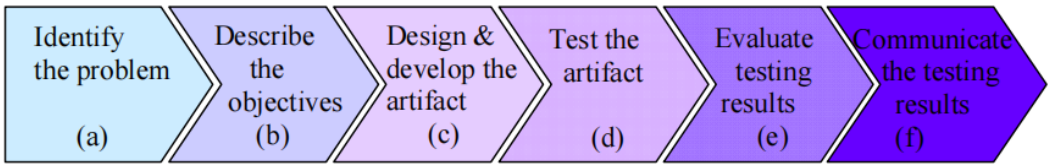
BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Metode penelitian yang akan diterapkan dalam penelitian ini adalah Model *Design and Development* (D&D). Ellis dan Levy (2010) menyatakan bahwa, "*two essential aspects of the defining characteristics of design and development research emerge. The design and development research results in production of some form of artifact, and the process is indeed research, not to be confused with product development*". Berdasarkan definisi tersebut, dapat disimpulkan bahwa model (D&D) merupakan pendekatan penelitian yang berfokus pada penciptaan artefak sebagai solusi terhadap permasalahan yang ada, dengan tetap menekankan aspek ilmiah agar kontribusinya tidak hanya bersifat praktis, tetapi juga akademis. Mengacu pada kerangka yang dikembangkan oleh Ellis dan Levy (2010), model ini terdiri dari enam tahapan utama, yaitu pertama *identify the problem motivating the research*, kedua *describe the objectives*, ketiga *design and develop the artifact*, keempat *subject the artifact to testing*, kelima *evaluate the results of testing*, dan yang terakhir adalah *communicate those results*. Tahapan model (D&D) tersebut dapat dilihat pada Gambar 3.1.



Gambar 3.1 Model *Design and Development* (D&D) (Ellis & Levy, 2010)

3.2 Identifikasi Masalah (*Identify the problem*)

Peningkatan penggunaan teknologi digital di berbagai sektor mendorong perpindahan sistem penyimpanan data dari media lokal menuju sistem berbasis internet yang dapat diakses dari mana saja. Hal ini memunculkan tantangan baru dalam menjaga keamanan data digital, terlebih dengan terus meningkatnya ancaman siber seperti *malware, trojan*, dan akses tidak sah. Berdasarkan data Badan Siber dan Sandi Negara (BSSN), terdapat lebih dari 122 juta anomali trafik internet yang terdeteksi pada periode Januari hingga Agustus 2024, menunjukkan tingginya risiko serangan terhadap data yang dikirim atau disimpan secara *online*. Hal ini diiringi juga dengan jumlah pengguna internet yang sangat banyak, yaitu lebih dari 221 juta jiwa menurut laporan APJII, sehingga memperluas potensi ancaman terhadap keamanan data.

Menanggapi tantangan tersebut, penelitian ini berfokus pada perancangan dan pengembangan aplikasi web berbasis *cloud* yang mampu menjaga kerahasiaan data. Salah satu kendala umum dalam penerapan sistem pengamanan data adalah bertambahnya ukuran *file* setelah melalui proses enkripsi, yang dapat berdampak pada efisiensi ukuran *file*. Untuk mengatasi hal tersebut, sistem ini dirancang dengan pendekatan yang tidak hanya berfokus pada aspek keamanan, tetapi juga mempertimbangkan efisiensi ukuran *file* dengan menerapkan proses kompresi sebelum data dienkripsi. Dalam penerapannya, aplikasi ini memanfaatkan layanan *cloud* sebagai media pendukung, sehingga dapat diakses secara *online*.

3.3 Mendeskripsikan Tujuan (*Describe the objectives*)

Sebagai upaya dalam menjawab tantangan keamanan data yang telah dijelaskan sebelumnya, maka perancangan dan pengembangan aplikasi ini dilakukan sebagai solusi yang mampu menjaga kerahasiaan *file* sekaligus mengatasi pembengkakan ukuran *file* yang umum terjadi setelah proses enkripsi. Aplikasi ini menerapkan pendekatan kompresi dan enkripsi secara terintegrasi, dengan tujuan menjaga data tetap aman, sekaligus mengoptimalkan ukuran *file* agar lebih efisien untuk disimpan secara *online*.

Untuk memastikan perlindungan data, sistem ini mengadopsi algoritma AES-256, yang dikenal karena ketahanannya terhadap serangan *brute force* dan tingkat keamanannya yang tinggi dalam berbagai implementasi sistem. Sementara itu, untuk mengatasi masalah efisiensi ukuran *file*, digunakan teknik kompresi Deflate, yang menggabungkan metode LZ77 dan Huffman coding, sehingga mampu mengurangi ukuran file secara signifikan sebelum dilakukan proses enkripsi.

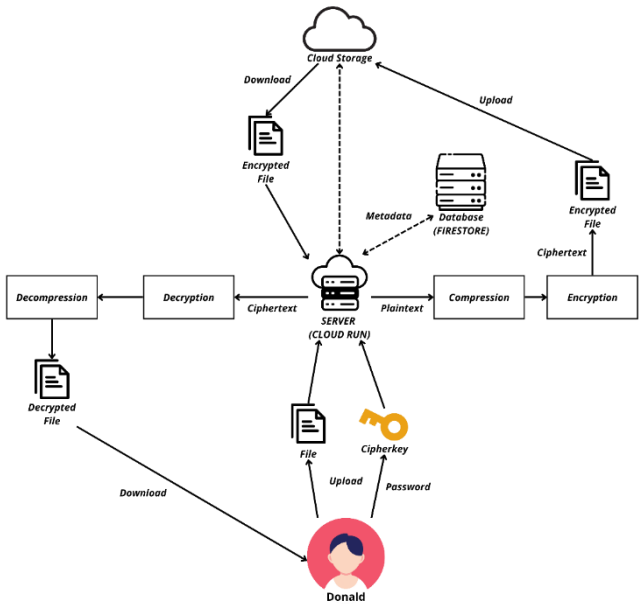
Aplikasi ini dijalankan menggunakan layanan GCP sebagai infrastruktur pendukung. Layanan Cloud Run digunakan untuk *deployment* sistem, Cloud Storage digunakan sebagai tempat penyimpanan *file* yang telah dikompresi dan dienkripsi, dan Firestore dimanfaatkan untuk menyimpan *metadata file* serta data pengguna.

3.4 Desain dan Pengembangan Sistem (*Design & develop the artifact*)

Setelah tujuan penelitian dijelaskan, langkah selanjutnya adalah merancang dan mengembangkan sistem. Bagian ini akan membahas perancangan dan pengembangan sistem yang akan diterapkan dalam penelitian ini, yang disusun kedalam beberapa poin utama sebagai berikut:

1. Diagram Arsitektur

   Diagram arsitektur merupakan representasi visual dari alur kerja sistem secara keseluruhan, yang menggambarkan bagaimana seluruh komponen dalam sistem saling berinteraksi untuk mencapai tujuan yang diharapkan. Dalam konteks penelitian ini, diagram arsitektur digunakan untuk menjelaskan proses pengamanan *file* melalui tahapan kompresi dan enkripsi, serta proses pemulihan *file* melalui dekripsi dan dekompresi. Pada Gambar 3.2 berikut menampilkan representasi arsitektur aplikasi web keamanan *file* yang dikembangkan.



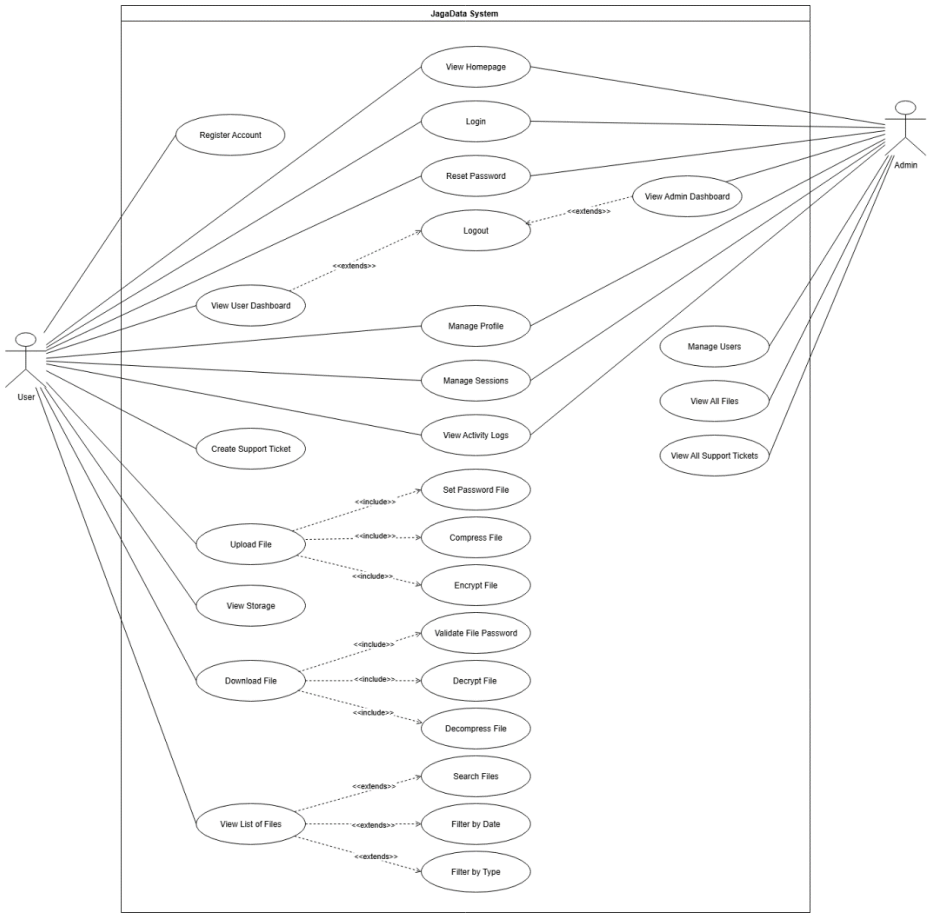
Gambar 3.2 Arsitektur Diagram Aplikasi Web Keamanan *File*

Pada Gambar 3.2, pengguna bernama Donald mengunggah *file* ke *server* (Cloud Run) disertai dengan *password* yang akan digunakan untuk mengamankan *file*. *Server* kemudian memproses *file* tersebut dengan cara mengompresinya dan mengenkripsinya menggunakan *cipher key* yang dibentuk dari *password* yang diberikan. Hasil akhirnya adalah berupa file terenkripsi *(encrypted file)* yang kemudian disimpan di Cloud Storage. Selain itu, *metadata* dari *file* seperti nama, dan ukuran disimpan di Firestore untuk mempermudah proses pencarian dan pengelolaan *file* di kemudian hari. Jika Donald ingin mengunduh *file* yang telah terenkripsi, Donald dapat memilih *file* yang akan didekripsi, kemudian memasukkan *password* yang sesuai. Jika *password* valid, sistem akan menggunakan *metadata* dari Firestore untuk menemukan *file* terenkripsi yang dimaksud, lalu mengunduhnya ke *server*. *File* tersebut kemudian akan didekripsi menggunakan *cipher key* yang dibentuk dari *password* yang sama, dan didekompresi hingga kembali ke bentuk aslinya. Setelah proses selesai, *file* hasil dekripsi dapat diunduh oleh Donald.

2. Desain *Use case* Diagram

   Bagian ini menjelaskan gambaran umum *use case* diagram dari sistem keamanan *file* yang dikembangkan. *Use case* diagram merupakan salah satu jenis *Unified*

Modeling Language (UML) yang digunakan untuk menggambarkan interaksi satu atau lebih aktor dengan sistem informasi yang akan dibagun, serta menjelaskan fungsi-fungsi apa saja yang tersedia dalam sistem tersebut sesuai dengan perannya masing-masing aktor (Nistrina & Sahidah, 2022). Desain *use case* diagram yang digunakan untuk membuat aplikasi website ini dapat dilihat pada Gambar 3.3.



Gambar 3.3 Desain *Use case* Diagram Keamanan *File*

Pada gambar 3.3 terdapat 2 aktor yaitu *User* dan *Admin*. Diagram ini menggambarkan interaksi pengguna dengan sistem dalam bentuk berbagai skenario penggunaan. *User* memiliki akses terhadap sejumlah fitur, seperti melihat beranda, registrasi akun, *login* dan *logout*, serta pengelolaan kata sandi. Selain itu, pengguna dapat mengakses *dashboard*, mengunggah dan mengunduh *file*, melihat kapasitas penyimpanan, serta mengelola daftar *file*. Proses unggah *file* mencakup tiga *use case* yang memiliki relasi *include*, yaitu kompresi *file*,

enkripsi *file*, dan pengaturan kata sandi. Sementara itu, proses unduh *file* juga memuat *use case* tambahan yang memiliki relasi *include*, yakni validasi kata sandi, dekripsi *file*, dan dekompresi *file*. Untuk pengelolaan *file*, pengguna dapat melihat daftar *file* dengan fitur tambahan berupa pencarian, penyaringan berdasarkan tanggal, dan jenis *file* yang masing-masing dihubungkan melalui relasi *extends*. Selain itu, pengguna juga dapat mengatur profil, mengelola sesi aktif, membuat tiket dukungan, dan melihat log aktivitas akun.

Sementara itu, *Admin* memiliki peran administratif dengan akses terhadap *dashboard admin*, pengelolaan data pengguna, *file*, profil dan sesi. *Admin* juga dapat memantau seluruh tiket dukungan yang masuk dan melihat log aktivitas akun.

3. Rencana Komponen Pengembangan Web

   Rencana komponen yang akan digunakan untuk pengembangan web mencakup penggunaan *Node.js* dengan *framework Express* di sisi *backend*, yang berfungsi untuk menangani logika aplikasi, pengelolaan data, dan interaksi dengan *database*. Untuk *frontend*, peneliti akan memanfaatkan *framework Next.js* dengan bahasa *TypeScript* dan *library React* guna membangun antarmuka pengguna *(User Interface)* yang modern, responsif, dan interaktif. *React* akan digunakan untuk menciptakan komponen UI yang dapat digunakan kembali *(reusable)* dan mengelola *state* aplikasi secara efisien. Tampilan visual akan diimplementasikan menggunakan *Tailwind* CSS, sebuah *framework* CSS *utility-first* yang memungkinkan perancangan antarmuka yang cepat, kustom, dan konsisten. Kombinasi *Node.js* dengan *Express* di *backend*, serta *Next.js (React/TypeScript)* dengan *Tailwind* CSS di *frontend,* menawarkan solusi pengembangan yang kuat, skalabel, dan modern, sekaligus memastikan pengalaman peneliti dan pengguna yang optimal.

   Dari sisi infrastruktur, pada penelitian ini akan memanfaatkan layanan dari Google Cloud Platform (GCP) untuk memastikan skalabilitas, efisiensi, dan kemudahan integrasi layanan. Firestore akan digunakan sebagai basis data NoSQL untuk menyimpan data pengguna dan metadata *file* secara *real-time*. Untuk penyimpanan *file* hasil proses enkripsi dan kompresi, sistem akan

menggunakan Google Cloud Storage, yang menyediakan solusi penyimpanan yang andal dan aman. Seluruh layanan *backend* dan *frontend* akan di-*deploy* menggunakan Google Cloud Run, yang mendukung penskalaan otomatis sesuai beban trafik dan tidak dikenakan biaya saat layanan tidak aktif, sehingga sangat sesuai untuk kebutuhan sistem berbasis penelitian. Detail konfigurasi dan spesifikasi layanan Cloud Run, Cloud Storage, dan Firestore yang digunakan dalam penelitian ini ditunjukkan pada Tabel 3.1, Tabel 3.2, dan Tabel 3.3.

Tabel 3.1 Spesifikasi Layanan Cloud Run

Parameter	**Spesifikasi**
Nama Layanan	jagadata-*frontend* dan jagadata-*backend*
Region	asia-*southeast*2 (Jakarta)
CPU	8 vCPUs
Memory	32 GiB
Concurrency	80
Min Instances	0
Max Instances	12
Timeout Request	300

Tabel 3.1 menunjukkan spesifikasi layanan Cloud Run yang digunakan untuk menjalankan *frontend* dan *backend* sistem, masing-masing dengan nama layanan jagadata-*frontend* dan jagadata-*backend*. Layanan ini ditempatkan di *region* asia-*southeast*2 (Jakarta) untuk meminimalkan latensi bagi pengguna di wilayah Indonesia. Konfigurasi CPU sebesar 8 vCPUs dan *memory* 32 GiB dipilih untuk mendukung proses komputasi berat, seperti kompresi dan enkripsi *file* berukuran besar. Nilai *concurrency* 80 berarti setiap *instance* dapat memproses hingga 80 permintaan secara bersamaan, sehingga meningkatkan efisiensi penggunaan sumber daya. *Min Instances* diatur pada 0 agar layanan tidak berjalan saat tidak digunakan sehingga menghemat biaya, sedangkan *Max Instances* dibatasi hingga 12 untuk mengontrol skala otomatis saat trafik tinggi. Parameter *Timeout Request* sebesar 300 detik memberikan waktu maksimum bagi setiap permintaan untuk diproses sebelum Cloud Run membatalkan eksekusi, yang penting untuk

mengakomodasi proses unggah, kompresi, dan enkripsi file dengan durasi relatif panjang.

Tabel 3.2 Spesifikasi Layanan Cloud Storage

Parameter	**Spesifikasi**
Nama *Bucket*	jagadata-*bucket*
Region	asia-*southeast*2 (Jakarta)
Storage Class	*Standard*
Public Access	*Blocked*

Tabel 3.2 menunjukkan spesifikasi layanan Cloud Storage yang digunakan untuk menyimpan *file* hasil proses, yaitu *file* yang telah terenkripsi. Nama *bucket* jagadata-*bucket* digunakan sebagai identitas unik pada Google Cloud Storage untuk memisahkan data dari proyek lain. Penempatan *bucket* di *region* asia-*southeast*2 (Jakarta) bertujuan mengurangi latensi dan mempercepat akses bagi pengguna di Indonesia. *Storage Class* yang digunakan adalah *Standard*, yang menawarkan kinerja tinggi dan akses data tanpa batasan frekuensi, sehingga sesuai untuk *file* terenkripsi yang mungkin perlu diunduh kapan saja. Sementara itu, *Public Access* diatur *Blocked* untuk memastikan tidak ada pihak eksternal yang dapat mengakses *file* secara langsung tanpa otorisasi, sehingga menjaga kerahasiaan dan keamanan data pengguna.

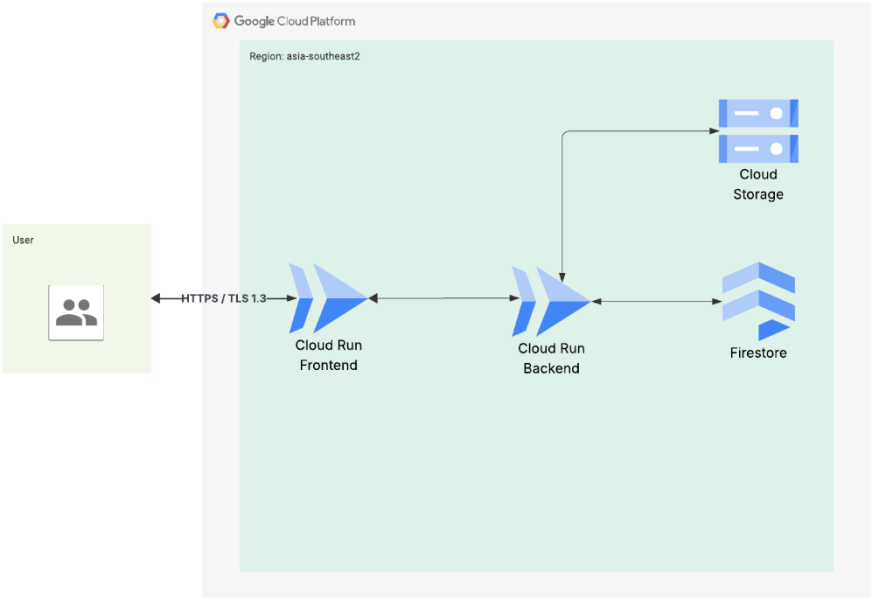
Tabel 3.3 Spesifikasi Layanan Firestore

Parameter	**Spesifikasi**
Nama *Database*	(*default*)
Configuration	Firestore *Native*
Region	asia-*southeast*2 (Jakarta)
Tipe Data	*NoSQL Document-Oriented*

Tabel 3.3 menampilkan spesifikasi layanan Firestore yang digunakan sebagai basis data utama sistem. Nama *database* menggunakan konfigurasi bawaan (*default*) yang disediakan oleh Google Cloud, sehingga terintegrasi langsung dengan proyek tanpa memerlukan pengaturan nama khusus. *Configuration*

dipilih dalam mode Firestore *Native*, yang dioptimalkan untuk aplikasi modern dengan dukungan *real-time update* dan skala otomatis sesuai beban. Layanan ini ditempatkan di region asia-s*outheast*2 (Jakarta) untuk mengurangi latensi akses data bagi pengguna di Indonesia. Tipe data yang digunakan adalah NoSQL *Document-Oriented*, di mana data disimpan dalam bentuk koleksi dan dokumen fleksibel, sehingga memudahkan penyimpanan *metadata file* dan informasi pengguna tanpa batasan skema yang kaku.

Untuk mendukung penjelasan mengenai layanan yang digunakan dalam sistem, peneliti menyusun diagram arsitektur *cloud* guna memberikan gambaran umum mengenai bagaimana layanan berjalan dan saling terhubung di lingkungan Google Cloud Platform yang akan dijelaskan pada gambar 3.4.



Gambar 3.4 Diagram Arsitektur Sistem Berbasis Google Cloud Platform

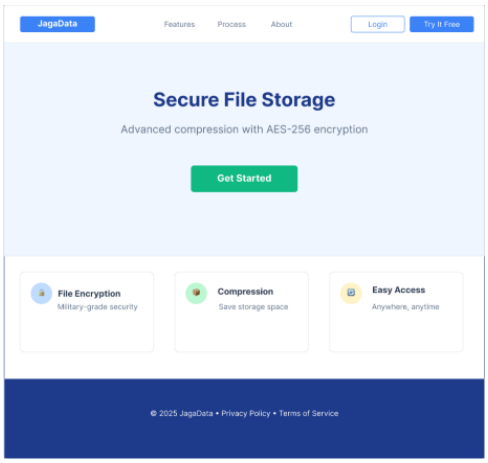
Pada gambar 3.4 menggambarkan arsitektur sistem berbasis Google Cloud Platform yang berjalan pada *region* asia-southeast2 (Jakarta). Sistem terdiri dari tiga layanan utama, yaitu Cloud Run, Cloud Storage, dan Firestore. Interaksi antara pengguna dan sistem dilakukan melalui Cloud Run *(Frontend)* dengan menggunakan koneksi aman berbasis protokol HTTPS/TLS 1.3. Permintaan dari pengguna diteruskan ke Cloud Run *(Backend)* untuk diproses lebih lanjut. *Backend* bertanggung jawab dalam menyimpan *file* terenkripsi ke Cloud Storage

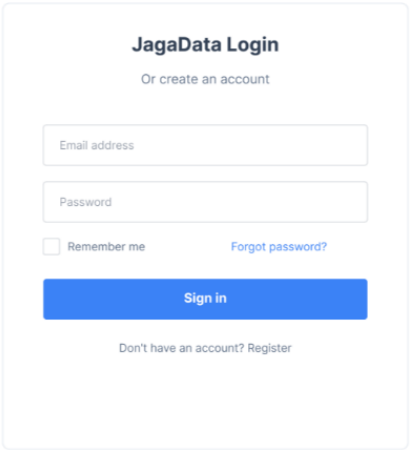
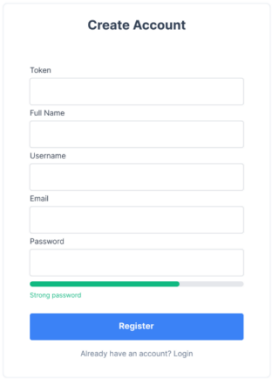
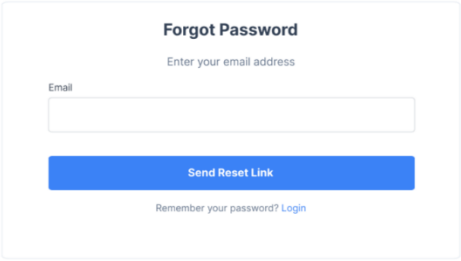
serta mencatat metadata *file* ke dalam Firestore guna mendukung pengelolaan dan pencarian data.

4. Desain Website

   Desain *website* merupakan bagian dari proses perancangan antarmuka yang ditujukan untuk menghadirkan pengalaman penggunaan yang nyaman, mudah dipahami, dan selaras dengan fungsi utama sistem. Dalam proses ini, digunakan alat bantu desain digital berbasis *website*, yaitu Figma, yang memungkinkan peneliti untuk menyusun *mockup* halaman secara visual sebelum memasuki tahap pengembangan aplikasi. Pada tahap ini menyajikan beberapa rancangan halaman utama yang disusun sebelum sistem diimplementasikan. Setiap rancangan halaman disusun berdasarkan hasil analisis peneliti terhadap kebutuhan fungsional sistem, dengan mempertimbangkan kejelasan navigasi, keteraturan tampilan, dan kemudahan dalam mengakses fitur utama. Adapun implementasi akhirnya dapat mengalami penyesuaian pada tahap pengembangan, menyesuaikan dengan kebutuhan fungsional yang ditemukan selama proses penelitian berlangsung. Rancangan desain *website* beserta keterangannya disajikan pada Tabel 3.4.

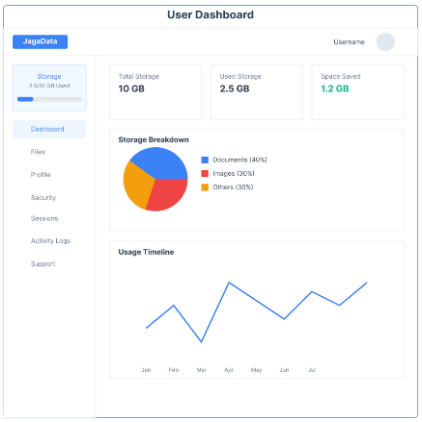
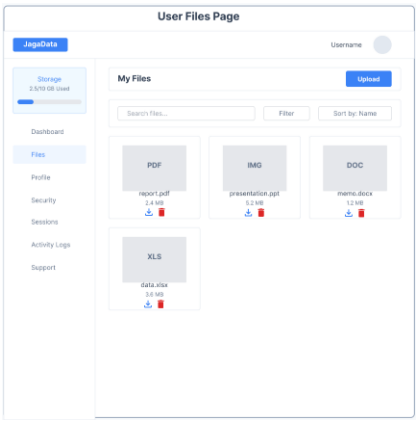
Tabel 3.4 Rancangan *Mockup* Halaman Utama dan *Authentication*

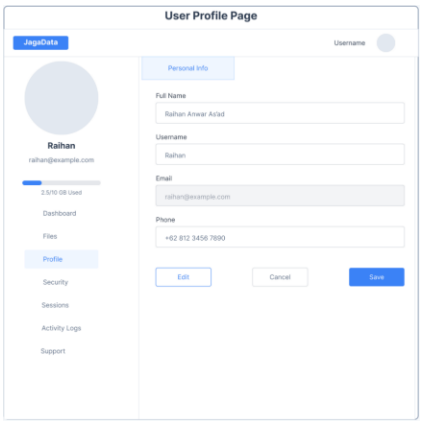
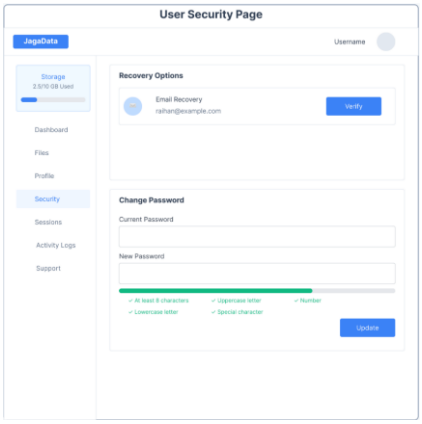
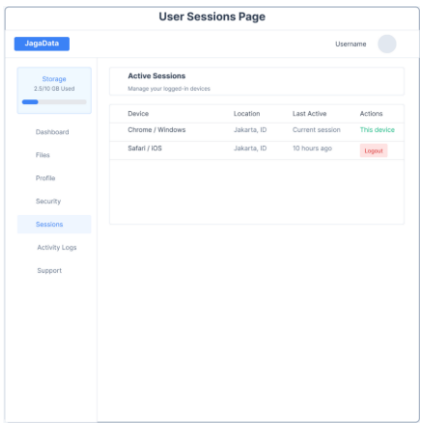
Keterangan	**Rancangan *Mockup***
Mockup halaman *landing page* dirancang sebagai tampilan awal saat pengguna pertama kali mengakses aplikasi web JagaData. Halaman ini menampilkan layanan utama pengamanan *file* dengan kompresi Deflate dan enkripsi AES-256, dilengkapi dengan menu navigasi serta tombol *Get Started* untuk mengarahkan pengguna ke halaman *login.*	

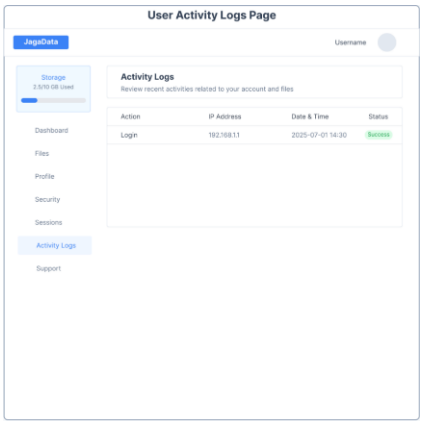
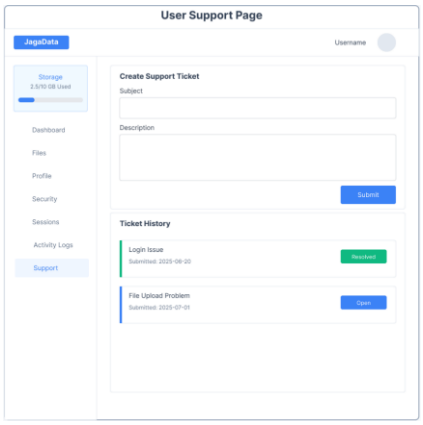
Keterangan	Rancangan *Mockup*
Mockup halaman *login* dirancang sebagai gerbang masuk bagi pengguna maupun *admin* untuk dapat mengakses fitur yang tersedia pada layanan JagaData.	
Mockup halaman *register* dirancang sebagai tempat pendaftaran akun baru bagi pengguna yang ingin mengakses layanan pada aplikasi web JagaData.	
Mockup halaman *forgot password* dirancang untuk membantu pengguna atau *admin* mereset kata sandi akun dengan memasukan alamat email yang telah terdaftar pada sistem.	

Rancangan *mockup* untuk halaman dengan *role user*, meliputi *dashboard*, manajemen *file*, sesi, dan aktivitas, yang disusun untuk menggambarkan interaksi utama pengguna dengan sistem. Desain ini memperhatikan kemudahan navigasi, kejelasan informasi, serta akses terhadap fitur utama. Rancangan detail *mockup* halaman untuk *role user* beserta dengan keterangannya dapat dilihat pada Tabel 3.5.

Tabel 3.5 Rancangan *Mockup* Halaman *User*

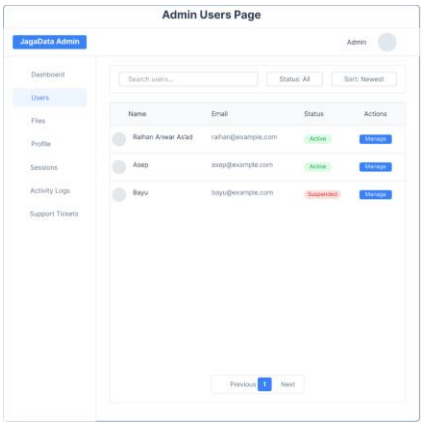
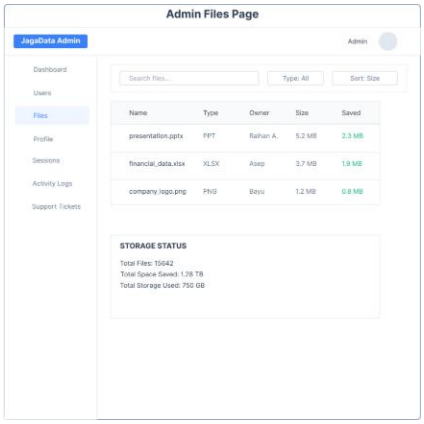
Keterangan	**Rancangan *Mockup***
Mockup halaman *user dashboard* dirancang untuk menampilkan informasi utama terkait penggunaan layanan JagaData. Pada halaman ini akan ditampilkan ringkasan kapasitas penyimpanan total, ruang yang sudah terpakai, serta sisa ruang yang tersedia. Selain itu, disertakan juga diagram persentase jenis *file* yang tersimpan dan grafik penggunaan berdasarkan waktu untuk membantu pengguna memantau aktivitas penyimpanan.	
Mockup halaman *user files* dirancang untuk menampilkan daftar *file* yang telah diunggah oleh pengguna. Pada halaman ini pengguna dapat melihat informasi *file* seperti nama, format, dan ukuran, serta melakukan pencarian, penyaringan, dan pengurutan *file*. Disediakan juga tombol *Upload* untuk menambahkan *file* baru. Selain itu, tersedia tombol *Download* dan *Delete* yang dapat digunakan pengguna untuk mengunduh *file* yang telah terenkripsi maupun *file* yang sudah didekripsi, serta menghapus *file* yang tidak diperlukan.	

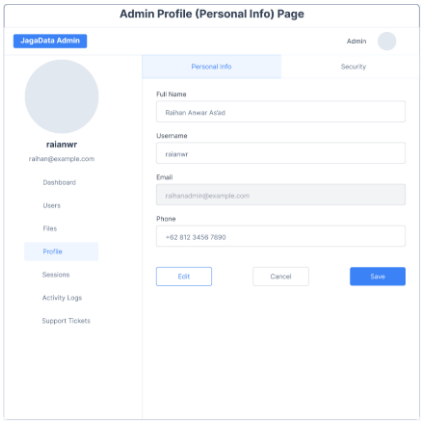
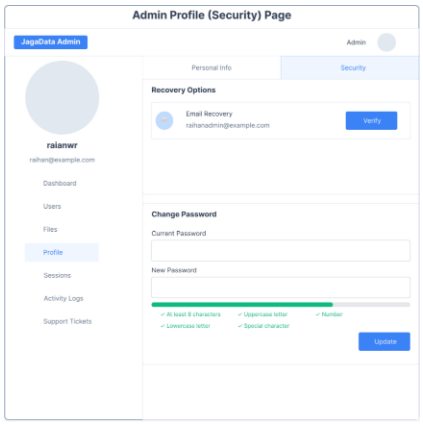
Keterangan	Rancangan *Mockup*
Mockup halaman *user profile* dirancang untuk menampilkan dan mengelola informasi akun pengguna. Pada halaman ini pengguna dapat melihat dan memperbarui data seperti nama lengkap, *username*, email, serta nomor telepon. Disediakan tombol *Edit*, *Cancel*, dan *Save* untuk memudahkan pengguna dalam melakukan perubahan data profil.	
Mockup halaman *user security* dirancang untuk memberikan opsi pengaturan keamanan akun. Halaman ini menyediakan fitur *Recovery Options* untuk memverifikasi email pemulihan serta fitur *Change Password* yang memungkinkan pengguna mengganti kata sandi lama dengan yang baru. Sistem juga menampilkan indikator kekuatan *password* berdasarkan kriteria yang telah ditentukan.	
Mockup halaman *user sessions* dirancang untuk menampilkan daftar perangkat yang sedang atau pernah melakukan *login* menggunakan akun tersebut. Informasi yang ditampilkan meliputi jenis perangkat, lokasi akses, waktu terakhir aktif, serta status sesi. Pengguna juga	

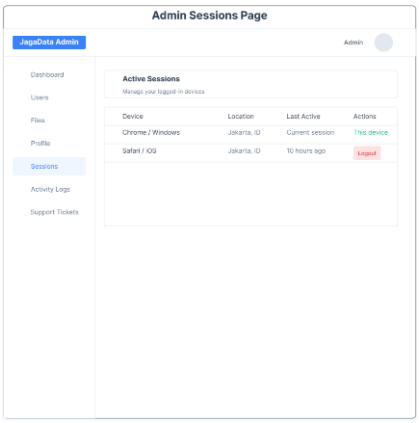
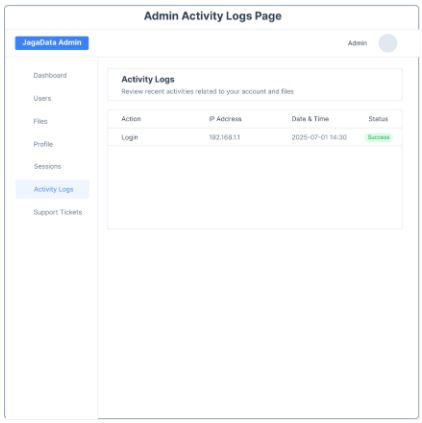
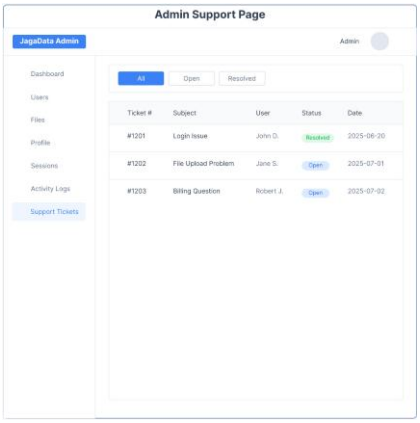
Keterangan	Rancangan *Mockup*
dapat mengakhiri sesi tertentu melalui tombol *Logout* pada bagian *Actions*.	
Mockup halaman *user activity logs* dirancang untuk menampilkan catatan aktivitas yang terjadi pada akun pengguna. Informasi yang disajikan meliputi jenis aktivitas, *Internet Protocol* (IP) *Address*, tanggal dan waktu, serta status yang menunjukkan berhasil atau tidaknya sistem memproses aktivitas tersebut.	
Mockup halaman *user support* dirancang untuk memfasilitasi pengguna dalam melaporkan masalah atau kendala yang dialami saat menggunakan layanan. Halaman ini menyediakan *form* untuk membuat tiket dukungan serta menampilkan riwayat tiket yang pernah diajukan pengguna, lengkap dengan status apakah masih aktif (*Open*) atau sudah ditangani (*Resolved*).	

Rancangan *mockup* untuk halaman dengan *role admin* meliputi *dashboard*, manajemen pengguna, pengelolaan *file*, dan pemantauan aktivitas sistem, yang disusun untuk menggambarkan fungsi utama *admin* dalam mengelola aplikasi. Desain ini disusun agar *admin* dapat memantau terkait performa sistem, sekaligus mengelola serta membalas tiket dukungan yang diajukan pengguna. Rancangan detail *mockup h*alaman untuk *role admin* beserta keterangannya dapat dilihat pada Tabel 3.6.

Tabel 3.6 Rancangan *Mockup* Halaman *Admin*

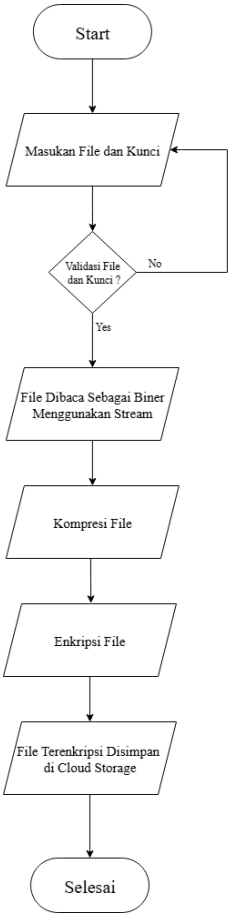
Keterangan	**Rancangan *Mockup***
Mockup halaman *admin dashboard* dirancang untuk menampilkan ringkasan informasi penting terkait performa sistem dan aktivitas pengguna, sehingga *admin* dapat memantau penggunaan layanan secara menyeluruh.	
Mockup halaman *admin users* dirancang untuk menampilkan daftar akun pengguna yang terdaftar pada sistem. Informasi yang ditampilkan meliputi nama, email, dan status akun, seperti *Active* atau *Suspended*. *Admin* juga dapat melakukan pengelolaan akun melalui tombol *Manage* yang tersedia pada setiap baris data.	
Mockup halaman *admin files* dirancang untuk menampilkan daftar *file* yang diunggah oleh pengguna ke dalam sistem. Informasi yang ditampilkan meliputi nama *file*, jenis *file*, pemilik, ukuran, serta ruang yang berhasil dihemat melalui kompresi dan enkripsi. Selain itu, tersedia bagian *Storage Status* yang memberikan ringkasan jumlah total *file*, kapasitas ruang yang	

Keterangan	Rancangan *Mockup*
dihemat, dan total penyimpanan yang digunakan.	
Mockup halaman *admin profile* (*personal* info) dirancang untuk menampilkan dan mengelola informasi akun. Pada halaman ini admin dapat melihat serta memperbarui data seperti nama lengkap, *username*, email, dan nomor telepon. Disediakan tombol *Edit*, *Cancel*, dan *Save* untuk memudahkan *admin* dalam melakukan perubahan data profil.	
Mockup halaman *admin profile* (*security*) dirancang untuk memberikan opsi pengaturan keamanan pada akun. Halaman ini menyediakan fitur *Recovery Options* untuk memverifikasi email pemulihan, serta fitur *Change Password* untuk mengganti kata sandi. Sistem juga menampilkan indikator kekuatan *password* berdasarkan kriteria yang telah ditentukan.	

Keterangan	Rancangan *Mockup*
Mockup halaman *admin sessions* dirancang untuk menampilkan daftar perangkat yang sedang atau pernah *login* menggunakan akun tersebut. Informasi yang tersedia meliputi jenis perangkat, lokasi akses, waktu terakhir aktif, serta status sesi. *Admin* juga dapat mengakhiri sesi tertentu melalui tombol *Logout* pada bagian *Actions*.	
Mockup halaman *admin activity logs* dirancang untuk menampilkan catatan aktivitas yang terjadi pada akun maupun *file* dalam sistem. Informasi yang ditampilkan meliputi jenis aktivitas, alamat IP, tanggal dan waktu, serta status yang menunjukkan berhasil atau tidaknya sistem memproses aktivitas tersebut.	
Mockup halaman *admin support* dirancang untuk membantu *admin* dalam mengelola tiket dukungan yang diajukan pengguna. Halaman ini menampilkan daftar tiket beserta nomor tiket, subjek permasalahan, nama pengguna, status, serta tanggal pengajuan.	

5. *Flowchart* Sistem

Flowchart atau bagan alur adalah diagram yang menggambarkan alur atau langkah-langkah dalam suatu proses secara sederhana menggunakan simbol-simbol tertentu untuk menunjukkan urutan setiap langkah. Dalam pemrograman dan sistem, *flowchart* sering digunakan untuk merancang, menganalisis, serta memahami tahapan yang diperlukan dalam menyelesaikan suatu tugas atau masalah. Adapun *flowchart* proses kompresi dan enkripsi pada penelitian ini ditampilkan pada Gambar 3.5.



Gambar 3.5 *Flowchart* Kompresi dan Enkripsi

Pada Gambar 3.5, proses dimulai dengan pengguna memasukkan *file* yang akan diproses beserta kunci enkripsi sebagai *input* utama. Setelah itu, sistem akan melakukan validasi terhadap *file* dan kunci yang diberikan untuk memastikan bahwa keduanya memenuhi persyaratan yang dibutuhkan. Jika validasi gagal, pengguna akan diarahkan kembali ke langkah awal untuk memperbaiki *input*. Namun, jika validasi berhasil, *file* akan diproses melalui tahap kompresi, di mana

ukuran *file* dikurangi untuk meningkatkan efisiensi penyimpanan dan pengiriman. Setelah proses kompresi selesai, langkah berikutnya adalah enkripsi *file* menggunakan kunci yang telah dimasukkan sebelumnya untuk menjaga kerahasiaan dan keamanan data. Proses ini menghasilkan *file* terenkripsi yang nantinya disimpan di cloud storage dan dapat dikembalikan ke bentuk *file* aslinya apabila menggunakan kunci yang sama dengan proses sebelumnya. Adapun *flowchart* proses dekripsi dan dekompresi pada penelitian ini, dapat dilihat pada Gambar 3.6.

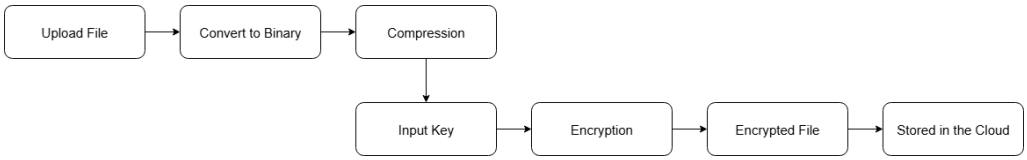
Gambar 3.6 *Flowchart* Dekripsi dan Dekompresi

Pada Gambar 3.6, menjelaskan alur proses dekripsi dan dekompresi *file* yang terkunci. Proses diawali dengan pengguna memilih *file* yang akan didekripsi beserta kunci yang digunakan untuk proses dekripsi. Sistem kemudian memvalidasi *file* dan kunci yang dimasukkan untuk memastikan keduanya sesuai. Jika validasi gagal, pengguna diminta untuk mengulangi langkah *input*.

Namun, jika validasi berhasil, proses dilanjutkan ke tahap dekripsi, di mana *file* terenkripsi diubah kembali ke format yang dapat dikenali. Setelah *file* berhasil didekripsi, langkah berikutnya adalah proses dekompresi untuk mengembalikan *file* yang telah dikompresi sebelumnya ke bentuk aslinya. Hasil akhirnya adalah *file* asli yang sepenuhnya dipulihkan dan dapat diakses kembali.

6. Diagram Blok Sistem

Diagram blok sistem merupakan representasi grafis yang digunakan untuk menggambarkan alur kerja dan hubungan antar komponen utama dalam sistem secara umum dan terstruktur. Diagram ini berfungsi sebagai alat bantu visual yang menyederhanakan pemahaman terhadap proses internal sistem, mulai dari *input* yang diterima, proses yang dijalankan, hingga *output* yang dihasilkan. Dalam konteks pengembangan sistem keamanan *file*, diagram blok sistem menjelaskan bagaimana data diproses melalui tahapan-tahapan penting seperti kompresi, enkripsi, penyimpanan, hingga proses dekripsi dan dekompresi.

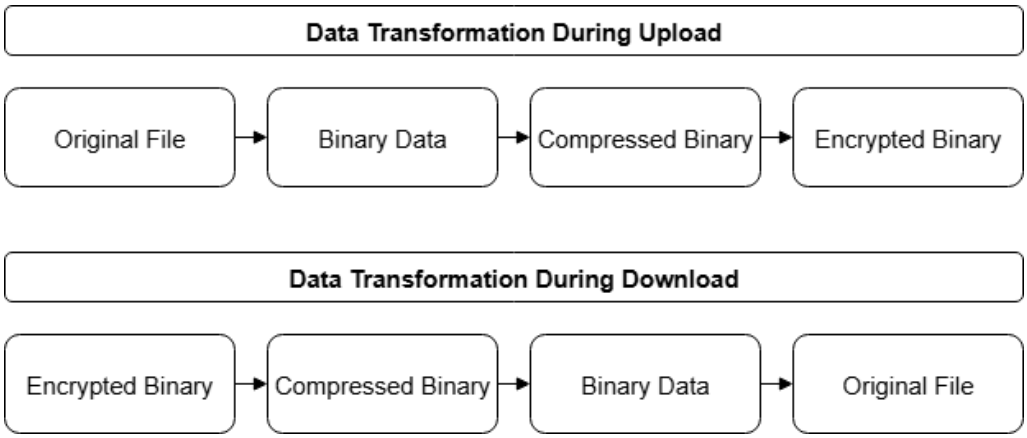


Gambar 3.7 Diagram Blok Kompresi-Enkripsi Keamanan *File*

Pada gambar 3.7 menampilkan alur proses utama dalam sistem kompresi dan enkripsi *file* yang dikembangkan. Proses diawali dengan sistem menerima *file* yang telah diunggah oleh pengguna. *File* tersebut kemudian dikonversi menjadi data biner agar dapat diproses oleh sistem. Tahap berikutnya adalah kompresi, di mana data biner dikompresi menggunakan algoritma Deflate untuk mengurangi ukuran *file*. Setelah proses kompresi selesai, data hasil kompresi kemudian dienkripsi menggunakan algoritma AES-256 dengan kunci yang telah diberikan oleh pengguna melalui antarmuka aplikasi web, sehingga menghasilkan *file* terenkripsi. *File* tersebut selanjutnya disimpan ke dalam layanan penyimpanan Google Cloud storage sebagai bentuk akhir dari proses, yang memastikan *file* yang telah diamankan tersedia secara *online*.

Gambar 3.8 Diagram Blok Dekripsi-Dekompresi Keamanan *File*

Pada gambar 3.8 menggambarkan urutan proses internal sistem dalam mengembalikan *file* terenkripsi ke bentuk aslinya. Proses dimulai ketika pengguna memasukkan kunci enkripsi yang diperlukan untuk membuka akses terhadap *file*. Setelah itu, sistem mengunduh *file* yang telah dienkripsi dari penyimpanan *cloud*. *File* tersebut kemudian diproses melalui tahap dekripsi menggunakan algoritma AES-256 dan kunci yang telah diberikan, sehingga menghasilkan data biner terdekripsi. Selanjutnya, data tersebut masuk ke tahap dekompresi untuk mengembalikan ukuran dan struktur *file* seperti semula. Hasil akhir dari keseluruhan proses ini adalah *file* asli yang dapat diunduh dan diakses kembali oleh pengguna.



Gambar 3.9 Diagram Blok Transformasi Data

Pada gambar 3.9 menunjukkan perubahan bentuk data secara sistematis selama proses unggah dan unduh *file* dalam sistem. Pada tahap unggah (*Data Transformation During Upload*), *file* asli terlebih dahulu diubah menjadi data biner agar dapat diproses oleh sistem. Data biner tersebut kemudian dikompresi menggunakan algoritma Deflate untuk mengurangi ukuran, menghasilkan *Compressed Binary*. Selanjutnya, data yang telah dikompresi dienkripsi menggunakan algoritma AES-256 sehingga terbentuk *Encrypted Binary* sebagai hasil akhir proses unggah.

Pada sisi sebaliknya, selama proses unduh (*Data Transformation During Download*), sistem melakukan rekonstruksi *file* dengan membalik tahapan sebelumnya. Proses dimulai dari *Encrypted Binary* yang didekripsi untuk memper*oleh* *Compressed Binary*, lalu didekompresi menjadi *Binary* Data, dan akhirnya dikonversi kembali ke format semula sebagai *Original File*.

7. Rencana Pengembangan Sistem

   Rencana pengembangan sistem web berbasis *cloud* akan menggunakan metode *Agile,* yang berfokus pada iterasi cepat, dan adaptasi terhadap perubahan kebutuhan selama proses pengembangan aplikasi web. *Agile* membagi prosesnya menjadi beberapa iterasi singkat atau disebut *sprint*, di mana setiap *sprint* mencakup perencanaan *(planning),* desain *(design*), pengembangan *(development),* pengujian *(testing),* penerapan *(deploy)* dan evaluasi *(review).* Pendekatan ini memungkinkan pengembangan sistem dilakukan secara bertahap dan dapat melakukan penyesuaian sesuai kebutuhan, sekaligus meminimalkan risiko kesalahan selama proses pengembangan. Seperti yang dijelaskan oleh Ariesta dkk. (2021), metode *Agile* mengutamakan pengembangan *incremental* yang cepat dengan melibatkan pengguna secara langsung, serta memfokuskan pada kualitas kode dan pengurangan *overhead* proses.

3.5 Desain Uji Coba Sistem (*Test the artifact*)

Pada tahap desain uji coba, sistem akan diuji menggunakan metode *black-box testing*. Metode ini berfokus pada pengujian fungsional sistem dengan menguji spesifikasi perangkat lunak tanpa melibatkan kode sumber, memastikan apakah fungsi, *input*, dan *output* sesuai dengan yang diharapkan (Pratama dkk., 2023). Dalam pengujian ini, setiap fitur pada sistem diverifikasi dengan memberikan berbagai *input* untuk memeriksa apakah *output* yang dihasilkan sesuai dengan spesifikasi yang telah ditentukan. Pendekatan ini memastikan bahwa pengujian dapat dilakukan dari perspektif pengguna, sehingga membantu mengidentifikasi kesalahan atau kekurangan yang mungkin terlewat oleh peneliti. Adapun ringkasan tahapan uji coba yang diterapkan menggunakan metode *black-box testing* dapat

dilihat pada Tabel 3.7, Tabel 3.8 dan Tabel 3.9, sedangkan tahapan uji coba secara lebih lengkapnya dapat dilihat pada Lampiran 6, Lampiran 7 dan Lampiran 8.

Tabel 3.7 Uji Coba Sistem Halaman Utama dan *Authentication*

No.	**Fitur yang Diuji**	**Jenis Fitur**	**Kebutuhan**	**Hasil yang Diharapkan**
1	Tombol "*Try It Free*" dan *"Get Started*"	*Landing Page*	Pengguna dapat mengklik tombol untuk ke halaman *login*	Sistem menavigasi pengguna ke halaman *login*
2	*Request Token Form*	*Register*	Pengguna dapat memasukkan email untuk memperoleh kode verifikasi	Sistem mengirimkan kode verifikasi ke email dan pengguna dialihkan ke halaman registrasi
3	*Registration Validation Form*	*Register*	Pengguna dapat mengisi seluruh kolom pada *form* registrasi (*verification code*, *username*, *password*, *confirm password*, *birthdate*, *gender*)	Sistem akan memunculkan pesan e*rror* apabila salah satu kolom kosong atau format salah
4	Kekuatan *Password*	*Register*	Pengguna dapat memasukkan kata sandi ke dalam kolom	Sistem menampilkan indikator kekuatan kata sandi di

No.	Fitur yang Diuji	Jenis Fitur	Kebutuhan	Hasil yang Diharapkan
			password dengan berbagai tingkat kompleksitas	bawah kolom *password*
5	Kecocokan *Password*	*Register*	Pengguna dapat memasukkan nilai yang sama atau berbeda pada kolom *password* dan *confirm password*	Sistem menampilkan pesan *error* jika nilai pada kedua kolom tidak cocok
…	…	…	…	…
18	Tombol "Reset *Password"*	Reset *Password*	Pengguna dapat menekan tombol "Reset *Password*" setelah mengisi *form* dengan *password* baru yang valid	Sistem menyimpan *password* baru dan mengarahkan pengguna ke halaman *login*

Pada Tabel 3.7 menyajikan ringkasan uji coba sistem pada fitur halaman utama dan *authentication* dengan total 18 pengujian yang lebih detailnya dapat dilihat pada Lampiran 6. Fokus pengujian mencakup validasi fungsionalitas tombol navigasi utama pada halaman beranda, proses registrasi pengguna baru, autentikasi *login*, serta pemulihan akun melalui fitur lupa kata sandi. Pengujian pada tahap registrasi meliputi verifikasi email, validasi isian *form*, penilaian kekuatan dan kecocokan kata sandi, serta pengiriman notifikasi email. Sementara itu, fitur *login* diuji dari

aspek validitas kredensial, pengelolaan sesi melalui opsi "*remember me*", pembatasan percobaan *login*, hingga otorisasi akses berdasarkan peran pengguna. Terakhir, prosedur pemulihan akun memastikan keandalan sistem dalam mengirim tautan reset, memverifikasi *token*, dan menyimpan perubahan kata sandi.

Tabel 3.8 Uji Coba Sistem dengan *Role User*

No.	Fitur yang Diuji	Jenis Fitur	Kebutuhan	Hasil yang Diharapkan
1	Statistik dan Indikator *Storage*	*User Dashboard*	Pengguna dapat membuka halaman *dashboard* untuk melihat statistik dan indikator penggunaan *storage*	Sistem menampilkan informasi *storage* (total *storage*, *used storage*, *available storage*, dan total *space saved*) serta indikator warna progres yang berubah merah jika penggunaan ≥85%
2	*Usage Timeline*	*User Dashboard*	Pengguna dapat membuka halaman *dashboard* untuk melihat visualisasi grafik *storage usage* bulanan	Sistem menampilkan grafik *timeline usage storage* per bulan dengan akurat
3	*Storage Breakdown*	*User Dashboard*	Pengguna dapat melihat grafik	Sistem menampilkan

No.	Fitur yang Diuji	Jenis Fitur	Kebutuhan	Hasil yang Diharapkan
			distribusi *file* berdasarkan jenis *file*	grafik distribusi *file* secara akurat berdasarkan jenis *file* yang tersimpan di basis data
4	Tampilan Daftar *File*	*File List*	Pengguna dapat membuka halaman *files* untuk melihat daftar *file* yang telah diunggah oleh pengguna	Sistem menampilkan daftar *file* dengan informasi lengkap seperti *name*, *size*, *type*, *uploaded*, dan *actions*
5	*Sorting, Search and Filter*	*File List*	Pengguna dapat memilih kriteria tertentu untuk mengurutkan atau memfilter daftar *file*, serta melakukan pencarian berdasarkan nama *file*	Sistem menampilkan *file* yang telah diurutkan atau difilter sesuai kriteria yang dipilih pengguna atau yang dicari berdasarkan nama *file*
…	…	…	…	…
37	Tombol "*Logout*"	*Session*	Pengguna dapat mengklik tombol "*Logout*" untuk keluar dari sesi *login*	Sistem mengakhiri sesi *login* dan mengarahkan pengguna kembali ke halaman *login*

Pada Tabel 3.8 menyajikan ringkasan uji coba sistem dari sisi pengguna (*role user*) dengan total 37 pengujian yang mencakup fitur *dashboard*, pengelolaan *file*, keamanan, profil, sesi, hingga dukungan pengguna, yang lebih detailnya dapat dilihat pada Lampiran 7. Pengujian ini meliputi tampilan statistik penyimpanan, aktivitas terakhir, dan distribusi tipe *file*, serta kemampuan sistem dalam menyaring, mengurutkan, dan melakukan *pagination* terhadap daftar *file* yang dimiliki oleh *user*. Proses unggah dan unduh diuji secara menyeluruh, termasuk validasi ukuran, progres visual, kompresi, enkripsi, dan dekripsi *file*. Selain itu, sistem diuji untuk fitur penghapusan *file*, pengelolaan profil, serta verifikasi email. Fitur keamanan diuji melalui pengelolaan sesi aktif dan penggantian *password*. Aspek dukungan pengguna diuji melalui sistem tiket dan notifikasi email, serta proses *logout* untuk mengakhiri sesi pengguna.

Tabel 3.9 Uji Coba Sistem dengan *Role Admin*

No.	Fitur yang Diuji	Jenis Fitur	Kebutuhan	Hasil yang Diharapkan
1	Statistik Utama	*Admin Dashboard*	*Admin* dapat membuka halaman *dashboard* untuk melihat informasi statistik sistem secara keseluruhan	Sistem menampilkan statistik *total users*, *total files*, *total space saved*, dan *storage used*
2	*File Type Distribution*	*Admin Dashboard*	*Admin* dapat melihat grafik distribusi *file* yang menampilkan seluruh *file* yang	Sistem menampilkan grafik distribusi *file* yang diklasifikasikan berdasarkan tipe

No.	Fitur yang Diuji	Jenis Fitur	Kebutuhan	Hasil yang Diharapkan
			diklasifikasikan berdasarkan tipe ekstensi	ekstensi secara akurat
3	*Platform Statistics*	*Admin Dashboard*	*Admin* dapat melihat informasi statistik platform yang mencakup data *active users*, *suspended users*, *average file size*, dan *storage breakdown*	Sistem menampilkan statistik platform secara akurat dan sesuai dengan data yang tersimpan di basis data
4	*Compression & Processing Performance*	*Admin Dashboard*	*Admin* dapat melihat informasi performa sistem yang mencakup *average compression ratio* dan *total space saved* berdasarkan tipe ekstensi *file*	Sistem menampilkan data performa secara akurat berdasarkan metrik hasil perhitungan yang telah dilakukan oleh sistem
5	Tampilan Daftar *User*	*Users Management*	*Admin* dapat membuka	Sistem menampilkan

No.	Fitur yang Diuji	Jenis Fitur	Kebutuhan	Hasil yang Diharapkan
			halaman *user management* untuk melihat seluruh akun yang telah terdaftar dalam sistem	daftar *user* lengkap dengan informasi akun dan status akun
…	…	…	…	…
37	Tombol "*Logout*"	*Session*	*Admin* dapat mengklik tombol "*Logout*" untuk keluar dari sesi *login*	Sistem mengakhiri sesi *login* dan mengarahkan *admin* kembali ke halaman *login*

Pada Tabel 3.9 menyajikan ringkasan uji coba sistem dari sisi *admin* (*role admin*) dengan total 37 pengujian yang mencakup pengelolaan data pengguna, *file*, tiket dukungan, serta pengaturan akun dan sesi, yang lebih detailnya dapat dilihat pada Lampiran 8. Fitur *dashboard* diuji untuk memastikan ketepatan informasi statistik *global*, performa kompresi, dan distribusi tipe *file*. Pengelolaan pengguna meliputi tampilan daftar *user*, pencarian berdasarkan status, pengeditan data, *suspend* dan reaktivasi akun, hingga penghapusan permanen. Pengujian terhadap manajemen *file* mencakup pencarian, penampilan detail, dan validasi integritas informasi. Fitur dukungan diuji melalui pengelolaan tiket, pengiriman balasan, lampiran, dan perubahan status tiket. Pengaturan akun *admin* meliputi profil, *password*, email, serta verifikasi perubahan kredensial. Selain itu, sistem diuji untuk fungsi *log* aktivitas, pencarian riwayat, dan pengelolaan sesi aktif.

3.6 Desain Evaluasi Hasil Uji (*Evaluate testing results*)

Evaluasi hasil uji coba dilakukan untuk menilai sejauh mana aplikasi web berbasis *cloud* yang dikembangkan mampu memenuhi tujuannya dalam mengamankan *file* serta meningkatkan efisiensi penyimpanan. Penilaian ini didasarkan pada proses inti yang diterapkan dalam sistem, meliputi kompresi, enkripsi, dekripsi, dan dekompresi. Evaluasi dilakukan untuk memastikan bahwa seluruh proses berjalan dengan baik serta memberikan dampak nyata terhadap keamanan data dan penghematan ruang penyimpanan. Pendekatan evaluasi dilakukan secara bertahap dengan mengamati efektivitas setiap algoritma yang digunakan, baik secara terpisah maupun ketika diterapkan secara berurutan dalam sistem.

Pengujian pertama difokuskan pada analisis kinerja algoritma Deflate. Algoritma ini diuji dengan mengompresi sejumlah *file* dalam empat format berbeda, yaitu .pdf, .csv, .docx, dan .pptx, di mana setiap *file* dibaca dalam bentuk biner sebelum diproses. Tujuan dari pengujian ini adalah untuk mengevaluasi efektivitas kinerja algoritma Deflate dalam mengompresi *file*, khususnya dalam mengurangi ukuran *file* secara signifikan. Efisiensi kompresi ini menjadi salah satu indikator penting dalam optimalisasi ruang penyimpanan *cloud*. *File* hasil kompresi kemudian digunakan dalam pengujian tahap berikutnya.

Tahap selanjutnya, *file* hasil kompresi dienkripsi menggunakan algoritma AES-256 untuk memastikan bahwa isi *file* tetap terlindungi, meskipun terjadi kebocoran atau akses oleh pihak yang tidak berwenang. Proses ini kemudian dilanjutkan dengan dekripsi untuk memverifikasi bahwa *file* dapat dikembalikan ke bentuk semula secara utuh, tanpa kehilangan atau perubahan informasi. Evaluasi kinerja AES-256 difokuskan pada dua aspek utama yaitu kemampuan menjaga kerahasiaan data melalui bentuk *ciphertext* yang tidak dapat dibaca tanpa kunci yang sah, serta keandalannya dalam mempertahankan integritas data selama proses enkripsi dan dekripsi.

Setelah proses kompresi dan enkripsi dilakukan, evaluasi dilanjutkan dengan membandingkan ukuran *file* sebelum dan sesudah melalui kedua proses tersebut.

Pengukuran efisiensi dilakukan menggunakan rumus *space saving*, yaitu rasio penghematan ruang penyimpanan yang dihasilkan oleh sistem.

Adapun rumus dasar dari *Space saving* adalah sebagai berikut:

$$Space\ saving = 1 - \frac{Compressed\ File\ Size}{Original\ File\ Size} \quad (2)$$

Pendekatan ini juga diterapkan dalam penelitian yang dilakukan oleh Aruna dkk. (2023), yang menggunakan rumus *space saving* untuk mengevaluasi efektivitas berbagai algoritma kompresi teks. Dalam studi tersebut, perbandingan antara ukuran *file* sebelum dan sesudah kompresi digunakan sebagai dasar untuk menilai sejauh mana algoritma dapat mengurangi redundansi data secara efisien tanpa mengorbankan isi informasi. Mengacu pada pendekatan tersebut, evaluasi dalam penelitian ini bertujuan untuk mengukur sejauh mana kombinasi algoritma Deflate dan AES-256 berkontribusi dalam efisiensi ukuran *file*, tanpa mengorbankan aspek keamanannya.

Langkah evaluasi selanjutnya adalah pengujian keamanan pada saat proses pengiriman *file* ke *server*. Pengujian ini akan menggunakan perangkat lunak Wireshark untuk memastikan bahwa seluruh data yang ditransmisikan antara pengguna dan *server* aplikasi berlangsung melalui saluran terenkripsi *Hypertext Transfer Protocol Secure* (HTTPS) berbasis protokol TLS 1.3. Rancangan pengujian ini mencakup tiga tahapan utama. Pertama, dilakukan verifikasi terhadap resolusi DNS dan alamat IP domain layanan menggunakan perintah *nslookup* pada domain yang telah di*deploy* melalui platform Cloud Run. Kedua, dilakukan perekaman lalu lintas jaringan *(packet capture)* menggunakan Wireshark selama proses interaksi penting, seperti *login*, registrasi, unggah *file*, dan unduh *file*. Ketiga, diterapkan filter tcp.port == 443 and (ipv6.addr == …) untuk menampilkan secara spesifik lalu lintas HTTPS antara klien dan *server*, sehingga dapat dipastikan bahwa data yang dikirim telah terenkripsi dan tidak bercampur dengan lalu lintas lainnya.

Langkah evaluasi selanjutnya adalah menganalisis *file* yang telah dikompresi dan enkripsi menggunakan analisis korelasi Pearson terhadap representasi biner *file* sebelum (*plaintext*) dan sesudah proses kompresi dan enkripsi (*ciphertext*). Evaluasi ini dilakukan untuk mengukur sejauh mana struktur data berubah akibat proses transformasi. Nilai korelasi yang mendekati nol menunjukkan bahwa hasil enkripsi tidak memiliki keterkaitan linier dengan data asli, yang menjadi indikator kuat bahwa algoritma berhasil menyamarkan pola data.

Terakhir, untuk memperoleh gambaran umum dari hasil pengujian, dilakukan perhitungan nilai rata-rata dari setiap parameter yang diuji, meliputi ukuran *file* sebelum proses, ukuran *file* sesudah proses, persentase *space saving*, serta nilai Korelasi Pearson. Rata-rata tersebut dihitung dengan cara menjumlahkan seluruh hasil pengujian pada masing-masing parameter, kemudian membaginya dengan jumlah data uji. Pendekatan ini digunakan agar hasil pengujian lebih mudah dipahami, sekaligus memberikan representasi menyeluruh terhadap performa sistem pada tiap format *file* yang diuji.

Dengan dilakukannya serangkaian evaluasi, baik dari sisi efisiensi ukuran *file*, keamanan data selama transmisi, hingga kekuatan kriptografi terhadap struktur data, penelitian ini diharapkan dapat memberikan gambaran menyeluruh mengenai kinerja sistem yang dikembangkan. Setiap metode evaluasi yang digunakan, mulai dari pengukuran *space saving*, pengamatan lalu lintas jaringan, hingga analisis korelasi, berperan sebagai indikator keberhasilan implementasi kombinasi algoritma Deflate dan AES-256 dalam konteks pengamanan *file*. Temuan dari evaluasi ini menjadi dasar penting dalam menilai efektivitas pendekatan yang digunakan pada penelitian ini, serta memberikan landasan bagi pengembangan sistem serupa di masa mendatang.

3.7 Pemaparan Hasil Uji (*Communicate testing results*)

Hasil uji coba sistem disampaikan secara rinci dalam laporan skripsi yang mencakup analisis data, temuan utama, dan evaluasi kinerja aplikasi web berbasis *cloud* yang dikembangkan. Laporan ini akan memaparkan kemampuan aplikasi web dalam mengurangi ukuran *file* melalui proses kompresi serta menjaga

kerahasiaan data melalui proses enkripsi. Data hasil uji coba akan disajikan dalam bentuk tabel untuk memberikan ilustrasi yang jelas dan mudah dipahami. Seluruh hasil yang diperoleh dikomunikasikan dan didiskusikan bersama dosen pembimbing melalui proses bimbingan skripsi secara bertahap. Penyampaian hasil ini bertujuan untuk mendokumentasikan pencapaian penelitian secara sistematis dan memberikan dasar evaluasi untuk pengembangan lebih lanjut.