentuk terenkripsi), satker ID, waktu pembuatan (created at), waktu update (updated at), dan peran pengguna (role). Dalam tabel tersebut terdapat satu akun admin (Admin Pusdatin) dan beberapa akun operator dengan nama yang berbeda-beda, dimana setiap baris data dilengkapi dengan tombol aksi untuk mengubah, menyalin, dan menghapus data. Data yang ditampilkan menunjukkan bahwa akun-akun tersebut dibuat antara periode Februari-Maret 2025 dengan pembaruan terakhir dilakukan pada Maret 2025.

Setelah desain database selesai, langkah selanjutnya adalah mengimplementasikan sistem autentikasi. Sistem ini dirancang untuk mendukung tiga peran pengguna: admin yang memiliki akses penuh untuk mengelola data dan pengaturan sistem, operator yang memiliki akses terbatas untuk mengelola data, serta public yang hanya dapat mengakses data publik. Implementasi sistem autentikasi meliputi penggunaan PHP untuk memproses login dan registrasi, hashing password sebelum disimpan di database untuk meningkatkan keamanan,

serta pembuatan sesi pengguna untuk menjaga status login saat berinteraksi dengan aplikasi.

Pada diagram *use case*, aktor-aktor yang terlibat dan interaksi mereka dengan sistem digambarkan secara jelas. Sebelum membuat diagram tersebut, penting untuk mengidentifikasi aktor dan aktivitas yang ada dalam sistem. Adapun hasil identifikasi aktor untuk diagram *use case* ditunjukkan dalam Tabel dibawah.

Tabel 3. 1 Identifikasi aktor use case diagram

No	Aktor	Deskripsi
1	Admin	Mengelola dataset secara keseluruhan dari satuan kerja (satker) Kemhan dan memverifikasi datasetnya. Kegiatan pengelolaan meliputi data keseluruhan di lingkup kemhan seperti dataset dan infografis, kemudian memverifikasi data, mengedit serta menambahkan satker dan user untuk operator (semua yang bisa dilkukan oleh operator bisa dilakukan oleh Admin)
2	Operator	Mengelola dataset dan infografis satuan kerja (satker) masing-masing yang mencakup proses unggah, pengeditan, dan pembaruan dataset, serta melihat aktivitas terkait dengan dataset dan infografis yang telah di upload
3	User	Aktor yang mengakses *website* layanan terbuka kemhan mulai dari *request* dataset, grafik dataset, melihat dan mengunduh dataset.

Tabel 3. 2 Identifikasi use case diagram

No	Usecase	Deskripsi	Aktor
1	Login	Admin dan operator masuk ke dalam sistem dengan cara memasukan *email* dan *password* agar dapat mengakses sistem. Setiap admin memiliki email dan passwordnya masing-masing yang telah dibuat dan diberikan.	Admin & Operator
2	Penyampaian pesan	User (publik) memasukan identitas dengan menyampaikan pesan ke	User

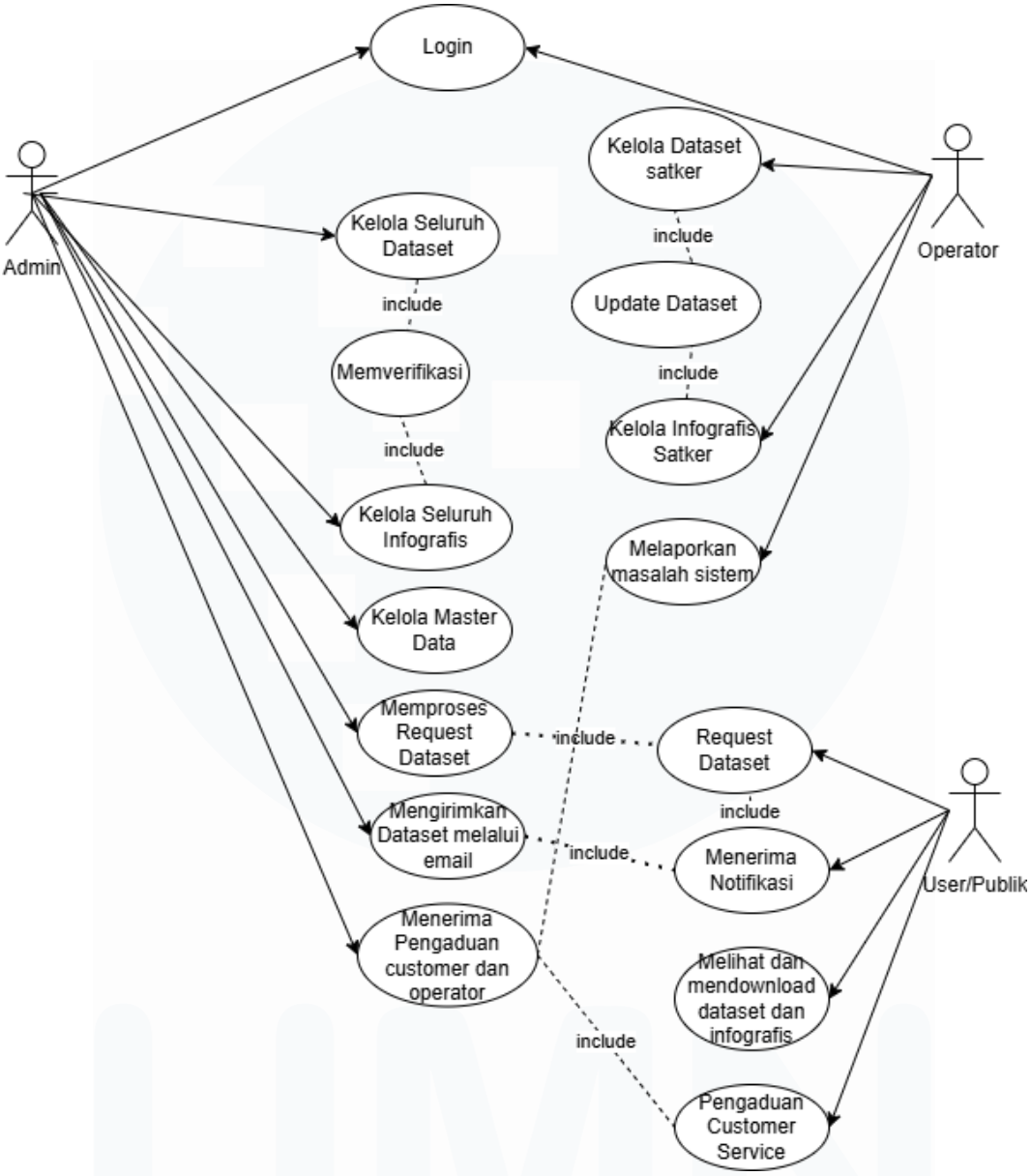
		tujuan satker yang akan di tuju.	
3	Kelola dataset	Dokumen diinput ke dalam sistem oleh admin dan operator, kemudian data tersebut di verifikasi oleh admin untuk memastikan tidak adanya kesalahan dalam proses input atau update. Dataset persatker juga dapat dikelola langsung oleh operator.	Admin & Operator
4	Kelola infografis	Admin dan operator mengupload file infografis sebagai acuan pengguna (publik) melihat informasi. Kemudian admin memverifikasi data yang telah di*input* atau di*update*. Dalam hal ini operator hanya mengelola infografis persatker.	Admin & Operator
5	Verifikasi data	Admin melakukan verifikasi dataset dan infografis yang telah di *upload* dan di *update*.	Admin
6	Update Data	Admin dan operator melakukan update data terkait dengan data yang telah di upload, setelah data di update maka operator harus menunggu lagi untuk admin memverifikasi data agar tampil di halaman user	Admin & Operator
7	Request dataset	Dokumen yang tidak bisa di akses secara terbuka oleh *user* (publik), dapat mengisi form permintaan dataset yang kemudian akan di setujui oleh admin dan dataset dapat dikirim melalui *email* user (publik).	User
8	Melihat dan mendowload dataset	Melihat kumpulan aktivitas dataset pada masing-masing satker dan dataset keseluruhan yang telah di	User

		input dan di verifikasi oleh operator serta admin	
9	Melihat dan mendownload infografis	Melihat kumpulan aktivitas infografis pada masing-masing satker dan infografis keseluruhan yang telah di input dan di verifikasi oleh operator serta admin	User
10	Logout	Admin dan operator keluar dari sistem	Admin & Operator

Setelah mengidentifikasi aktor dan kegiatan use case, maka dibuat diagram use case untuk website Portal data Satu Kementerian Pertahanan seperti ditunjukkan pada gambar dibawah.

Pada akhir minggu pertama, desain user database telah selesai dan sistem autentikasi dasar telah berhasil diimplementasikan. Langkah selanjutnya adalah melakukan pengujian terhadap sistem yang telah dibangun dan melanjutkan pengembangan fitur-fitur tambahan yang diperlukan untuk website open data Kemhan. Dengan fondasi yang kuat ini, diharapkan website dapat berfungsi dengan baik dan aman untuk pengguna dari berbagai kalangan.

3.2.2 Minggu Ke 2 (UseCase, Login, Logout, dan Implementasi Role)



Gambar 3. 2 Diagram Use case

Gambar 3.4 merupakan *use case yang* dibuat berdasarkan workflow yang bisa dikerjakan oleh user, tahap berikutnya adalah menyusun narasi *use case* yang merinci aktivitas yang dilakukan oleh aktor serta respons dari sistem. Adapun hasil narasi *use case* dapat dilihat dalam tabel dibawah berikut.

Tabel 3. 3 Narasi use case login

Use case name	Login

Use case ID	1	
Description	Admin & operator masuk ke dalam sistem dengan menggunakan *email* dan *password* agar dapat mengelola dataset.	
Precondition	Admin & operator memiliki akun yang telah dibuat.	
Trigger	Admin & operator ingin mengakses sistem untuk mengelola dataset.	
Typical course of event	*Actor actions*	*System response*
	1. Admin & operator memasukkan email dan password pada halaman *login*.	1. Sistem memverifikasi kredensial admin.
	2. Admin & operator menekan tombol "*Login*".	2. Jika valid, sistem mengarahkan admin ke halaman *dashboard*. Jika tidak valid, sistem menampilkan pesan kesalahan.
Alternative course	Jika admin ingin mengganti password, dapat menggunakan fitur "akun" untuk mereset kredensialnya.	
Conclusion	Admin & operator berhasil masuk ke dalam sistem dan dapat mengelola dataset sesuai hak aksesnya.	
Post condition	Admin & operator dalam keadaan masuk (*logged in*) dan memiliki akses penuh ke pengelolaan dataset.	

Tabel 3. 4 Narasi use case penyampaian pesan

Use case name	Penyampaian pesan	
Use case ID	2	
Description	*User* mengirimkan pesan kepada satuan kerja yang dituju.	
Precondition	*User* sudah mengakses sistem dan memiliki informasi satuan kerja tujuan.	
Trigger	*User* ingin mengajukan pertanyaan atau menyampaikan pesan dan informasi kepada satuan kerja terkait.	
Typical course of event	*Actor actions*	*System response*
	1. *User* memasukkan identitas dan isi pesan.	1. Sistem memverifikasi data input *user*.
	2. *User* menekan tombol "kirim pesan"	2. Sistem mengirimkan pesan ke satuan kerja yang dituju dan menampilkan

		konfirmasi bahwa pesan telah dikirim.
Alternative course	-	
Conclusion	*User* berhasil mengirimkan pesan ke satuan kerja terkait.	
Post condition	Pesan *user* berhasil tersimpan dan terkirim ke satuan kerja tujuan.	

Tabel 3. 5 Narasi kelola fungsi visualisasi

Use case name	Kelola Visualisasi	
Use case ID	4	
Description	Admin & operator dapat melakukan proses filer sort time berdasarkan waktu yang diinginkan dan nantinya data akan muncul sesuai yang mereka inginkan	
Precondition	Admin & operator sudah masuk ke dalam sistem dan memiliki hak akses untuk melakukan proses sort time	
Trigger	Admin & operator ingin menampilkan data sesuai dengan waktu yang mereka inginkan	
Typical course of event	*Actor actions*	*System response*
	1. Admin & operator memilih menu "Dashboard"	1. Sistem menampilkan fitur filter data, total data dan visualisasi diagram garis
	2. Admin & operator melakukan filter dataset berdasarkan waktu yang diinginkan	2. Sistem menampilkan pop up calender dan juga quick filter berdasarkan 1 Minggu, 2 Minggu, 1 Bulan, 3 Bulan dan 6 Bulan yang lalu
	3. Admin & operator menerapkan filter yang telah diinginkan	3. Sistem menyimpan infografis dan menampilkan notifikasi bahwa perubahan telah disimpan.
Alternative course	-	
Conclusion	Admin & operator berhasil mengelola infografis untuk menampilkan informasi dalam bentuk visual.	
Post condition	Sistem menyimpan dan menampilkan infografis terbaru.	

Tabel 3. 6 Narasi kelola dataset

Use case name	Kelola dataset	
Use case ID	3	
Description	Admin & operator mengelola dataset dalam sistem untuk memastikan data yang disajikan valid dan akurat.	
Precondition	Admin & operator memiliki akses ke sistem dan ingin mengelola dataset.	
Trigger	Admin & operator ingin memperbarui, menambah, atau memverifikasi dataset.	
Typical course of event	*Actor actions*	*System response*
	1. Admin & operator mengakses menu "Kelola Dataset"	1. Sistem menampilkan daftar dataset yang tersedia.
	2. Admin & operator memilih untuk menambah, melihat statistic dan mengedit data	2. Sistem memberikan formulir input atau konfirmasi penghapusan data.
	3. Admin & operator menyimpan perubahan.	3. Sistem memperbarui database dan menampilkan notifikasi bahwa perubahan berhasil dilakukan.
Alternative course	-	
Conclusion	Dataset berhasil dikelola oleh admin & operator sesuai kebutuhan.	
Post condition	Dataset dalam sistem diperbarui sesuai dengan perubahan yang dilakukan oleh admin & operator.	

Tabel 3. 7 Narasi kelola infografis

Use case name	Kelola data grafis	
Use case ID	4	
Description	Admin & operator mengelola data grafis dalam sistem untuk menampilkan informasi dalam bentuk visualisasi yang lebih mudah dipahami	
Precondition	Admin & operator sudah masuk ke dalam sistem dan memiliki hak akses untuk mengelola data grafis.	
Trigger	Admin & operator ingin menampilkan atau memperbarui data dalam bentuk grafis.	
Typical course of event	*Actor actions*	*System response*

	4. Admin & operator memilih menu "Kelola Infografis".	4. Sistem menampilkan halaman pengelolaan data grafis.
	5. Admin & operator mengunggah atau memilih dataset yang ingin divisualisasikan, melihat statistik dan mengedit	5. Sistem memproses dataset, menampilkan *preview* grafik, serta pilihan terkait dengan update data.
	6. Admin & operator menyimpan perubahan.	6. Sistem menyimpan infografis dan menampilkan notifikasi bahwa perubahan telah disimpan.
Alternative course	-	
Conclusion	Admin & operator berhasil mengelola infografis untuk menampilkan informasi dalam bentuk visual.	
Post condition	Sistem menyimpan dan menampilkan infografis terbaru.	

Tabel 3. 8 Narasi verifikasi data

Use case name	Verifikasi data	
Use case ID	5	
Description	Admin melakukan verifikasi terhadap data (dataset atau infografis) yang diunggah oleh operator dan mereview terkait data yang masuk ke sistem	
Precondition	Admin sudah masuk ke dalam sistem dan memiliki hak akses untuk melakukan verifikasi data.	
Trigger	Admin ingin melakukan pengecekan dan verifikasi data yang telah diunggah oleh operator.	
Typical course of event	*Actor actions*	*System response*
	1. Admin masuk ke halaman verifikasi data.	1. Sistem menampilkan daftar data yang di *approved*, *rejected*, dan *pending*.
Alternative course	-	
Conclusion	Admin berhasil melakukan verifikasi, sehingga data yang valid dapat digunakan dalam sistem dan data yang tidak valid dapat dikoreksi atau ditolak.	

Post condition	Data yang disetujui tersedia untuk digunakan dalam sistem, sedangkan data yang ditolak akan tersimpan di table data_upload dengan status rejected

Tabel 3. 9 Narasi request dataset

Use case name	*Request* dataset	
Use case ID	6	
Description	*User* mengajukan permintaan untuk mendapatkan dataset tertentu dari sistem.	
Precondition	*User* telah masuk ke dalam sistem dan ingin mengakses dataset tertentu yang tidak tersedia untuk unduhan langsung.	
Trigger	*User* membutuhkan dataset tertentu yang belum tersedia di sistem atau membutuhkan akses khusus.	
Typical course of event	*Actor actions*	*System response*
	1. User mencari dataset mengklik menu "*Request* Dataset" di halaman utama.	1. Sistem menampilkan formulir permintaan dataset.
	2. User mengisi detail permintaan, seperti nama dataset dan satker tujuan.	2. Sistem memverifikasi input dan menampilkan tombol "Kirim".
	3. *User* menekan tombol "Kirim".	3. Sistem menyimpan permintaan dan mengirimkan notifikasi ke admin untuk ditinjau.
	4. Admin meninjau permintaan dan menyetujui/menolak akses.	4. Sistem mengirimkan status permintaan kepada *user*.
Alternative course	-	
Conclusion	*User* berhasil mengajukan permintaan dataset, dan admin dapat menyetujui atau menolak permintaan tersebut.	
Post condition	Dataset dalam sistem diperbarui sesuai dengan perubahan yang dilakukan oleh admin & operator.	

Tabel 3. 10 Narasi melihat dan mendownload dataset

Use case name	Melihat dan mendownload dataset
Use case ID	7

Description	*User* melihat daftar dataset yang tersedia dan mengunduh dataset yang diinginkan.	
Precondition	*User* sudah masuk ke dalam sistem dan memiliki hak akses untuk melihat atau mengunduh dataset.	
Trigger	*User* ingin mengakses dataset untuk dianalisis atau digunakan dalam penelitian.	
Typical course of event	*Actor actions*	*System response*
	1. *User* memilih menu "organisasi/satker"	1. Sistem menampilkan nama-nama satker
	2. *User* mencari atau memilih satker.	2. Sistem menampilkan deskripsi dan detail dataset.
	3. *User* mengklik tombol "*Download*".	3. Sistem memproses permintaan dan mengunduh *file* dataset ke perangkat *user*.
Alternative course	-	
Conclusion	*User* berhasil melihat dan mengunduh dataset sesuai dengan kebutuhan.	
Post condition	Dataset telah berhasil diunduh dan tersimpan di perangkat *user*.	

Tabel 3. 11 Narasi melihat dan mendownload infografis

Use case name	Melihat dan mendownload infografis	
Use case ID	8	
Description	*User* mengakses halaman utama untuk melihat infografis publik dan aktivitas *file*.	
Precondition	*User* memiliki akses ke sistem dan terhubung ke internet.	
Trigger	*User* ingin melihat informasi infografis dan aktivitas *file*.	
Typical course of event	*Actor actions*	*System response*
	2. *User* memilih menu "organisasi/satker"	1. Sistem menampilkan nama-nama satker
	3. User mencari atau memilih satker.	4. Sistem menampilkan deskripsi dan detail infografis.
	5. User mengklik tombol "*Download*".	4. Sistem memproses permintaan dan mengunduh *file*

		infografis ke perangkat *user*.
Alternative course	-	
Conclusion	*User* mendapatkan akses ke halaman infografis publik dan aktivitas *file*.	
Post condition	*User* telah melihat informasi yang ditampilkan pada halaman.	

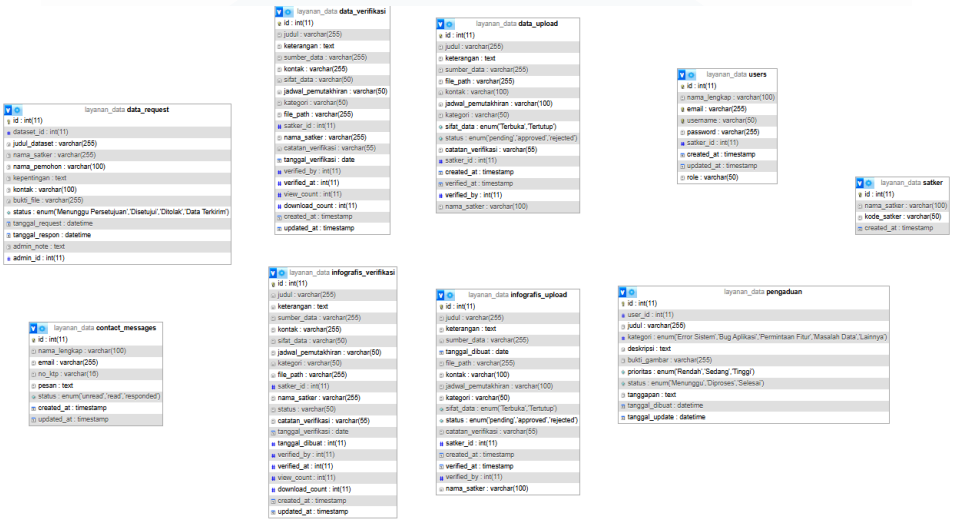
Tabel 3. 12 Narasi Logout

Use case name	*Logout*	
Use case ID	9	
Description	Admin & operator keluar dari sistem untuk mengakhiri sesi penggunaannya.	
Precondition	Admin & operator sudah masuk ke dalam sistem.	
Trigger	Admin & operator ingin mengakhiri sesi dan keluar dari akun.	
Typical course of event	*Actor actions*	*System response*
	1. Admin & operator mengklik tombol "*logout* akun".	1. Sistem menampilkan konfirmasi *logout*.
	2. Admin & operator mengonfirmasi *logout*.	2. Sistem mengakhiri sesi dan mengarahkan admin & operator kembali ke halaman utama.
Alternative course	-	
Conclusion	Admin & operator berhasil keluar dari sistem, dan sesi penggunaannya berakhir.	
Post condition	Admin & operator harus *login* kembali untuk mengakses sistem.	

Dari Tabel diatas merupakan Narasi yang akan dilakukan oleh user selama menggunakan website Portal Data satu Kementrian Pertahanan, disaat user melakukan aksi maka system akan merespon aksi yang dilakukan oleh user, semua ini tentunya berdasarkan usecasenya juga.

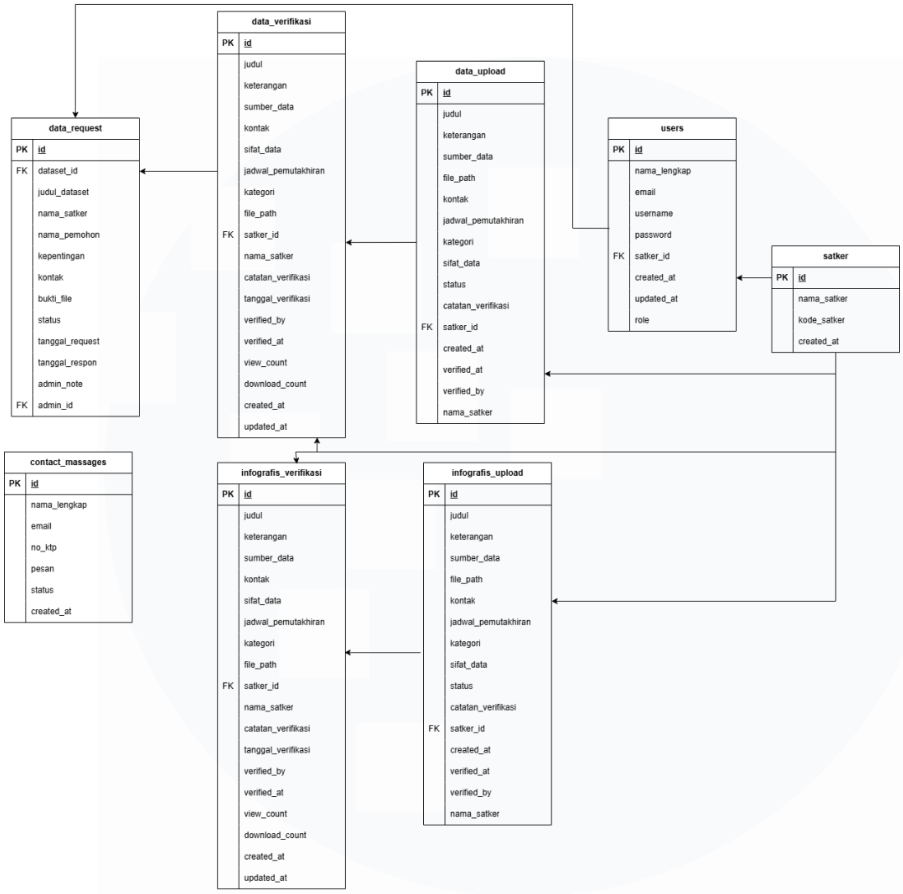
Sebelum menjelaskan tentang ERD Diagram berikut merupakan table database layanan_data yang digunakan secara langsung didalam pembuatan website portal

data satu Kementrian pertahanan RI yang diambil dari tampilan mysql PHPmyadmin untuk keperluan laporan.



Gambar 3. 3 Tabel Database di PHPmyadmin

Gambar 3.5 merupakan Tabel database yang diambil langsung dari PHPmyadmin sesuai dengan table yang ada yaitu berjumlah 9 Tabel yang saling berkoneksi satu sama lain berikut merupakan ERD diagramnya.



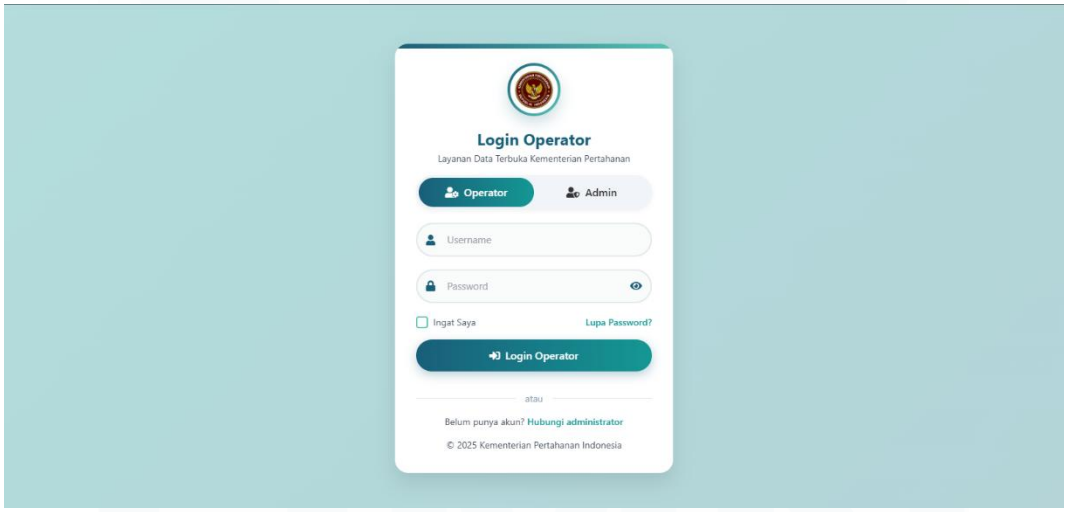
Gambar 3. 4 ERD Diagram layanan_data

Gambar 3.6 merupakan ERD Diagram yang menggambarkan struktur basis data yang dirancang untuk mendukung sistem manajemen unggahan dan verifikasi data pada suatu instansi. Terdapat delapan entitas utama dalam sistem ini, yaitu users, satker, data_upload, data_verifikasi, infografis_upload, infografis_verifikasi, data_request, dan contact_messages. Setiap entitas memiliki atribut yang menggambarkan informasi yang diperlukan dalam proses unggah, verifikasi, serta interaksi pengguna. Entitas users merupakan pusat utama interaksi dalam sistem, karena memiliki relasi langsung ke berbagai entitas lain, baik sebagai pengunggah maupun sebagai verifikator.

Fitur keamanan yang diterapkan dalam sistem login mencakup penggunaan mysqli_real_escape_string() untuk mencegah SQL injection pada username, serta penggunaan Prepared Statement untuk query database. Selain itu, password_verify() digunakan untuk memverifikasi password yang di-hash, dan

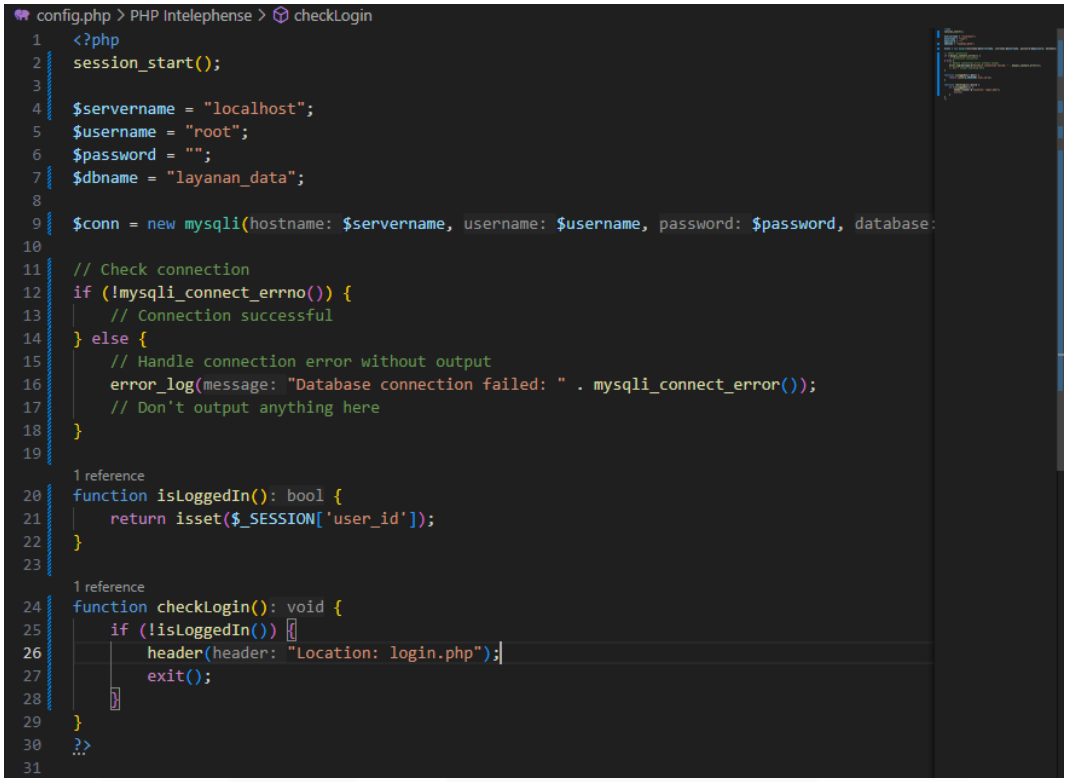
terdapat validasi role user sesuai tipe login. Manajemen session juga diterapkan untuk menyimpan data user yang login, dengan sistem yang melakukan redirect ke dashboard sesuai role user.

Alur proses dimulai ketika user mengirim form login melalui POST request, di mana username dan password divalidasi. Sistem kemudian memeriksa keberadaan username di database, memverifikasi kesesuaian role dengan tipe login, dan memverifikasi password. Jika semua langkah berhasil, session dibuat dan user diarahkan ke dashboard; jika gagal, pesan error ditampilkan. Namun, terdapat inkonsistensi pada bagian redirect, di mana baik admin maupun operator diarahkan ke dashboardadmin.php, padahal seharusnya operator diarahkan ke halaman dashboard yang berbeda.



Gambar 3. 5 Tampilan Login Admin dan Operator

Gambar 3.7 ini merupakan tampilan login untuk operator disaat operator dan admin login maka ada field tersendiri yang bisa di switch, jika data yang dimasukan admin tetapi di field operator maka akan terjadi eror dan tidak bisa masuk.



Gambar 3. 6 Code config.php

Gambar 3.8 tersebut merupakan config.php untuk mengkoneksikan antara code yang dimasukan di VS code dengan database yang ada di phpMyAdmin. Kode dimulai dengan session_start() untuk memulai sesi pengguna, kemudian mendefinisikan parameter koneksi ke database MySQL seperti servername, username, password, dan dbname. Selanjutnya, objek $conn dibuat menggunakan new mysqli() untuk menghubungkan ke database. Koneksi diperiksa dengan mysqli_connect_errno() dan jika gagal, pesan kesalahan akan dicatat menggunakan error_log. Terdapat juga dua fungsi: isLoggedIn() yang mengembalikan true jika $_SESSION['user_id'] sudah diset, dan checkLogin() yang akan mengarahkan pengguna ke halaman login.php jika belum login, serta menghentikan eksekusi dengan exit(). Kode ini berguna untuk mengamankan halaman dari akses yang tidak sah.

logout.php

```php
<?php
// Tambahkan file logout.php baru
session_start();
session_destroy();
header(header: "Location: index.php");
exit();
?>
```

Gambar 3. 7 Code logout.php

Gambar 3.9 tersebut menampilkan kode program PHP pada file logout.php yang berfungsi untuk mengakhiri sesi pengguna (logout). Pertama, session_start() digunakan untuk memulai atau melanjutkan sesi yang sedang aktif. Kemudian, session_destroy() dijalankan untuk menghapus semua data sesi yang tersimpan. Setelah sesi dihancurkan, perintah header("Location: index.php") digunakan untuk mengarahkan pengguna kembali ke halaman utama (index.php). Terakhir, fungsi exit() dipanggil untuk menghentikan eksekusi skrip agar proses redirect berjalan dengan benar. Komentar di baris kedua menunjukkan bahwa file ini baru saja ditambahkan untuk menangani proses logout.

```php
        <?php if ($_SESSION['role'] === 'admin') : ?>
            <li class="mb-4">
                <a class="flex items-center <?php echo $current_page === 'verifikasi.php' ? 'text-blue-600 font-bold relative' : ''; ?>" h
                    <i class="fas fa-check-circle mr-3"></i> Verifikasi Data
                </a>
            </li>

            <li class="mb-4">
                <a class="flex items-center <?php echo $current_page === 'request_data.php' ? 'text-blue-600 font-bold relative' : ''; ?>"
                    <i class="fas  fa-folder-open mr-3"></i> Request Data
                </a>
            </li>

            <li class="mb-4">
                <a class="flex items-center <?php echo $current_page === 'master_data.php' ? 'text-blue-600 font-bold relative' : ''; ?>"
                    <i class="fas fa-cogs mr-3"></i> Akses Kontrol dan Pengelolaan
                </a>
            </li>
        <?php endif; ?>

        <li class="mb-4">
            <a class="flex items-center <?php echo $current_page === 'akun.php' ? 'text-blue-600 font-bold relative' : ''; ?>" href="akun.
                <i class="fas fa-user mr-3"></i> Akun
            </a>
        </li>
    </ul>
</nav>
```

Gambar 3. 8 Code sidebar.php role admin

Gambar 3.10 tersebut merupakan implementasi sidebar untuk sistem Portal Data Satu Kementerian Pertahanan yang menggunakan PHP sebagai back-end.

Sistem ini mengandalkan manajemen session PHP untuk kontrol akses, di mana $_SESSION['role'] digunakan untuk membedakan hak akses antara admin dan operator. Nama halaman aktif disimpan dalam variabel $current_page menggunakan basename($_SERVER['PHP_SELF']), yang kemudian digunakan untuk memberikan penanda visual pada menu yang sedang aktif.

Aplikasi ini menggunakan arsitektur Multi-Page Application (MPA) di mana setiap menu mengarah ke file PHP terpisah. Sistem autentikasi diimplementasikan melalui manajemen session PHP, dengan fitur logout yang mengarahkan ke logout.php untuk menghancurkan session dan mengarahkan user kembali ke halaman login. Setiap halaman terproteksi perlu memiliki validasi session dan role untuk mencegah akses tidak sah.

Ketika login sebagai Admin, user dapat mengakses semua menu berikut:

1. Dashboard
2. Kontrol Dataset
3. Kontrol Infografis
4. Aktivitas
5. Pengaduan Umum
6. Verifikasi Data
7. Request Data
8. Akses Kontrol dan Pengelolaan
9. Akun

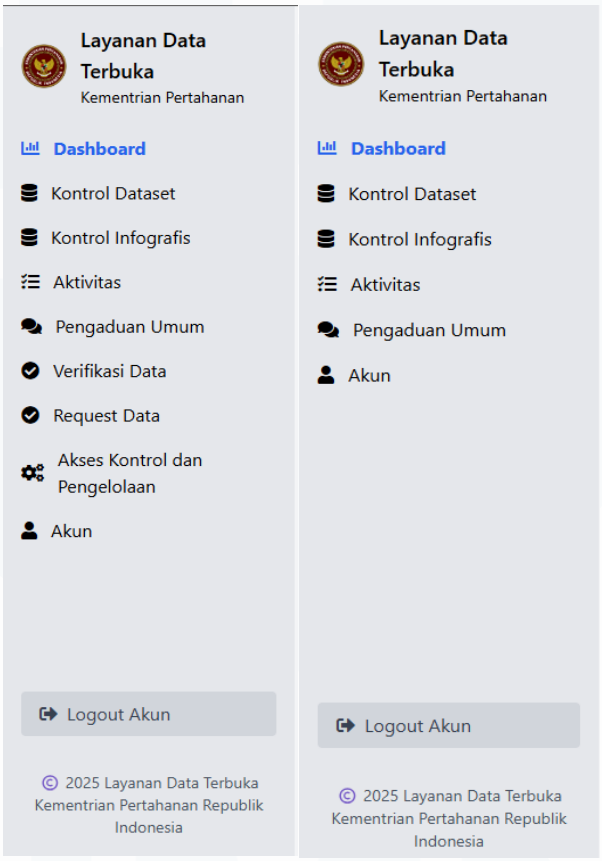
Sedangkan ketika login sebagai Operator, user hanya dapat mengakses menu:

1. Dashboard
2. Kontrol Dataset
3. Kontrol Infografis

4. Aktivitas

5. Pengaduan Umum

6. Akun

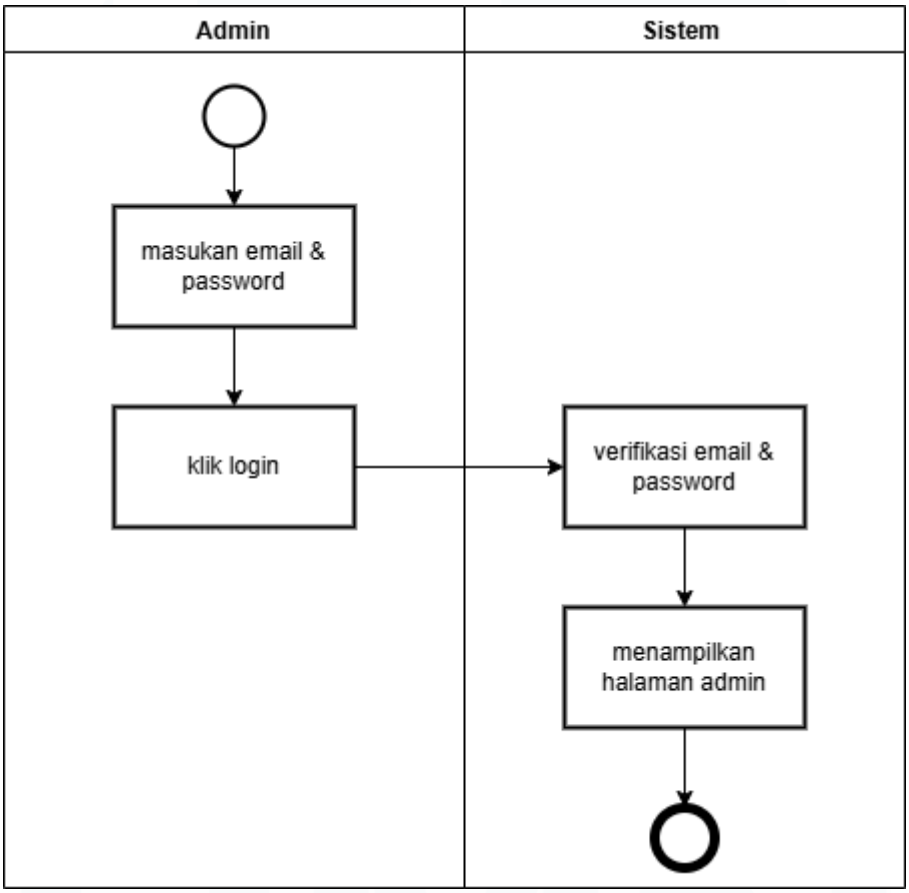


Gambar 3. 9 Kiri sidebar untuk Admin, Kanan untuk Operator

Gambar 3.11 Merupakan Perbedaan tampilan Interface pada halaman admin dan operator perbedaan ini tentunya dibuat karena Admin memiliki fungsi yang lebih leluasa sedangkan Operator cakupannya hanya sebatas untuk Kelola dataset dan Infografis, Semua fungsi Operator dimiliki oleh Admin, tetapi semua fungsi Admin belum tentu dimiliki oleh Operator.

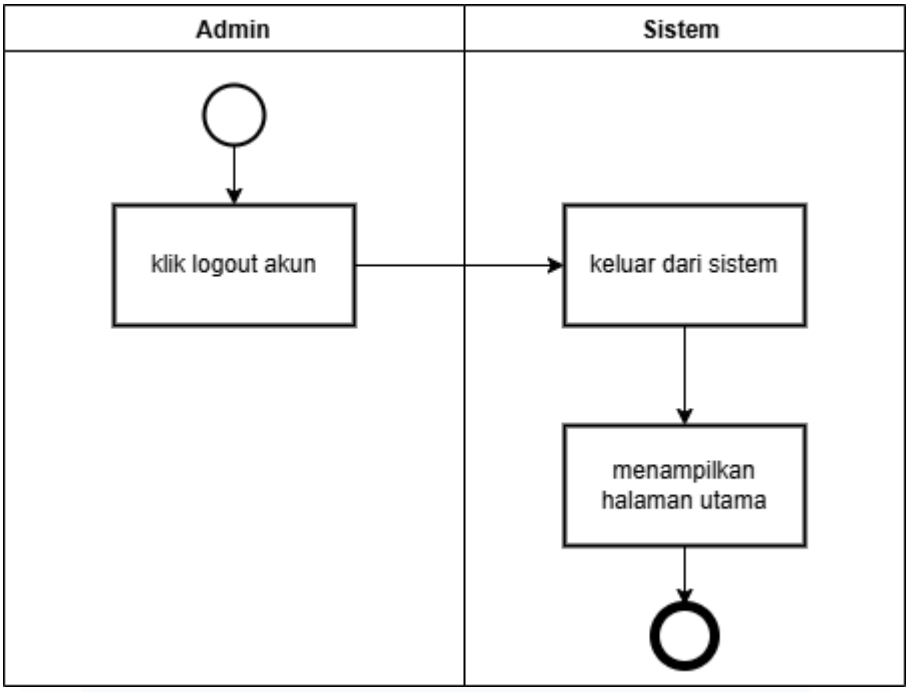
3.2.3 Minggu Ke 3 (Activity, Sequence dan Implementasi dataset)

Pada tahap ini, *activity* diagram dibuat untuk menggambarkan alur proses yang terjadi dalam sistem. Berikut ini adalah gambaran *activity diagram* pada *website* Portal Data Satu Kementerian pertahanan RI.



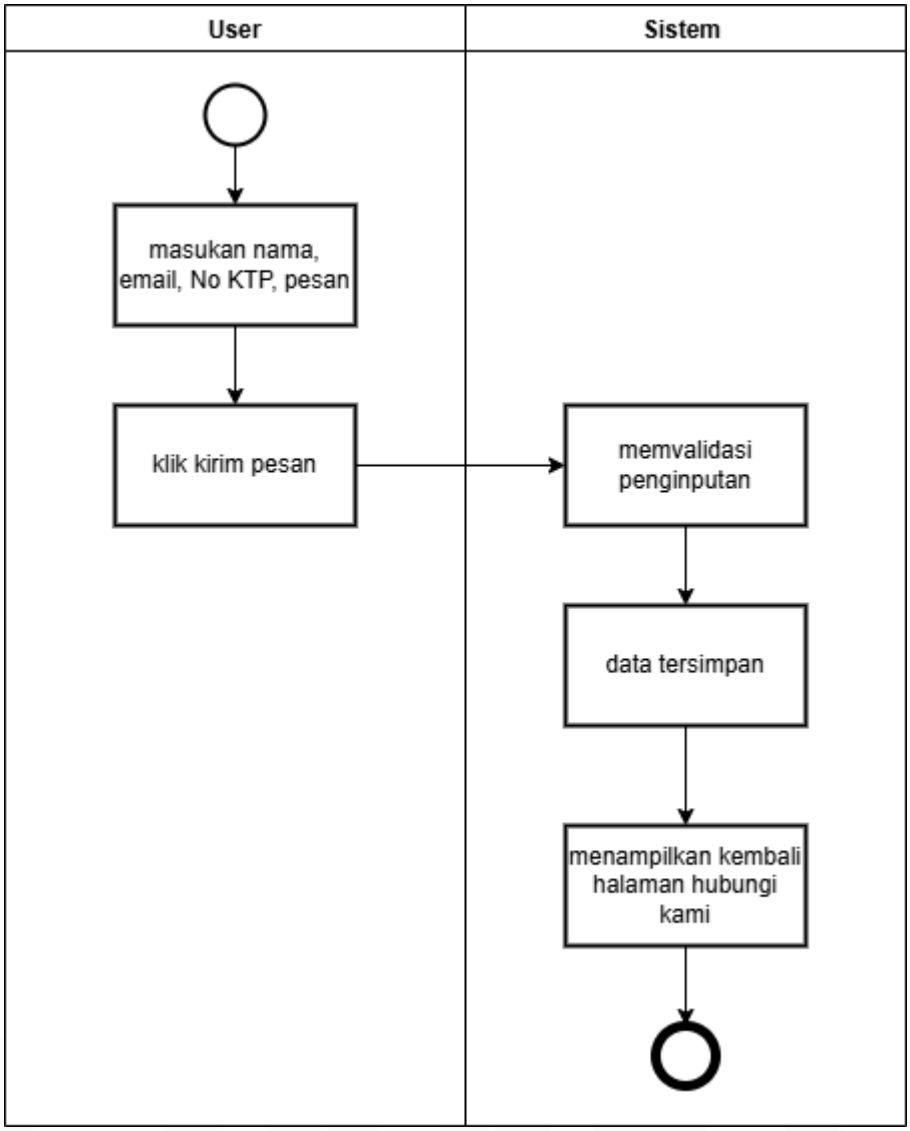
Gambar 3. 10 Activity diagram login

Pada Gambar 3.12 activity diagram login, admin diminta untuk mengisi *email* dan *password* yang telah diberikan. Setelah itu sistem akan melakukan verifikasi *email* dan *password*, kemudian sistem akan menampilkan halaman admin dan operator.



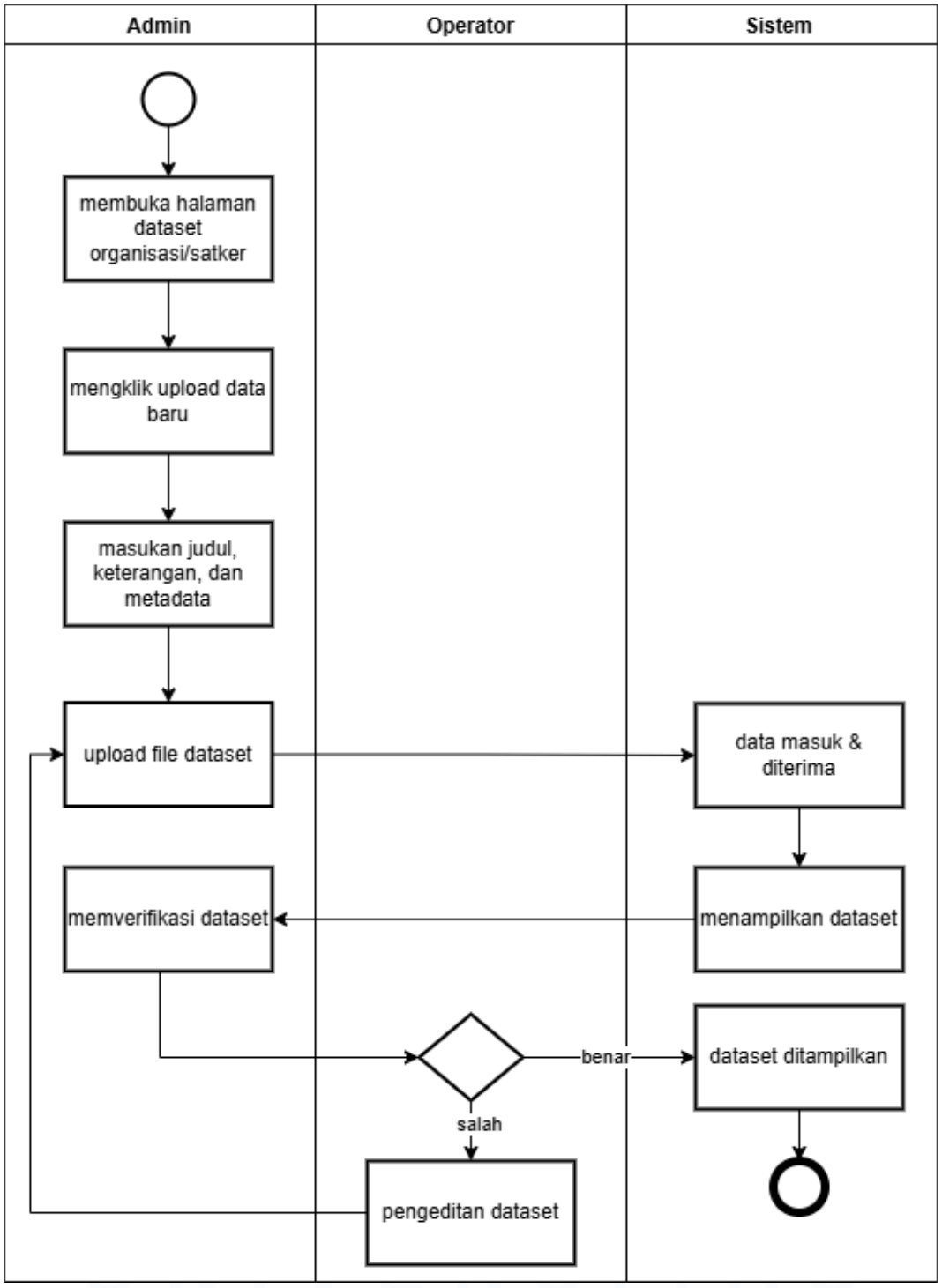
Gambar 3. 11 Activity diagram logout

Pada Gambar 3.13 activity ini, menggambarkan admin dan operator keluar dari sistem dengan mengklik tombol *logout* akun maka system akan otomatis mengeluarkan user dari halaman admin ke halaman publik diharuskan memasukan username dan password Kembali saat Login akun.



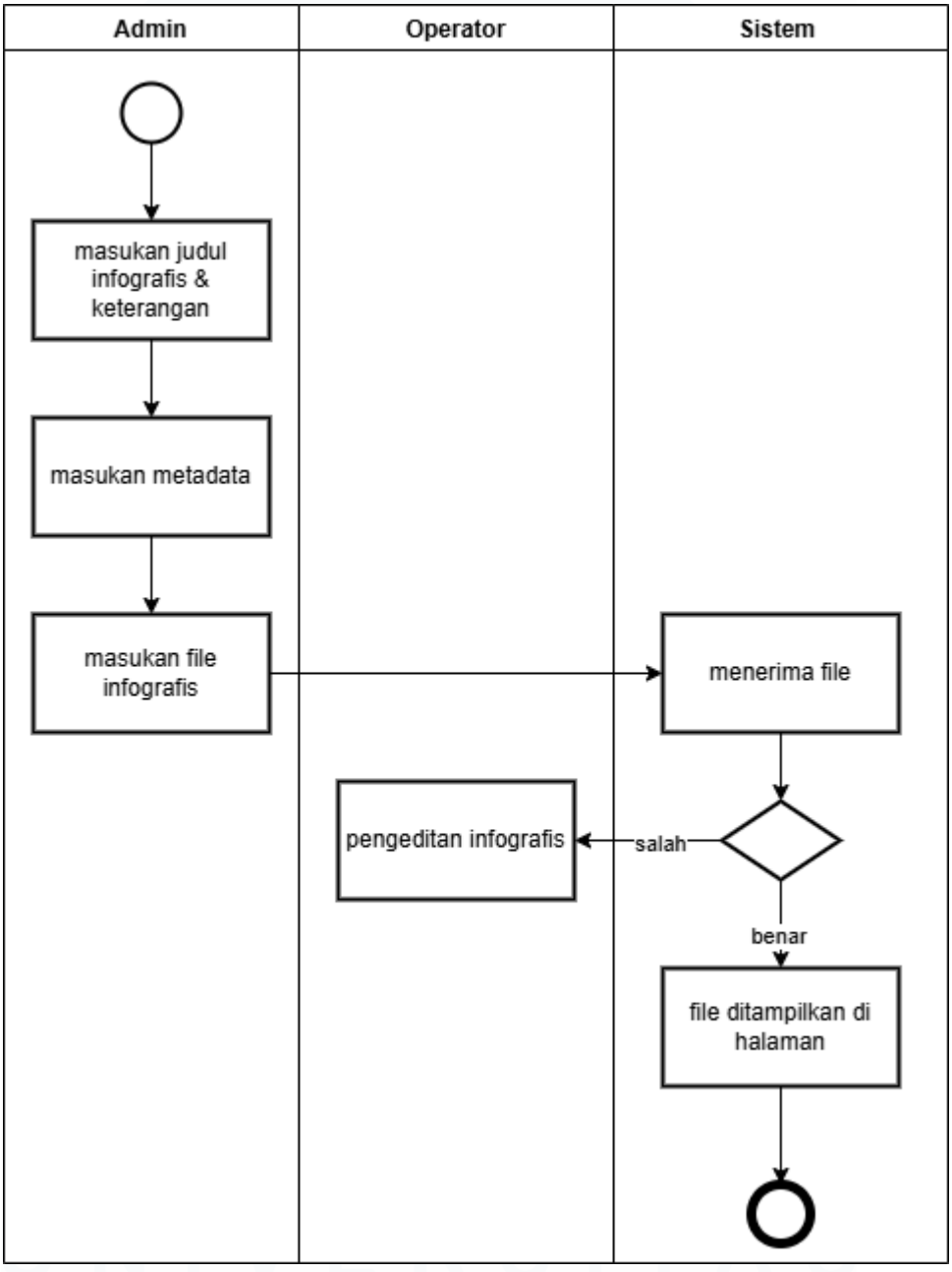
Gambar 3. 12 Activity diagram penyampaian pesan

Pada Gambar 3.14 activity ini, menggambarkan *user* (publik) mengirimkan pesan ke satker tujuannya biasanya digunakan untuk memberikan saran dan keperluan terkait dengan satuan kerja yang mereka tuju.



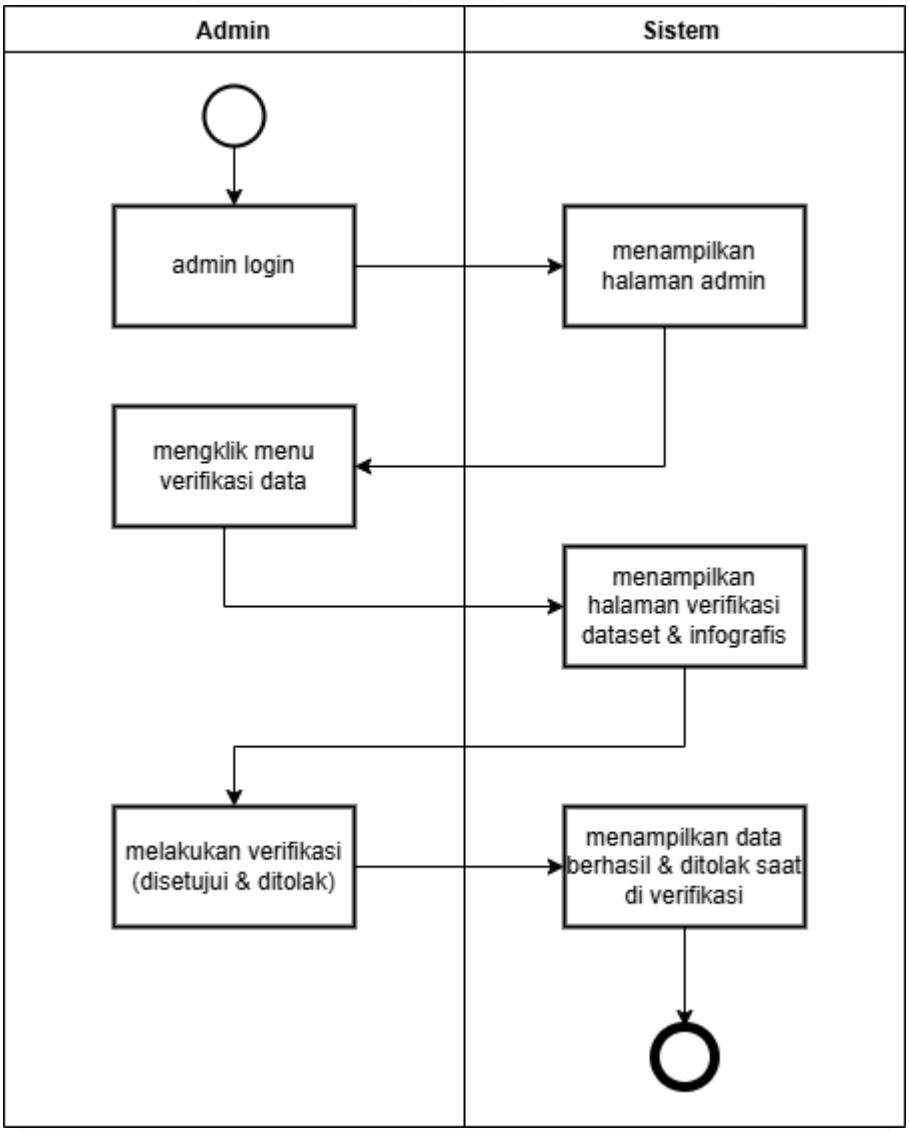
Gambar 3. 13 Activity diagram kelola dataset

Pada Gambar 3.15 activity ini, menggambarkan bahwa sebelum admin memasukan *file* dataset, admin mengisi spesifikasi data seperti judul dataset, sumber data, keteranagan, kontak, ,jadwal pemutakhiran, sifat data, dan kategori.=



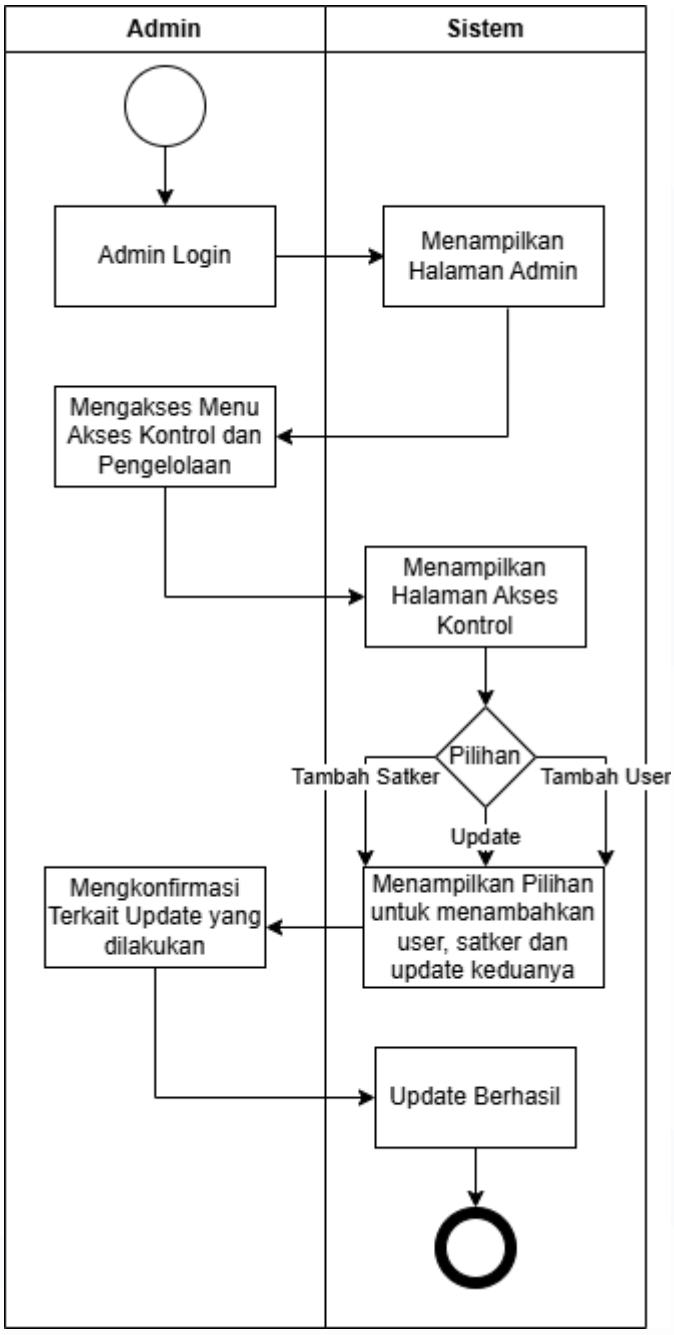
Gambar 3. 14 Activity diagram kelola infografis

Pada Gambar 3.16 activity diagram ini menggambarkan infografis bahwa, admin dapat mengedit dan memperbaharui infografis secara keseluruhan dan operator dapat mengelola infografis tiap satker masing-masing.



Gambar 3. 15 Activity diagram verifikasi data

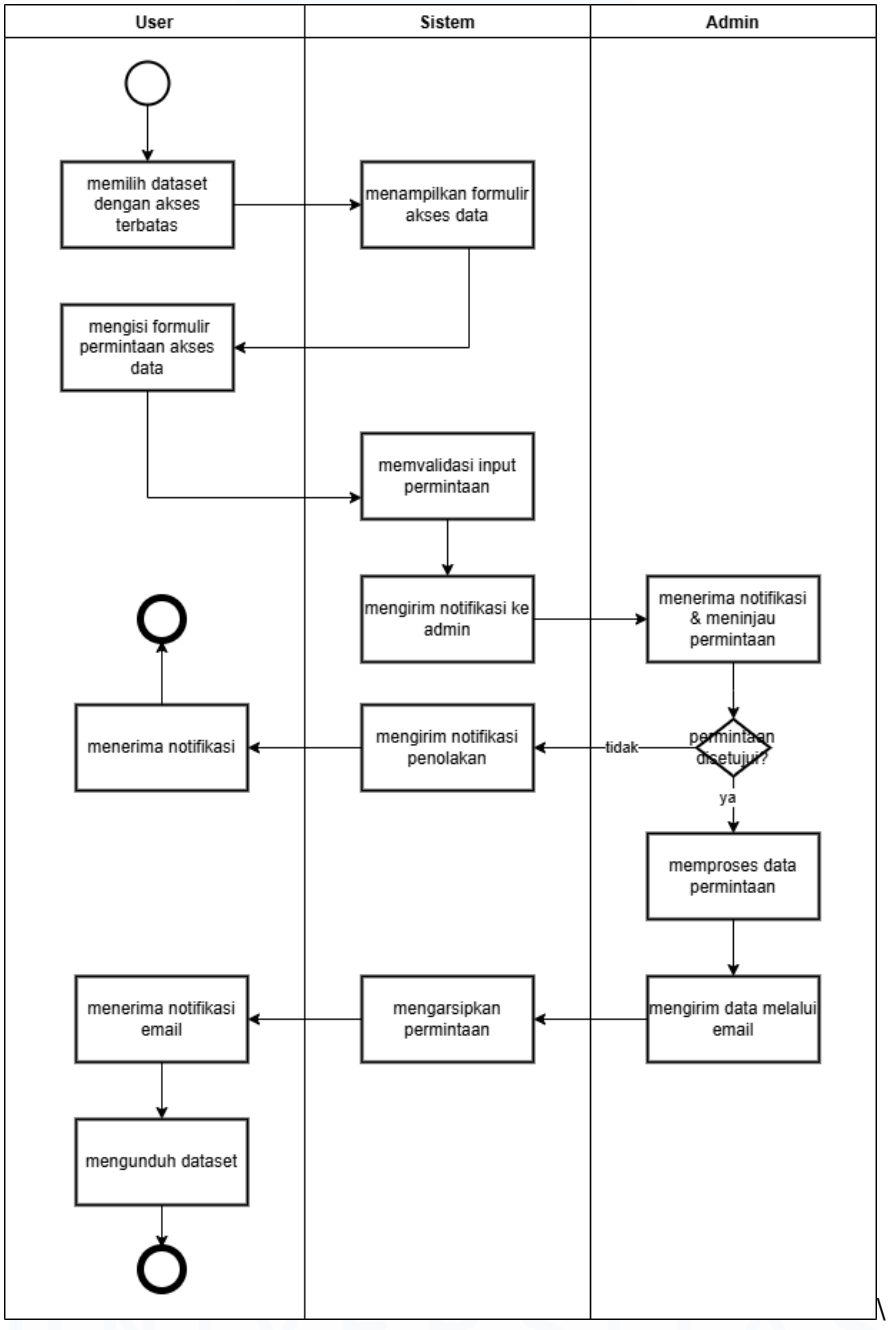
Pada Gambar 3.17 activity diagram ini, menggambarkan bahwa admin melakukan Verifikasi terhadap data dan infografis yang telah diupload oleh operator sebelumnya, dengan memverifikasi datanya maka, data akan berubah statusnya menjadi approved atau rejected, Jika Approve maka data akan dilanjutkan untuk ditampilkan di halaman publik



Gambar 3. 16 Activity diagam akses kontrol

Gambar 3.18 Akses Kontrol dan Pengelolaan Admin bisa menambahkan dan mengedit satker ataupun user, Jika ada Satker baru yang memungkinkan pada satuan kerja Kementrian pertahanan yang mengharuskan untuk menggunakan

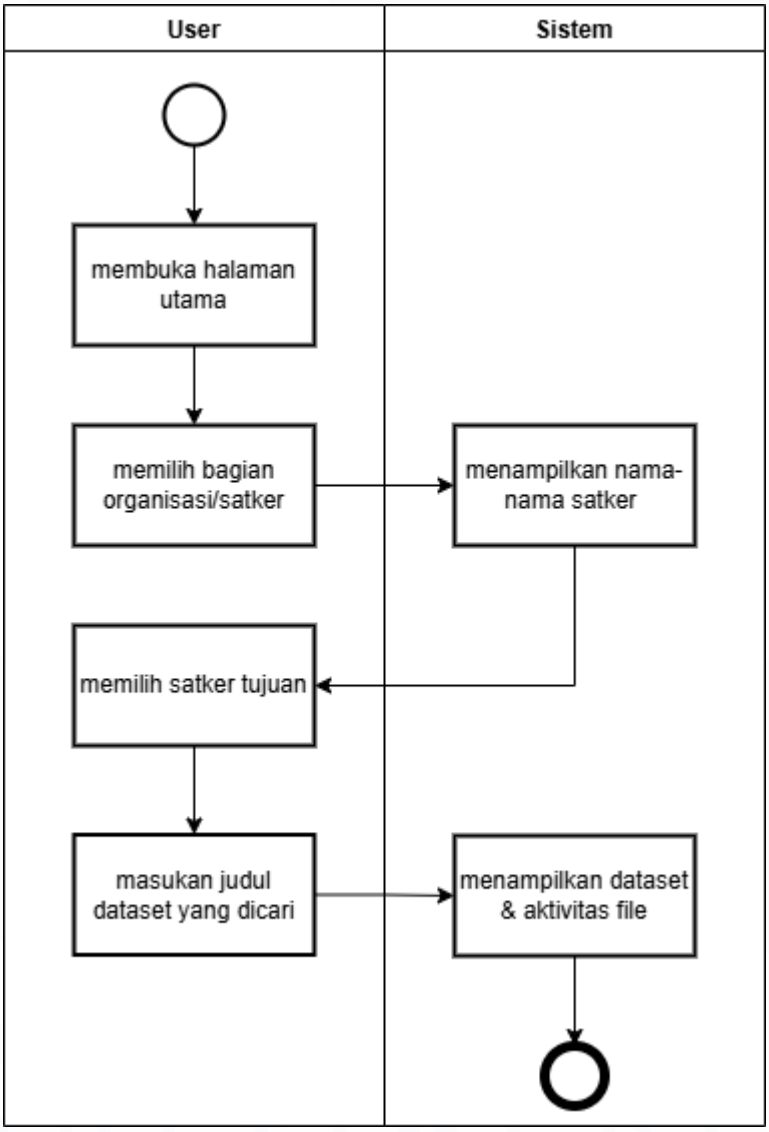
sistem pada website ini maka halaman ini yang dibutuhkan oleh admin untuk menambahkan satker dan operator baru.



Gambar 3. 17 Activity diagram request dataset

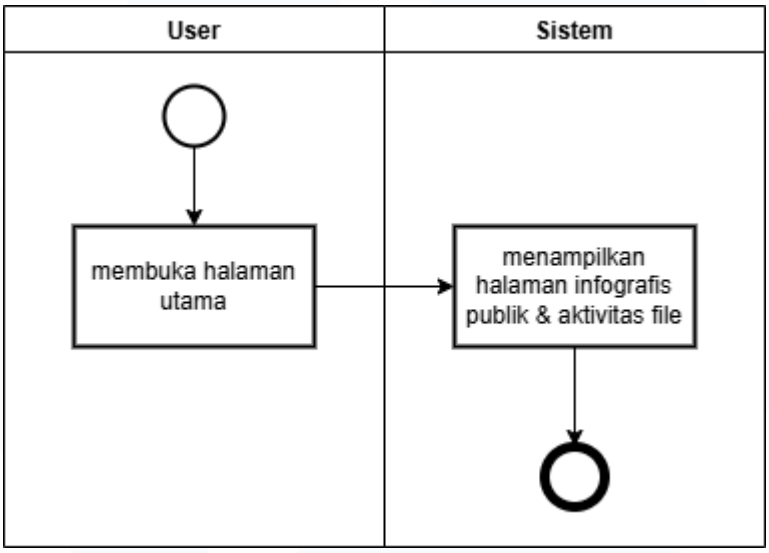
Gambar 3.19 activity diagram ini, menggambarkan bahwa *user* meminta dataset yang tidak dapat diakses secara terbuka dengan mengisi formulir permintaan data kepada admin. Kemudian sistem memvalidasi permintaan dengan

mengirimkan notifikasi kepada admin dan admin meninjau permintaan. Dalam hal ini dikatakan jika permintaan ditolak maka dataset tersebut tidak dapat diberikan dan jika permintaan disetujui, maka admin memproses data dan mengirimkan melalui *email* dengan sistem mengarsipkan permintaan. Kemudian user menerima notifkasi *email* beserta pemberian dataset yang diminta. Dan *user* dapat mengunduh dataset tersebut.



Gambar 3. 18 Activity diagram melihat dan mendownload dataset

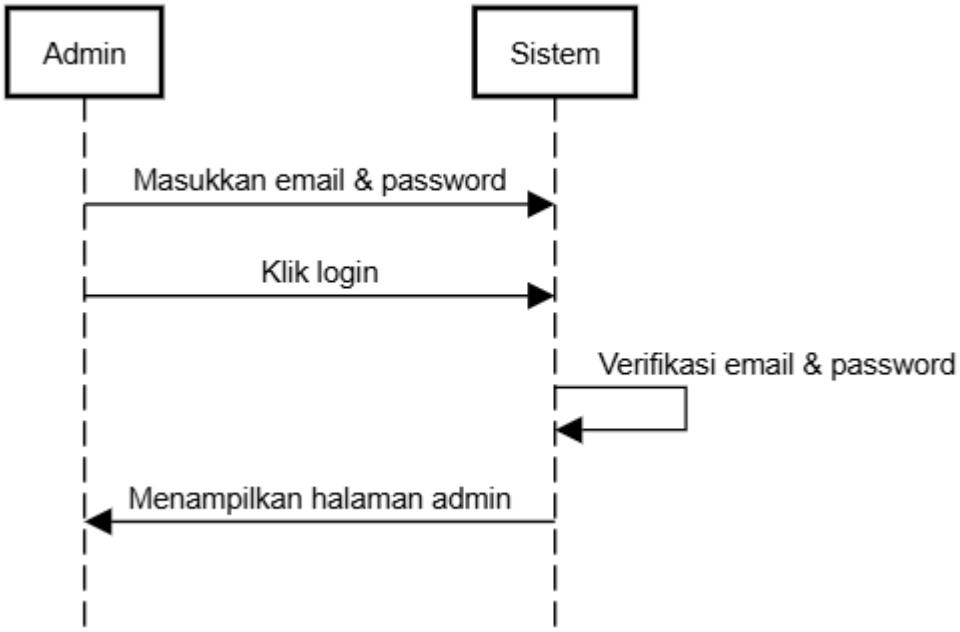
Pada Gambar 3.20 activity diagram ini, *user* dapat melihat aktivitas *file* dataset seperti sudah diliat dan sudah di *download* pada pengaksesan di sistem. Adapun data yang sudah ada dimenu sesuai dengan filter satker yang dituju oleh user.



Gambar 3. 19 Activity diagram melihat dan mendownload infografis

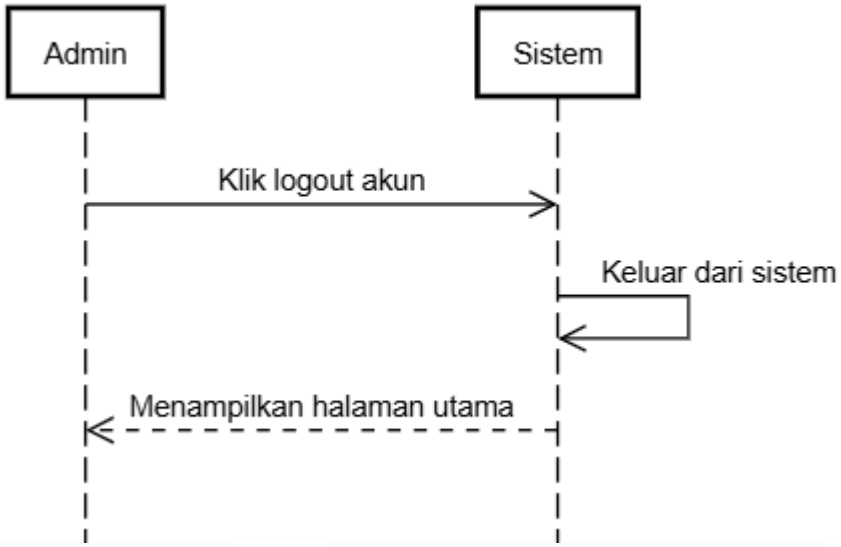
Pada Gambar 3.21 activity diagram ini, user juga dapat melihat aktivitas *file* infografis seperti sudah diliat dan sudah di *download* pada pengaksesan di sistem Adapun pada menu yang sudah difilter sesuai dengan satker masing-masing.

Selanjutnya merupakan Sequence diagram yang merupakan Gambaran alur dan Interaksi antar objek/ User dan juga system, sequence diagram dibagi menjadi 10 macam gambar yang tentunya sesuai dengan Activity diagram yang telah dibuat berikut merupakan Sequence diagram dan penjelesannya



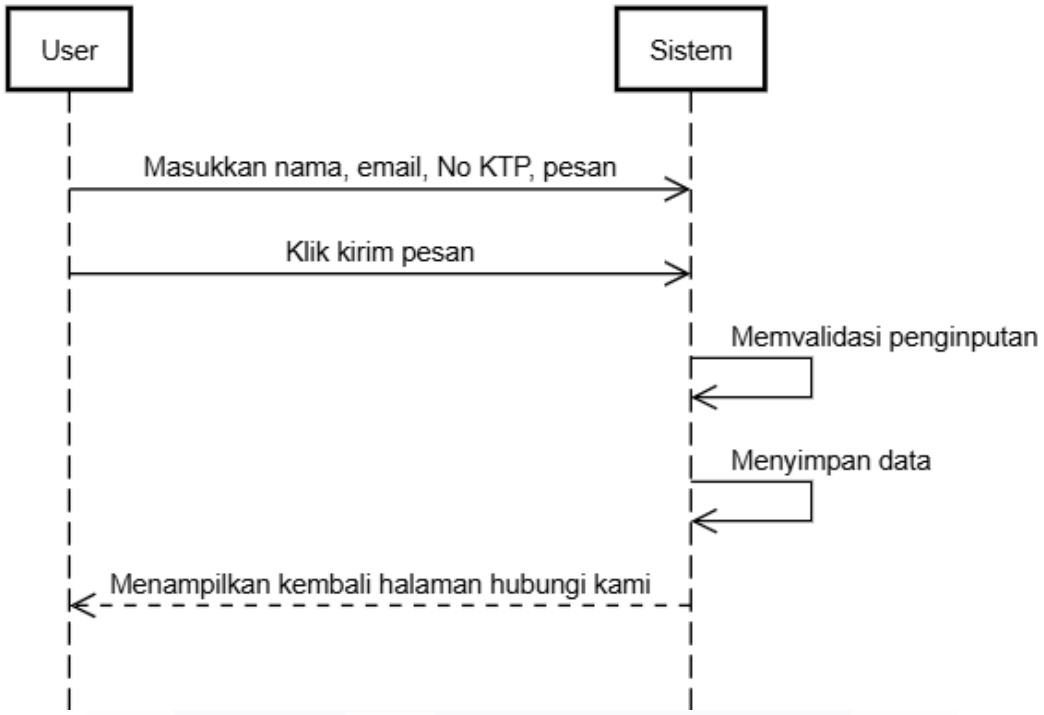
Gambar 3. 20 Sequence diagram login

Gambar 3.22 tersebut merupakan diagram urutan (sequence diagram) yang menggambarkan proses login seorang admin ke dalam sistem. Proses dimulai ketika admin memasukkan email dan password, lalu menekan tombol login untuk mengirimkan data tersebut ke sistem. Setelah menerima permintaan login, sistem melakukan verifikasi terhadap email dan password yang dimasukkan. Jika data yang diberikan valid, sistem akan menampilkan halaman admin sebagai respon atas keberhasilan login. Diagram ini berfungsi untuk memperjelas alur interaksi antara admin dan sistem, serta membantu pengembangan sistem dalam memahami proses autentikasi pengguna secara visual.



Gambar 3. 21 Sequence diagram logout

Gambar 3.23 tersebut menunjukkan sequence diagram yang menggambarkan alur proses logout dalam sebuah sistem dengan dua aktor utama yaitu Admin sebagai pengguna dan Sistem sebagai aplikasi yang memproses permintaan. Prosesnya dimulai ketika Admin mengirimkan permintaan "Klik logout akun" ke Sistem, kemudian Sistem memproses permintaan tersebut dengan mengeluarkan Admin dari sistem ("Keluar dari sistem"), dan terakhir Sistem mengirimkan konfirmasi kembali ke Admin dengan menampilkan "halaman utama" sebagai tanda bahwa proses logout telah berhasil dilakukan. Diagram ini menggunakan garis putus-putus untuk menunjukkan lifeline dari setiap aktor dan panah solid untuk menunjukkan aliran komunikasi yang dibaca secara berurutan dari atas ke bawah sesuai dengan kronologi waktu kejadian.

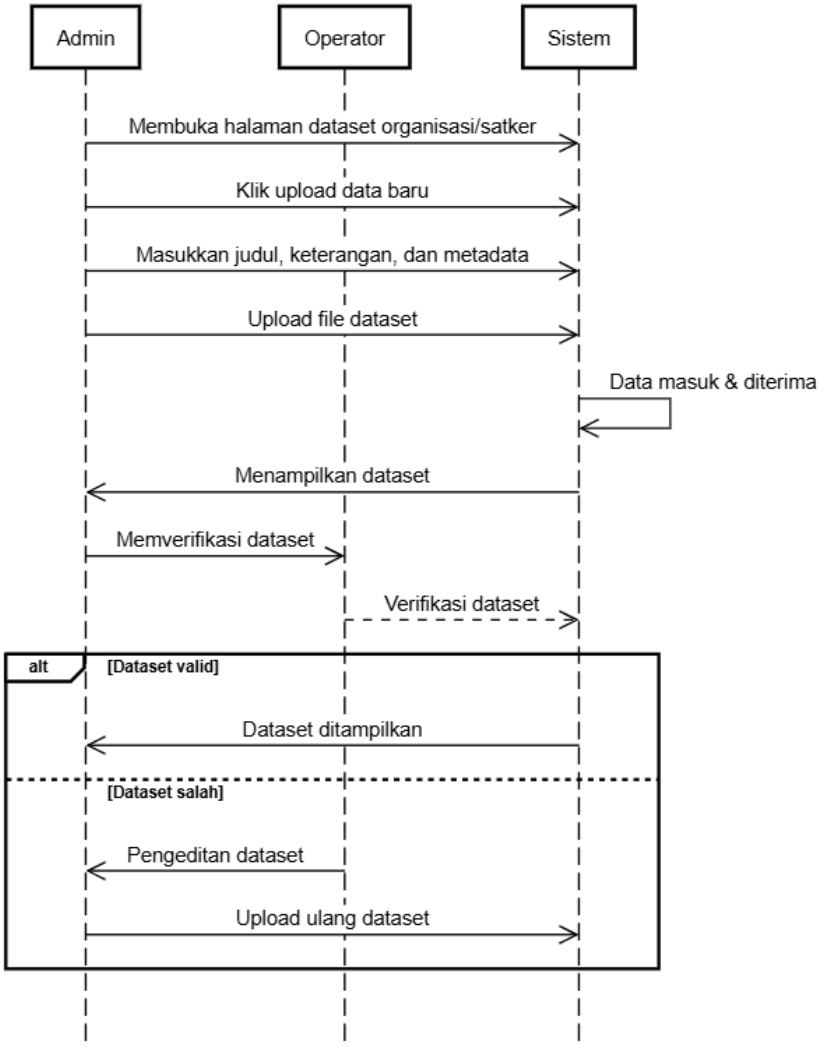


Gambar 3. 22 Sequence diagram penyampaian pesan

Gambar 3.24 tersebut menunjukkan sequence diagram yang menggambarkan alur interaksi antara User dan Sistem dalam sebuah aplikasi contact atau messaging. Prosesnya dimulai ketika User memasukkan data berupa nama, email, nomor KTP, dan pesan, kemudian mengklik tombol "Kirim pesan" untuk mengirimkan informasi tersebut ke Sistem. Setelah menerima data, Sistem melakukan validasi untuk memastikan format dan kelengkapan data sudah sesuai, lalu menyimpan data tersebut ke dalam database atau storage. Akhirnya, Sistem mengirimkan feedback berupa halaman konfirmasi atau pesan konfirmasi kembali kepada User untuk memberitahu bahwa proses telah berhasil diselesaikan.

Diagram ini menggunakan garis putus-putus untuk menunjukkan lifeline dari setiap aktor dan panah untuk menunjukkan arah komunikasi, dengan sequence yang berjalan dari atas ke bawah secara kronologis. Sequence diagram seperti ini biasanya digunakan dalam tahap perancangan sistem untuk memvisualisasikan bagaimana komponen-komponen sistem berinteraksi dalam menyelesaikan sebuah

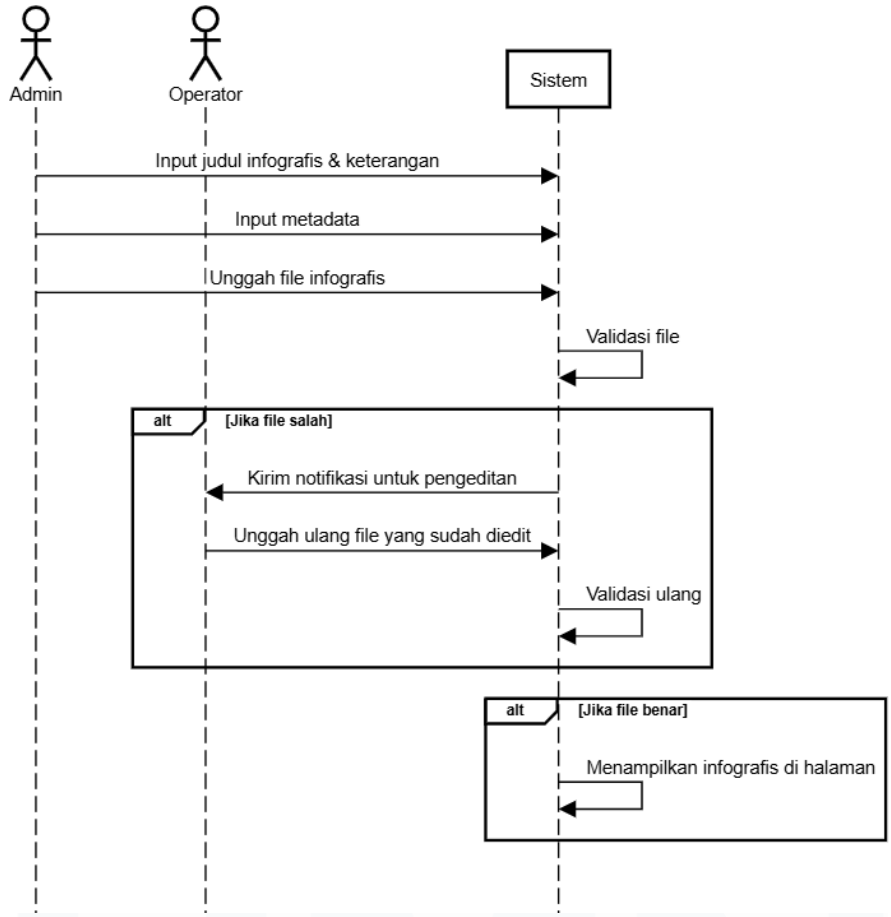
use case atau skenario tertentu, sehingga memudahkan developer dan stakeholder untuk memahami alur proses bisnis yang akan diimplementasikan.



Gambar 3. 23 Sequence diagram upload dataset

Gambar 3.25 menggambarkan alur kerja sistem manajemen dataset yang melibatkan tiga entitas utama: Admin, Operator, dan Sistem. Proses dimulai ketika Admin membuka halaman dataset organisasi/satker, kemudian mengklik tombol untuk upload data baru. Admin memasukkan informasi penting seperti judul dataset, keterangan, dan metadata yang diperlukan. Setelah itu, Admin melakukan

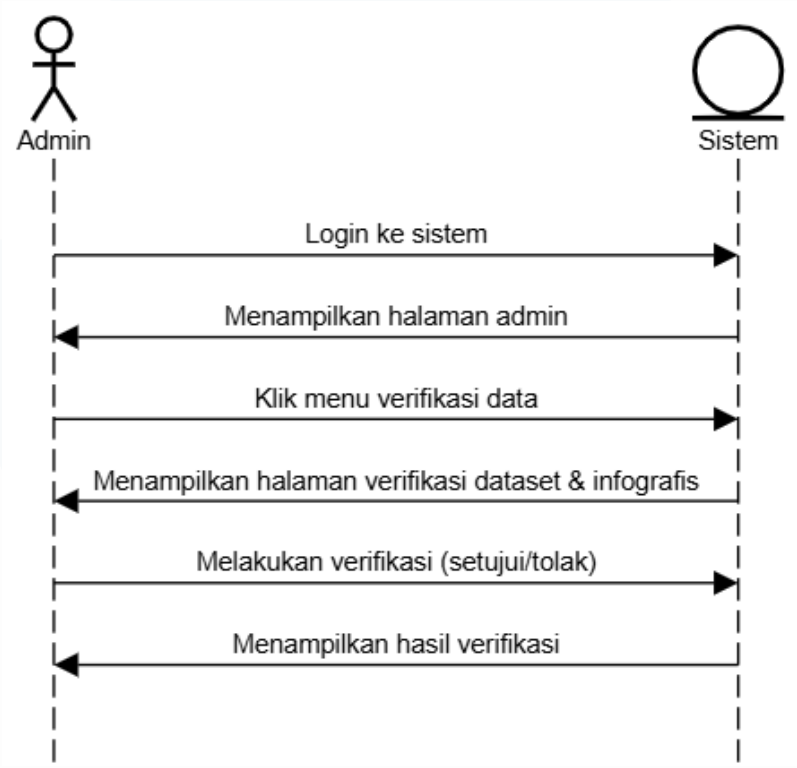
upload file dataset ke dalam sistem, dan sistem akan menerima serta menyimpan data tersebut.



Gambar 3. 24 Sequence diagram upload infografis

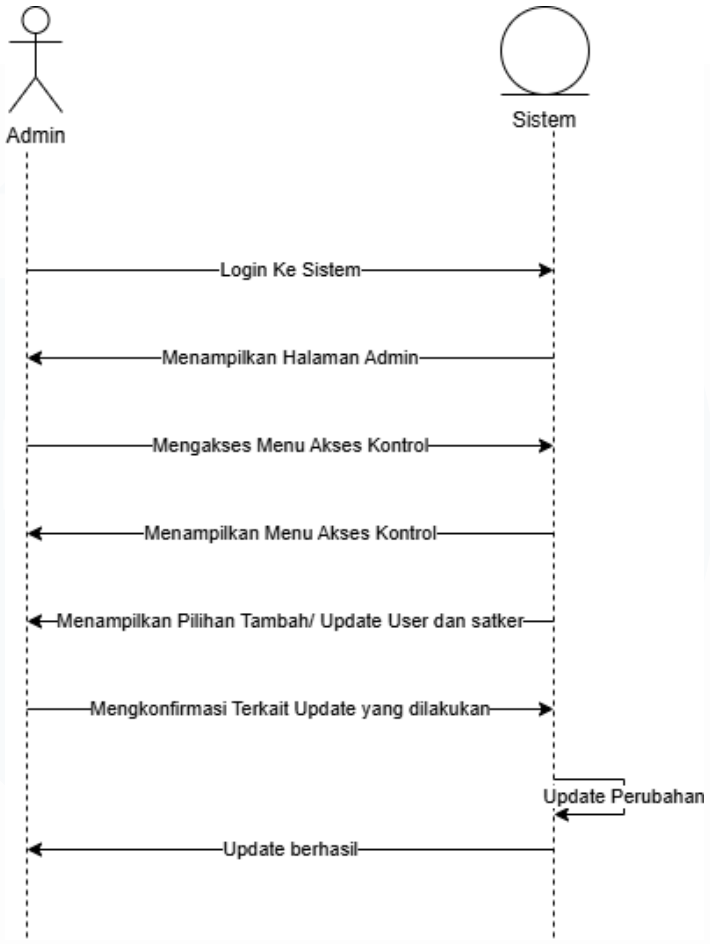
Gambar 3.26 ini menunjukkan sequence diagram untuk sistem pengelolaan file infografis yang melibatkan tiga entitas utama: Admin, Operator, dan Sistem. Proses dimulai dengan Admin menginput judul infografis dan keterangan ke sistem, dilanjutkan dengan Operator menginput metadata dan mengunggah file infografis. Setelah file berhasil diunggah, sistem melakukan validasi file dan memberikan konfirmasi kepada Admin. Admin kemudian dapat mengirim notifikasi untuk pengeditan dan mengunggah ulang file yang telah diedit, yang kembali divalidasi oleh sistem. Terakhir, Admin dapat melihat file terbaru dan sistem menampilkan

infografis di halaman utama, menciptakan alur kerja yang terstruktur untuk manajemen konten infografis dari input awal hingga publikasi final.



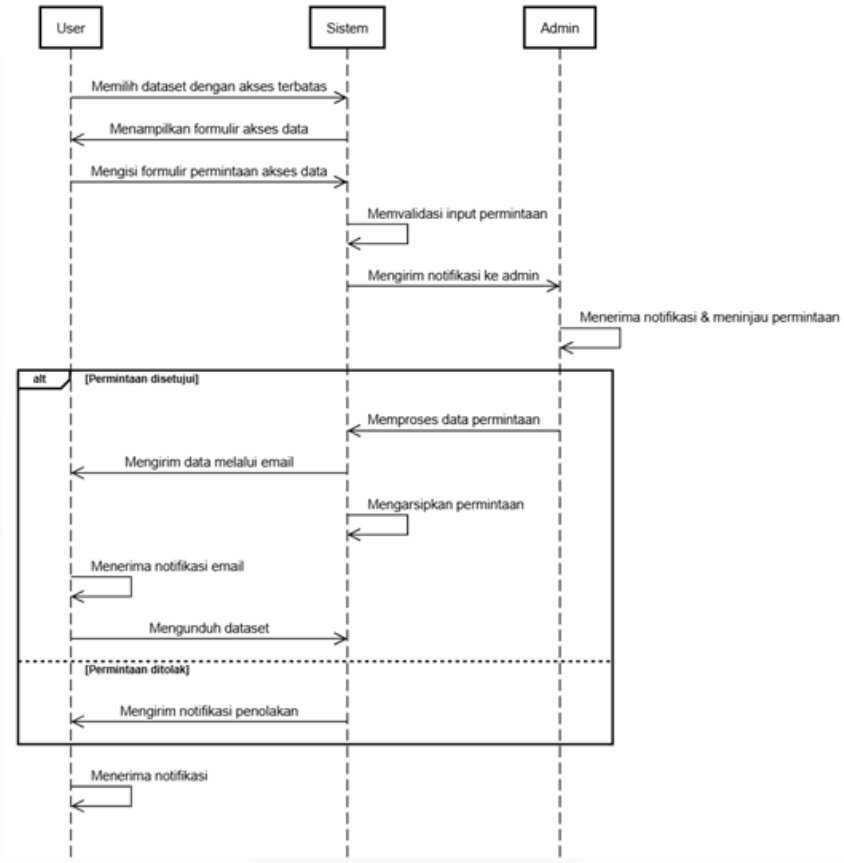
Gambar 3. 25 Sequence diagram verifikasi data

Gambar 3.27 tersebut menunjukkan diagram sequence yang menggambarkan proses verifikasi data antara Admin dan Sistem. Prosesnya dimulai dengan Admin melakukan login ke sistem, kemudian sistem menampilkan halaman admin. Setelah itu, Admin mengklik menu verifikasi data yang memicu sistem untuk menampilkan halaman verifikasi dataset dan infografis. Admin kemudian melakukan proses verifikasi dengan menyetujui atau menolak data, dan terakhir sistem menampilkan hasil verifikasi kepada Admin. Diagram ini mengilustrasikan alur kerja standar untuk proses validasi data dalam sebuah sistem administrasi.



Gambar 3. 26 Sequence diagram akses kontrol

Gambar 3.28 tersebut menunjukkan diagram sequence yang menggambarkan alur interaksi antara Admin dan Sistem dalam proses pengelolaan user dan akses kontrol. Proses dimulai dengan Admin melakukan login ke sistem, kemudian sistem menampilkan halaman admin yang memberikan akses untuk mengatur menu akses kontrol. Admin dapat menampilkan dan mengelola menu akses kontrol, serta melakukan update data user beserta informasi terkait seperti menambah atau mengubah user dan kata sandi. Setelah Admin melakukan perubahan, sistem akan mengkonfirmasi update yang telah dilakukan dan memberikan feedback berupa status "Update berhasil" kepada Admin, menunjukkan bahwa proses pengelolaan user dan akses kontrol telah selesai dilaksanakan dengan sukses.

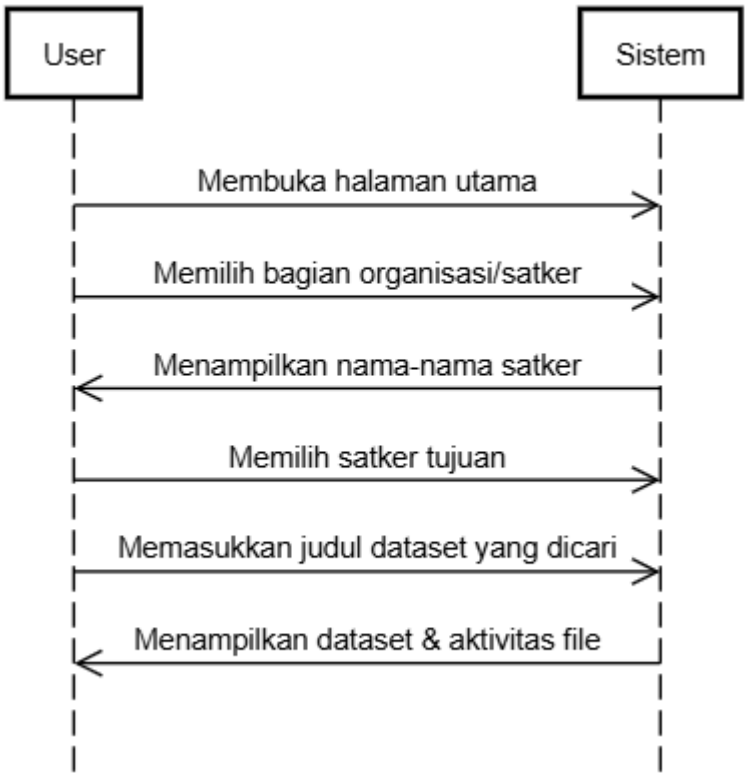


Gambar 3. 27 Sequence diagram request data

Gambar 3.29 merupakan diagram yang menggambarkan sistem akses data dengan tiga aktor utama: User, System, dan Admin. Proses dimulai ketika User meminta dataset dengan akses terbatas, kemudian System menampilkan formulir akses data dan User mengisi formulir permintaan akses data. Setelah itu, System memvalidasi input permintaan dan mengirim notifikasi kepada Admin untuk ditinjau dan disetujui.

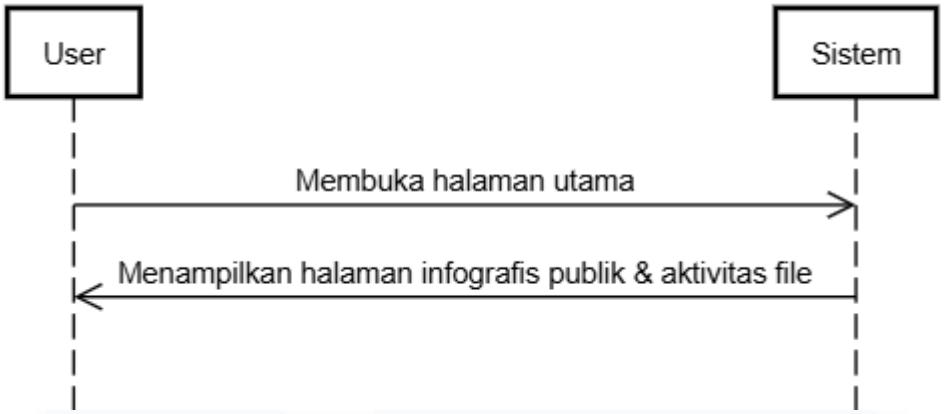
Dalam tahap selanjutnya, Admin menerima notifikasi dan meninjau permintaan tersebut. Jika disetujui, System akan memproses data permintaan dan mengirim data melalui email kepada User. User kemudian menerima notifikasi email dan mengunduh dataset yang diminta. Proses ini juga mencakup pengiriman notifikasi penolakan jika permintaan tidak disetujui, serta notifikasi kepada User

ketika seluruh proses selesai. Sistem ini dirancang untuk memastikan kontrol akses yang tepat terhadap data sensitif melalui persetujuan administratif.



Gambar 3. 28 Sequence diagram melihat dan mendownload dataset

Gambar 3.30 menunjukkan interaksi sequence antara User dan System dalam proses pencarian dataset. Alur dimulai dengan User membuka halaman utama, kemudian memilih bagian organisasi atau satker yang diinginkan. System merespons dengan menampilkan nama-nama satker yang tersedia, dan User memilih satker tujuan. Selanjutnya User memasukkan judul dataset yang ingin dicari, dan System akan menampilkan dataset beserta file aktivitas yang sesuai dengan kriteria pencarian tersebut. Proses ini merupakan alur pencarian data yang memungkinkan User untuk menemukan dataset spesifik berdasarkan organisasi/satker dan judul yang diinginkan.



Gambar 3. 29 Sequence diagram melihatr dan mendownload infografis

Gambar 3.31 Merupakan activity yang dilakukan user public Ketika ingin mengakses infografis yang terdapat di beberapa satker maka didalam menu tersebut terdapat statistic terkait data infografis yang dituju oleh user.

Sistem backend untuk modul data upload telah diimplementasikan dengan fokus pada keamanan dan validasi data. Implementasi ini mencakup proses CRUD (Create, Read, Update, Delete) untuk tabel data_upload dengan fitur utama sebagai berikut:

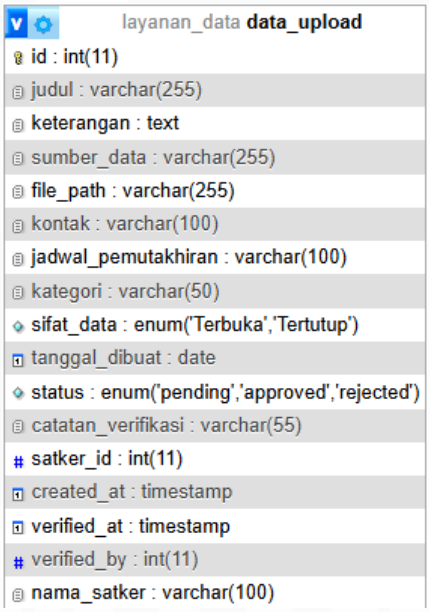
Modul upload dataset ini memungkinkan operator dari berbagai satuan kerja (satker) untuk mengunggah file dataset dalam format CSV dan XLSX dengan batas ukuran maksimal. Sistem menyediakan validasi ketat terhadap format file yang diizinkan untuk mencegah upload file berbahaya. Flow proses dimulai dengan user melakukan login, kemudian sistem mengambil informasi user termasuk satker dari database untuk mencegah manipulasi data antar satker.

Proses upload file menggunakan pendekatan dua tahap: file pertama disimpan ke direktori sementara (/uploads/temp), kemudian dipindahkan ke direktori final (/uploads/datasets) setelah semua validasi berhasil. Data metadata dataset seperti judul, keterangan, sumber data, kontak, sifat data, jadwal

pemutakhiran, dan kategori disimpan ke dalam database dengan status awal 'pending' untuk menunggu proses verifikasi oleh admin.

Implementasi menggunakan prepared statements untuk mencegah SQL injection, serta validasi tipe file dan ukuran untuk mencegah serangan melalui file upload. Sistem juga menangani berbagai skenario kesalahan seperti kegagalan upload, format file tidak sesuai, dan masalah dalam penyimpanan database, dengan memberikan pesan error yang informatif kepada pengguna.

Fitur CRUD yang telah diimplementasikan memungkinkan pengguna melihat daftar dataset yang telah diupload beserta status verifikasinya (pending, approved, atau rejected), serta catatan verifikasi dari admin. Pendekatan keamanan berlapis ini memastikan integritas data yang diupload ke dalam sistem Portal Data satu Kementerian Pertahanan.



Gambar 3. 30 Tabel data_upload

Gambar 3.32 merupakan Tabel data_upload dalam basis data layanan_data digunakan untuk menyimpan informasi terkait unggahan data oleh satuan kerja, mencakup berbagai atribut penting seperti id sebagai identifikasi unik, judul, keterangan, sumber_data, file_path, dan kontak yang menjelaskan rincian data dan

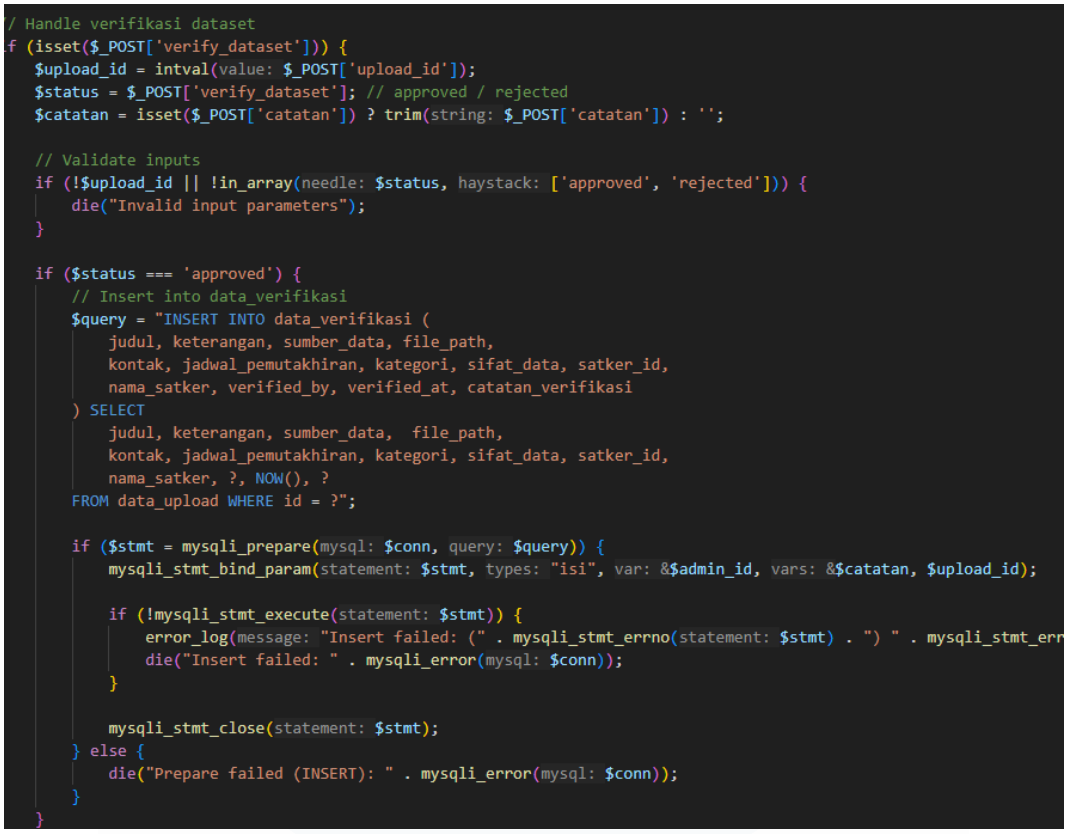
pengunggahnya. Tabel ini juga mencatat jadwal_pemutakhiran, kategori, serta sifat_data yang menentukan apakah data bersifat 'Terbuka' atau 'Tertutup'. Kolom tanggal_dibuat mencatat waktu pembuatan, sementara status menunjukkan tahapan verifikasi data—'pending', 'approved', atau 'rejected'—yang dapat disertai catatan_verifikasi. Identitas satuan kerja tercatat melalui satker_id dan nama_satker, sedangkan proses verifikasi direkam lewat verified_at, verified_by, dan created_at. Struktur ini mendukung sistem manajemen data yang terintegrasi, akuntabel, dan siap pakai untuk keperluan publik atau internal.

1. Pemeriksaan Otentikasi dan Otorisasi

```php
if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'admin') {
    // Redirect atau tampilkan pesan
    header(header: "Location: login.php");
    exit();
}



// Check if user is logged in
if (!isset($_SESSION['user_id'])) {
    header(header: "Location: login.php");
    exit;
}
```

Gambar 3. 31 Code session role verifikasi

Gambar 3.33 terssebut merupakan kode yang memastikan bahwa hanya pengguna yang sudah login dan memiliki peran sebagai 'admin' yang dapat mengakses halaman, dengan syarat bahwa user_id telah tersimpan dalam session; jika salah satu dari kondisi tersebut tidak terpenuhi, maka pengguna akan secara otomatis dialihkan ke halaman login.2. Proses Verifikasi Dataset.



```php
// Handle verifikasi dataset
if (isset($_POST['verify_dataset'])) {
    $upload_id = intval(value: $_POST['upload_id']);
    $status = $_POST['verify_dataset']; // approved / rejected
    $catatan = isset($_POST['catatan']) ? trim(string: $_POST['catatan']) : '';

    // Validate inputs
    if (!$upload_id || !in_array(needle: $status, haystack: ['approved', 'rejected'])) {
        die("Invalid input parameters");
    }

    if ($status === 'approved') {
        // Insert into data_verifikasi
        $query = "INSERT INTO data_verifikasi (
            judul, keterangan, sumber_data, file_path,
            kontak, jadwal_pemutakhiran, kategori, sifat_data, satker_id,
            nama_satker, verified_by, verified_at, catatan_verifikasi
        ) SELECT
            judul, keterangan, sumber_data,  file_path,
            kontak, jadwal_pemutakhiran, kategori, sifat_data, satker_id,
            nama_satker, ?, NOW(), ?
        FROM data_upload WHERE id = ?";

        if ($stmt = mysqli_prepare(mysql: $conn, query: $query)) {
            mysqli_stmt_bind_param(statement: $stmt, types: "isi", var: &$admin_id, vars: &$catatan, $upload_id);

            if (!mysqli_stmt_execute(statement: $stmt)) {
                error_log(message: "Insert failed: (" . mysqli_stmt_errno(statement: $stmt) . ") " . mysqli_stmt_err
                die("Insert failed: " . mysqli_error(mysql: $conn));
            }

            mysqli_stmt_close(statement: $stmt);
        } else {
            die("Prepare failed (INSERT): " . mysqli_error(mysql: $conn));
        }
    }
```

Gambar 3. 32 Code verifikasi dataset

Gambar 3.34 merupakan kode PHP pada gambar tersebut berfungsi untuk menangani proses verifikasi dataset yang dikirim melalui metode POST. Pertama, kode memeriksa apakah parameter verify_dataset ada dalam request. Jika ada, nilai upload_id, status (antara 'approved' atau 'rejected'), dan catatan diambil dari input. Validasi dilakukan untuk memastikan upload_id tidak kosong dan status valid. Jika tidak memenuhi syarat, proses dihentikan dengan pesan error.

Jika statusnya 'approved', maka data dari tabel data_upload akan disalin ke tabel data_verifikasi menggunakan perintah SQL INSERT INTO ... SELECT. Query ini dipersiapkan menggunakan mysqli_prepare, lalu di-bind parameter admin_id, catatan, dan upload_id. Jika eksekusi gagal, maka pesan error akan dicatat ke log dan eksekusi dihentikan dengan die(). Jika sukses, statement ditutup dengan mysqli_stmt_close. Jika query tidak bisa dipersiapkan, error juga akan ditampilkan.

3. Proses Verifikasi Infografis

erifikasi.php > ...

```php
// Handle verifikasi infografis
if (isset($_POST['verify_infografis'])) {
    $upload_id = $_POST['upload_id'];
    $status = $_POST['verify_infografis']; // approved / rejected
    $catatan = isset($_POST['catatan']) ? $_POST['catatan'] : '';

    if ($status === 'approved') {
        $query = "INSERT INTO infografis_verifikasi (
            judul, keterangan, sumber_data, file_path,
            kontak, jadwal_pemutakhiran, kategori, sifat_data, satker_id,
            nama_satker, verified_by, verified_at, catatan_verifikasi
        ) SELECT
            judul, keterangan, sumber_data, file_path,
            kontak, jadwal_pemutakhiran, kategori, sifat_data, satker_id,
            nama_satker, ?, NOW(), ?
        FROM infografis_upload WHERE id = ?";

        $stmt = mysqli_prepare(mysql: $conn, query: $query);

        if (!$stmt) {
            die("Prepare failed (INSERT infografis_verifikasi): " . mysqli_error(mysql: $conn));
        }

        mysqli_stmt_bind_param(statement: $stmt, types: "isi", var: &$admin_id, vars: &$catatan, $upload_id);
        mysqli_stmt_execute(statement: $stmt);
    }
```

Gambar 3. 33 Code verifikasi infografis

Gambar 3.35 merupakan Proses verifikasi infografis mirip dengan verifikasi dataset, yaitu dimulai dengan menerima ID upload, status, dan catatan verifikasi melalui metode POST. Jika status yang diterima adalah 'approved', maka sistem akan menyalin data ke tabel *infografis_verifikasi*. Selanjutnya, sistem akan memperbarui status, nama admin yang melakukan verifikasi, waktu verifikasi, dan catatan ke tabel *infografis_upload*. Seluruh proses ini menggunakan *prepared statement* untuk menjaga keamanan data, dan setelah selesai, pengguna akan diarahkan kembali ke halaman verifikasi.

4. Mengambil Data Pending

```php
// Ambil data dataset pending
$query_datasets = "SELECT * FROM data_upload WHERE status = 'pending'";
$result_datasets = mysqli_query(mysql: $conn, query: $query_datasets);

if (!$result_datasets) {
    die("Query Error (datasets): " . mysqli_error(mysql: $conn));
}

// Ambil data infografis pending
$query_infografis = "SELECT * FROM infografis_upload WHERE status = 'pending'";
$result_infografis = mysqli_query(mysql: $conn, query: $query_infografis);

if (!$result_infografis) {
    die("Query Error (infografis): " . mysqli_error(mysql: $conn));
}

// Get verification counts
$query_counts = "SELECT
                    (SELECT COUNT(*) FROM data_upload WHERE status = 'pending') as dataset_pending,
                    (SELECT COUNT(*) FROM infografis_upload WHERE status = 'pending') as infografis_pending,
                    (SELECT COUNT(*) FROM data_verifikasi) as dataset_verified,
                    (SELECT COUNT(*) FROM infografis_verifikasi) as infografis_verified";
$result_counts = mysqli_query(mysql: $conn, query: $query_counts);
$counts = mysqli_fetch_assoc(result: $result_counts);
?>
```

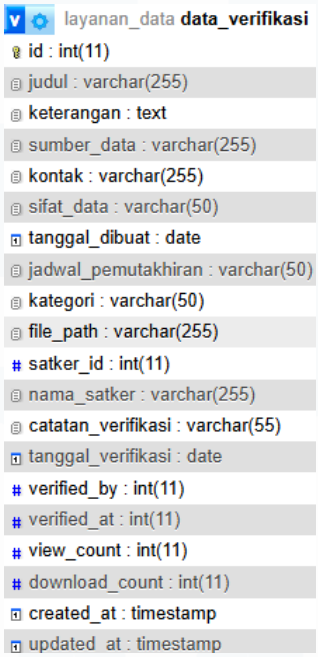
Gambar 3. 34 Code SELECT data pending

Gambar 3.36 kode ini berfungsi untuk mengambil seluruh data yang terdapat pada tabel dataset dan infografis yang memiliki status 'pending', yang berarti data tersebut masih menunggu untuk ditinjau atau diproses lebih lanjut. Setelah data berhasil diambil melalui query, hasilnya akan disimpan dalam variabel tertentu agar dapat digunakan dan ditampilkan pada bagian frontend aplikasi, sehingga pengguna dapat melihat data yang belum diverifikasi. Selain itu, kode ini juga dilengkapi dengan mekanisme penanganan error yang berguna untuk mendeteksi dan menginformasikan apabila terjadi kegagalan dalam proses pengambilan data, sehingga sistem tetap dapat berjalan dengan baik dan memberikan umpan balik yang sesuai kepada pengembang atau pengguna.

Sistem ini mengimplementasikan proses verifikasi data yang diunggah dengan mengutamakan keamanan, kejelasan alur, dan pengalaman pengguna. Fitur utamanya meliputi kontrol akses berbasis peran yang membatasi proses verifikasi hanya untuk admin, penyimpanan data terverifikasi di tabel terpisah (dual storage), serta mekanisme persetujuan atau penolakan yang disertai catatan verifikasi.

Selain itu, sistem mencatat audit berupa siapa dan kapan proses verifikasi dilakukan, menggunakan prepared statement untuk menjaga keamanan dari serangan injeksi, menangani error dengan baik, serta menyajikan statistik verifikasi

untuk analisis lebih lanjut. Dari sisi pengguna, sistem memberikan pengalaman yang baik melalui redirect dan notifikasi status yang jelas. Alur kerja yang dibentuk memungkinkan operator untuk mengunggah data, kemudian admin melakukan verifikasi sebelum data disetujui dan dipindahkan ke tabel final untuk digunakan lebih lanjut.

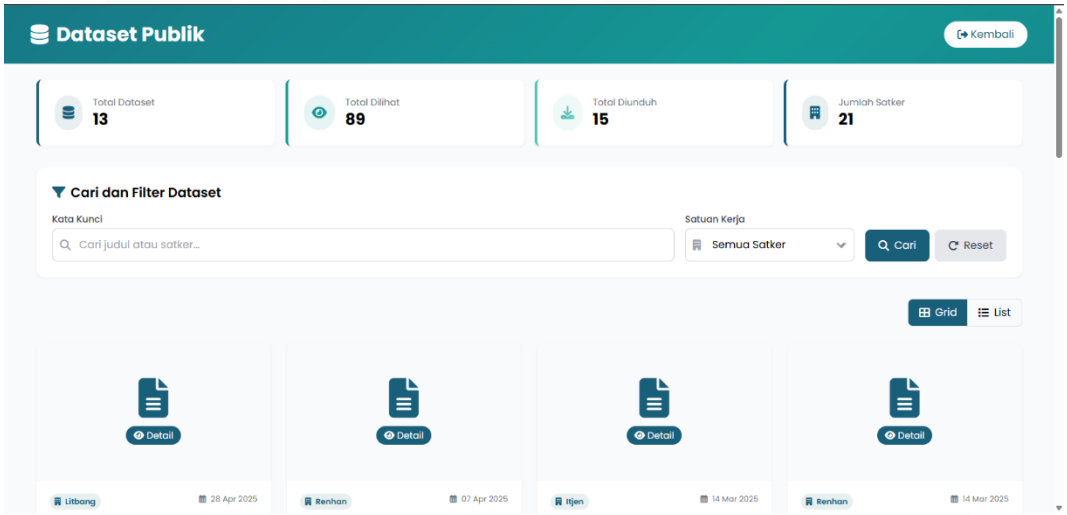


Gambar 3. 35 Tabel data_verifikasi

Gambar 3.37 merupakan Tabel data_verifikasi dalam database layanan_data berfungsi untuk menyimpan informasi terkait proses verifikasi data atau dokumen. Setiap entri memiliki ID unik (id), judul (judul), keterangan (keterangan), sumber data (sumber_data), dan kontak (kontak). Kolom sifat_data menunjukkan tingkat kerahasiaan data, sementara tanggal_dibuat mencatat waktu data dibuat dan jadwal_pemutakhiran menunjukkan jadwal pembaruannya. Data juga dikategorikan melalui kolom kategori, dan dokumen terkait dapat diakses melalui file_path. Informasi mengenai satuan kerja dicatat dalam satker_id dan nama_satker. Proses verifikasi dilengkapi dengan catatan (catatan_verifikasi),

tanggal verifikasi (tanggal_verifikasi), ID verifikator (verified_by), dan waktu verifikasi (verified_at). Aktivitas pengguna tercermin dalam jumlah tampilan (view_count) dan unduhan (download_count). Terakhir, kolom created_at dan updated_at mencatat waktu penciptaan dan pembaruan data dalam sistem.

3.2.4 Minggu ke 4 (Pengembangan Dataset untuk API Publik)



Gambar 3. 36 Tampilan dataset publik

Gambar 3.38 Merupakan tampilan dataset public, Total Dataset diambil berdasarkan jumlah data yang masuk ke table data_verifikasi, total dilihat merupakan view_count yang dijumlah secara keseluruhan, total diunduh berdasarkan download_count yang dijumlahkan, Jumlah satker merupakan jumlah satker yang ada di table satker. Lalu filter satker yang memungkinkan user untuk memilih data berdasarkan satker yang ada.

```php
// Total dataset
$count_query = "SELECT COUNT(*) as total FROM data_verifikasi";
$count_result = mysqli_query(mysql: $conn, query: $count_query);
$total = mysqli_fetch_assoc(result: $count_result)['total'];

// Total views
$views_query = "SELECT SUM(view_count) as total_views FROM data_verifikasi";
$views_result = mysqli_query(mysql: $conn, query: $views_query);
$total_views_row = mysqli_fetch_assoc(result: $views_result);
$total_views = $total_views_row['total_views'] ?? 0; // Use null coalescing to handle NULL

// Total downloads - FIX HERE: using correct result variable
$downloads_query = "SELECT SUM(download_count) as total_downloads FROM data_verifikasi";
$downloads_result = mysqli_query(mysql: $conn, query: $downloads_query); // Corrected from $views_query
$total_downloads_row = mysqli_fetch_assoc(result: $downloads_result);
$total_downloads = $total_downloads_row['total_downloads'] ?? 0; // Use null coalescing to handle NULL

// Total satker
$satker_count_query = "SELECT COUNT(DISTINCT nama_satker) as total_satker FROM data_verifikasi";
$satker_count_result = mysqli_query(mysql: $conn, query: $satker_count_query);
$total_satker = mysqli_fetch_assoc(result: $satker_count_result)['total_satker'];
?>
```

Gambar 3. 37 Code dataset by data_verifikasi

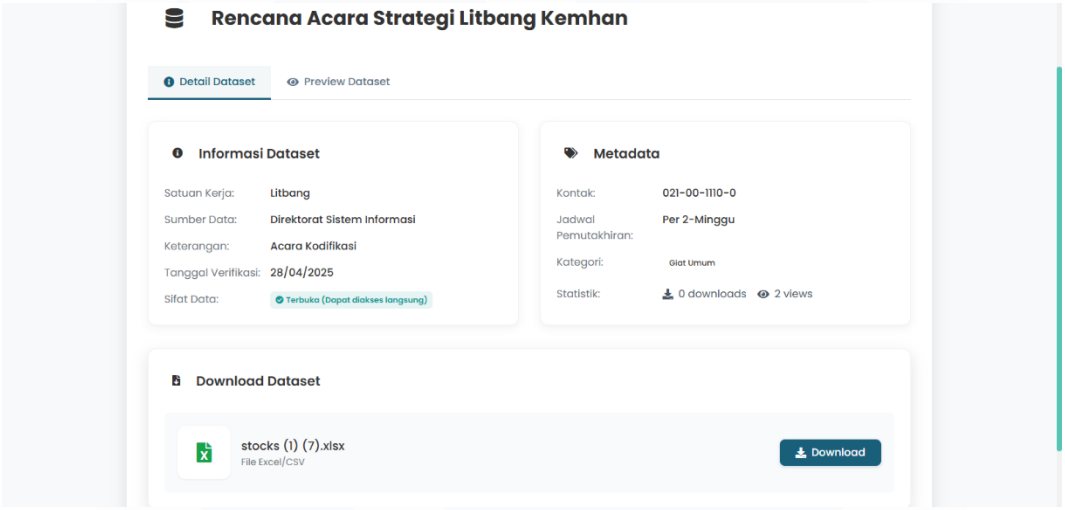
Gambar 3.39 ini dimulai dengan menginklusi file config.php yang berisi informasi koneksi ke database. Ini memungkinkan aplikasi untuk berinteraksi dengan data yang tersimpan. Koneksi ini penting untuk menjalankan berbagai query SQL yang diperlukan untuk mendapatkan informasi dari tabel data_verifikasi.

Fungsi utama dari kode ini adalah menghitung statistik dataset dengan menjalankan beberapa query SQL. Misalnya, untuk menghitung total dataset, total tampilan, dan total unduhan, kode ini menggunakan perintah COUNT dan SUM. Selain itu, terdapat query untuk menghitung jumlah unik satuan kerja (satker) yang terdaftar dalam database. Semua data statistik ini kemudian ditampilkan di bagian depan aplikasi.

Selanjutnya, data dataset diambil menggunakan query yang mengurutkan dataset berdasarkan tanggal verifikasi terbaru. Dengan memanfaatkan fungsi mysqli_fetch_assoc, hasil query diubah menjadi array asosiatif yang memudahkan pengolahan dan penampilan informasi di halaman web. Ini memastikan bahwa pengguna mendapatkan data terbaru dan relevan setiap kali mereka mengakses halaman.

Terakhir, dalam dataset_publik.php ini juga menyertakan fungsi JavaScript untuk menyaring dan memfilter dataset berdasarkan input pengguna. Fungsi

filterDatasets() memungkinkan pengguna untuk mencari berdasarkan kata kunci dan satuan kerja, sementara setViewMode(mode) mengubah tampilan dataset antara mode grid atau list. Dengan cara ini, kode ini tidak hanya menyajikan data, tetapi juga memberikan interaksi yang dinamis bagi pengguna.



Gambar 3. 38 Tampilan detail dataset

Gambar 3.40 Tersebut merupakan tampilan detail dataset yang didalamnya menampilkan meta data yang diambil dari table data_verifikasi Adapun sifat data yang didalamnya bisa berupa Terbuka maupun Terbatas, data terbatas akan dikirimkan dengan mengisi form request data ke Admin untuk admin kemudian memprosesnya.

```php
if (!isset($_GET['id'])) {
    header(header: 'Location: dataset_publik.php');
    exit();
}

$id = intval(value: $_GET['id']);

// Update view count
mysqli_query(mysql: $conn, query: "UPDATE data_verifikasi SET view_count = view_count + 1 WHERE id = $id");

// Get dataset details
$query = "SELECT * FROM data_verifikasi WHERE id = $id";
$result = mysqli_query(mysql: $conn, query: $query);
$data = mysqli_fetch_assoc(result: $result);
```

Gambar 3. 39 Code GET data berdasarkan id

Gambar 3.41 kode PHP pada gambar digunakan untuk menampilkan detail data berdasarkan parameter id yang dikirim melalui URL. Pertama, sistem memeriksa apakah parameter id tersedia; jika tidak, pengguna akan dialihkan ke halaman dataset_publik.php. Setelah itu, nilai id dikonversi menjadi bilangan bulat untuk mencegah serangan injeksi. Kode kemudian memperbarui jumlah tampilan (view count) pada tabel data_verifikasi dengan menambahkan satu ke kolom view_count untuk data yang sesuai. Terakhir, sistem menjalankan query untuk mengambil semua data dari tabel data_verifikasi berdasarkan id tersebut, dan hasilnya disimpan dalam variabel $data.

```php
// Check if the form for restricted access request is submitted
if (isset($_POST['submit_request'])) {
    $nama = mysqli_real_escape_string(mysql: $conn, string: $_POST['nama']);
    $kepentingan = mysqli_real_escape_string(mysql: $conn, string: $_POST['kepentingan']);
    $kontak = mysqli_real_escape_string(mysql: $conn, string: $_POST['kontak']);
    $dataset_id = $id;
    $judul_dataset = mysqli_real_escape_string(mysql: $conn, string: $data['judul']);
    $nama_satker = mysqli_real_escape_string(mysql: $conn, string: $data['nama_satker']);

    // Validate email format
    if (!filter_var(value: $kontak, filter: FILTER_VALIDATE_EMAIL)) {
        $error_message = "Alamat email tidak valid. Silakan masukkan email yang benar.";
    } else {
        // Handle file upload
        $bukti_file = '';
        if (isset($_FILES['bukti']) && $_FILES['bukti']['error'] == 0) {
            $target_dir = "Uploads/requests/";
            if (!file_exists(filename: $target_dir)) {
                mkdir(directory: $target_dir, permissions: 0777, recursive: true);
            }

            $file_extension = strtolower(string: pathinfo(path: $_FILES['bukti']['name'], flags: PATHINFO_EXTENSION));
            $new_filename = 'bukti_' . time() . '_' . uniqid() . '.' . $file_extension;
            $target_file = $target_dir . $new_filename;
```

Gambar 3. 40 Code identifikasi file bukti upload request

Gambar 3.42 merupakan code potongan kode PHP tersebut dimulai dengan pengecekan apakah formulir permintaan akses telah dikirim melalui tombol submit_request. Jika ya, maka data dari formulir seperti nama, kepentingan, kontak,

judul_dataset, dan nama_satker diambil dari variabel $_POST dan dibersihkan menggunakan mysqli_real_escape_string() untuk mencegah serangan SQL Injection. Selain itu, nilai dataset_id diambil dari variabel $id, kemungkinan sebelumnya sudah didefinisikan. Selanjutnya, kode melakukan validasi format email pada kolom kontak menggunakan filter_var() dengan filter FILTER_VALIDATE_EMAIL. Jika email tidak valid, maka akan menampilkan pesan kesalahan.

Jika email valid, maka bagian selanjutnya menangani proses unggah file bukti. Awalnya, variabel $bukti_file diinisialisasi kosong. Lalu dicek apakah file dengan nama bukti ada dan tidak mengalami error. Jika kondisi terpenuhi, maka direktori target (Uploads/requests/) disiapkan. Jika direktori belum ada, maka akan dibuat menggunakan mkdir() dengan izin akses 0777. Ekstensi file diambil dan diubah ke huruf kecil, kemudian dibuat nama file baru yang unik menggunakan fungsi time() dan uniqid(). Terakhir, path lengkap file target disusun dengan menggabungkan nama folder dan nama file baru.

3.2.5 Minggu ke 5 (Kategorisasi Infografis)

Gambar 3. 41 Tabel infografis_upload

Gambar 3.43 Merupakan table infografis_upload yaitu table yang berisikan data infografis yang dimasukan oleh operator, Setelah Infografis di Verifikasi oleh admin, Maka data yang masuk ke infografis_upload maka akan masuk ke infografis_verifikasi, dengan isi tabel penambahan seperti view_count dan download_count sebagai fitur baru, untuk operator dalam melihat statistic data mereka.

```php
// Move the file to the final destination
$final_path = __DIR__ . '/uploads/infografis/' . basename(path: $file_name);
if (!is_dir(filename: 'uploads/infografis')) {
    mkdir(directory: 'uploads/infografis', permissions: 0777, recursive: true);
}
if (rename(from: $temp_file_path, to: $final_path)) {
    // Insert record into the database
    $query = "INSERT INTO infografis_upload (
        judul, keterangan, sumber_data, kontak, sifat_data,
        jadwal_pemutakhiran, kategori, file_path, satker_id, nama_satker, status
    ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 'pending')";

    $stmt = mysqli_prepare(mysql: $conn, query: $query);
```

Gambar 3. 42 Code Directory file infografis

Gambar 3.44 merupakan Kode tersebut bertujuan untuk memindahkan file yang diunggah ke direktori tujuan di server, yaitu uploads/infografis/, dengan memastikan folder tersebut sudah ada atau membuatnya jika belum tersedia. Nama file ditentukan dengan menggunakan basename, dan path lengkap dibuat dengan __DIR__. Jika folder belum ada, maka dibuat dengan hak akses penuh (0777). Setelah itu, file dipindahkan dari lokasi sementara ke direktori tujuan menggunakan fungsi rename.

Jika pemindahan file berhasil, maka data terkait infografis tersebut disiapkan untuk disimpan ke dalam database. Informasi seperti judul, keterangan, sumber data, kontak, kategori, dan path file disimpan menggunakan query INSERT INTO. Nilai status secara default diisi sebagai 'pending', menandakan file tersebut menunggu proses verifikasi atau persetujuan. Untuk keamanan, query SQL disiapkan dengan mysqli_prepare untuk mencegah SQL injection.

3.2.6 Minggu ke 6 (Infografis Publik, Preview dan Download)

Gambar 3. 43 Tabel infografis_verifikasi

Gambar 3.45 Merupakan Tabel infografis_verifikasi yang merupakan data dari table infografis_upload yang sudah diverifikasi yang kemudian akan masuk ke table infografis_verifikasi dengan penambahan isi table berupa view_count dan download_count serta updated_at.

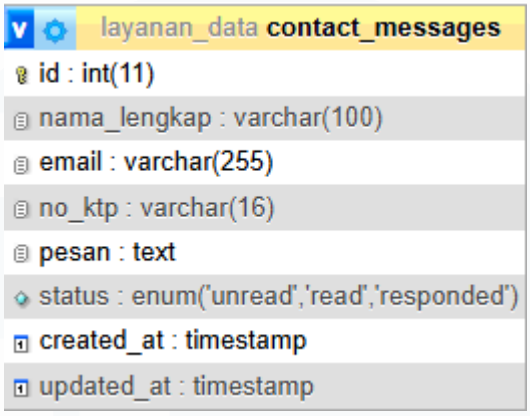
```php
<?php
// Total infografis
$count_query = "SELECT COUNT(*) as total FROM infografis_verifikasi";
$count_result = mysqli_query(mysql: $conn, query: $count_query);
$total = mysqli_fetch_assoc(result: $count_result)['total'];

// Total views
$views_query = "SELECT SUM(view_count) as total_views FROM infografis_verifikasi";
$views_result = mysqli_query(mysql: $conn, query: $views_query);
$total_views_row = mysqli_fetch_assoc(result: $views_result);
$total_views = $total_views_row['total_views'] ?? 0;

// Total downloads
$downloads_query = "SELECT SUM(download_count) as total_downloads FROM infografis_verifikasi";
$downloads_result = mysqli_query(mysql: $conn, query: $downloads_query);
$total_downloads_row = mysqli_fetch_assoc(result: $downloads_result);
$total_downloads = $total_downloads_row['total_downloads'] ?? 0;

// Total satker
$satker_count_query = "SELECT COUNT(DISTINCT nama_satker) as total_satker FROM infografis_verifikasi";
$satker_count_result = mysqli_query(mysql: $conn, query: $satker_count_query);
$total_satker = mysqli_fetch_assoc(result: $satker_count_result)['total_satker'];
?>
```

Gambar 3. 44 Code statistik infografis

Gambar 3.46 merupakan code PHP yang digunakan untuk mengambil data statistik dari tabel infografis_verifikasi, seperti total infografis, total views, total downloads, dan jumlah satuan kerja (satker) yang unik. Pertama, dilakukan query untuk menghitung jumlah total infografis menggunakan SELECT COUNT(*), kemudian hasilnya disimpan ke dalam variabel $total. Untuk menghitung jumlah view, digunakan SELECT SUM(view_count), dan hasilnya disimpan di $total_views, dengan fallback ke 0 jika tidak ada data.

Selanjutnya, untuk mengetahui jumlah total unduhan, digunakan query SELECT SUM(download_count) dan hasilnya disimpan dalam $total_downloads. Terakhir, untuk menghitung berapa banyak satker yang berbeda, digunakan SELECT COUNT(DISTINCT nama_satker) yang menyimpan hasilnya di $total_satker. Semua hasil query ini disiapkan untuk ditampilkan sebagai statistik ringkasan, misalnya pada dashboard aplikasi.

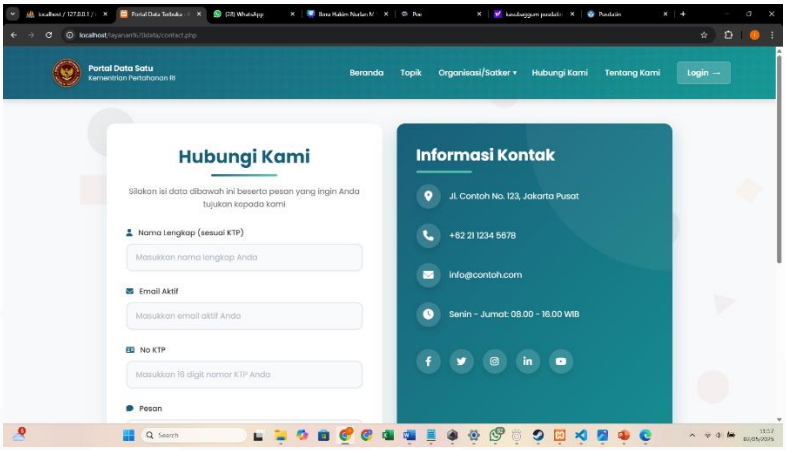
3.2.7 Minggu ke 7 (Sistem Pengaduan)



Gambar 3. 45 Tabel contact_massages

Gambar 3.47 Merupakan Bagian back-end dari kode contact.php dimulai dengan memuat file konfigurasi config.php, yang berisi pengaturan database yang diperlukan untuk menghubungkan aplikasi ke database. Setelah itu, judul halaman ditetapkan dan file header dimasukkan. Dua variabel yaitu $error dan $success diinisialisasi untuk menyimpan pesan yang akan ditampilkan kepada pengguna, tergantung pada hasil pengiriman formulir.

Saat pengguna mengirimkan formulir, kode ini memeriksa apakah metode pengiriman adalah POST. Data yang diterima dari formulir, termasuk nama lengkap, email, nomor KTP, dan pesan, kemudian disanitasi menggunakan fungsi mysqli_real_escape_string() untuk mencegah serangan SQL injection. Selain itu, validasi input dilakukan untuk memastikan semua field terisi, format email yang benar, serta validitas nomor KTP yang harus berisi 16 digit angka.



Gambar 3. 46 Interface contact.php

Gambar 3.48 tersebut merupakan tampilan form yang akan diisi user untuk dikirimkan ke Admin, Jika semua validasi berhasil, proses penyimpanan data dilakukan dengan menggunakan prepared statement untuk meningkatkan keamanan. Kueri SQL INSERT digunakan untuk menyimpan pesan kontak ke dalam tabel contact_messages, di mana parameter kueri diikat dengan tipe data string. Jika eksekusi kueri berhasil, pesan sukses akan ditampilkan kepada pengguna, dan formulir akan dikosongkan. Sebaliknya, jika terjadi kesalahan selama eksekusi, pesan error akan ditampilkan.

Keamanan dalam kode ini diterapkan melalui beberapa langkah, termasuk sanitasi input menggunakan mysqli_real_escape_string(), penggunaan prepared statement untuk mencegah SQL injection, serta validasi input untuk memastikan bahwa data yang dimasukkan sesuai dengan format yang diinginkan. Dengan menerapkan praktik keamanan yang baik, kode back-end ini membantu menjaga integritas data dan melindungi aplikasi dari potensi ancaman.

Gambar 3. 47 Tabel pengaduan

Pada Gambar 3.49 Merupakan Tabel Pengaduan yang digunakan oleh Operator untuk diadukan ke admin terkait dengan system dalam website. Sistem back-end dibangun menggunakan PHP dengan arsitektur prosedural dan MySQL sebagai database. Koneksi database dan konfigurasi utama sistem disimpan dalam file config.php yang di-include di awal script. Sistem menggunakan mekanisme session untuk manajemen autentikasi, dengan fungsi checkLogin() yang memastikan hanya pengguna yang sudah login yang dapat mengakses sistem. Data pengguna seperti user_id dan role disimpan dalam session setelah login berhasil dan digunakan untuk mengontrol akses dan fungsionalitas berdasarkan peran pengguna.

Untuk pengelolaan pengaduan, sistem memiliki fungsi determinePriority() yang secara otomatis menentukan prioritas berdasarkan jumlah kata dalam deskripsi pengaduan menggunakan str_word_count(). Proses submit pengaduan baru dilakukan melalui handler POST yang mencakup validasi input, sanitasi data menggunakan mysqli_real_escape_string(), dan penanganan upload file gambar. Sistem mendukung multiple file upload dengan batasan maksimal 3 gambar, melakukan validasi tipe file (hanya menerima image/jpeg, image/png, image/jpg, image/gif) dan ukuran file (maksimal 2MB per file).

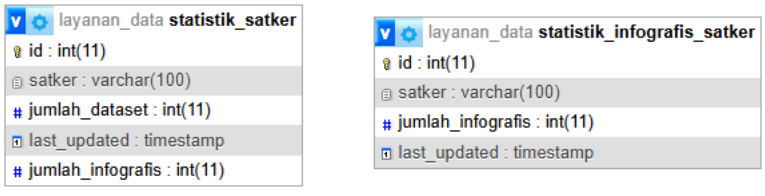
Pada bagian pengelolaan file, sistem menggunakan fungsi-fungsi PHP file handling seperti move_uploaded_file() untuk memindahkan file yang diupload ke direktori tertentu, dengan generate nama file unik menggunakan kombinasi uniqid()

dan timestamp untuk mencegah konflik nama file. File gambar disimpan dalam direktori uploads/pengaduan/ dengan informasi path file disimpan di database dalam format string yang dipisahkan dengan karakter '|' untuk multiple gambar. Sistem juga melakukan pengecekan dan pembuatan direktori secara otomatis jika belum ada.

Query database dirancang untuk mengoptimalkan performa dengan penggunaan JOIN dan ORDER BY yang kompleks untuk mengurutkan pengaduan berdasarkan status dan prioritas. Untuk admin, query menggabungkan tabel pengaduan dengan tabel users untuk mendapatkan informasi lengkap termasuk nama operator. Sistem menggunakan prepared statements untuk query yang membutuhkan parameter dari input pengguna, mencegah SQL injection. Setiap operasi database (insert, update) dilengkapi dengan penanganan error dan feedback melalui session messages yang ditampilkan ke pengguna setelah redirect.

3.2.8 Minggu ke 8 (Dashboard admin dan Statistik Visualisasi)

Laporan ini dimulai dengan pengimporan konfigurasi database melalui include 'config.php'. Proses autentikasi dilakukan dengan memeriksa sesi pengguna menggunakan $_SESSION['user_id']. Jika pengguna belum melakukan login, mereka akan diarahkan ke halaman login. Data pengguna kemudian diambil dari database dengan menggunakan prepared statement untuk memastikan keamanan data dan mencegah potensi serangan SQL injection.

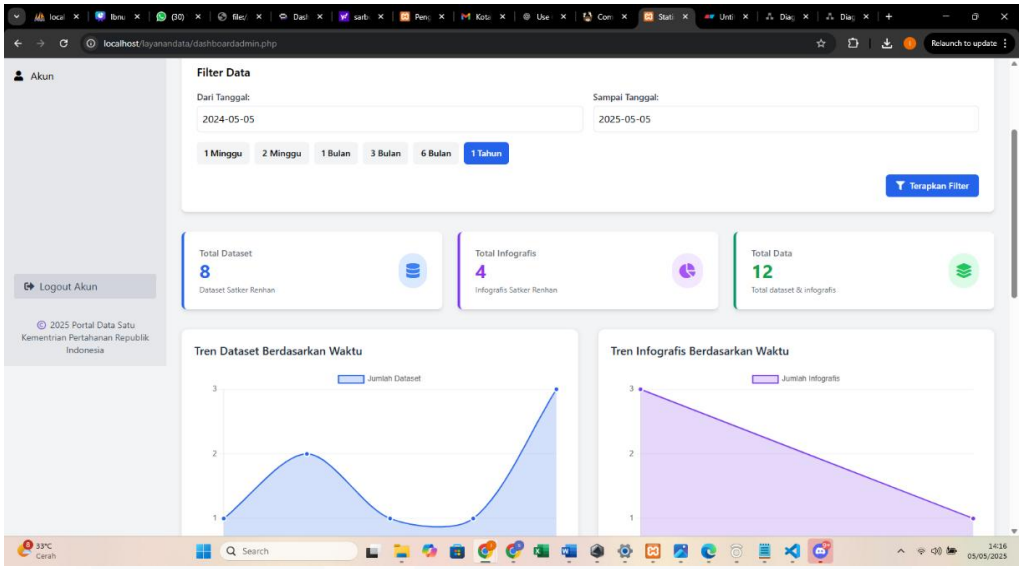


Gambar 3. 48 Tabel statistik data satker

Gambar 3.50 tersebut menampilkan struktur dua tabel dalam basis data dengan skema layanan_data, yaitu statistik_satker dan statistik_infografis_satker. Tabel statistik_satker digunakan untuk menyimpan data statistik berdasarkan satuan kerja (satker), mencakup kolom id sebagai primary key, satker untuk nama satuan kerja, jumlah_dataset untuk jumlah dataset yang dimiliki, jumlah_infografis untuk jumlah infografis yang tersedia, serta last_updated untuk mencatat waktu pembaruan terakhir. Tabel ini memberikan informasi lengkap mengenai dataset dan infografis yang dimiliki setiap satker.

Sementara itu, tabel statistik_infografis_satker lebih terfokus pada data infografis saja, dengan kolom id, satker, jumlah_infografis, dan last_updated. Meskipun kedua tabel memiliki beberapa atribut yang mirip, tabel statistik_satker memuat data yang lebih lengkap karena juga mencakup jumlah dataset. Redundansi kolom jumlah_infografis di kedua tabel dapat menyebabkan inkonsistensi data jika tidak dikelola dengan baik. Oleh karena itu, penting untuk memastikan integritas data antar tabel agar informasi yang ditampilkan tetap akurat.

Selanjutnya, dalam pengambilan data statistik, dilakukan perhitungan jumlah dataset dan infografis yang telah disetujui dengan menggunakan query COUNT. Hasil dari query ini diolah menggunakan mysqli_fetch_assoc untuk mendapatkan jumlah total. Untuk visualisasi data per satuan kerja (satker), dilakukan query yang menghitung jumlah dataset dan infografis dengan menggabungkan tabel satker dan data_upload menggunakan LEFT JOIN. Hasil dari query ini diproses dalam loop dan adisimpan dalam array untuk dikirimkan ke frontend.



Gambar 3. 49 Interface statistik data satker

Gambar 3.51 merupakan Statistik mengenai lima satker teratas juga dihimpun melalui query kompleks yang menggabungkan tiga tabel: satker, data_upload, dan infografis_upload. Dalam proses ini, jumlah dataset dan infografis untuk setiap satker dihitung, diurutkan berdasarkan jumlah total, dan dibatasi pada lima satker teratas. Data ini disimpan dalam array $top_satkers dengan struktur yang memisahkan antara dataset, infografis, dan total, sehingga memudahkan analisis lebih lanjut.

Dari segi keamanan, penggunaan prepared statement sangat penting untuk mencegah SQL injection, sedangkan fungsi htmlspecialchars() digunakan untuk melindungi aplikasi dari serangan XSS saat menampilkan data di frontend. Visualisasi data dilakukan menggunakan Chart.js, yang mencakup pembuatan grafik bar horizontal untuk dataset dan infografis secara terpisah. Selain itu, kartu statistik menampilkan total dataset dan infografis dengan desain yang menarik, sementara tabel detail menyajikan data lengkap statistik per satker dengan dua tabel terpisah untuk dataset dan infografis. Konfigurasi Chart.js diatur untuk responsif dan mudah dibaca, mendukung pemahaman yang lebih baik atas data yang disajikan.

Gambar 3. 50 Code filter by date

Gambar 3.52 kode PHP tersebut dimulai dengan proses penetapan filter waktu default jika parameter filter belum diberikan melalui URL ($_GET). Variabel $filter_days akan diset menjadi 365 hari secara default, yang berarti sistem akan menampilkan data dari 365 hari terakhir. Kemudian, tanggal awal ($date_from) dan tanggal akhir ($date_to) ditentukan berdasarkan filter hari tersebut, dan dijadikan parameter untuk query SQL dengan format waktu lengkap menggunakan kondisi BETWEEN pada kolom created_at.

Selanjutnya, bagian kedua kode melakukan query ke database untuk menghitung statistik jumlah data yang disetujui. Terdapat dua query, yaitu untuk menghitung jumlah dataset ($query_approved_dataset) dan jumlah infografis ($query_approved_infografis) yang memiliki status 'approved' untuk satker_id tertentu dalam rentang waktu yang telah difilter. Nilai yang diambil dari query ini disimpan ke dalam variabel $approved_dataset dan $approved_infografis, lalu dijumlahkan menjadi $total_approved. Selain itu, terdapat juga query tambahan untuk mengambil data dataset dalam format time series harian, yang menghitung jumlah data berdasarkan tanggal pembuatan (created_at) dan digunakan untuk visualisasi tren data.

Selain statistik agregat dan tren harian, sistem juga menampilkan daftar detail data dan infografis yang telah disetujui, termasuk judul, sumber data, kategori, dan tanggal pembuatan. Data ini diambil dalam urutan tanggal terbaru, sehingga memudahkan pengguna untuk memantau dan meninjau kembali kontribusi yang

telah mereka unggah. Dengan demikian, kode ini mendukung transparansi dan pengawasan terhadap kinerja satker dalam menyuplai informasi terverifikasi secara berkala.

```php
// Dataset pagination
$page_dataset  = isset($_GET['page_dataset']) ? (int)$_GET['page_dataset'] : 1;
$start_dataset = ($page_dataset - 1) * $per_page;
$query_count_dataset  = "SELECT COUNT(*) as total FROM data_upload WHERE status = 'approved' AND satker_id = $satker_id $time_filter";
$total_dataset        = mysqli_fetch_assoc(result: mysqli_query(mysql: $conn, query: $query_count_dataset))['total'];
$total_pages_dataset  = ceil(num: $total_dataset / $per_page);
$query_dataset_details = "SELECT id, judul, sumber_data, kategori, created_at, file_path
                          FROM data_upload
                          WHERE status = 'approved' AND satker_id = $satker_id $time_filter
                          ORDER BY created_at DESC
                          LIMIT $start_dataset, $per_page";
$result_dataset_details = mysqli_query(mysql: $conn, query: $query_dataset_details);

// Infografis pagination
$page_infografis  = isset($_GET['page_infografis']) ? (int)$_GET['page_infografis'] : 1;
$start_infografis = ($page_infografis - 1) * $per_page;
$query_count_infografis  = "SELECT COUNT(*) as total FROM infografis_upload WHERE status = 'approved' AND satker_id = $satker_id $time
$total_infografis        = mysqli_fetch_assoc(result: mysqli_query(mysql: $conn, query: $query_count_infografis))['total'];
$total_pages_infografis  = ceil(num: $total_infografis / $per_page);
$query_infografis_details = "SELECT id, judul, sumber_data, kategori, created_at, file_path
                             FROM infografis_upload
                             WHERE status = 'approved' AND satker_id = $satker_id $time_filter
                             ORDER BY created_at DESC
                             LIMIT $start_infografis, $per_page";
$result_infografis_details = mysqli_query(mysql: $conn, query: $query_infografis_details);
?>
```

Gambar 3. 51 Dataset dan infografis pagination

Gambar 3.53 ini menangani **pagination** (pembagian halaman) untuk dua jenis data: dataset dan infografis. Pada bagian pertama (Dataset pagination), sistem mengecek apakah parameter page_dataset telah dikirim melalui URL. Jika tidak, maka halaman default adalah 1. Variabel $start_dataset menghitung data awal yang akan diambil berdasarkan halaman aktif dan jumlah data per halaman ($per_page). Kemudian, sistem melakukan query untuk menghitung total dataset yang berstatus approved dan sesuai satker_id serta filter waktu. Jumlah total halaman ($total_pages_dataset) dihitung menggunakan fungsi ceil(), dan kemudian query detail dataset dijalankan dengan batas (LIMIT) berdasarkan pagination.

Bagian kedua (Infografis pagination) bekerja hampir identik, hanya saja digunakan untuk data dari tabel infografis_upload. Sistem mengecek parameter page_infografis, menghitung data awal dengan $start_infografis, lalu menghitung total data infografis yang telah disetujui dan sesuai dengan satker_id dan rentang waktu. Jumlah halaman total dihitung, dan query detail infografis pun dijalankan berdasarkan urutan created_at DESC, kemudian dibatasi (LIMIT) berdasarkan

pagination. Dengan sistem ini, pengguna hanya akan melihat sebagian data di setiap halaman, sehingga antarmuka lebih efisien dan tidak terlalu berat.

3.2.9 Minggu ke 9-10 (Activity Dashboard Admin dan Kategorisasi Topik)

```
$count_query = "
    SELECT COUNT(*) as total FROM (
        SELECT
            'Dataset' AS jenis_upload,
            du.judul,
            du.nama_satker,
            du.created_at,
            u.nama_lengkap,
            du.status
        FROM data_upload du
        LEFT JOIN users u ON du.satker_id = u.satker_id

        UNION ALL

        SELECT
            'Infografis' AS jenis_upload,
            iu.judul,
            iu.nama_satker,
            iu.created_at,
            u.nama_lengkap,
            iu.status
        FROM infografis_upload iu
        LEFT JOIN users u ON iu.satker_id = u.satker_id
    ) as combined_data
";

$count_result    = mysqli_query(mysql: $conn, query: $count_query);
$count_row       = mysqli_fetch_assoc(result: $count_result);
$total_records   = $count_row['total'];
$total_pages     = ceil(num: $total_records / $records_per_page);
```

Gambar 3. 52 Code fungsi SELECT as jenis_upload untuk activity user

Gambar 3.54 merupakan program di atas yang berfungsi untuk menampilkan data gabungan dari dua jenis unggahan, yaitu Dataset dan Infografis, dalam satu tampilan halaman. Proses dimulai dengan menyertakan file config.php untuk

memastikan koneksi ke basis data, serta pemeriksaan autentikasi pengguna melalui sesi user_id. Jika pengguna belum login, maka secara otomatis akan diarahkan ke halaman login. Tahapan ini bertujuan untuk menjaga keamanan akses data dan memastikan bahwa hanya pengguna yang memiliki izin yang dapat mengakses halaman tersebut.

```php
$query = "
    SELECT * FROM (
        SELECT
            'Dataset' AS jenis_upload,
            du.judul,
            du.nama_satker,
            du.created_at,
            u.nama_lengkap,
            du.status
        FROM data_upload du
        LEFT JOIN users u ON du.satker_id = u.satker_id

        UNION ALL

        SELECT
            'Infografis' AS jenis_upload,
            iu.judul,
            iu.nama_satker,
            iu.created_at,
            u.nama_lengkap,
            iu.status
        FROM infografis_upload iu
        LEFT JOIN users u ON iu.satker_id = u.satker_id
    ) AS combined
    ORDER BY created_at DESC
    LIMIT {$offset}, {$records_per_page}
";

$result = mysqli_query(mysql: $conn, query: $query);
?>
```

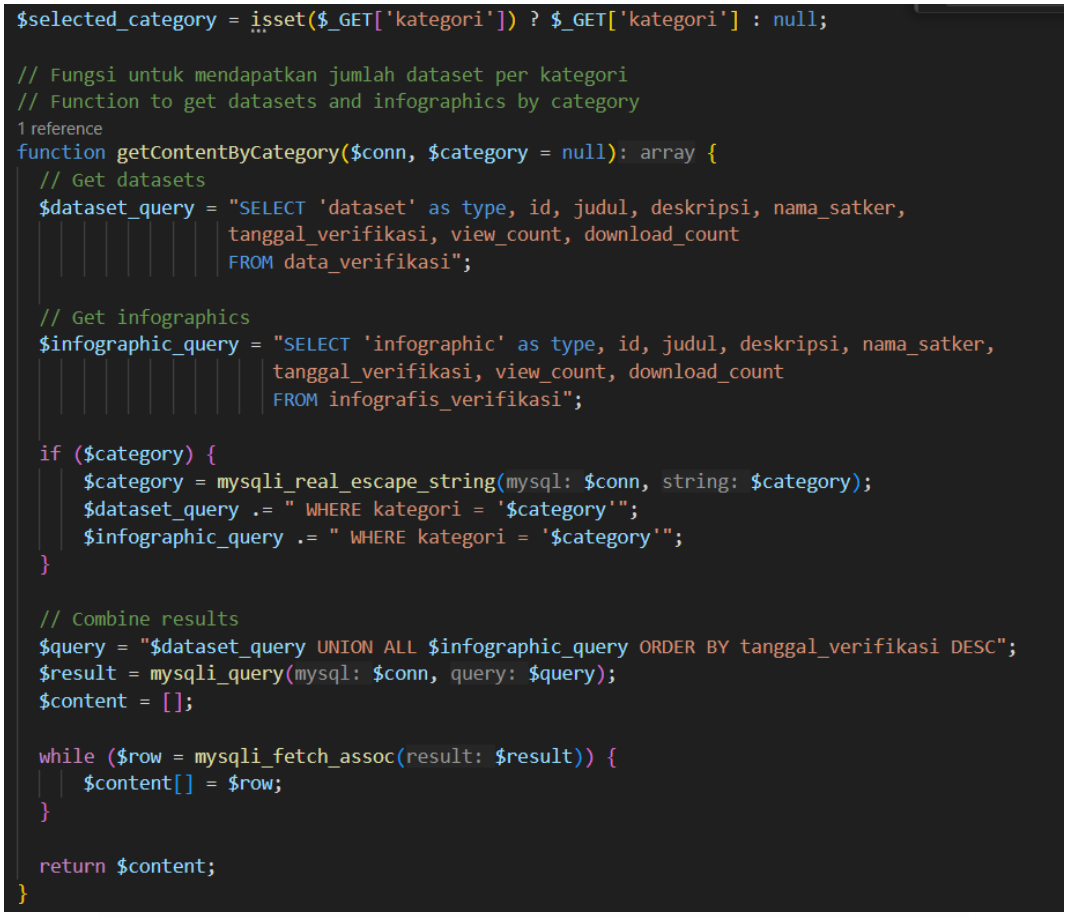
Gambar 3. 53 Code mekanisme pagination activity

Gambar 3.55 menerapkan mekanisme pagination (penomoran halaman) guna membatasi jumlah data yang ditampilkan pada setiap halaman, dengan ketentuan menampilkan 10 data per halaman. Nomor halaman yang aktif diambil dari

parameter URL ($_GET['page']). Berdasarkan nilai ini, sistem menghitung offset atau posisi awal data yang akan ditampilkan dalam query, agar data dapat diambil secara bertahap sesuai halaman yang dipilih oleh pengguna.

Pada bagian utama, dilakukan proses penggabungan data menggunakan perintah UNION ALL dari dua tabel utama, yaitu data_upload dan infografis_upload. Kedua tabel tersebut memiliki struktur data yang serupa, dan masing-masing diberi label tambahan jenis_upload untuk membedakan jenis unggahan. Selain itu, kedua tabel juga di-join dengan tabel users melalui kolom satker_id, sehingga data pengguna yang melakukan unggahan juga dapat ditampilkan, seperti nama satuan kerja dan nama lengkap pengunggah.

Setelah proses penghitungan total data selesai dilakukan untuk keperluan pagination, maka dilanjutkan dengan query utama untuk mengambil data sesuai halaman yang aktif. Data yang diambil disusun berdasarkan tanggal unggahan (created_at) dari yang terbaru ke yang terlama, kemudian dibatasi jumlahnya berdasarkan offset dan jumlah data per halaman. Hasil dari query ini disimpan dalam variabel $result dan selanjutnya dapat ditampilkan di antarmuka pengguna.



```php
$selected_category = isset($_GET['kategori']) ? $_GET['kategori'] : null;

// Fungsi untuk mendapatkan jumlah dataset per kategori
// Function to get datasets and infographics by category
1 reference
function getContentByCategory($conn, $category = null): array {
    // Get datasets
    $dataset_query = "SELECT 'dataset' as type, id, judul, deskripsi, nama_satker,
                      tanggal_verifikasi, view_count, download_count
                      FROM data_verifikasi";

    // Get infographics
    $infographic_query = "SELECT 'infographic' as type, id, judul, deskripsi, nama_satker,
                          tanggal_verifikasi, view_count, download_count
                          FROM infografis_verifikasi";

    if ($category) {
        $category = mysqli_real_escape_string(mysql: $conn, string: $category);
        $dataset_query .= " WHERE kategori = '$category'";
        $infographic_query .= " WHERE kategori = '$category'";
    }

    // Combine results
    $query = "$dataset_query UNION ALL $infographic_query ORDER BY tanggal_verifikasi DESC";
    $result = mysqli_query(mysql: $conn, query: $query);
    $content = [];

    while ($row = mysqli_fetch_assoc(result: $result)) {
        $content[] = $row;
    }

    return $content;
}
```

Gambar 3. 54 Code fungsi jumlah dataset per kategori

Gambar 3.56 merupakan Fungsi getContentByCategory digunakan untuk mengambil data dari dua tabel utama: data_verifikasi (berisi dataset) dan infografis_verifikasi (berisi infografis). Kedua jenis data ini digabung menggunakan query UNION ALL agar dapat ditampilkan bersama, dan hasilnya diurutkan berdasarkan tanggal verifikasi terbaru. Sementara itu, fungsi getCategoryStats dan getTotalViewsByCategory digunakan untuk menghitung jumlah konten dan total kunjungan (views) berdasarkan kategori yang dipilih, berguna untuk statistik tampilan yang lebih informatif.

Selain itu, fungsi getAllCategories mengambil semua kategori unik dari kedua tabel, menghitung jumlah dataset dan total view untuk masing-masing kategori, kemudian mengurutkannya berdasarkan jumlah view terbanyak. Ini berguna untuk membuat daftar kategori populer. Di sisi lain, fungsi getDatasetsByCategory

difokuskan untuk mengambil dataset berdasarkan kategori tertentu, serta menghitung selisih hari antara tanggal verifikasi dan tanggal saat ini agar dapat diukur seberapa baru data tersebut.

```php
// Function to get category statistics
0 references
function getCategoryStats($conn, $category = null): array {
    $stats = [
        'dataset_count' => 0,
        'infographic_count' => 0,
        'total_views' => 0
    ];

    // Get dataset stats
    $dataset_query = "SELECT COUNT(*) as count, SUM(view_count) as views
                      FROM data_verifikasi";

    // Get infographic stats
    $infographic_query = "SELECT COUNT(*) as count, SUM(view_count) as views
                          FROM infografis_verifikasi";

    if ($category) {
        $category = mysqli_real_escape_string(mysql: $conn, string: $category);
        $dataset_query .= " WHERE kategori = '$category'";
        $infographic_query .= " WHERE kategori = '$category'";
    }

    $result = mysqli_query(mysql: $conn, query: $dataset_query);
    $dataset_stats = mysqli_fetch_assoc(result: $result);
    $stats['dataset_count'] = $dataset_stats['count'];
    $stats['total_views'] += $dataset_stats['views'];

    $result = mysqli_query(mysql: $conn, query: $infographic_query);
    $infographic_stats = mysqli_fetch_assoc(result: $result);
    $stats['infographic_count'] = $infographic_stats['count'];
    $stats['total_views'] += $infographic_stats['views'];

    return $stats;
}
```

Gambar 3. 55 Fungsi code statistik per kategori

Gambar 3.57 merupakan fungsi getCategoryStats yang bertujuan untuk mengambil statistik berdasarkan kategori dari dua tabel berbeda: data_verifikasi (untuk dataset) dan infografis_verifikasi (untuk infografis). Fungsi ini mengembalikan array berisi tiga informasi utama: jumlah dataset (dataset_count), jumlah infografis (infographic_count), dan total jumlah tampilan (total_views). Pada awalnya, query disusun untuk menghitung jumlah data dan menjumlahkan

view_count dari kedua tabel tersebut. Jika parameter kategori diberikan, maka query akan difilter berdasarkan kolom kategori.

Selanjutnya, fungsi melakukan eksekusi query menggunakan mysqli_query() dan mengambil hasilnya dengan mysqli_fetch_assoc(). Nilai-nilai yang didapat kemudian disimpan dalam array $stats. Total tampilan (total_views) akan dijumlahkan dari kedua sumber data (dataset dan infografis), sedangkan jumlah dataset dan infografis masing-masing diambil dari hasil query mereka. Fungsi ini cukup fleksibel karena dapat digunakan baik untuk seluruh data maupun data berdasarkan kategori tertentu.

```php
// First get a list of all categories from both tables
$query = "SELECT kategori
          FROM (
              SELECT kategori FROM data_verifikasi
              UNION
              SELECT kategori FROM infografis_verifikasi
          ) AS all_categories
          WHERE kategori IS NOT NULL AND kategori != ''
          GROUP BY kategori";

$result = mysqli_query(mysql: $conn, query: $query);
$categories = [];

while ($row = mysqli_fetch_assoc(result: $result)) {
    $categoryName = $row['kategori'];

    // Now get the counts and views for this category
    $countQuery = "SELECT
        (SELECT COUNT(*) FROM data_verifikasi WHERE kategori = '$categoryName') as dataset_count,
        (SELECT COALESCE(SUM(view_count), 0) FROM data_verifikasi WHERE kategori = '$categoryName') +
        (SELECT COALESCE(SUM(view_count), 0) FROM infografis_verifikasi WHERE kategori = '$categoryName') as total_views";

    $countResult = mysqli_query(mysql: $conn, query: $countQuery);
    $countData = mysqli_fetch_assoc(result: $countResult);

    $categories[] = [
        'kategori' => $categoryName,
        'dataset_count' => $countData['dataset_count'],
        'total_views' => $countData['total_views'] ?? 0
    ];
}
```

Gambar 3. 56 Code daftar kategori unik dan jumlah dataset

Gambar 3.58 ini digunakan untuk mengambil daftar semua kategori yang unik dari dua tabel, yaitu data_verifikasi dan infografis_verifikasi, serta menghitung statistik masing-masing kategori. Pertama, query UNION digunakan untuk menggabungkan kategori dari kedua tabel dan menyaring entri yang tidak null atau kosong. Hasil query ini kemudian dieksekusi, dan setiap kategori disimpan dalam array $categories melalui perulangan while.

Di dalam perulangan, setiap kategori akan diproses untuk mendapatkan dua data statistik: jumlah dataset (dataset_count) dari tabel data_verifikasi, serta total tampilan (total_views) yang dijumlahkan dari dua tabel (data_verifikasi dan infografis_verifikasi). Fungsi COALESCE digunakan agar jika hasil penjumlahan view_count bernilai null, akan dikembalikan sebagai 0. Hasil akhirnya dimasukkan ke dalam array $categories[] yang berisi nama kategori, jumlah dataset, dan total tampilan.

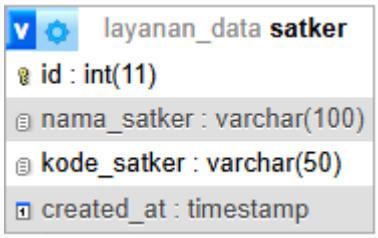
Setelah kondisi filter terbentuk, query utama dijalankan untuk mengambil seluruh data dari tabel data_verifikasi yang sesuai dengan filter yang telah ditentukan. Selain mengambil semua kolom, query juga menghitung selisih hari antara tanggal verifikasi dan tanggal saat ini, yang disimpan dalam kolom tambahan days_old. Data hasil query kemudian diurutkan berdasarkan tanggal verifikasi secara menurun (data terbaru ditampilkan lebih dahulu). Seluruh hasil query ini kemudian disimpan dalam variabel $result untuk diproses lebih lanjut, misalnya ditampilkan dalam bentuk tabel di halaman web.

Selain menampilkan data utama, kode ini juga menjalankan beberapa query tambahan untuk menghasilkan statistik dari data yang telah difilter. Statistik yang dihitung meliputi jumlah total data, total jumlah view, total jumlah unduhan, dan jumlah satuan kerja (satker) yang unik. Semua query statistik ini menggunakan kondisi WHERE yang sama agar hasil statistik tetap relevan dengan data yang sedang ditampilkan. Dengan adanya fitur ini, pengguna dapat melihat ringkasan data secara real-time sesuai dengan kriteria filter yang dipilih.

3.2.10 Minggu 11-12 (Master Data dan Update Data)

Master Data/Akses Kontrol Pengelolaan merupakan fungsi yang hanya dimiliki oleh role Admin untuk menambahkan serta mengubah user dan satker, disaat ada satuan kerja baru maka admin harus menambahkan dihalaman ini, begitupun juga dengan user baru, jika ada satker baru otomatis juga akan ada user

baru yang ditambahkan, data satker diambil dari tabel satker berikut merupakan tabelnya.



Gambar 3. 57 Tabel satker

Gambar 3.59 Merupakan Tabel satker yang berisikan id, nama_satker, kode_satker dan juga created_at. Tabel satker ini yang menjadi penghubung atau primary key dari user yang hanya dapat mengakses berdasarkan hanya satkernya masing-masing saja.

Selanjutnya merupakan bagian dari sistem manajemen data yang digunakan untuk mengelola data satuan kerja (satker) dan pengguna (user) yang memiliki peran sebagai operator. Akses ke halaman ini dibatasi hanya untuk pengguna dengan role admin, yang dicek melalui sesi login. Jika pengguna tidak memiliki akses sebagai admin, maka secara otomatis akan diarahkan ke halaman login. Hal ini menjadi langkah awal untuk memastikan keamanan sistem agar hanya pihak yang berwenang yang dapat melakukan perubahan data.

Script ini mendukung fungsi CRUD (Create, Read, Update, Delete) untuk dua entitas utama yaitu satker dan user. Fungsi ini dijalankan ketika sistem menerima request dengan metode POST dan disertai parameter action. Admin dapat menambahkan, mengedit, atau menghapus data satker dan user operator. Khusus untuk user, saat penambahan data, sistem akan mengecek apakah username atau email sudah digunakan. Jika tidak, data baru akan disimpan dengan password yang sudah di-hash untuk keamanan. Sementara itu, dalam pengeditan user, sistem juga memungkinkan pembaruan password jika diinginkan.

Selain penanganan perubahan data, sistem ini juga mengambil data dari database untuk ditampilkan. Data satker diambil seluruhnya untuk ditampilkan dalam daftar atau sebagai isian dropdown pada form. Untuk user, data yang diambil hanya yang memiliki role operator dan disertai dengan informasi satker masing-masing melalui proses join antar tabel. Hasil query tersebut disimpan dalam variabel untuk kemudian dapat digunakan di bagian tampilan (frontend) dari halaman web.

Secara keseluruhan, kode ini mengatur backend manajemen user dan satuan kerja dalam sebuah sistem berbasis web. Dengan pengamanan login admin, validasi data input, serta penggunaan query SQL yang telah difilter untuk menghindari injeksi, script ini cukup efektif sebagai fondasi sistem administratif. Namun, untuk pengembangan lebih lanjut, disarankan menambahkan fitur notifikasi kesalahan atau keberhasilan yang lebih jelas ke tampilan HTML, serta sanitasi input yang lebih kuat dan penggunaan prepared statements untuk meningkatkan keamanan.

```php
$id = $_GET['id'];
$type = $_GET['type'];

// Get data based on type
if ($type == 'data') {
    $table_verify = 'data_verifikasi';
    $table_upload = 'data_upload';
    $upload_folder = 'datasets';
    $allowed_extensions = ['csv', 'xlsx'];
} else {
    $table_verify = 'infografis_verifikasi';
    $table_upload = 'infografis_upload';
    $upload_folder = 'infografis';
    $allowed_extensions = ['png', 'jpg', 'jpeg'];
}

// Get verified data
$query = "SELECT * FROM $table_verify WHERE id = ?";
$stmt = mysqli_prepare(mysql: $conn, query: $query);
mysqli_stmt_bind_param(statement: $stmt, types: "i", var: &$id);
mysqli_stmt_execute(statement: $stmt);
$result = mysqli_stmt_get_result(statement: $stmt);

if (mysqli_num_rows(result: $result) == 0) {
    header(header: "Location: keterangan_data.php");
    exit();
}

$data = mysqli_fetch_assoc(result: $result);
```

Gambar 3. 58 Code prepared statement update data

Gambar 3.60 merupakan bagian dari sistem manajemen data berbasis web yang digunakan untuk memproses data terverifikasi oleh pengguna yang telah login. Proses dimulai dengan pengecekan sesi untuk memastikan bahwa pengguna telah login, kemudian mengambil informasi lengkap pengguna termasuk satuan kerjanya (satker). Parameter id dan type yang dikirim lewat URL digunakan untuk menentukan data atau infografis mana yang akan diproses, serta menetapkan tabel dan direktori upload yang sesuai berdasarkan tipe tersebut.

Setelah data diverifikasi ditemukan di dalam tabel (data_verifikasi atau infografis_verifikasi), pengguna dapat mengirimkan form berisi informasi terbaru seperti judul, keterangan, dan jadwal pemutakhiran. Jika pengguna juga

mengunggah file baru, sistem akan memvalidasi ekstensi file tersebut, mengunggahnya sementara ke folder temp, lalu memindahkannya ke folder final yang sesuai (uploads/datasets atau uploads/infografis). Ekstensi file yang diperbolehkan tergantung pada jenis data yang diproses (misalnya CSV/XLSX untuk data, JPG/PNG untuk infografis).

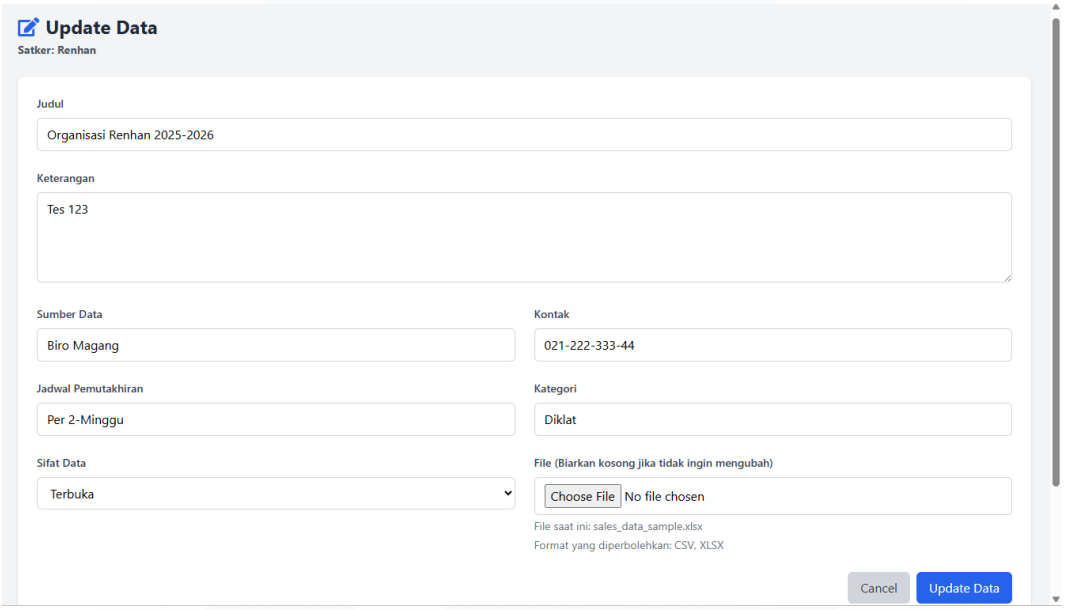
```php
// Check if data exists in upload table - we need to find by some identifying information
// Let's try to find by title and source from the same satker
$check_query = "SELECT id FROM $table_upload WHERE judul = ? AND sumber_data = ? AND satker_id = ?";
$stmt = mysqli_prepare(mysql: $conn, query: $check_query);
mysqli_stmt_bind_param(statement: $stmt, types: "sss", var: &$data['judul'], vars: &$data['sumber_data'], $satker_id);
mysqli_stmt_execute(statement: $stmt);
$check_result = mysqli_stmt_get_result(statement: $stmt);

// If data exists in upload table, delete it
if (mysqli_num_rows(result: $check_result) > 0) {
    $upload_data = mysqli_fetch_assoc(result: $check_result);
    $upload_id = $upload_data['id'];

    $delete_upload_query = "DELETE FROM $table_upload WHERE id = ?";
    $stmt = mysqli_prepare(mysql: $conn, query: $delete_upload_query);
    mysqli_stmt_bind_param(statement: $stmt, types: "i", var: &$upload_id);
    mysqli_stmt_execute(statement: $stmt);
}

// Delete from verification table
$delete_query = "DELETE FROM $table_verify WHERE id = ?";
$stmt = mysqli_prepare(mysql: $conn, query: $delete_query);
mysqli_stmt_bind_param(statement: $stmt, types: "i", var: &$id);
(global variable) string $insert_query
@var string $insert_query                    status
$insert_query = "INSERT INTO $table_upload (judul, keterangan, sumber_data, file_path, kontak,
                 jadwal_pemutakhiran, kategori, sifat_data, status, satker_id, nama_satker)
                 VALUES (?, ?, ?, ?, ?, ?, ?, ?, 'pending', ?, ?)";
```

Gambar 3. 59 Code proses transaksi update data

Gambar 3.61 merupakan proses selanjutnya dilakukan dalam transaksi database agar data tetap konsisten. Sistem akan memeriksa apakah data dengan judul dan sumber data yang sama dari satker tersebut sudah ada di tabel upload. Jika ya, maka data tersebut akan dihapus terlebih dahulu. Kemudian data dari tabel verifikasi juga dihapus dan data baru dengan status "pending" dimasukkan ke tabel upload. Jika semua langkah berhasil, transaksi dikonfirmasi dengan commit; jika terjadi kesalahan, transaksi dibatalkan dengan rollback.

Akhirnya, setelah proses selesai, pengguna akan diarahkan kembali ke halaman utama (keterangan_data.php) dengan parameter yang menandakan bahwa proses berhasil. Kode ini juga telah menggunakan prepared statements untuk mencegah serangan SQL Injection, sehingga cukup aman dari sisi keamanan database. Selain

itu, struktur folder dan pengecekan file memberikan kontrol tambahan agar file yang diunggah sesuai format dan tidak menyebabkan masalah di sistem.



Gambar 3. 60 Tampilan interface update data

Gambar 3.62 Merupakan tampilan Operator saat mereka ingin mengupdate Data mereka. Mereka bisa mengubah semua meta data yang tersedia didalam Database untuk mereka ubah, setelah diubah maka operator harus menunggu Kembali untuk Admin mereview data yang diupdate, setelah admin approve maka data akan muncul dipublik

3.2.11 Minggu 13 (Testing Eror dan Presentasi Project)

Testing dan debugging error sangat penting dilakukan setelah website selesai dikerjakan karena bertujuan memastikan bahwa seluruh fitur dan fungsi berjalan sesuai harapan, bebas dari bug yang dapat mengganggu pengguna, serta memberikan pengalaman pengguna (user experience) yang optimal. Melalui testing, pengembang dapat mengevaluasi kinerja website, menemukan kesalahan pada tampilan, fungsi, maupun keamanan seperti celah pada sistem login atau potensi serangan SQL injection. Sementara itu, debugging memungkinkan perbaikan terhadap error atau kesalahan logika yang ditemukan selama pengujian.

Selain itu, testing juga membantu memastikan performa website tetap cepat dan responsif di berbagai perangkat dan kondisi penggunaan. Tanpa proses ini, website berisiko mengalami gangguan fungsi, lambat diakses, atau bahkan rentan terhadap serangan, yang pada akhirnya merugikan pemilik maupun pengguna website. Berikut merupakan beberapa tahap yang dilakukan dalam testing dan debugging:

1. Pengujian Manajemen Pengguna dan Hak Akses: Dari tabel users dan satker, perlu dilakukan testing mendalam pada sistem autentikasi dan otorisasi. Khususnya untuk membedakan akses antara role 'operator' dan 'admin'. Pengujian harus mencakup validasi login, keamanan password (yang menggunakan hash), dan pembatasan akses berdasarkan satker_id. Potensi error yang perlu diantisipasi termasuk duplikasi email/username, invalid password hash, atau ketidaksesuaian relasi antara users dengan satker.
2. Verifikasi Data dan Infografis: Alur kerja dari data_upload ke data_verifikasi (dan infografis_upload ke infografis_verifikasi) memerlukan pengujian komprehensif. Testing harus memastikan status perubahan ('pending', 'approved', 'rejected') berfungsi dengan benar, file_path tersimpan dan dapat diakses, serta catatan verifikasi dan timestamp terekam dengan akurat. Bug yang sering muncul termasuk file corrupt selama upload, inconsistent status updates, atau masalah dalam tracking view_count dan download_count.
3. Sistem Request dan Pengaduan: Tabel data_request dan pengaduan perlu diuji untuk memastikan tracking status berjalan baik. Untuk data_request, alur dari 'Menunggu Persetujuan' hingga 'Data Terkirim' harus konsisten. Pada pengaduan, prioritas dan status harus dikelola dengan benar, termasuk attachment bukti_gambar. Error yang perlu diwaspadai meliputi missing file attachments, status yang tidak ter-update, atau tanggapan yang tidak tersimpan.
4. Contact Messages dan Statistik: Pengujian pada contact_messages harus memverifikasi proper handling dari pesan masuk dan status updates. Tabel statistik_satker dan statistik_infografis_satker perlu diuji untuk memastikan auto-update jumlah dataset dan infografis berjalan akurat. Potential bugs

termasuk counter yang tidak ter-update, timestamp yang tidak sinkron, atau masalah dalam kalkulasi statistik.

5. Database Relations dan Integrity: Testing perlu memastikan foreign key constraints (seperti antara pengaduan dan users) berfungsi dengan baik. Perlu juga memverifikasi bahwa operasi CASCADE pada delete user berjalan sesuai ekspektasi. Error handling harus diuji untuk kasus seperti orphaned records, constraint violations, atau inconsistent data states antara tabel yang berelasi. Khusus untuk file handling, perlu dipastikan proper cleanup ketika record dihapus atau diupdate.

   Setelah itu selesainya project ini selama 13 Minggu Akhirnya saya melakukan presentasi dan paparan terkait dengan project website Portal Data Satu Kemhan di hadapan para pegawai negri sipil,

3.3 Kendala yang ditemukan

Penggunaan PHP native tanpa framework modern memiliki sejumlah keterbatasan, terutama dari sisi struktur proyek. Dalam pengembangan aplikasi berskala besar, struktur yang tidak modular akan menyulitkan dalam pengorganisasian kode. Dengan rancangan sistem tersebut diharapkan perusahaan akan memperoleh beberapa kemudahan dalam menginput data sekaligus membantu pihak perusahaan untuk menyusun laporan penggajian menjadi lebih cepat dan lebih efisien[5]. Hal ini menyebabkan proyek menjadi kurang rapi dan sulit untuk dikembangkan lebih lanjut seiring meningkatnya kompleksitas aplikasi.

Di samping itu, ketiadaan fitur bawaan seperti ORM (Object Relational Mapping), middleware, dan sistem keamanan otomatis menyebabkan pengembangan dengan PHP native lebih rentan terhadap kesalahan pemrograman. Developer harus menulis semua koneksi database, query SQL, hingga filter keamanan secara manual. Hal ini tidak hanya menyita waktu, tetapi juga membuka potensi terjadinya celah keamanan seperti SQL Injection, XSS (Cross-Site

Scripting), dan CSRF (Cross-Site Request Forgery) jika tidak ditangani dengan benar.

Masalah lain yang muncul adalah dalam aspek maintenance dan kolaborasi tim. Tanpa standar arsitektur yang jelas, kode program menjadi sulit dibaca dan dipelajari oleh developer lain, terutama dalam tim yang besar atau berpindah tangan. Pengujian otomatis (automated testing) dan dokumentasi pun seringkali diabaikan dalam proyek PHP native, yang membuat proses debugging dan deployment menjadi lebih lambat dan tidak efisien.

3.4 Solusi atas kendala yang ditemukan

Salah satu solusi utama adalah beralih menggunakan framework PHP modern seperti Laravel atau CodeIgniter. Framework ini menawarkan struktur proyek yang sudah tertata dengan baik serta menyediakan fitur-fitur esensial seperti routing otomatis, middleware keamanan, dan ORM, sehingga proses pengembangan menjadi lebih cepat dan aman. Dengan adanya standar arsitektur seperti MVC (Model-View-Controller), kolaborasi tim menjadi lebih mudah karena alur kerja dan penempatan kode sudah ditentukan secara sistematis.

Namun, jika tetap ingin menggunakan PHP native karena keterbatasan resource atau kebutuhan spesifik, solusi lainnya adalah dengan membangun mini-framework sendiri secara bertahap. Misalnya, mulai dengan membuat sistem routing sederhana, library koneksi database yang aman (menggunakan PDO dan prepared statement), serta helper untuk validasi dan sanitasi input. Hal ini bisa membantu menanamkan kebiasaan struktur yang rapi sekaligus menjaga keamanan aplikasi meskipun tidak menggunakan framework besar.

Selain itu, penting untuk menerapkan praktik terbaik dalam pengembangan perangkat lunak seperti version control (menggunakan Git), penulisan dokumentasi kode, dan pengujian unit. PHP Native Framework untuk memudahkan programmer pemula yang belum terbiasa dengan konsep MVC dan OOP[6]. Dengan begitu, meskipun berbasis PHP native, proyek tetap dapat dikelola secara profesional dan

scalable dalam jangka panjang. Penggunaan tools eksternal seperti Composer untuk manajemen dependensi dan PHPUnit untuk testing juga sangat disarankan guna menutupi kekurangan bawaan dari PHP native itu sendiri.

Meskipun tidak menggunakan framework, proyek ini telah dilengkapi dengan fitur-fitur yang mendukung fungsionalitas dan keamanan sistem. Beberapa fitur utama seperti pengelolaan data dari 21 Satker, API untuk integrasi front-end, sistem statistik interaksi, dan dashboard interaktif telah diimplementasikan dengan baik. Dari sisi keamanan, telah diterapkan praktik terbaik seperti password hashing**,** validasi input**,** autentikasi berbasis session**,** serta penggunaan prepared statements untuk mencegah serangan SQL Injection**.** Dengan pendekatan ini, sistem tetap dapat berjalan secara aman dan stabil, meskipun dibangun menggunakan PHP native.