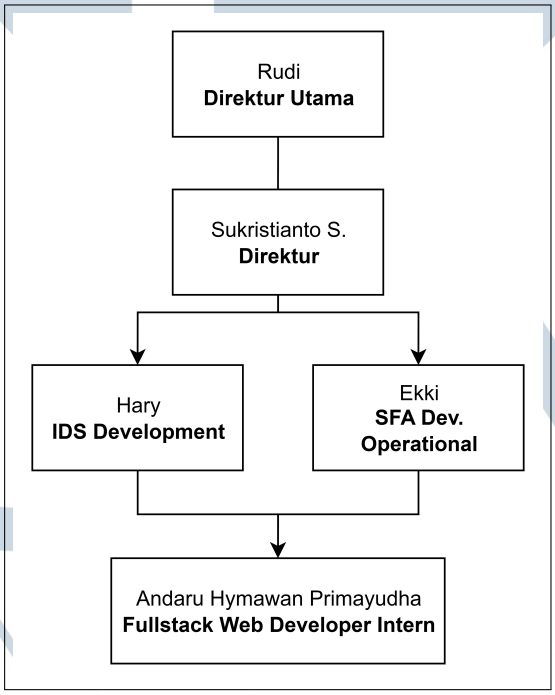
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama program kerja magang di PT ESKA LINK, posisi yang diberikan sebagai Fullstack Web Developer Intern di bawah koordinasi divisi IDS Development dan SFA Development and Operational. Pada awalnya, pekerjaan ada di bawah divisi IDS Development saja, namun dikarenakan adanya kebutuhan sumber daya pada divisi lain maka dialihkan ke divisi tersebut. Gambar 3.1 merupakan kedudukan tanggung jawab selama melakukan kegiatan magang di PT ESKA LINK. Tanggung jawab yang diemban mencangkup pengembangan modul pada aplikasi IDS dan SFA, serta melakukan perbaikan bug jika terjadi permasalahan pada aplikasi tersebut.



Gambar 3.1. Kedudukan pada organisasi perusahaan PT ESKA LINK

Sumber: [4]

Dalam proses magang, beberapa media digunakan untuk mendukung kegiatan kerja dan mempermudah komunikasi, yaitu:

1. Telegram digunakan sebagai platform komunikasi utama untuk keperluan

koordinasi harian.

2. Zoom digunakan sebagai platform komunikasi jika pembagian tugas oleh supervisi tidak memungkinkan adanya pertemuan secara tatap muka.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang sebagai Fullstack Web Developer Intern di PT ESKA LINK, tugas yang dilakukan adalah pengembangan modul distribusi. Aktivitas pengembangan tersebut mencakup penambahan fitur baru serta perbaikan bug pada sistem yang telah berjalan. Proses pengembangan dan perbaikan bug dilakukan pada dua produk utama, yaitu IDS dan SFA, dengan karakteristik pekerjaan yang berbeda pada masing-masing produk.

Pada produk IDS, pekerjaan yang dilakukan umumnya berupa penambahan dan perbaikan berskala minor yang berasal dari berbagai kebutuhan klien. Beberapa tugas yang dikerjakan meliputi perbaikan kesalahan perhitungan data, penambahan variabel pada variabel global, penyesuaian posisi dokumen PDF yang dihasilkan oleh suatu fungsi, penambahan validasi pada proses perubahan data, perbaikan ikon antarmuka yang tidak muncul, serta perubahan pada *query* SQL akibat adanya ketentuan baru dari klien atau kesalahan penulisan sebelumnya. Selain pekerjaan berskala minor, terdapat pula tugas yang memiliki tingkat kompleksitas lebih tinggi, seperti penambahan tabel baru untuk mendukung implementasi mekanisme penyimpanan data. Pada implementasi tersebut, selain membuat tabel baru, diperlukan penyesuaian pada fungsi yang telah ada sebelumnya agar sistem dapat melakukan pengecekan terhadap tabel lama, serta menentukan alur penyimpanan data apakah tetap menggunakan tabel lama atau dialihkan ke tabel baru sesuai dengan kondisi yang ditetapkan.

Pada produk SFA, tugas yang diberikan sebagian besar berupa *change request* (CR) dari klien yang disampaikan melalui tim support SFA secara harian. Setiap CR yang diterima perlu diimplementasikan sesuai dengan kebutuhan klien. Permintaan yang diajukan umumnya berkaitan dengan kebutuhan laporan, seperti penyesuaian data yang ditampilkan pada menu antarmuka pengguna serta penambahan fitur agar data tersebut dapat diekspor ke dalam bentuk file Excel atau PDF. Salah satu tugas utama dengan tingkat kompleksitas tinggi pada produk SFA adalah implementasi sistem penyimpanan file berbasis *cloud*. Dalam implementasi ini, pihak supervisi telah menyiapkan kredensial untuk tiga penyedia *cloud*, yaitu AWS S3, Google Cloud Storage, dan Eranya. Proses

implementasi mencakup pembuatan tabel baru serta pengembangan handler khusus untuk mengatur berbagai aktivitas yang berkaitan dengan *cloud storage*, seperti penyimpanan file melalui API, penyimpanan file melalui antarmuka pengguna, serta pengambilan data yang telah disimpan. Selain itu, diperlukan penyediaan variabel global berupa konfigurasi *bucket storage* agar klien dapat memilih penyedia *cloud* yang akan digunakan. Pada tahap akhir, mekanisme penyimpanan dan pengambilan data berbasis *cloud* diintegrasikan ke seluruh modul yang terkait, yaitu modul penyimpanan melalui API, penyimpanan melalui UI, penampilan data pada laporan UI, penampilan data pada proses persetujuan, serta penampilan data pada fitur ekspor Excel.

3.3 Uraian Pelaksanaan Magang

Berikut ini disajikan uraian pelaksanaan kegiatan kerja magang di PT ESKA LINK yang dilakukan setiap minggu selama periode magang. Rincian aktivitas dan pekerjaan yang dikerjakan pada masing-masing minggu dapat dilihat pada Tabel 3.1.

Tabel 3.1. Pekerjaan tiap minggu selama kegiatan magang

Minggu Ke -	**Pekerjaan yang dilakukan**
1	Onboarding dan mulai mengerjakan tugas memperbaiki bug
2	Memperbaiki bug dan melakukan eksplorasi *source code* sistem
3	Memperbaiki bug dan menambah tampilan antarmuka pengguna yang diminta oleh klien
4	Memperbaiki bug dan menambah fitur dinamis yang diminta oleh klien
5	Melanjutkan implementasi fitur dan memperbaiki bug
6	Menerima pengarahan dan mulai mengerjakan *Change Request* (CR) yang diminta oleh klien
7	Memperbaiki fitur antarmuka pengguna dan mengembangkan fitur ekspor Excel sesuai kebutuhan klien
8	Mengembangkan fitur sesuai permintaan klien dan melakukan perpindahan ke divisi lain
	Lanjut pada halaman berikutnya

Tabel 3.1 Pekerjaan tiap minggu selama kegiatan magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
9	Melanjutkan pengembangan fitur penyimpanan dan implementasi fitur terkait
10	Implementasi fitur terkait dan melakukan pengujian
11	Memunculkan data foto dan mengubah pemilihan distributor
12	Melanjutkan fitur pemilihan distributor dan mempelajari Laravel dan Vue
13	Membuat dynamic payload mapping
14	Implementasi fitur penyimpanan pada SFA
15	Melanjutkan implementasi penyimpanan pada SFA dan CR klien
16	Mengerjakan CR klien
17	Melanjutkan implementasi penyimpanan pada SFA dan CR klien
18	Melanjutkan implementasi penyimpanan pada SFA dan CR klien

3.3.1 Software yang Digunakan

Berikut beberapa software yang digunakan dalam proses pengembangan modul penyimpanan file selama pelaksanaan program magang:

1. Internet Information Services (IIS)

   Internet Information Services (IIS) adalah perangkat lunak web server yang dikembangkan oleh Microsoft, dirancang khusus untuk meng-host situs web dan menjalankan aplikasi berbasis web di sistem operasi Windows. Fungsi utama dari IIS adalah sebagai jembatan yang menerima permintaan pengguna melalui berbagai protokol seperti HTTP, HTTPS, dan FTP, kemudian memproses dan mengirimkan konten yang sesuai dari server kembali ke *client* [5].

2. PhpStorm

   PhpStorm adalah *Integrated Development Environment* (IDE) *premium* dan *cross-platform* dari JetBrains yang dirancang khusus untuk pengembangan PHP, menyediakan fitur canggih seperti bantuan pengodean, debugging

visual, dan integrasi penuh dengan *framework* PHP serta teknologi *Frontend* terkait [6].

3. Navicat

   Navicat berfungsi sebagai alat *Graphical User Interface* (GUI) terpadu yang sangat kuat untuk administrasi dan pengembangan basis data, memfasilitasi konektivitas simultan ke berbagai sistem *Database Management System* (DBMS) serta mempermudah operasi penting seperti migrasi data, sinkronisasi skema, dan eksekusi kueri kompleks [7].

4. FileZilla Client dan Server

   FileZilla menyediakan solusi transfer file *open-source* yang terdiri dari dua komponen utama, yaitu FileZilla Client sebagai aplikasi klien untuk menghubungkan dan mentransfer file secara aman ke server jarak jauh menggunakan protokol FTP/FTPS/SFTP. Sementara itu, FileZilla Server berfungsi sebagai perangkat lunak server di Windows yang memungkinkan sistem lokal untuk menerima koneksi dan menyediakan akses file kepada pihak lain melalui protokol FTP dan FTPS [8].

5. OpenVPN

   OpenVPN adalah protokol *Virtual Private Network* (VPN) *open-source* yang dirancang untuk menciptakan koneksi terenkripsi yang aman antara perangkat melalui internet, seringkali menggunakan protokol SSL/TLS untuk pertukaran kunci yang kuat [9]. Alat ini digunakan dalam lingkungan perusahaan untuk menyediakan akses jarak jauh yang aman (*secure remote access*) serta mengamankan komunikasi data antara situs (*site-to-site*).

6. S3 Browser

   S3 Browser adalah aplikasi klien desktop berbasis Windows yang menyediakan *Graphical User Interface* (GUI) untuk mengelola dan memfasilitasi transfer data pada layanan penyimpanan objek *Amazon Simple Storage Service* (Amazon S3) [10].

7. Bitbucket

   Bitbucket Cloud adalah platform hosting kode berbasis Git yang dirancang untuk kolaborasi tim, menyediakan layanan *version control*, alur kerja *Continuous Integration/Continuous Delivery* (CI/CD) terintegrasi melalui

Pipelines, serta integrasi yang erat dengan produk Atlassian lainnya seperti Jira [11].

8. Insomnia Insomnia adalah *software open-source* yang berfungsi sebagai *API client cross-platform* yang dirancang untuk membantu pengembang dalam merancang, menguji, mendokumentasikan, dan berinteraksi dengan API, termasuk REST, SOAP, GraphQL, gRPC, dan WebSockets, dengan fitur-fitur yang mendukung kolaborasi tim [12].

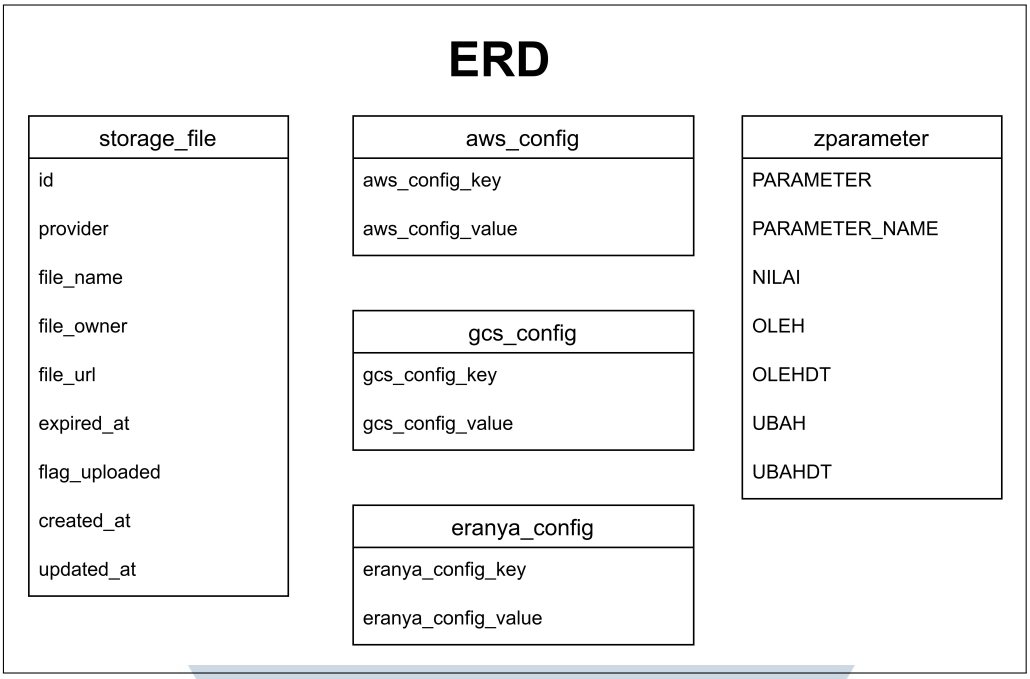
3.3.2 Perancangan Modul Penyimpanan File

Bagian ini akan menjelaskan alur perancangan dan implementasi modul penyimpanan file menggunakan *cloud* yang selanjutnya akan diintegrasikan ke dalam modul-modul terkait.

A Perancangan Basis Data (ERD)

Perancangan basis data untuk sistem *multi-storage* dinamis ini menggunakan sistem manajemen basis data relasional SQL Server. Struktur data dirancang khusus untuk mendukung seluruh kebutuhan fungsional, mulai dari manajemen kredensial *cloud*, penentuan penyedia *cloud* yang aktif secara dinamis, hingga mekanisme penyimpanan dan validasi metadata file.

Gambar 3.2 menunjukkan seluruh tabel yang digunakan pada proses ini. Tidak terdapat relasi antar tabel karena setiap tabel berfungsi sebagai tabel mandiri untuk menyimpan data tertentu. Pemanggilan tabel dilakukan secara langsung di dalam kode program sesuai dengan kebutuhan sistem.



Gambar 3.2. ERD penyimpanan *cloud*

A.1 Tabel *zparameter*

Tabel *zparameter* digunakan sebagai media penyimpanan seluruh parameter global yang digunakan pada sistem SFA. Pada konteks pengembangan modul ini, tabel *zparameter* berfungsi untuk menyimpan informasi mengenai penyedia layanan *cloud* yang sedang aktif digunakan oleh sistem. Nilai parameter tersebut bersifat dinamis dan dapat diubah sesuai kebutuhan, sehingga setiap proses penyimpanan file akan mengacu pada penyedia layanan *cloud* yang telah ditetapkan. Tabel 3.2 menjelaskan setiap atribut yang terdapat pada tabel *zparameter*.

Tabel 3.2. Struktur tabel *zparameter*

Nama Kolom	**Tipe Data**	**Ketentuan**	**Keterangan**
PARAMETER	VARCHAR	PRIMARY KEY, NOT NULL	Nama parameter pada global parameter
PARAMETER_NAME	VARCHAR	NULL	Deskripsi setiap parameter
NILAI	VARCHAR	NULL	Nilai dari parameter
OLEH	VARCHAR	NULL	Aktor yang membuat parameter
OLEHDT	DATETIME	NULL	Waktu pembuatan parameter
UBAH	VARCHAR	NULL	Aktor yang mengubah parameter
UBAHDT	DATETIME	NULL	Waktu pembaruan parameter

A.2 Tabel *storage_file*

Tabel *storage_file* berfungsi untuk menyimpan konfigurasi file yang berhasil disimpan di *cloud*. Mulai dari nama, penyedia *cloud*, sampai url yang digunakan untuk file di *cloud*. Pada Tabel 3.3 dijelaskan setiap atribut yang ada pada tabel dan apa saja kegunaannya.

Tabel 3.3. Struktur tabel *storage_file*

Nama Kolom	**Tipe Data**	**Ketentuan**	**Keterangan**
id	INTEGER	PRIMARY KEY, NOT NULL	Id setiap file
provider	NVARCHAR	NOT NULL	Penyedia *cloud* yang digunakan
file_name	NVARCHAR	NOT NULL	Nama file
file_owner	NVARCHAR	NOT NULL	Nama folder yang digunakan
file_url	NVARCHAR	NOT NULL	Url file di *cloud*
expired_at	DATETIME	NULL	Batas waktu file di *cloud*
flag_uploaded	INT	NOT NULL	Tanda bahwa file sudah disimpan di *cloud*
created_at	DATETIME	NULL	Waktu file disimpan di *cloud*
updated_at	DATETIME	NULL	Waktu file diperbarui di *cloud*

A.3 Tabel *aws_config*, *eranya_config*, dan *gcs_config*

Tabel *aws_config*, *eranya_config*, dan *gcs_config* memiliki fungsi yang sama, yaitu menyimpan seluruh konfigurasi yang diperlukan untuk proses penyimpanan file ke *cloud*. Perbedaan dari ketiga tabel tersebut terletak pada penyedia layanan *cloud* yang digunakan. Setiap kali sistem melakukan akses ke layanan *cloud*, tabel konfigurasi ini dipanggil untuk memperoleh parameter yang dibutuhkan.

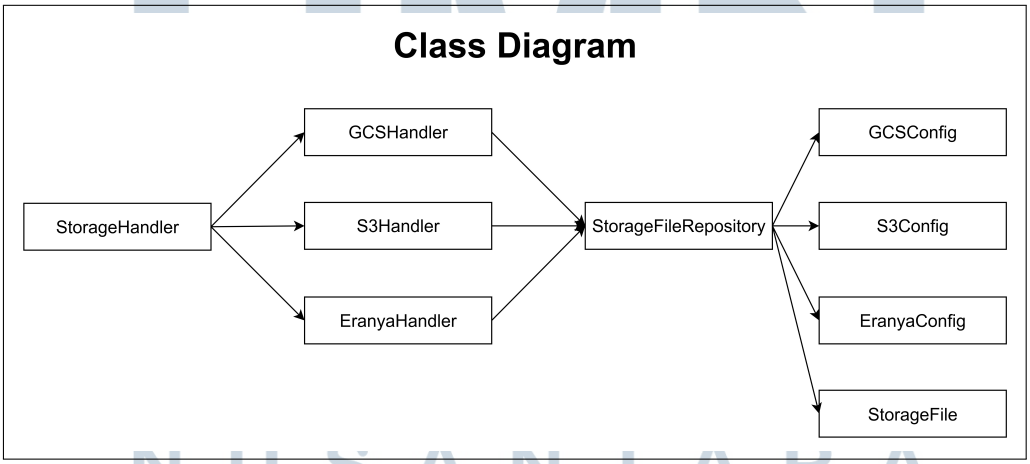
Konfigurasi yang disimpan meliputi *access key id*, *secret access key*, *bucket*, *region*, dan *url*. Pada Tabel 3.4, seluruh konfigurasi dari berbagai penyedia *cloud* digabungkan ke dalam satu tabel karena struktur dan jenis konfigurasinya bersifat seragam.

Tabel 3.4. Struktur tabel *aws_config*, *eranya_config*, dan *gcs_config*

Nama Kolom	Tipe Data	Ketentuan	Keterangan
cloud_config_key	VARCHAR	PRIMARY KEY, NOT NULL	Daftar variabel yang diperlukan untuk mengakses cloud
clouds_config_value	VARCHAR	NOT NULL	Nilai dari setiap variabel

B Perancangan Komponen (Class Diagram)

Arsitektur komponen sistem ini dirancang dengan memisahkan tanggung jawab setiap kelas, seperti diilustrasikan pada Gambar 3.3. *StorageHandler* bertindak sebagai pengelola utama dan merupakan satu-satunya kelas yang dihubungi oleh *controller* atau *service* lain ketika memerlukan operasi penyimpanan atau pengambilan file. Untuk menjalankan tugasnya, *StorageHandler* bergantung pada *StorageFileRepository* untuk mendapatkan informasi mengenai penyedia *cloud* aktif yang tersimpan dalam database. Setelah mengetahui penyedua *cloud* mana yang harus digunakan (misalnya AWS, GCS, atau Eranya), *StorageHandler* akan mendelegasikan perintah (menyimpan atau memanggil file) secara penuh kepada salah satu dari tiga kelas handler spesifik: *S3Handler*, *GCSHandler*, atau *EranyaHandler*.



Gambar 3.3. Class diagram pada penyimpanan menggunakan *cloud*

Setiap *handler* ini memegang seluruh logika koneksi dan interaksi unik dengan layanan *cloud* masing-masing. Sementara itu, *StorageFileRepository*

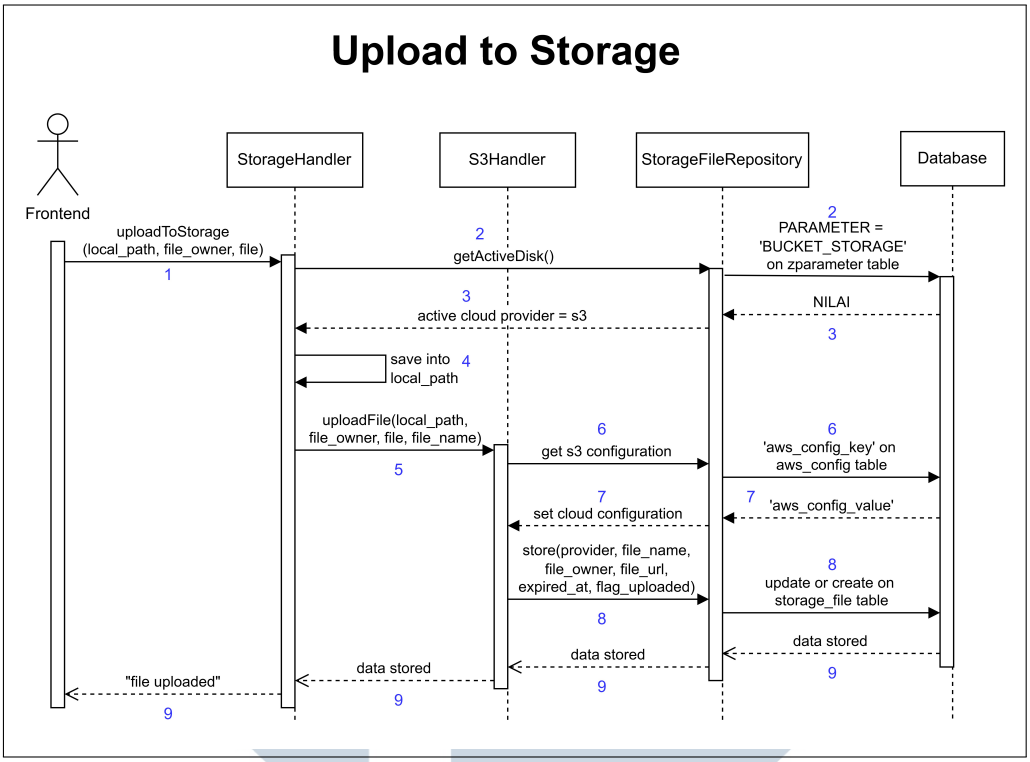
bertanggung jawab penuh atas semua interaksi data, seperti mengambil dan mendekripsi kredensial dari model konfigurasi (*GCSConfig*, *S3Config*, *EranyaConfig*) serta menyimpan dan memperbarui metadata file (seperti URL dan waktu kedaluwarsa) di model *StorageFile*. Struktur ini memastikan logika inti sistem terpusat di *StorageHandler* dan terisolasi dari detail implementasi teknis setiap penyedia *cloud*.

C Perancangan Proses (Sequence Diagram)

Pemodelan alur proses sistem ini menggunakan Sequence Diagram untuk menggambarkan urutan komunikasi dan interaksi antar komponen sistem *Backend*, yang terutama berpusat pada kelas *StorageHandler* sebagai manajer proses.

C.1 Proses Penyimpanan File ke *Cloud*

Proses penyimpanan file ke *cloud* dapat dilakukan melalui dua mekanisme, yaitu melalui antarmuka web dan melalui API. Gambar 3.4 menunjukkan sequence diagram proses penyimpanan file melalui API. Secara umum, alur proses penyimpanan file melalui antarmuka web maupun API adalah sama. Pada mekanisme API, file yang dapat dikirimkan dibatasi hanya pada format JPG yang telah dikonversi ke dalam bentuk base64, sedangkan ukuran file tidak dibatasi secara khusus oleh sistem. Sementara itu, pada antarmuka web mendukung penyimpanan file dengan format JPG, PNG, serta PDF. Pemisahan mekanisme dilakukan untuk memudahkan implementasi serta menyesuaikan penggunaan dengan kebutuhan sistem.



Gambar 3.4. Sequence diagram *upload to storage*

Tahapan proses menyimpan file ke *cloud* dijelaskan sebagai berikut:

1. Proses dimulai ketika *Frontend* mengirimkan permintaan menyimpan file ke *StorageHandler* dengan membawa parameter *local_path*, *file_owner*, dan *file*.
2. *StorageHandler* kemudian memanggil fungsi *getActiveDisk()* untuk menentukan penyedia layanan *cloud* yang sedang aktif melalui *StorageFileRepository*. *StorageFileRepository* melakukan pencarian ke dalam tabel *zparameter* pada database untuk memperoleh nilai dari parameter *BUCKET_STORAGE*. Parameter ini menyimpan informasi mengenai penyedia *cloud* yang aktif, seperti *s3*, *eranya*, atau *gcs*.
3. Setelah nilai *BUCKET_STORAGE* diperoleh, misalnya bernilai *s3*, maka informasi tersebut dikembalikan ke *StorageHandler* sebagai penanda bahwa penyedia *cloud* yang digunakan adalah AWS S3.
4. File yang diterima kemudian disimpan terlebih dahulu ke dalam penyimpanan lokal. Penyimpanan sementara ini bertujuan sebagai mekanisme pengamanan, sehingga apabila terjadi kegagalan pada proses upload ke *cloud*, file masih tersedia secara lokal.

5. Karena penyedia *cloud* yang aktif adalah AWS S3, maka *StorageHandler* akan memanggil *S3Handler* untuk menangani proses penyimpanan file ke AWS S3.

6. *S3Handler* selanjutnya mengambil seluruh konfigurasi dan kredensial AWS S3 yang tersimpan pada tabel *aws_config*. Data ini mencakup seluruh parameter yang dibutuhkan untuk proses koneksi dan penyimpanan file ke S3.

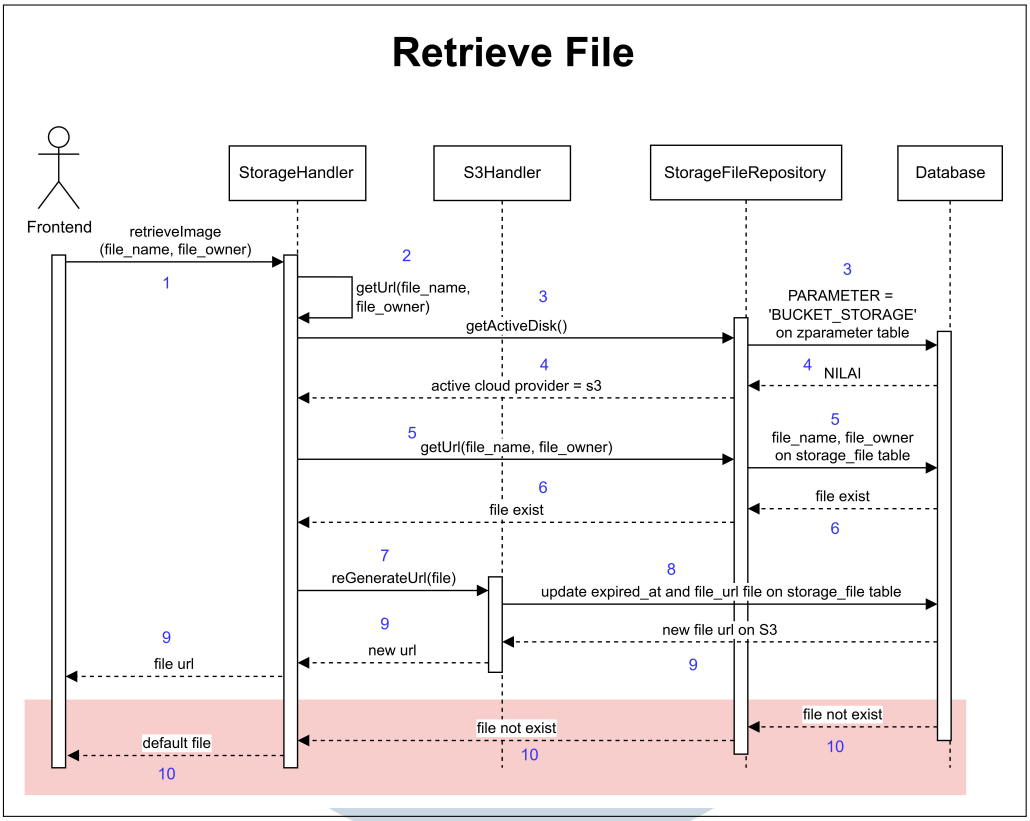
7. Setelah konfigurasi berhasil diperoleh, data tersebut digunakan oleh *S3Handler* sebagai acuan untuk menentukan *bucket* dan layanan AWS S3 yang akan digunakan.

8. *S3Handler* kemudian melakukan proses upload file ke AWS S3. Setelah proses berhasil, sistem akan menghasilkan *file_url* serta menentukan waktu kedaluwarsa file berdasarkan konfigurasi yang berlaku, yaitu tujuh hari sejak file disimpan. File yang telah disimpan secara lokal juga akan dihapus karena proses penyimpanan file ke *cloud* sudah berhasil. Seluruh informasi tersebut kemudian disimpan ke dalam tabel *storage_file*, yang meliputi *provider*, *file_name*, *file_owner*, *file_url*, *expired_at*, dan *flag_uploaded*.

9. Setiap komponen yang terlibat dalam proses ini akan mengembalikan status bahwa file telah berhasil disimpan di *cloud*. Informasi keberhasilan tersebut juga dikirimkan kembali ke *Frontend*.

C.2 Proses Pengambilan File dari *Cloud*

Proses pengambilan file dari *cloud* ditunjukkan pada Gambar 3.5. Alur proses ini dimulai dari permintaan yang dikirimkan oleh *Frontend* untuk mengakses file tertentu hingga sistem mengembalikan URL file yang valid. Selain itu, sistem juga menangani kondisi apabila URL file telah melewati masa kedaluwarsa dengan melakukan pembaruan URL sebelum dikirimkan kembali ke *Frontend*.



Gambar 3.5. Sequence diagram *retrieve file*

Tahapan proses pengambilan file dari *cloud* dijelaskan sebagai berikut:

1. Proses dimulai ketika *Frontend* mengirimkan permintaan pengambilan file ke *StorageHandler* dengan membawa parameter *file_name* dan *file_owner*.
2. *StorageHandler* kemudian memanggil fungsi *getUrl()* untuk meneruskan permintaan tersebut ke *StorageFileRepository*. Fungsi ini bertujuan untuk melakukan pencarian data file pada database.
3. *StorageHandler* kemudian memanggil fungsi *getActiveDisk()* untuk menentukan penyedia layanan *cloud* yang sedang aktif melalui *StorageFileRepository*. *StorageFileRepository* melakukan pencarian ke dalam tabel *zparameter* pada database untuk memperoleh nilai dari parameter *BUCKET_STORAGE*. Parameter ini menyimpan informasi mengenai penyedia *cloud* yang aktif, seperti *s3*, *eranya*, atau *gcs*.
4. Setelah nilai *BUCKET_STORAGE* diperoleh, misalnya bernilai *s3*, maka informasi tersebut dikembalikan ke *StorageHandler* sebagai penanda bahwa penyedia *cloud* yang digunakan adalah AWS S3.

5. *StorageFileRepository* melakukan pengecekan pada tabel *storage_file* berdasarkan parameter *file_name* dan *file_owner* yang diberikan, untuk memastikan apakah data file yang dimaksud tersedia di dalam sistem.

6. Apabila data file ditemukan, maka informasi file yang bersesuaian akan dikembalikan ke *StorageHandler* untuk diproses lebih lanjut.

7. Setiap file memiliki masa kedaluwarsa URL. Oleh karena itu, *StorageHandler* akan melakukan pengecekan apakah file tersebut telah melewati masa kedaluwarsa. Jika file telah kedaluwarsa, maka *StorageHandler* akan memanggil *S3Handler* untuk melakukan proses pembaruan URL file. Pada tahap ini, sistem akan menghasilkan URL baru dengan masa kedaluwarsa yang diperpanjang selama tujuh hari sejak proses pembaruan dilakukan.

8. URL baru beserta pembaruan nilai *expired_at* kemudian disimpan kembali ke dalam tabel *storage_file* dengan tetap mempertahankan informasi file yang lainnya.

9. Setelah proses pembaruan selesai, *StorageHandler* akan mengembalikan URL file yang valid kepada *Frontend*. Apabila URL file belum melewati masa kedaluwarsa, maka sistem akan langsung mengembalikan URL yang sudah tersimpan di database tanpa melakukan proses pembaruan.

10. Apabila data file tidak ditemukan pada tabel *storage_file*, maka sistem akan mengembalikan file bawaan (*default file*). Dalam implementasi ini, file bawaan umumnya berupa gambar *image not found* yang sebelumnya telah disimpan pada penyimpanan lokal.

Pada proses pengambilan file dari *cloud*, sistem tidak hanya digunakan untuk mengembalikan URL file guna kebutuhan tampilan pada antarmuka, tetapi juga dapat dimanfaatkan untuk kebutuhan lain, seperti proses ekspor data ke dalam file Excel. Pada proses ekspor data, khususnya untuk laporan (*report*), file gambar umumnya tidak ditampilkan dalam bentuk URL, melainkan langsung disematkan ke dalam file Excel menggunakan URL yang diperoleh dari sistem.

Meskipun demikian, terdapat perbedaan perlakuan antara pengambilan file untuk kebutuhan tampilan antarmuka dan untuk kebutuhan ekspor Excel. Pada tampilan antarmuka, apabila data file tidak ditemukan di database, sistem akan mengembalikan file bawaan (*default file*) sebagai pengganti, misalnya berupa gambar *image not found*. Sebaliknya, pada proses ekspor ke Excel, sistem tidak

mengembalikan file apa pun apabila data file tidak tersedia. Pendekatan ini diterapkan untuk menghindari penampilan elemen yang tidak relevan pada laporan Excel, sehingga hanya data yang valid dan dibutuhkan saja yang disertakan.

Selain mekanisme pengambilan URL file, sistem juga menyediakan opsi untuk memperbarui URL file dengan disertai perubahan nama file. Proses pembaruan ini memiliki alur yang hampir sama dengan proses pembuatan URL baru. Perbedaannya terletak pada atribut data yang diperbarui. Pada pembaruan URL akibat masa kedaluwarsa, atribut yang diperbarui meliputi *file_url* dan *expired_at*. Sementara itu, pada kasus pembaruan nama file, atribut yang diperbarui hanya mencakup *file_url* dan *file_name*, tanpa mengubah masa kedaluwarsa file.

3.3.3 Implementasi Modul Penyimpanan Secara *Cloud*

Bagian ini membahas implementasi modul penyimpanan file berbasis *cloud* serta perubahan yang diterapkan pada sistem. Seluruh gambar dan hasil pengujian pada bagian ini diambil pada tanggal 15 Desember 2025, sehingga status validitas data yang ditampilkan bergantung pada masa kedaluwarsa masing-masing file terhadap tanggal tersebut.

A Penyimpanan Konfigurasi AWS S3

Pada tahap ini dilakukan implementasi penyimpanan konfigurasi kredensial AWS S3 yang digunakan oleh sistem. Konfigurasi tersebut disimpan pada tabel *aws_config*, yang berfungsi sebagai media penyimpanan parameter-parameter koneksi ke layanan AWS S3 secara terpusat dan terstruktur.

Struktur tabel *aws_config* dibuat mengacu pada hasil perancangan basis data yang telah dibahas pada bagian sebelumnya. Oleh karena itu, pada tahap implementasi ini tidak dilakukan pembahasan ulang terhadap fungsi masing-masing atribut, melainkan difokuskan pada realisasi teknis dalam bentuk migrasi database. Kode 3.1 merupakan potongan kode untuk membuat tabel baru di database.

```
Schema::create('aws_config', function (Blueprint $table) {
    $table->string('aws_config_key', 200);
    $table->longText('aws_config_value');
    $table->primary('aws_config_key');
});
```

Kode 3.1: Migrasi tabel *aws_config*

Setelah struktur tabel dibuat, tahap selanjutnya adalah pengisian data awal menggunakan mekanisme *database seeder*. Proses ini bertujuan untuk menyimpan nilai-nilai konfigurasi AWS S3 yang diperlukan oleh sistem, seperti *access key*, *secret key*, nama *bucket*, dan *region*. Kode 3.2 merupakan proses pengisian data untuk tabel *aws_config* di database. Untuk menjaga keamanan data sensitif, nilai konfigurasi disimpan dalam kondisi terenkripsi sebelum dimasukkan ke dalam database.

```
AwsConfig::create([
    'aws_config_key' => 'AWS_ACCESS_KEY_ID',
    'aws_config_value' => Encrypt('awsAccessKey')
],
[
    'aws_config_key' => 'AWS_SECRET_ACCESS_KEY',
    'aws_config_value' => Encrypt('awsSecretAccessKey')
],
[
    'aws_config_key' => 'AWS_BUCKET',
    'aws_config_value' => Encrypt('awsBucket')
],
[
    'aws_config_key' => 'AWS_DEFAULT_REGION',
    'aws_config_value' => Encrypt('awsDefaultRegion')
]);
```

Kode 3.2: *Seeder* tabel *aws_config*

Selain pembuatan tabel baru, tahap implementasi selanjutnya adalah penambahan parameter global yang digunakan untuk menentukan penyedia layanan *cloud* yang aktif pada sistem. Parameter global ini disimpan pada tabel *zparameter* dan berfungsi sebagai konfigurasi terpusat yang dapat diubah sesuai dengan kebutuhan pengguna.

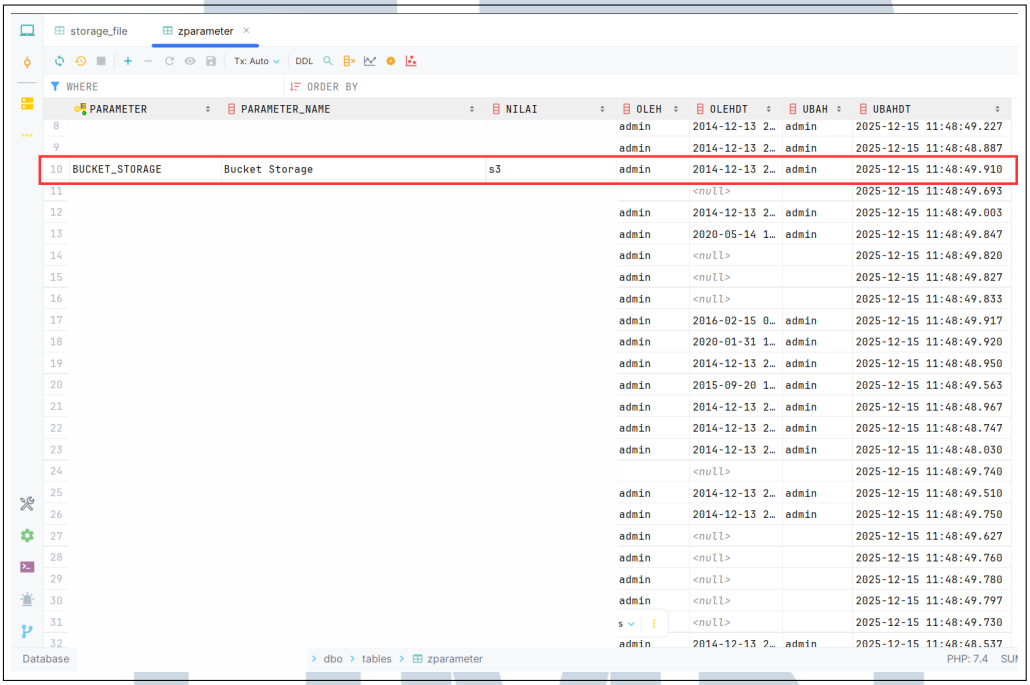
Kode 3.3 menunjukkan proses penambahan parameter *BUCKET_STORAGE* ke dalam tabel *zparameter*. Nilai dari parameter tersebut merepresentasikan penyedia *cloud* yang digunakan oleh sistem. Pada implementasi ini, nilai *s3* menunjukkan bahwa penyedia layanan *cloud* yang aktif adalah AWS S3.

```
Parameter::create([
    'PARAMETER' => 'BUCKET_STORAGE',
    'PARAMETER_NAME' => 'Bucket Storage',
    'NILAI' => 's3',
    'OLEH' => 'admin',
    'UBAH' => 'admin',
```

```
7 ]) ;
```

Kode 3.3: *Seeder* parameter *BUCKET_STORAGE* pada tabel zparameter

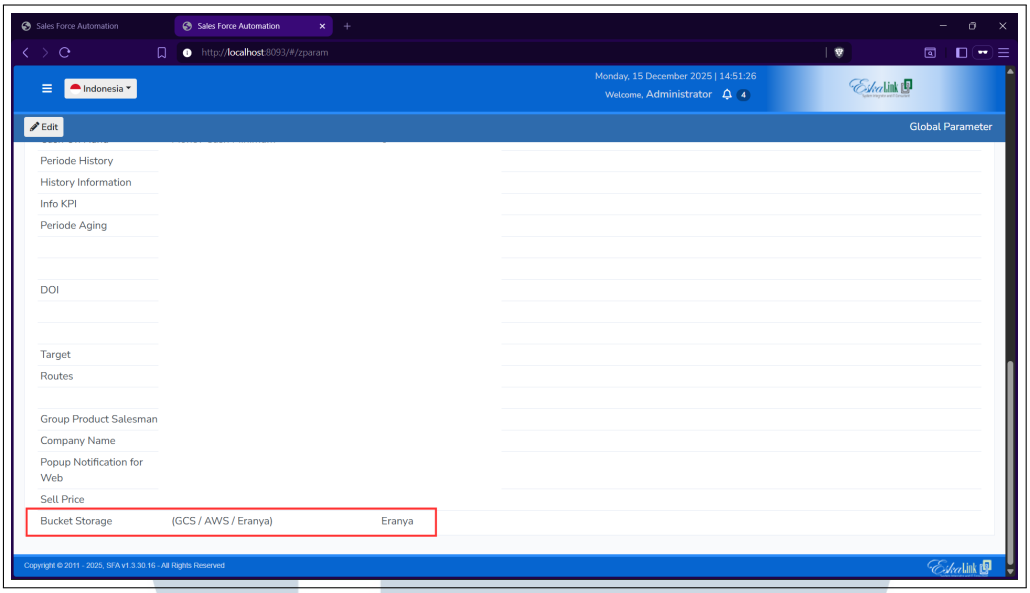
Setelah proses *seeder* berhasil dijalankan, dilakukan verifikasi untuk memastikan bahwa parameter *BUCKET_STORAGE* telah tersimpan dengan benar di dalam tabel *zparameter*. Gambar 3.6 menampilkan kondisi tabel *zparameter*, yang menunjukkan bahwa nilai parameter *BUCKET_STORAGE* telah terisi dengan nilai *s3*.



Gambar 3.6. Tabel *zparameter* untuk verifikasi parameter *BUCKET_STORAGE*

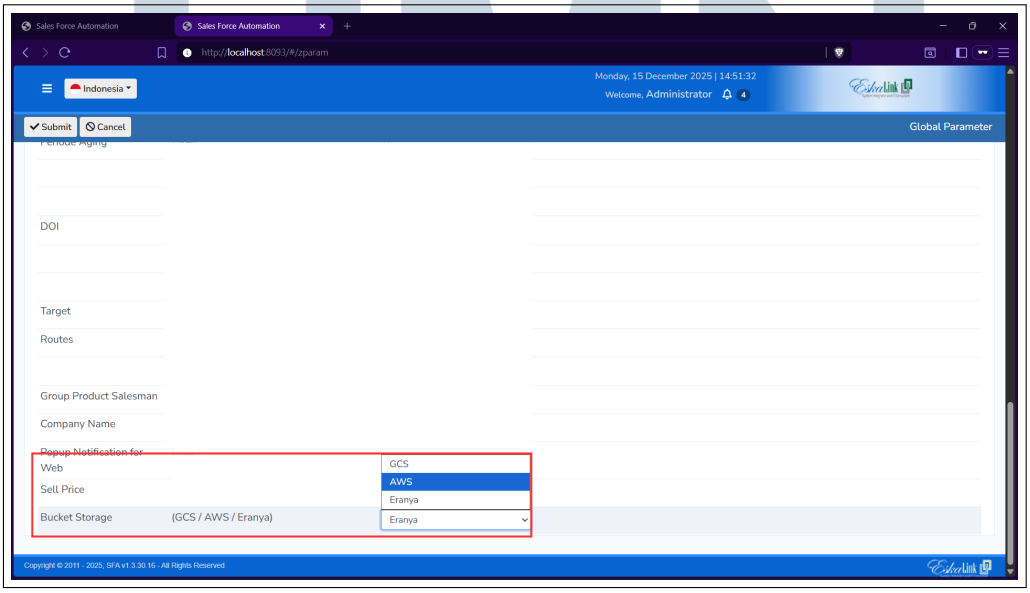
Keberadaan parameter *BUCKET_STORAGE* pada database selanjutnya dimanfaatkan pada sisi antarmuka pengguna. Pada menu pengaturan Global Parameter, ditambahkan satu konfigurasi baru yang memungkinkan pengguna untuk memilih penyedia layanan *cloud* yang akan digunakan oleh sistem.

Gambar 3.7 menunjukkan tampilan antarmuka Global Parameter yang menampilkan penyedia *cloud* yang sedang aktif. Apabila sistem memiliki lebih dari satu penyedia *cloud* yang terdaftar, maka user bisa mengganti penyedia *cloud* tersebut.



Gambar 3.7. Tampilan antarmuka pengguna menu Global Parameter untuk menampilkan penyedia *cloud* aktif

Selain menampilkan informasi penyedia *cloud* yang aktif, sistem juga menyediakan fitur untuk mengubah penyedia layanan *cloud*. Gambar 3.8 memperlihatkan tampilan antarmuka ketika pengguna melakukan perubahan penyedia *cloud*. Pada implementasi ini, pengguna dapat memilih penyedia *cloud* yang tersedia, seperti AWS S3, Google Cloud Storage (GCS), atau Eranya.

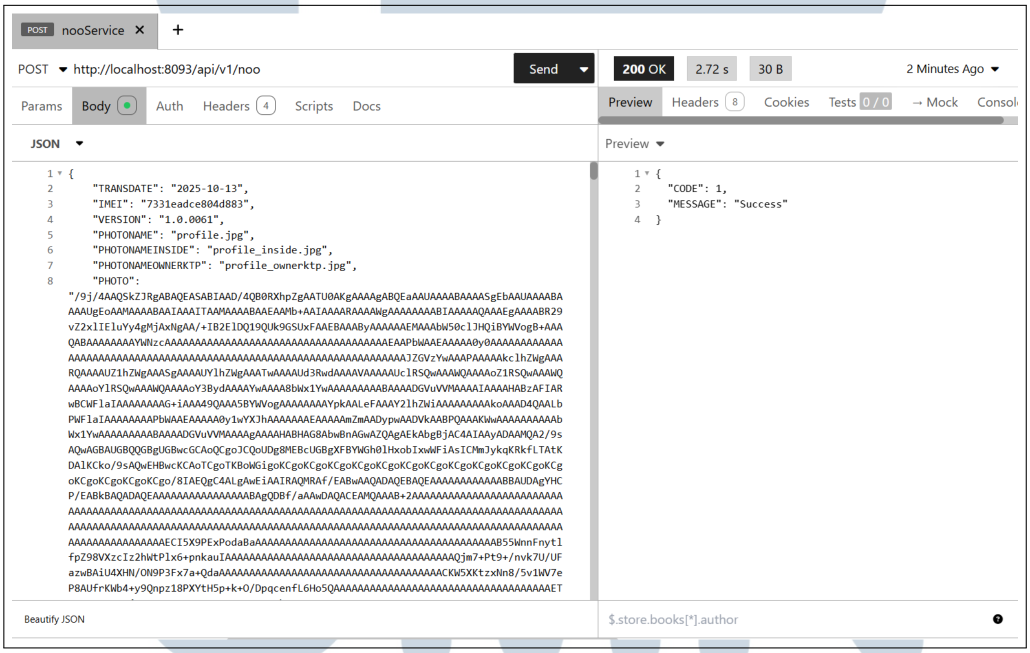


Gambar 3.8. Tampilan antarmuka pengguna menu Global Parameter untuk mengubah penyedia layanan *cloud*

B Penyimpanan File Melalui API dan Antarmuka Pengguna

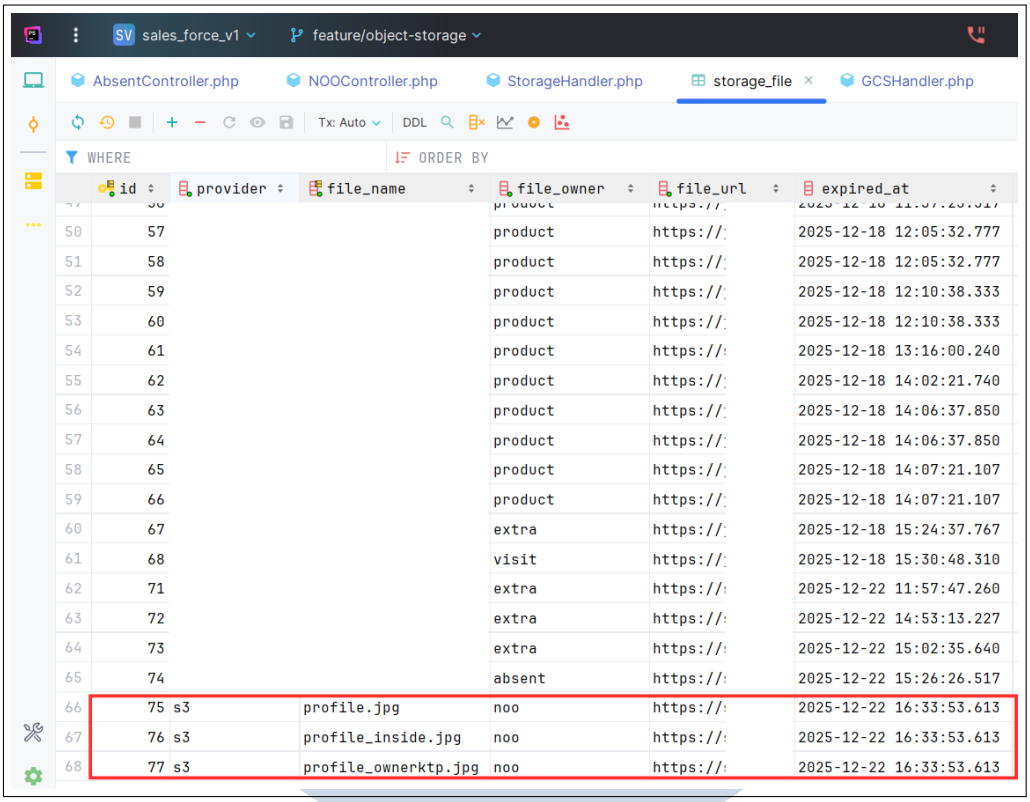
Sesuai dengan penjelasan sebelumnya, proses penyimpanan file pada sistem dapat dilakukan melalui dua mekanisme, yaitu melalui *Application Programming Interface* (API) dan melalui antarmuka pengguna (*user interface*). Pada bagian ini ditunjukkan contoh implementasi penyimpanan file melalui API.

Gambar 3.9 memperlihatkan percobaan pengiriman file melalui API *NooService* menggunakan *software* Insomnia. Pada percobaan tersebut, terdapat tiga file gambar yang dikirimkan, yaitu *profile.jpg*, *profile_inside.jpg*, dan *profile_ownerktp.jpg*. File yang dikirim melalui API terlebih dahulu dikonversi ke dalam format *base64*. Selanjutnya, data *base64* yang diterima oleh sistem akan dikonversi kembali ke dalam format file aslinya sebelum disimpan.



Gambar 3.9. Pengujian pengiriman file melalui API menggunakan Insomnia

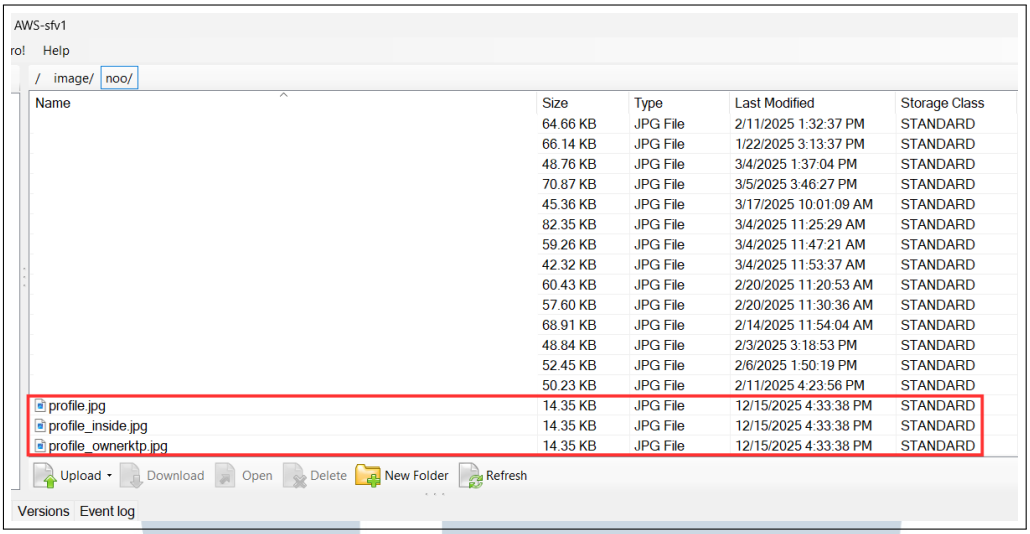
Setelah proses pengiriman file melalui API dilakukan, tahap selanjutnya adalah melakukan verifikasi penyimpanan data pada sisi database. Sistem terlebih dahulu memastikan bahwa informasi file telah tercatat pada tabel *storage_file*. Gambar 3.10 menunjukkan kondisi tabel *storage_file* setelah proses penyimpanan, di mana ketiga file berhasil tersimpan dan penyedia layanan *cloud* yang digunakan tercatat sebagai *s3* pada kolom *provider*.



	id	provider	file_name	file_owner	file_url	expired_at
50	57			product	https://	2025-12-18 12:05:32.777
51	58			product	https://	2025-12-18 12:05:32.777
52	59			product	https://	2025-12-18 12:10:38.333
53	60			product	https://	2025-12-18 12:10:38.333
54	61			product	https://	2025-12-18 13:16:00.240
55	62			product	https://	2025-12-18 14:02:21.740
56	63			product	https://	2025-12-18 14:06:37.850
57	64			product	https://	2025-12-18 14:06:37.850
58	65			product	https://	2025-12-18 14:07:21.107
59	66			product	https://	2025-12-18 14:07:21.107
60	67			extra	https://	2025-12-18 15:24:37.767
61	68			visit	https://	2025-12-18 15:30:48.310
62	71			extra	https://	2025-12-22 11:57:47.260
63	72			extra	https://	2025-12-22 14:53:13.227
64	73			extra	https://	2025-12-22 15:02:35.640
65	74			absent	https://	2025-12-22 15:26:26.517
66	75	s3	profile.jpg	noo	https://	2025-12-22 16:33:53.613
67	76	s3	profile_inside.jpg	noo	https://	2025-12-22 16:33:53.613
68	77	s3	profile_ownerktp.jpg	noo	https://	2025-12-22 16:33:53.613

Gambar 3.10. Data file yang tersimpan pada tabel *storage_file*

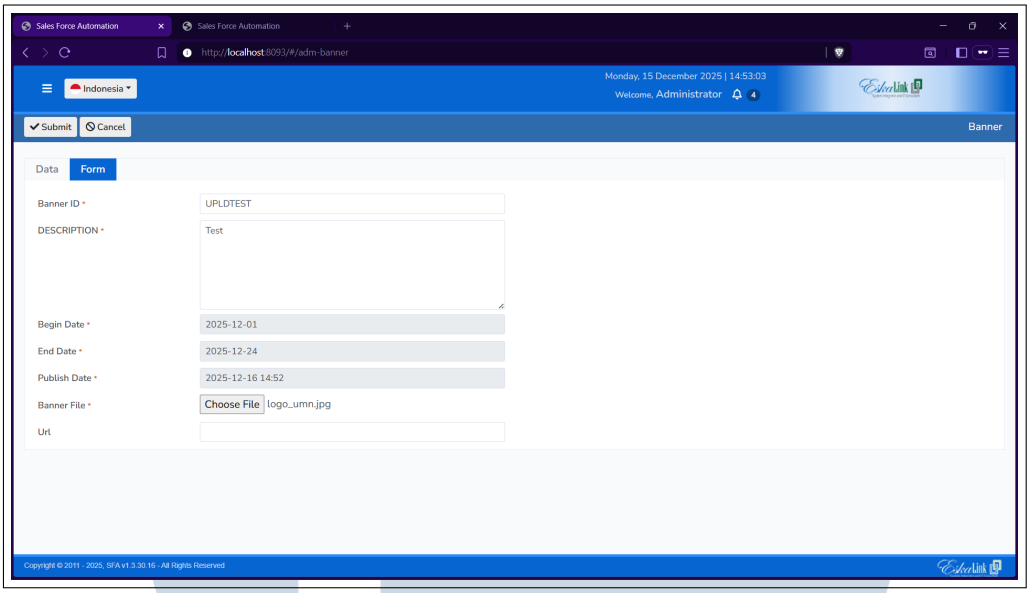
Sebagai tahap akhir, dilakukan verifikasi langsung pada layanan penyimpanan AWS S3 untuk memastikan bahwa file benar-benar telah tersimpan pada *bucket* yang dituju. Verifikasi ini dilakukan menggunakan aplikasi S3 Browser, yaitu perangkat lunak antarmuka grafis untuk mengelola penyimpanan *cloud*. Gambar 3.11 memperlihatkan bahwa ketiga file yang dikirimkan melalui API telah berhasil tersimpan pada AWS S3 sesuai dengan data yang tercatat di database.



Gambar 3.11. Verifikasi penyimpanan file pada AWS S3 menggunakan S3 Browser

Setelah dilakukan pengujian penyimpanan file melalui API, tahap selanjutnya adalah melakukan pengujian penyimpanan file melalui antarmuka pengguna. Pengujian ini bertujuan untuk memastikan bahwa proses unggah file melalui antarmuka pengguna dapat berjalan dengan baik dan terintegrasi dengan mekanisme penyimpanan *cloud* yang sama.

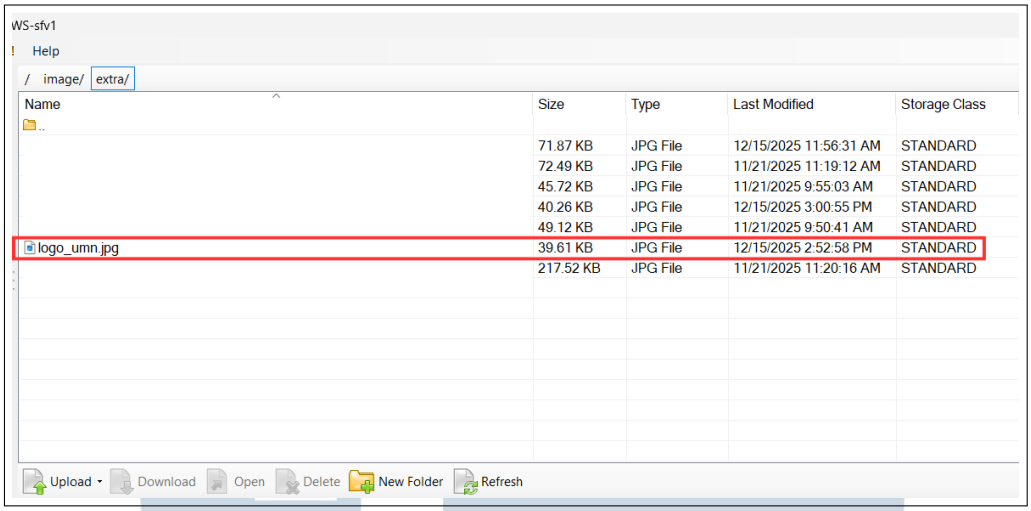
Gambar 3.12 memperlihatkan tampilan menu pada antarmuka pengguna ketika melakukan pemilihan file yang akan disimpan. Pada pengujian ini, file dengan nama *logo_umn.jpg* dipilih untuk diunggah ke sistem. Setelah pengguna melakukan aksi *submit*, sistem akan memproses file tersebut dan meneruskannya ke layanan penyimpanan *cloud* yang aktif, yaitu AWS S3.



Gambar 3.12. Tampilan menu pengunggahan file melalui antarmuka pengguna

Untuk memastikan bahwa file benar-benar tersimpan pada layanan AWS S3, dilakukan pengecekan langsung menggunakan aplikasi S3Browser. Pengecekan ini bertujuan untuk memverifikasi bahwa file yang diunggah melalui antarmuka pengguna tidak hanya tercatat pada sistem, tetapi juga telah tersimpan secara fisik pada *bucket* AWS S3.

Gambar 3.13 menunjukkan bahwa file dengan nama *logo_umn.jpg* telah berhasil tersimpan pada AWS S3. Keberadaan file tersebut pada *bucket* yang sesuai menandakan bahwa proses unggah file melalui antarmuka pengguna telah berhasil diteruskan dan diproses oleh layanan penyimpanan *cloud* tanpa terjadi kesalahan.



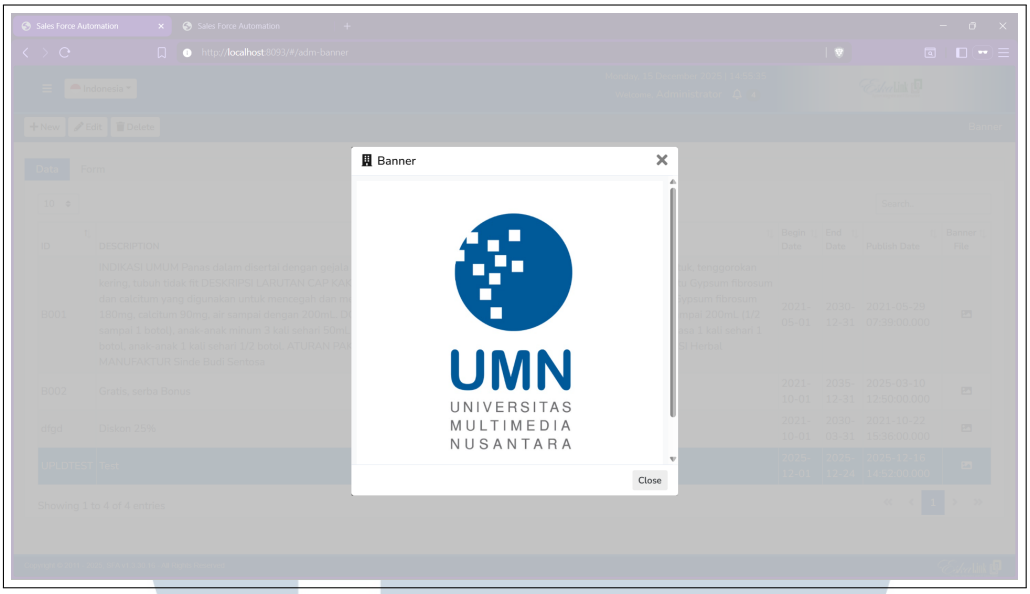
Gambar 3.13. Verifikasi penyimpanan file melalui antarmuka pengguna pada AWS S3 menggunakan S3Browser

C Pengambilan File dari *Cloud*

Setelah proses penyimpanan file gambar ke layanan *cloud* berhasil dilakukan, tahap selanjutnya adalah pengambilan dan penampilan file tersebut pada antarmuka pengguna. Tahap ini bertujuan untuk memastikan bahwa file gambar yang telah disimpan dapat diakses kembali dan ditampilkan dengan benar oleh sistem.

Pada proses penyimpanan sebelumnya, file gambar dengan nama *logo_umn.jpg* diunggah melalui antarmuka pengguna. Melalui menu yang sama, sistem menyediakan fitur untuk menampilkan kembali file gambar yang telah disimpan. Hal ini memungkinkan pengguna untuk memverifikasi secara langsung bahwa file gambar benar-benar tersimpan pada layanan *cloud*.

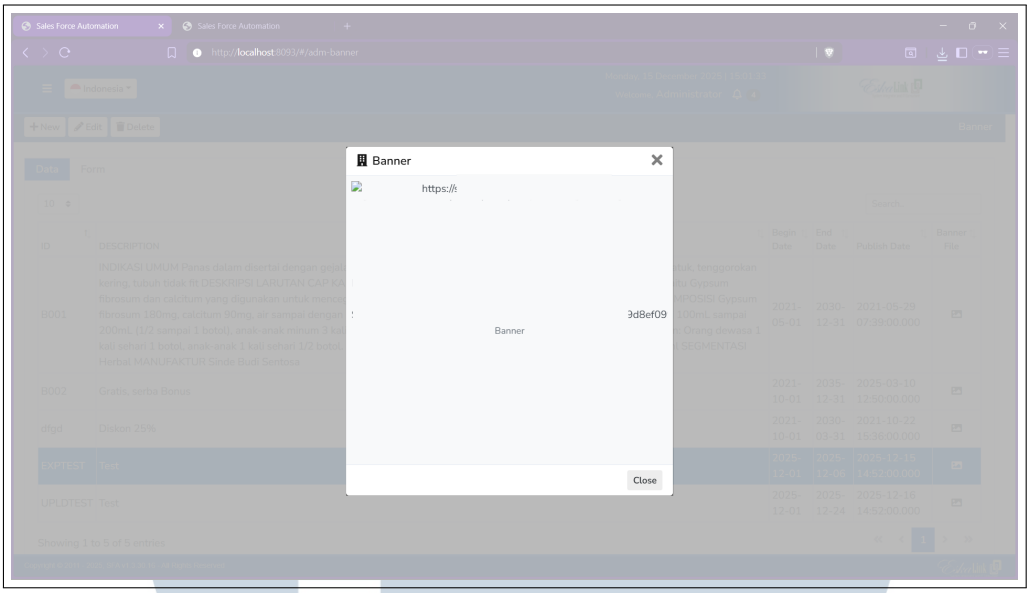
Gambar 3.14 menunjukkan tampilan file gambar *logo_umn.jpg* yang berhasil diambil dari layanan *cloud* dan ditampilkan pada antarmuka pengguna. Tampilan tersebut menandakan bahwa proses pengambilan file dari *cloud* telah berjalan dengan baik dan file dapat diakses sesuai dengan yang diharapkan.



Gambar 3.14. Tampilan file gambar *logo_umn.jpg* pada antarmuka pengguna

Pada sequence diagram pengambilan file (*retrieve file*) dijelaskan mekanisme *reGenerateUrl*, yaitu proses pembaruan URL file beserta masa kedaluwarsanya apabila file telah melewati batas waktu akses. Pada bagian ini dijelaskan implementasi mekanisme tersebut ketika file yang disimpan telah memasuki masa kedaluwarsa.

Secara default, masa kedaluwarsa file ditentukan tujuh hari setelah proses penyimpanan dilakukan. Untuk keperluan pengujian, masa kedaluwarsa file sengaja diatur menjadi satu hari sebelumnya, yaitu pada tanggal 14 Desember 2025. Dengan pengaturan tersebut, file gambar yang telah melewati masa kedaluwarsa tidak dapat ditampilkan pada antarmuka pengguna, sebagaimana ditunjukkan pada Gambar 3.15.



Gambar 3.15. Tampilan file gambar yang tidak dapat diakses karena telah melewati masa kedaluwarsa

Ketika kondisi tersebut terjadi, sistem akan menjalankan proses pengambilan file dengan melakukan pembaruan (*regenerate*) terhadap URL file dan nilai masa kedaluwarsa. Proses ini dilakukan secara otomatis saat file diakses kembali. Gambar 3.16a menunjukkan data file sebelum dilakukan pembaruan, sedangkan Gambar 3.16b menunjukkan data file setelah proses pembaruan berhasil dilakukan.

61	68				https://	2025-12-18 15:30:48.310
62	71				https://	2025-12-22 11:57:47.260
63	72				https://	2025-12-22 14:53:13.227
64	73	s3	cat_flower.jpg	extra	https://	2025-12-14 15:01:12.520

(a) Data file sebelum pembaruan masa kedaluwarsa

61	68				https://	2025-12-18 15:30:48.310
62	71				https://	2025-12-22 11:57:47.260
63	72				https://	2025-12-22 14:53:13.227
64	73	s3	cat_flower.jpg	extra	https://	2025-12-22 15:02:35.640

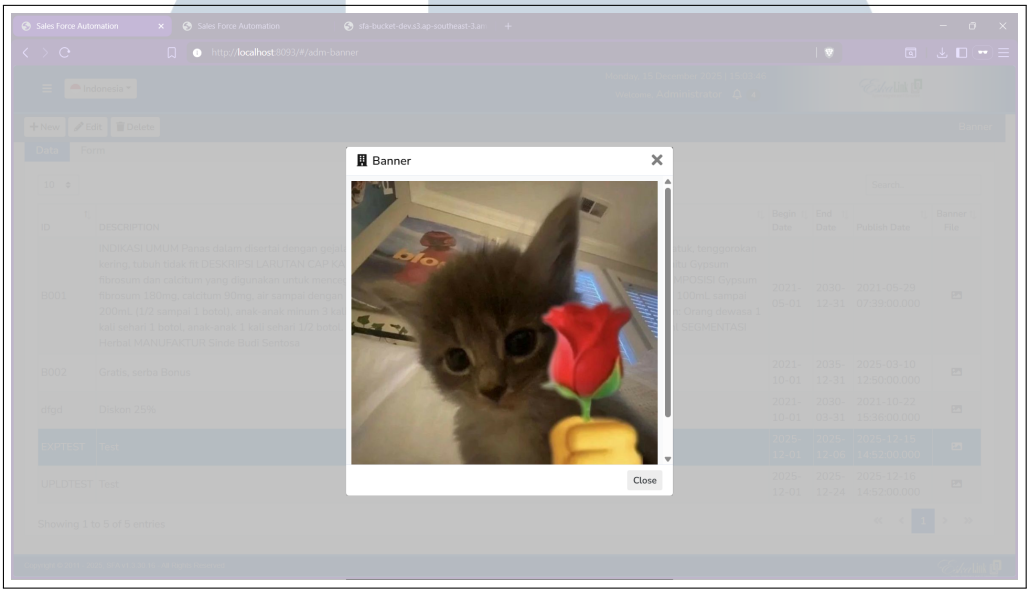
(b) Data file setelah pembaruan masa kedaluwarsa

Gambar 3.16. Perbandingan data file sebelum dan sesudah proses pembaruan URL

Berdasarkan perbandingan tersebut, terlihat bahwa nilai *expired_date* yang sebelumnya tercatat sebagai 14 Desember 2025 berubah menjadi 22 Desember 2025. Perubahan ini menunjukkan bahwa proses pembaruan URL dan masa

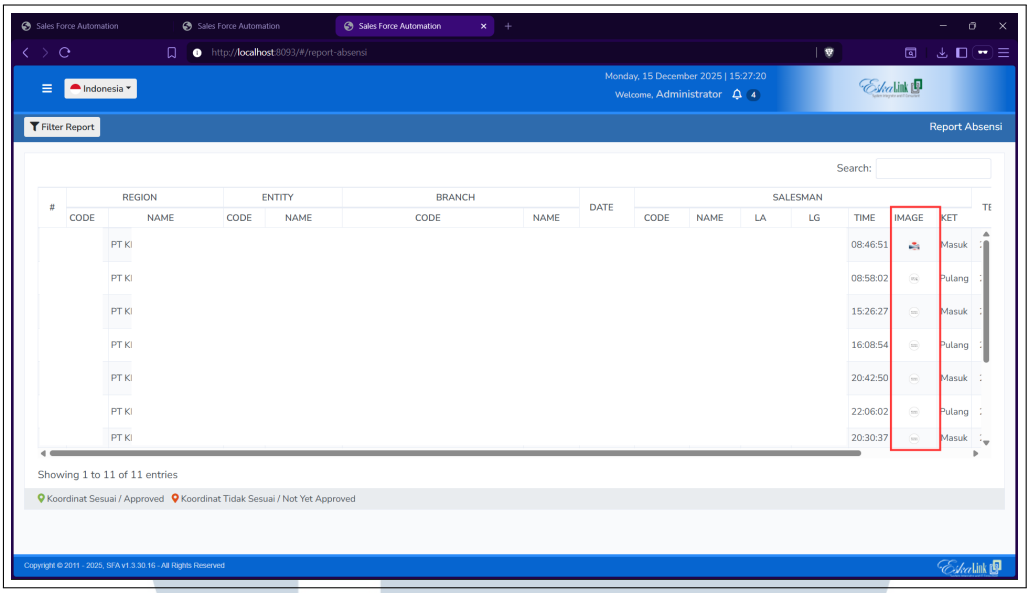
kedaluwarsa file telah berjalan dengan baik.

Setelah proses pembaruan dilakukan, file gambar kembali diakses melalui antarmuka pengguna. Gambar 3.17 memperlihatkan bahwa file gambar yang sebelumnya tidak dapat diakses kini telah berhasil ditampilkan kembali, menandakan bahwa URL file pada layanan *cloud* telah diperbarui dan kembali valid.



Gambar 3.17. Tampilan file gambar setelah pembaruan masa kedaluwarsa

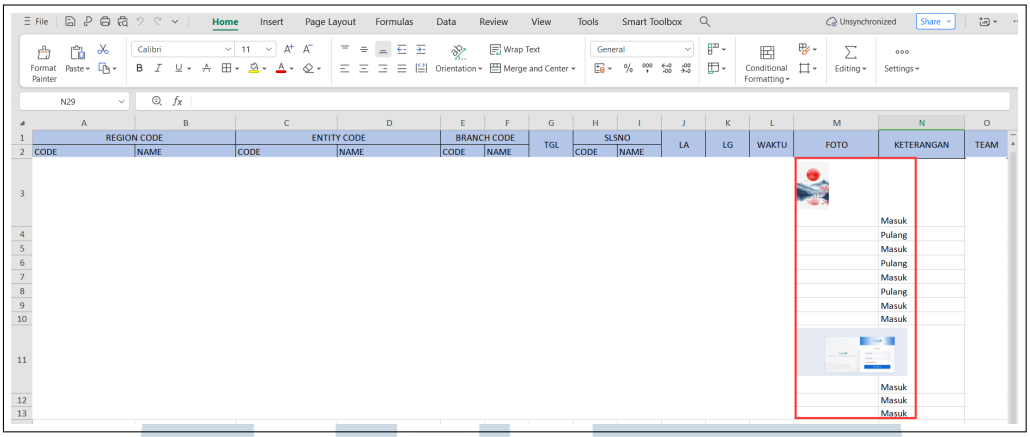
Selain ditampilkan pada antarmuka pengguna, file gambar yang disimpan pada layanan *cloud* juga digunakan pada fitur laporan. Gambar 3.18 menunjukkan menu laporan absensi yang memuat kolom *IMAGE*, di mana file gambar ditampilkan secara langsung pada antarmuka pengguna menggunakan URL yang diperoleh dari layanan *cloud*.



Gambar 3.18. Tampilan laporan pada antarmuka pengguna yang memuat file gambar dari *cloud*

Salah satu kebutuhan dari pengguna sistem adalah kemampuan untuk mengekspor data laporan ke dalam format Excel. Pada proses ekspor ini, file gambar tidak ditampilkan dalam bentuk URL, melainkan disematkan secara langsung ke dalam file Excel. Dengan pendekatan ini, tampilan laporan pada file Excel dapat menyerupai tampilan laporan pada antarmuka pengguna.

Gambar 3.19 menunjukkan hasil ekspor laporan ke dalam format Excel, di mana file gambar yang tersimpan pada layanan *cloud* berhasil ditampilkan secara langsung. Apabila file gambar tersedia dan dapat diakses pada layanan *cloud*, maka gambar akan ditampilkan pada file Excel. Sebaliknya, apabila file gambar tidak ditemukan atau tidak dapat diakses, maka kolom gambar pada file Excel akan dibiarkan kosong.



Gambar 3.19. Hasil ekspor laporan ke file Excel dengan file gambar

D Implementasi dan Pengujian pada Modul Terkait

Setelah seluruh fitur pada modul penyimpanan file berbasis *cloud* selesai diimplementasikan dan berhasil melalui berbagai skenario pengujian, tahap selanjutnya adalah menerapkan mekanisme tersebut pada modul-modul lain yang telah ada sebelumnya. Tahap ini bertujuan untuk memastikan bahwa seluruh proses penyimpanan dan pengambilan file pada sistem menggunakan mekanisme berbasis *cloud* secara konsisten.

Modul-modul yang terdampak oleh perubahan ini dikelompokkan ke dalam beberapa kategori, yaitu penyimpanan file melalui API, penyimpanan file melalui antarmuka pengguna, menu laporan (*report*), menu persetujuan (*approval*), serta proses ekspor data ke dalam file Excel. Secara keseluruhan, terdapat sekitar 60 berkas program yang disesuaikan agar mendukung mekanisme penyimpanan dan pengambilan file berbasis *cloud*.

Modul-modul tersebut sebelumnya masih menggunakan mekanisme penyimpanan file secara lokal. Oleh karena itu, dilakukan penyesuaian pada setiap modul agar seluruh proses pengelolaan file mengacu pada penyedia layanan *cloud* yang sedang aktif. Meskipun jumlah modul yang diubah cukup banyak, sebagian besar modul memiliki struktur dan alur proses yang serupa.

Dengan adanya keseragaman struktur dan mekanisme tersebut, proses implementasi dan pengujian dapat dilakukan secara lebih efisien. Pengujian dilakukan dengan cara mencoba secara langsung setiap modul yang telah disesuaikan, khususnya untuk memastikan bahwa proses penyimpanan file benar-benar berhasil dan file tersimpan pada *bucket storage* milik penyedia layanan

cloud yang digunakan. Mengingat sistem mendukung tiga penyedia layanan, yaitu AWS S3, Google Cloud Storage (GCS), dan Eranya, maka pengujian dilakukan pada masing-masing penyedia untuk memastikan kompatibilitas dan konsistensi perilaku sistem. Apabila dalam proses pengujian ditemukan kesalahan, maka kesalahan tersebut dicatat dan dianalisis lebih lanjut, mengingat pada beberapa kasus ditemukan kondisi di mana proses penyimpanan mengalami kegagalan pada percobaan awal namun berhasil pada percobaan berikutnya.

Adapun untuk proses pengambilan file, pengujian tidak dilakukan pada seluruh modul secara menyeluruh. Hal ini disebabkan karena mekanisme pengambilan file yang diterapkan memiliki alur dan struktur yang sama pada setiap modul. Oleh karena itu, pengujian dilakukan secara perwakilan pada beberapa modul, dengan asumsi bahwa keberhasilan pada satu modul dapat merepresentasikan keberhasilan modul lain yang menggunakan mekanisme serupa.

3.4 Kendala dan Solusi yang Ditemukan

Pada bagian ini dijelaskan berbagai kendala yang ditemui selama pelaksanaan program magang, khususnya dalam proses pengembangan dan implementasi sistem, serta solusi yang diterapkan untuk mengatasi kendala tersebut.

3.4.1 Kendala

Berdasarkan pelaksanaan program magang di PT ESKA LINK, terdapat beberapa kendala yang ditemui selama proses pengembangan sistem, di antaranya sebagai berikut:

1. Struktur alur kerja (*flow*) dan fungsi-fungsi pada produk IDS relatif kompleks dan sulit dipahami, terutama karena minimnya dokumentasi yang tersedia. Kondisi ini menyulitkan proses penyesuaian dan pengembangan fitur baru pada modul yang sudah ada.
2. Pada beberapa kasus, setelah proses pengembangan suatu menu atau modul selesai dilakukan, data yang dibutuhkan untuk proses pengujian tidak tersedia. Hal ini menyulitkan proses verifikasi apakah fitur yang dikembangkan telah berjalan sesuai dengan kebutuhan atau belum.

3.4.2 Solusi

Berdasarkan kendala-kendala yang telah diidentifikasi, beberapa solusi yang diterapkan untuk mengatasinya adalah sebagai berikut:

1. Untuk memahami alur kerja dan fungsi pada setiap modul, dilakukan penelusuran kode secara bertahap dengan membaca setiap fungsi dan file-file yang terkait dengan modul utama. Proses ini dibantu dengan pemanfaatan teknologi *Artificial Intelligence* (AI), seperti ChatGPT, untuk memperoleh pemahaman awal mengenai fungsi dan tujuan dari modul yang dikembangkan.
2. Untuk mengatasi keterbatasan data pada proses pengujian, dibuat data *dummy* yang disesuaikan dengan kebutuhan pengujian. Data *dummy* tersebut dibentuk berdasarkan struktur dan pola data yang sudah ada pada tabel database, dengan menerapkan variasi nilai pada kondisi tertentu agar sesuai dengan skenario pengujian yang diinginkan.