BAB III
METODOLOGI PENELITIAN

3.1. Metode Penelitian

Penelitian ini termasuk dalam kategori penelitian rekayasa perangkat lunak dengan pendekatan *Research and Development*. Pendekatan rekayasa perangkat lunak umum digunakan dalam penelitian yang berfokus pada pembangunan sistem sekaligus pengujian efektivitas implementasi keamanan data (Hillalia & Asri, 2026). Penelitian R&D dipilih karena tujuan utama penelitian ini adalah merancang, membangun, dan menguji sebuah sistem perangkat lunak yang fungsional sebagai solusi atas permasalahan yang telah diidentifikasi, bukan sekadar mendeskripsikan atau menganalisis fenomena yang sudah ada.

Pendekatan yang digunakan bersifat eksperimental. Pendekatan eksperimental umum digunakan pada penelitian sistem berbasis *cloud computing*, yaitu melalui implementasi sistem dan pengujian langsung terhadap komponen infrastruktur yang digunakan (G. Wijaya, 2022). Sistem dibangun secara nyata kemudian diuji dalam lingkungan yang terkontrol untuk mengukur fungsionalitas, performa, dan keamanannya secara kuantitatif. Pengujian dilakukan terhadap beberapa aspek kritis sistem, meliputi kecepatan transfer berkas, waktu enkripsi dan dekripsi AES-256, akurasi verifikasi integritas berkas menggunakan SHA-256, serta keberhasilan mekanisme *file chunking* pada berkas berukuran di atas 1,8 GB.

Dari sisi metodologi pengembangan sistem, penelitian ini menggunakan metode Prototyping. Metode ini dipilih karena karakteristik sistem yang dibangun melibatkan integrasi beberapa komponen teknis sekaligus *Web UI*, koneksi Pyrogram MTProto ke *server* Telegram, mekanisme kriptografi AES-256, dan logika *file chunking* yang memerlukan validasi fungsionalitas secara bertahap melalui siklus pembangunan *prototype*, evaluasi, dan penyempurnaan secara berulang hingga sistem mencapai kualitas yang diharapkan.

3.2. Lokasi dan Waktu Penelitian

Penelitian ini dilaksanakan secara mandiri di lingkungan pengembangan lokal milik peneliti, yang berfungsi sebagai laboratorium pribadi berbasis perangkat

komputer. Seluruh tahapan pengembangan sistem, mulai dari konfigurasi koneksi Pyrogram MTProto hingga pengujian performa berkas berukuran besar, dilakukan pada perangkat lokal tersebut tanpa memerlukan penyewaan infrastruktur *server* eksternal selama fase pengembangan. Pendekatan ini dipilih karena seluruh komponen sistem dapat dioperasikan sepenuhnya dalam lingkungan lokal.

Penelitian ini direncanakan berlangsung selama bulan Februari 2026 hingga Juli 2026. Rentang waktu tersebut mencakup seluruh tahapan penelitian mulai dari penyusunan Laporan Akhir, perancangan sistem, implementasi, pengujian, revisi, hingga penyusunan laporan hasil penelitian.

3.3. Objek dan Subjek Penelitian

3.3.1. Objek Penelitian

Objek penelitian dalam penelitian ini adalah sistem perangkat lunak *Private Cloud storage* yang dirancang dan dibangun oleh peneliti. Secara lebih spesifik, objek penelitian mencakup beberapa komponen utama sistem berikut:

1. Antarmuka *Web UI* berbasis *virtual file manager*, mencakup fungsionalitas unggah, unduh, pengelolaan direktori, pencarian, dan penghapusan berkas.
2. Mekanisme enkripsi dan dekripsi berkas menggunakan algoritma AES-256-CBC yang dijalankan secara lokal di sisi *backend* sebelum data dikirimkan ke infrastruktur Telegram.
3. Sistem verifikasi integritas berkas menggunakan fungsi *hash SHA-256* untuk memastikan berkas yang diunduh identik dengan berkas asli.
4. *Library* Pyrogram yang berkomunikasi langsung dengan *server* Telegram melalui protokol MTProto sebagai mekanisme komunikasi ke *backend* penyimpanan dengan kapasitas unggahan hingga 2 GB per berkas, didukung oleh TgCrypto untuk akselerasi enkripsi protokol.
5. Mekanisme *file chunking* otomatis untuk berkas yang ukurannya melebihi batas 1,8 GB per berkas.

Mekanisme manajemen kredensial secara *stateless* menggunakan enkripsi AES-256 dan PBKDF2, di mana *server* memfasilitasi penggunaan kredensial

Telegram secara dinamis tanpa pernah menyimpan kunci dekripsinya secara permanen.

3.3.2. Subjek Penelitian

Subjek penelitian adalah sejumlah pengguna yang terlibat dalam proses evaluasi sistem pada tahap akhir pengembangan. Pengguna yang dilibatkan merupakan individu yang representatif terhadap target pengguna sistem, yaitu mahasiswa atau profesional muda yang terbiasa menggunakan layanan *cloud storage* seperti Google Drive atau Dropbox, namun tidak harus memiliki latar belakang teknis di bidang kriptografi atau pemrograman. Evaluasi dilakukan untuk menilai tingkat kemudahan penggunaan antarmuka *Web UI* dari perspektif pengguna akhir.

Evaluasi *usability* sistem dilakukan menggunakan metode kuesioner dengan skala Likert 1 hingga 5. Kuesioner ini disebarkan kepada 5 orang responden untuk mengukur tingkat kelayakan sistem berdasarkan tiga parameter utama: Fungsi, Performa, dan Kemudahan Akses.

3.4.Alat dan Bahan Penelitian

Penelitian ini menggunakan perangkat keras dan perangkat lunak sebagai sarana utama dalam pembangunan, penerapan, dan pengujian sistem. Karena sistem ini berbasis *Cloud*, perangkat keras dibagi menjadi dua lingkungan, yaitu lingkungan *development* dan lingkungan *server*. Rincian alat dan bahan disajikan pada tabel berikut:

Tabel 3. 1 Perangkat Keras Klien & Pengembangan

Perangkat Keras	**Keterangan**
Prosesor (CPU)	AMD Ryzen 5
Memori (RAM)	16 GB
Penyimpanan	1 TB
Sistem Operasi	Windows 11 64-bit
Koneksi	100 Mbps

Tabel 3. 2 Perangkat Keras *Server* (*Virtual Private Server)*

Komponen	**Spesifikasi**
Prosesor (vCPU)	2 Core
Memori (RAM)	2 GB
Penyimpanan	60 GB NVMe SSD
Sistem Operasi	Ubuntu 22.04 LTS
Lokasi *Server*	Jakarta

Tabel 3. 3 Perangkat Lunak

Komponen	**Keterangan**
Bahasa Pemrograman	Python
Web *Framework*	Flask
Library Kriptografi	PyCryptodome (AES-256-CBC)
Library Telegram	Pyrogram (MTProto API)
Database	SQLite3
Web *Server* Gateway	Gunicorn (pada VPS)
Browser Pengujian	Google Chrome / Mozilla Firefox
Code Editor	Visual Studio Code

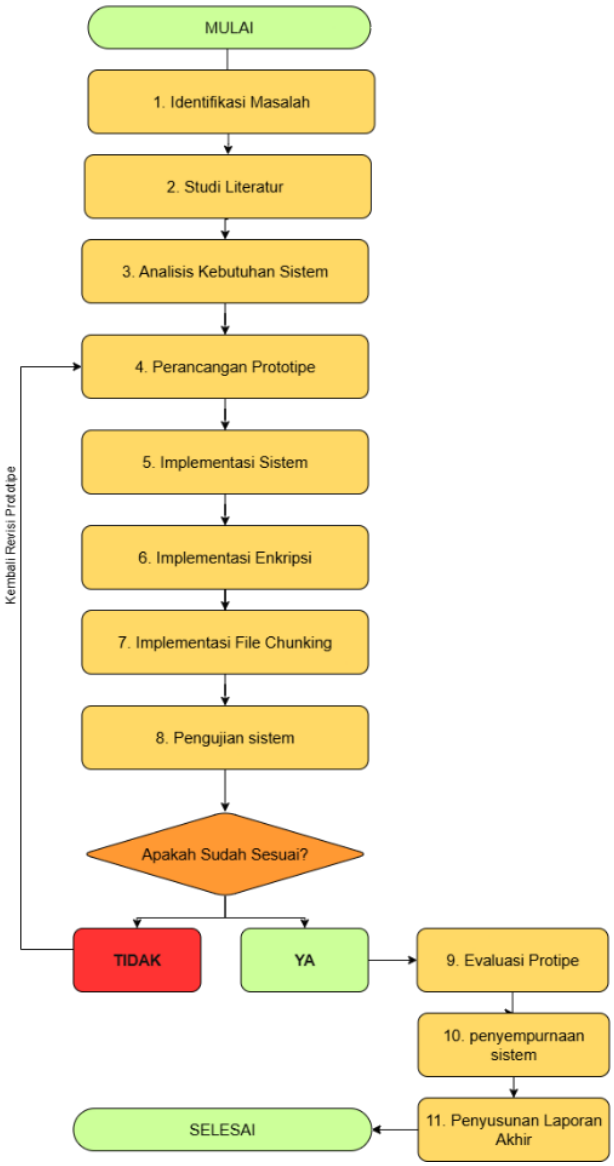
3.5. Teknik Pengumpulan Data

Pengumpulan data dalam penelitian ini dilakukan menggunakan beberapa teknik yang saling melengkapi, sebagai berikut:

1. Penelitian dimulai dengan mengumpulkan berbagai referensi seperti jurnal ilmiah, skripsi, artikel, dan dokumentasi resmi yang berkaitan dengan *cloud storage,* Telegram Bot API, MTProto algoritma AES-256, SHA-256, teknik *file chunking*, dan metode Prototyping. Dengan mempelajari referensi-referensi tersebut, peneliti memperoleh landasan teori yang kuat untuk merancang sistem yang diusulkan.
2. Selanjutnya, peneliti melakukan observasi pada layanan *cloud storage* yang umum digunakan, seperti Google Drive atau Dropbox. Dalam melakukan observasi ini, peneliti mengamati fitur, keterbatasan kapasitas, biaya layanan, serta aspek privasi data dari layanan *cloud storage*. Hasil observasi ini digunakan sebagai dasar analisis kebutuhan sistem yang dikembangkan.
3. Dalam melakukan penelitian, peneliti juga mengumpulkan data berupa catatan, gambar, dokumentasi teknis, serta spesifikasi perangkat keras dan perangkat lunak yang digunakan. Peneliti menggunakan dokumentasi ini untuk mencatat proses perancangan dan pengembangan sistem yang dibangun dalam penelitian.
4. Tahap terakhir dari penelitian ini adalah melakukan eksperimen langsung terhadap rancangan sistem yang dibangun dalam penelitian. Dalam melakukan eksperimen ini, peneliti menguji proses unggah file, enkripsi data, verifikasi integritas file, dan pengujian *file* berukuran besar. Hasil dari eksperimen ini digunakan sebagai data analisis performa dan keberhasilan sistem yang dibangun dalam penelitian.

3.6. Prosedur Penelitian

Penelitian ini dilaksanakan mengikuti alur metode Prototyping yang bersifat iteratif. Prosedur penelitian terdiri dari lima tahapan yang berurutan, di mana siklus dari tahap perancangan hingga evaluasi dapat diulang sebanyak yang diperlukan hingga sistem memenuhi seluruh kebutuhan yang telah ditetapkan. Alur prosedur penelitian secara keseluruhan disajikan pada Gambar 3.1 berikut, diikuti penjelasan rinci setiap tahapan.



Gambar 3. 1 Flowchart Tahapan Penelitian dengan Metode Prototyping

Pengembangan perangkat lunak dalam penelitian ini menggunakan metode *Prototyping*. Pemilihan metode ini didasari oleh tingginya kompleksitas teknis integrasi antara antarmuka Web, protokol MTProto Telegram, mekanisme enkripsi AES-256, dan logika *file chunking*. Metode *Prototyping* memungkinkan setiap fungsi diuji dan divalidasi secara bertahap. Berikut adalah penjabaran detail dari setiap tahapan yang dilakukan:

1. Analisis Kebutuhan Sistem (*Requirements Gathering*) Pada tahap ini, dilakukan pengumpulan spesifikasi teknis yang wajib dimiliki oleh sistem. Kebutuhan tersebut meliputi: kemampuan mengunggah berkas melampaui 2 GB, penerapan keamanan berlapis (AES-256-CBC, fungsi hash SHA-256, derivasi kunci PBKDF2), ketiadaan penyimpanan kunci permanen (*stateless key management*), serta kebutuhan infrastruktur seperti *framework* Flask, pustaka Pyrogram, dan basis data SQLite.
2. Perancangan Cepat (Quick Design) Berdasarkan analisis kebutuhan, tahap perancangan awal dilakukan dengan memodelkan arsitektur perangkat lunak menggunakan Unified Modeling Language (UML). Perancangan ini mencakup pembuatan Use Case Diagram, Activity Diagram, dan Sequence Diagram untuk memetakan alur kerja pengguna, serta perancangan Entity Relationship Diagram (ERD) untuk skema basis data SQLite yang akan menampung metadata berkas secara lokal.
3. Pembangunan Prototipe (*Build Prototype*) Desain yang telah dirancang kemudian diterjemahkan ke dalam bentuk kode program (implementasi). Pada tahap ini, dibangun sebuah *Proof of Concept* berupa antarmuka *Web User Interface* sebagai *virtual file manager* dan *backend* berbasis Python Flask. Fokus utama pada tahap ini adalah menyatukan logika enkripsi AES-256 lokal dengan jembatan transmisi data menuju API Telegram menggunakan sesi Pyrogram.
4. Evaluasi Prototipe (*User Evaluation*) Prototipe awal yang telah berjalan kemudian dievaluasi menggunakan metode *Black-Box Testing* untuk memvalidasi fungsionalitas CRUD (*Create, Read, Update, Delete*) dan pengujian performa enkripsi. Sistem juga diuji langsung (*User Acceptance Test*) oleh beberapa responden pengguna untuk menilai tingkat *usability* dari antarmuka web, kecepatan respons, dan stabilitas pengunggahan *file chunking*.
5. Penyempurnaan Prototipe (*Refining Prototype*) Masukan, anomali, serta *bug* seperti kegagalan *reassembly chunk* berkas atau terputusnya koneksi akibat *API Rate Limiting* yang teridentifikasi pada tahap evaluasi diperbaiki pada tahap ini. Siklus perbaikan ini dilakukan secara berulang (iteratif) hingga

sistem dinyatakan stabil. Setelah kriteria keberhasilan terpenuhi secara komprehensif, prototipe final *deployment* ke dalam lingkungan produksi menggunakan *Web Server Gateway Interface* Gunicorn agar dapat digunakan secara nyata.

3.6.1. Identifikasi Masalah

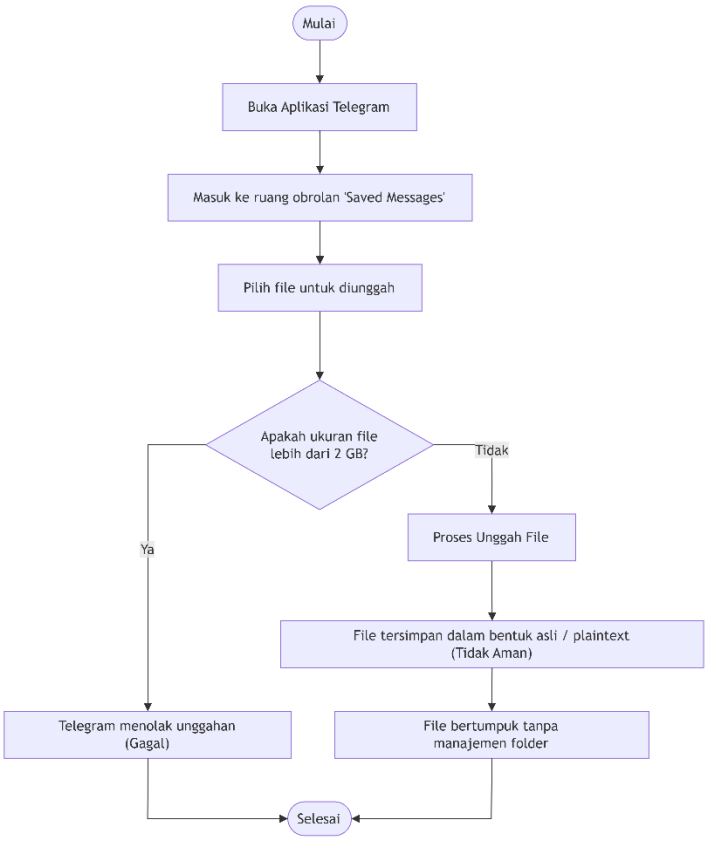
Tahap ini merupakan titik awal penelitian. Peneliti mengidentifikasi permasalahan utama yang menjadi latar belakang penelitian, yaitu keterbatasan kapasitas dan biaya layanan *public cloud storage*, risiko privasi data pada penyimpanan berbasis pihak ketiga, serta tidak adanya solusi penyimpanan berbasis Telegram yang memiliki antarmuka terstruktur sekaligus mekanisme enkripsi berkas yang komprehensif. Hasil identifikasi ini menjadi dasar perumusan masalah penelitian.

3.6.2. Studi Literatur

Peneliti mempelajari referensi ilmiah yang relevan mencakup konsep *Private Cloud storage*, spesifikasi teknis *Telegram Bot API* dan protokol MTProto melalui *library* Pyrogram, standar kriptografi AES-256, teknik *file chunking*, serta metodologi pengembangan perangkat lunak Prototyping. Studi literatur juga mencakup penelitian-penelitian terdahulu yang memiliki keterkaitan topik untuk memetakan celah penelitian yang akan diisi.

3.6.3.Analisis Sistem yang Berjalan

Sebelum sistem yang diusulkan pada penelitian ini dibangun, proses penyimpanan berkas pribadi berbasis Telegram masih dilakukan secara manual dan konvensional. Alur sistem eksisting tersebut digambarkan pada *Flowchart* berikut:



Gambar 3. 2 Alur Sistem Eksisting Pengunggahan Berkas secara Manual

Berdasarkan gambar di atas, sistem eksisting mengharuskan pengguna mengirimkan berkas secara manual ke dalam fitur *Saved Messages* pada aplikasi Telegram. Kondisi ini memiliki tiga kelemahan arsitektural yang fatal:

1. Tanpa Enkripsi Klien, berkas dikirim dan disimpan dalam bentuk asli (*plaintext*), sehingga rentan terhadap pelanggaran privasi data.
2. Tanpa Manajemen Direktori, Karena basis antarmukanya adalah ruang obrolan (*chat*), berkas akan terus menumpuk ke bawah tanpa adanya fitur *folder*, sehingga menyulitkan pengorganisasian data.
3. Terhalang Limitasi Protokol, Pengguna secara mutlak tidak dapat menyimpan berkas yang ukurannya melebihi batasan ketat server Telegram (maksimal 2 GB per berkas).

Ketiga kelemahan mendasar pada sistem eksisting inilah yang kemudian diselesaikan dan menjadi landasan pengembangan aplikasi *Private Cloud Storage* pada penelitian ini.

3.6.4. Analisis Kebutuhan Sistem

Berdasarkan hasil identifikasi masalah dan studi literatur, peneliti menentukan kebutuhan fungsional dan non-fungsional sistem secara rinci. Kebutuhan fungsional mencakup: (1) kemampuan unggah, unduh, dan pengelolaan berkas melalui *Web UI*; (2) enkripsi AES-256-CBC otomatis sebelum berkas dikirimkan ke Telegram; (3) verifikasi integritas SHA-256; serta (4) *file chunking* otomatis untuk berkas di atas batas operasional sebagaimana parameter operasional yang telah ditetapkan sejak awal penelitian. Kebutuhan non-fungsional mencakup performa yang dapat diterima, keamanan data terenkripsi, dan kemudahan penggunaan antarmuka.

Selain kebutuhan utama, sistem ini juga memiliki beberapa fitur keamanan dan pendukung lainnya yang mengikuti standar praktik terbaik dalam pengembangan aplikasi web modern. Sistem ini menggunakan autentikasi JWT (*JSON Web Token)* melalui *library* Flask-JWT-Extended untuk mengotentikasi pengguna. Setiap kali pengguna mengirimkan permintaan ke *REST API backend, backend* harus menyertakan *token* JWT yang valid di *header Authorization*. *Token* ini hanya berlaku selama 24 jam, dan jika pengguna melakukan *logout*, *token* tersebut akan dimasukkan ke dalam daftar *blocklist* di *database* untuk mencegah penggunaan ulang. Konfigurasi CORS (*Cross-Origin Resource Sharing*) menggunakan *library* Flask-CORS diterapkan untuk memungkinkan frontend dan *backend* berkomunikasi secara lancar meskipun berjalan pada domain atau port yang berbeda. Ini sangat berguna selama proses pengembangan dan *deployment.* Untuk menjaga keamanan password pengguna, sistem ini menggunakan algoritma Bcrypt untuk meng-*hash password* sebelum menyimpannya ke *database*. Ini sesuai dengan standar keamanan yang direkomendasikan oleh OWASP. Sistem ini juga menggunakan *PBKDF2 Key Derivation* dengan 600.000 iterasi dan *salt random 16-byte* untuk mengenkripsi kunci enkripsi AES per-file. Konfigurasi ini sesuai dengan rekomendasi OWASP 2023 untuk mencegah serangan *brute-force* terhadap kunci enkripsi. Selain itu, sistem ini juga memiliki fitur *streaming download* yang memungkinkan pengguna mengunduh file secara *streaming*, tanpa perlu memuat

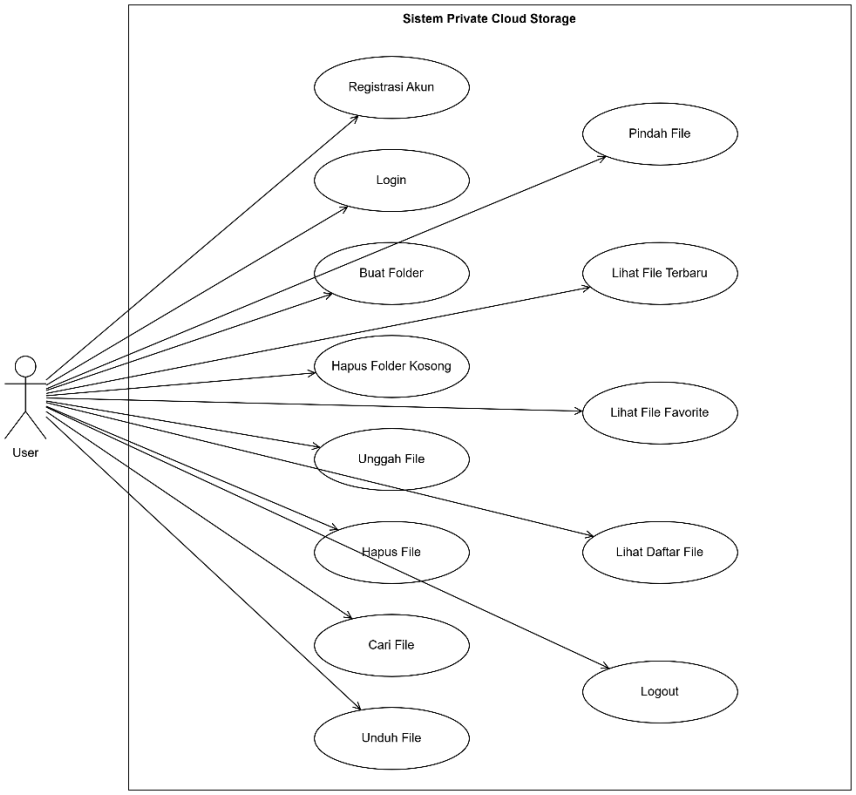
seluruh isi file ke dalam RAM *server*. Ini sangat berguna untuk mengunduh file berukuran besar dengan penggunaan memori yang efisien. Antarmuka *Web UI* juga dilengkapi dengan beberapa fitur tambahan, seperti pencarian file, penandaan file favorit, daftar file terbaru, pemindahan file antar folder, dan penghapusan berkas secara permanen**.**

Lebih lanjut, secara fungsional, sistem harus menyediakan antarmuka bagi pengguna untuk mendaftarkan kredensial Telegram pribadi (API ID, API *Hash*, dan Chat ID) sebagai syarat wajib penggunaan layanan. Secara non-fungsional, kredensial Telegram pengguna menuntut tingkat keamanan yang ketat. Sistem diwajibkan mengenkripsi data kredensial tersebut sejak dari proses registrasi hingga penggunaan sesi (*session*) secara dinamis, di mana kunci dekripsi dilarang keras untuk disimpan secara persisten di sisi *server* (*Non-Persistent Key Architecture*)**.**

3.6.5. Perancangan *Prototype*

Dalam penelitian ini, dirancang arsitektur sistem secara menyeluruh. Meliputi yang pertama desain antarmuka *Web UI* sebagai *virtual file manager*. Kedua arsitektur *backend* yang menghubungkan *Web UI* dengan *server* Telegram melalui Pyrogram MTProto dan REST API berbasis Python dengan Flask. Ketiga skema basis data *metadata* untuk menyimpan informasi berkas, urutan *chunk*, dan parameter kriptografi. Serta keempat alur kerja enkripsi-dekripsi AES-256-CBC dan mekanisme *chunking/reassembly*. Seluruh rancangan ini dituangkan dalam bentuk diagram *usecase, Entity Relationship Diagram, Class Diagram.*

3.6.5.1 *Use case Diagram*



Gambar 3. 3 *Use case Diagram* Sistem Private *Cloud storage*

Use case Diagram pada Gambar 3.2 menggambarkan interaksi yang terjadi antara aktor dengan sistem Private *Cloud storage*. Sistem ini dirancang untuk digunakan oleh satu jenis aktor utama, yaitu Pengguna (User), yang memiliki akses penuh terhadap fitur pengelolaan berkas pribadi.

Sebagai catatan, pra-syarat (pre-condition) utama bagi setiap aktor Pengguna adalah telah memiliki kredensial API Telegram (API ID dan API *Hash*) yang didapatkan secara mandiri melalui portal resmi Telegram, serta memiliki saluran atau grup privat yang akan difungsikan sebagai wadah penyimpanan yang ditandai dengan Chat ID. Tanpa ketiga kredensial ini, pengguna tidak dapat meregistrasikan akun maupun mengakses fitur-fitur utama sistem. Berikut adalah penjelasan mengenai aktor yang terlibat dalam sistem:

Tabel 3. 4 Pengertian *Actor Usecase*

No	**Aktor**	**Deskripsi**
1	Pengguna (User)	Orang yang menggunakan sistem untuk menyimpan, mengunduh, mencari, dan mengelola berkas pribadi mereka dalam Private *Cloud storage* secara aman.
2	Registrasi Akun	Pengguna mendaftarkan akun baru dengan memasukkan username, email, password, beserta kredensial Telegram (API ID, API *Hash*, Chat ID) milik masing-masing untuk mendapatkan akses ke dalam sistem.
3	Login	Pengguna melakukan autentikasi dengan memasukkan kredensial (username dan password) untuk masuk ke dalam dasbor sistem.
4	*Upload* File	Pengguna mengunggah berkas ke sistem. Sistem akan mengenkripsi berkas menggunakan AES-256 dan mengunggahnya ke Telegram melalui MTProto.
5	*Download* File	Pengguna mengunduh berkas yang telah disimpan. Sistem akan mengambil data dari Telegram, mendekripsi secara *real-time*, dan mengirimkannya ke pengguna.
6	Hapus File	Sistem akan menghapus pesan yang mengandung *file* di *server* Telegram secara permanen. Selanjutnya, sistem akan memperbarui *metadata* di basis data lokal sehingga berkas dihilangkan dari antarmuka pengguna. Mengingat infrastruktur penyimpanan utama (Telegram) telah menghapus *file* tersebut, proses ini bersifat permanen (*hard-delete logic*) dan berkas tidak dapat dipulihkan kembali.
7	Kelola Folder	Pengguna membuat atau menghapus folder untuk mengorganisasi berkas-berkas mereka.

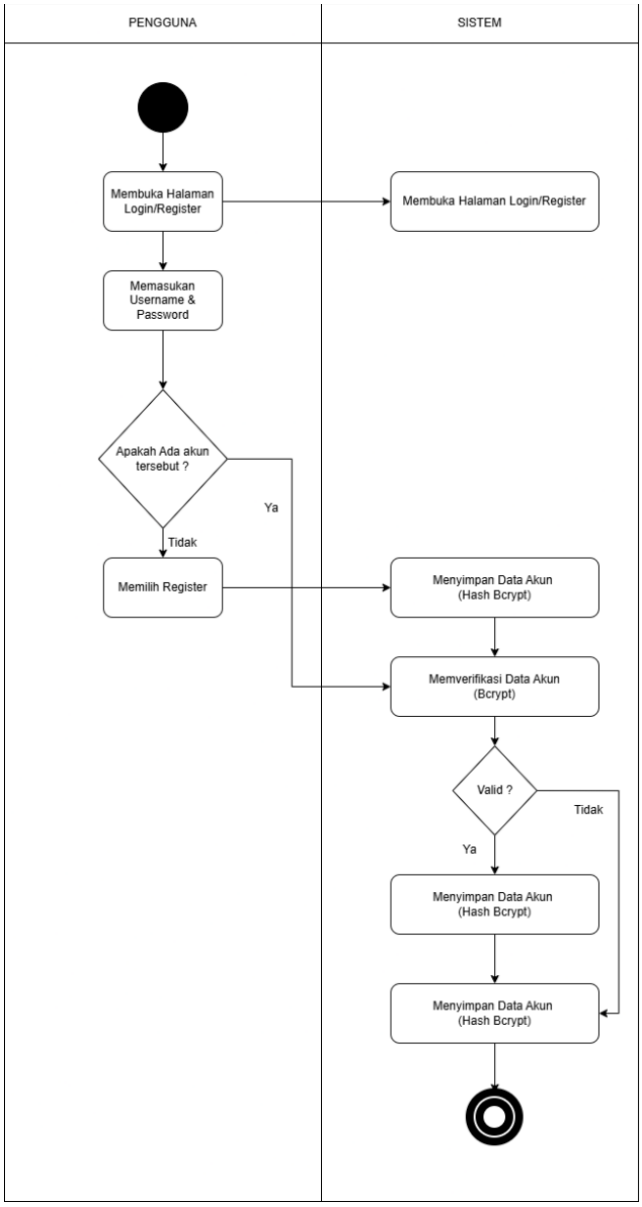
No	Aktor	Deskripsi
8	Pindah File	Pengguna memindahkan berkas dari satu direktori folder ke folder lainnya di dalam sistem.
9	Pencarian File	Pengguna mencari berkas tertentu berdasarkan nama berkas menggunakan fitur pencarian.
10	Logout	Pengguna keluar dari sistem dan sistem akan menghapus *token* sesi untuk keamanan.

3.6.5.2 *Activity Diagram*

Activity Diagram pada Gambar 3.3 menggambarkan alur kerja sistem untuk proses autentikasi, yang meliputi akses halaman utama, login, dan registrasi. Alur ini dibagi menjadi dua *swimlane*, yaitu bagian Pengguna yang melakukan interaksi, dan bagian Sistem yang melakukan pemrosesan dan validasi data. Berikut adalah rincian aktivitas pada alur tersebut:

Tabel 3. 5 Pengertian Actor *Activity Diagram*

No	**Aktivitas**	**Deskripsi**
1	Akses aplikasi via web	Pengguna membuka aplikasi Private *Cloud storage* melalui peramban web.
2	Menampilkan halaman utama	Sistem memuat dan menampilkan halaman utama (landing page) kepada pengguna.
3	Validasi Status Login	Sistem memeriksa apakah pengguna sudah login. Jika belum, tetap di halaman utama. Jika memilih login, lanjut ke halaman login.
4	Menampilkan halaman login	Sistem menampilkan formulir login untuk pengguna memasukkan kredensial.
5	Pilihan Autentikasi	Sistem memberikan opsi: jika sudah punya akun, pengguna mengisi form login; jika belum, pengguna diarahkan ke halaman register.
6	Isi form registrasi	Pengguna mengisi data diri (username, password) pada halaman register.
7	Isi form login	Pengguna mengisi username dan password pada halaman login.
8	Validasi Kredensial	Sistem memvalidasi data masukan pengguna. Jika tidak valid, dikembalikan ke halaman login/register. Jika valid, sesi dibuat.
9	Menampilkan halaman *dashboard*	Sistem mengarahkan pengguna yang telah terautentikasi ke halaman *dashboard* utama.

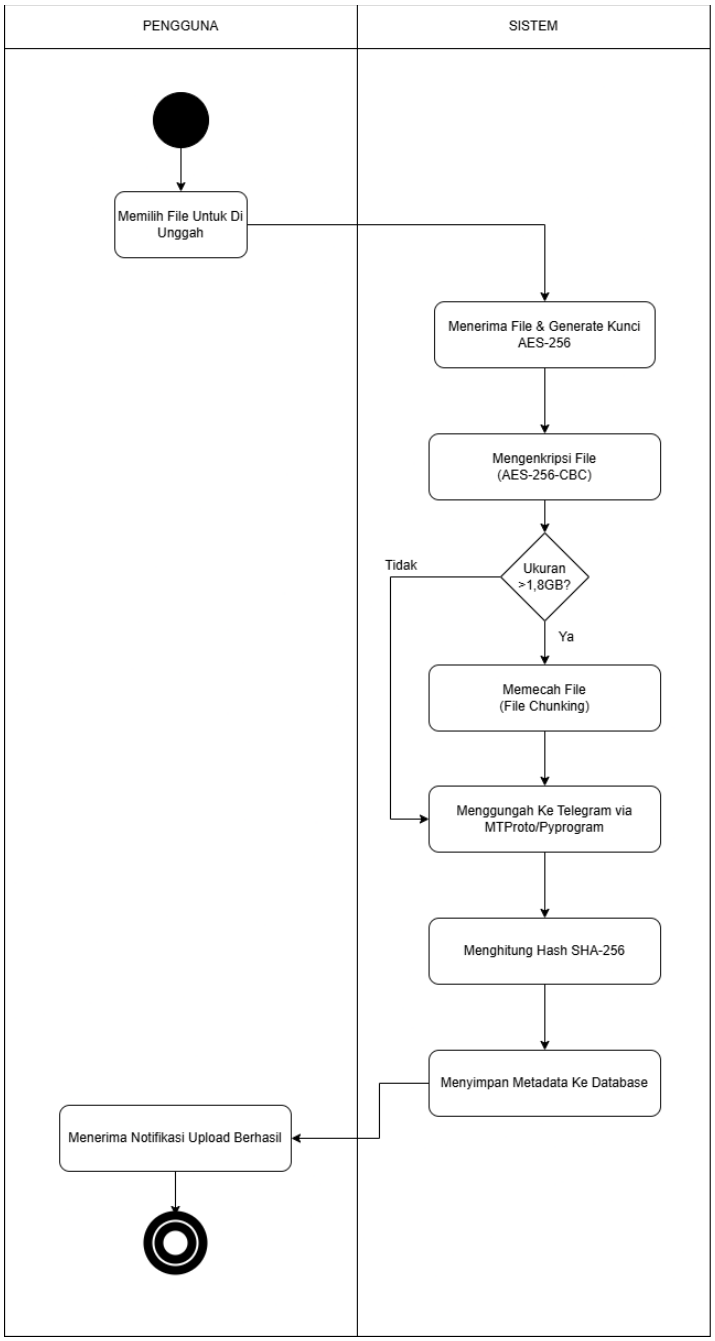


Gambar 3. 4 *Activity Diagram* Autentikasi

Selain proses autentikasi, sistem juga memiliki alur kerja utama untuk pengelolaan berkas, yaitu proses unggah (*upload*) dan unduh (*download*). *Activity Diagram* pada Gambar 3.4 menggambarkan alur kerja proses unggah berkas. Proses ini melibatkan mekanisme kriptografi enkripsi AES-256 dan pemecahan berkas (*chunking*) jika ukuran berkas melebihi 1,8 GB. Berikut adalah rincian aktivitas pada alur unggah berkas:

Tabel 3. 6 Pengertian Actor *Activity Diagram Upload* File

No	**Aktivitas**	**Deskripsi**
1	Pilih file dan klik *Upload*	Pengguna memilih berkas yang ingin diunggah dari peramban web dan menekan tombol *upload*.
2	Terima file	Sistem (*Backend*) menerima file dari sisi klien.
3	Hitung *Hash SHA-256*	Sistem menghitung nilai *hash SHA-256* dari berkas asli (*plaintext*) sebagai referensi integritas data.
4	Pengecekan Ukuran File	Sistem mengevaluasi ukuran berkas. Jika ukuran melebihi 1,8 GB, sistem akan mengarahkan ke alur *chunking*.
5	Enkripsi Single File	Jika file kurang dari 1,8 GB, sistem men-generate satu AES Key dan IV, lalu mengenkripsi seluruh file menggunakan mode AES-256-CBC, kemudian mengunggahnya ke Telegram.
6	Enkripsi *File chunking*	Jika file lebih dari 1,8 GB, sistem memecahnya menjadi beberapa segmen (*chunk*). Setiap *chunk* dienkripsi menggunakan AES Key dan IV yang unik, lalu diunggah ke Telegram secara berurutan.
7	Simpan *Metadata* & Key ke DB	Sistem mengamankan AES Key dengan PBKDF2 (key wrapping) lalu menyimpannya bersama *metadata* (file_id, *hash*, nama file) ke dalam SQLite lokal.
8	Tampilkan sukses	Sistem memberikan respon ke pengguna bahwa unggah berkas telah berhasil.

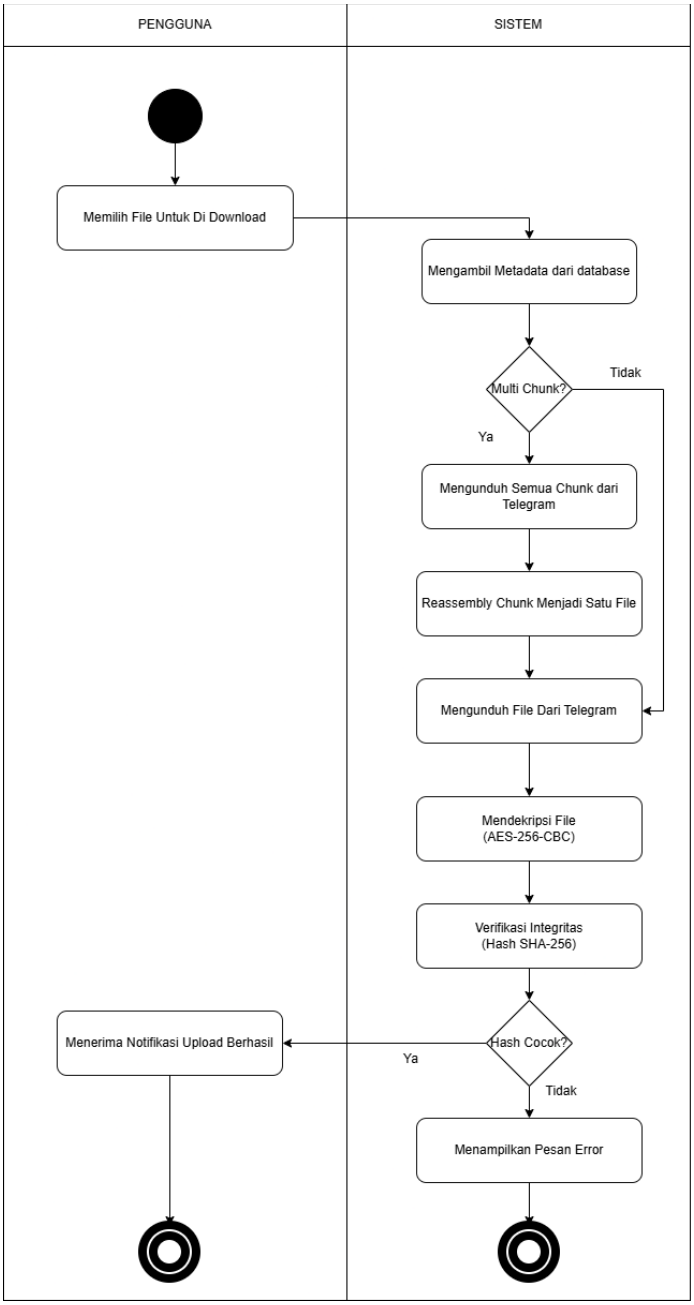


Gambar 3. 5 *Activity Diagram Upload File*

Activity Diagram pada Gambar 3.5 menggambarkan alur kerja proses unduh (*download*) berkas. Pada proses ini, sistem memanfaatkan teknik *streaming* untuk mengambil data dari Telegram dan mendekripsinya secara langsung tanpa harus menyimpannya secara penuh di memori *server*, sehingga meningkatkan efisiensi. Berikut adalah rincian aktivitas pada alur unduh berkas:

Tabel 3. 7 *Activity Diagram Download* File

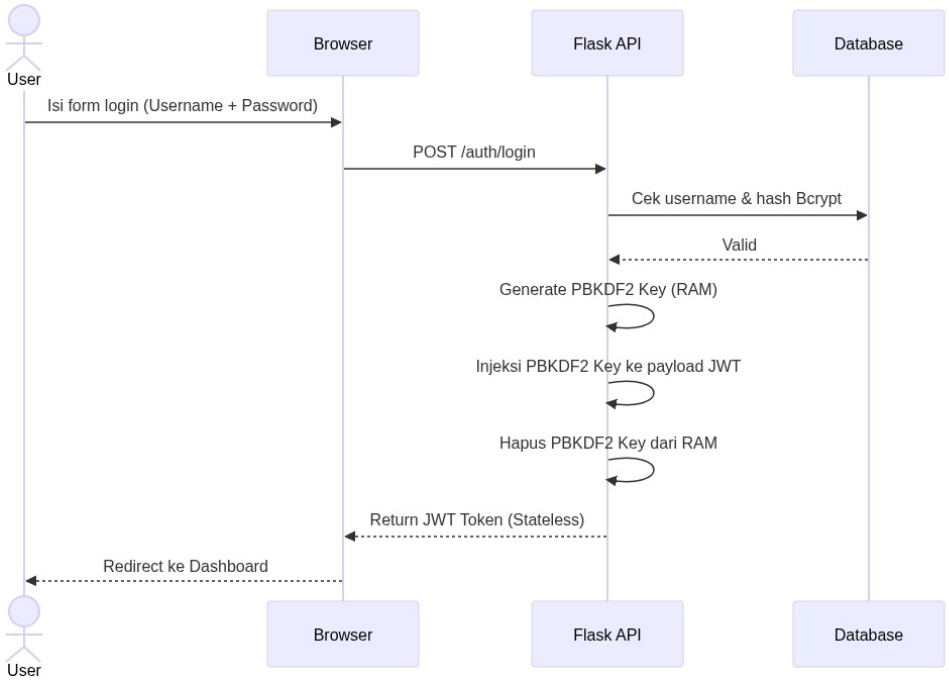
No	**Aktivitas**	**Deskripsi**
1	Klik tombol *Download*	Pengguna menekan tombol unduh pada berkas tertentu di antarmuka *Web UI*.
2	Terima request *download*	Sistem menerima permintaan unduh dan memvalidasi hak akses pengguna terhadap berkas tersebut.
3	Ambil Kunci AES dari DB	Sistem mengambil *metadata* berkas, mendapatkan file_id Telegram, dan melakukan unwrap pada Kunci AES menggunakan *Master* Key.
4	*Stream* file dari Telegram	Sistem mengunduh berkas terenkripsi (*ciphertext*) dari infrastruktur Telegram secara *streaming* (*chunk*-by-*chunk*).
5	Dekripsi AES-256 *real-time*	Sistem mendekripsi *stream ciphertext* yang diterima dari Telegram menggunakan AES-256-CBC secara *real-time* di memori.
6	*Stream* file ke pengguna	Sistem langsung mengirimkan hasil dekripsi (*plaintext*) secara *streaming* ke peramban pengguna.
7	Menerima file asli	Peramban pengguna menerima aliran data dan menyusunnya kembali menjadi berkas utuh di perangkat pengguna.



Gambar 3. 6 *Activity Diagram Download* File

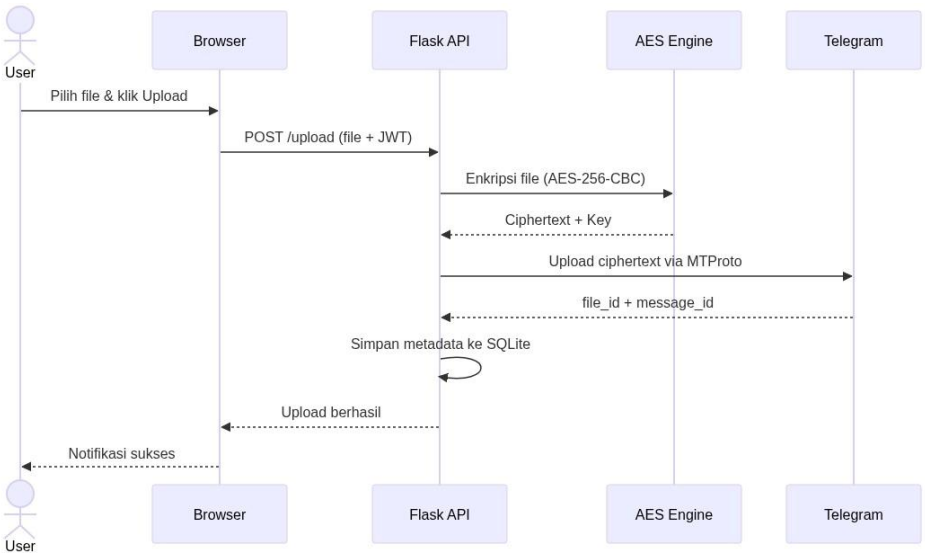
3.6.5.3 *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan urutan interaksi antar komponen sistem dalam menjalankan satu skenario fungsional. Gambar 3.6 menggambarkan urutan interaksi pada proses autentikasi (login). Pengguna mengisi formulir login di *browser*, kemudian Flask API memverifikasi kredensial ke *database* dan mengembalikan JWT *Token* sebagai bukti autentikasi yang valid.



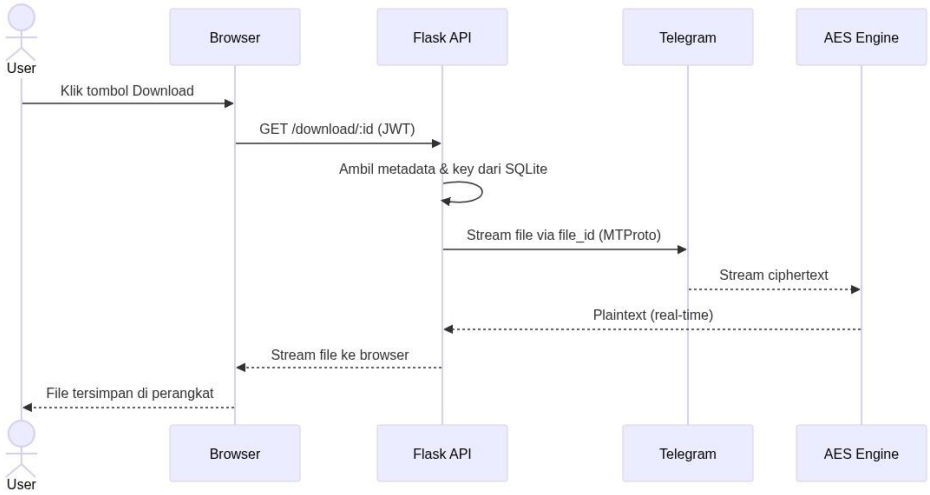
Gambar 3. 7 *Sequence Diagram* Autentikasi (Login)

Sequence Diagram pada Gambar 3.7 menggambarkan urutan interaksi antar komponen dalam proses unggah berkas. Berkas diterima oleh Flask API, dienkripsi melalui AES Engine, dikirim ke Telegram via protokol MTProto, dan *metadata* beserta kunci enkripsi yang telah di-wrap dengan PBKDF2 disimpan ke dalam basis data SQLite.



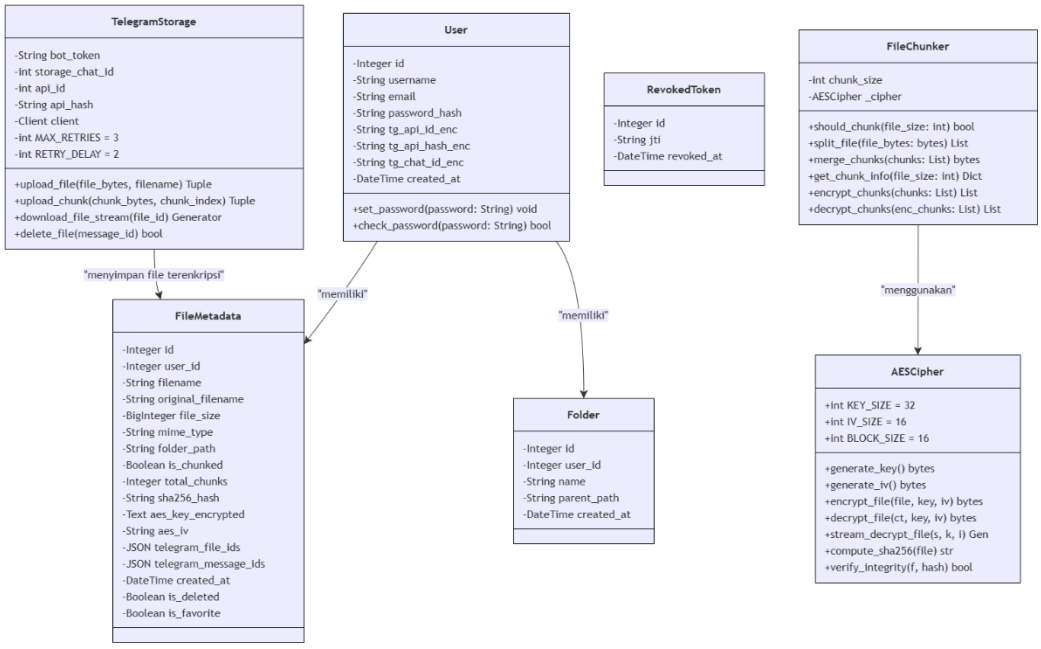
Gambar 3. 8 *Sequence Diagram Upload* File

Sequence Diagram pada Gambar 3.8 menggambarkan urutan interaksi antar komponen dalam proses unduh berkas. Sistem mengambil kunci enkripsi dari *database* lokal, kemudian mengunduh berkas terenkripsi dari Telegram secara *streaming*, mendekripsinya menggunakan AES-256, dan mengirimkan hasilnya langsung ke *browser* pengguna.



Gambar 3. 9 *Sequence Diagram Download* File

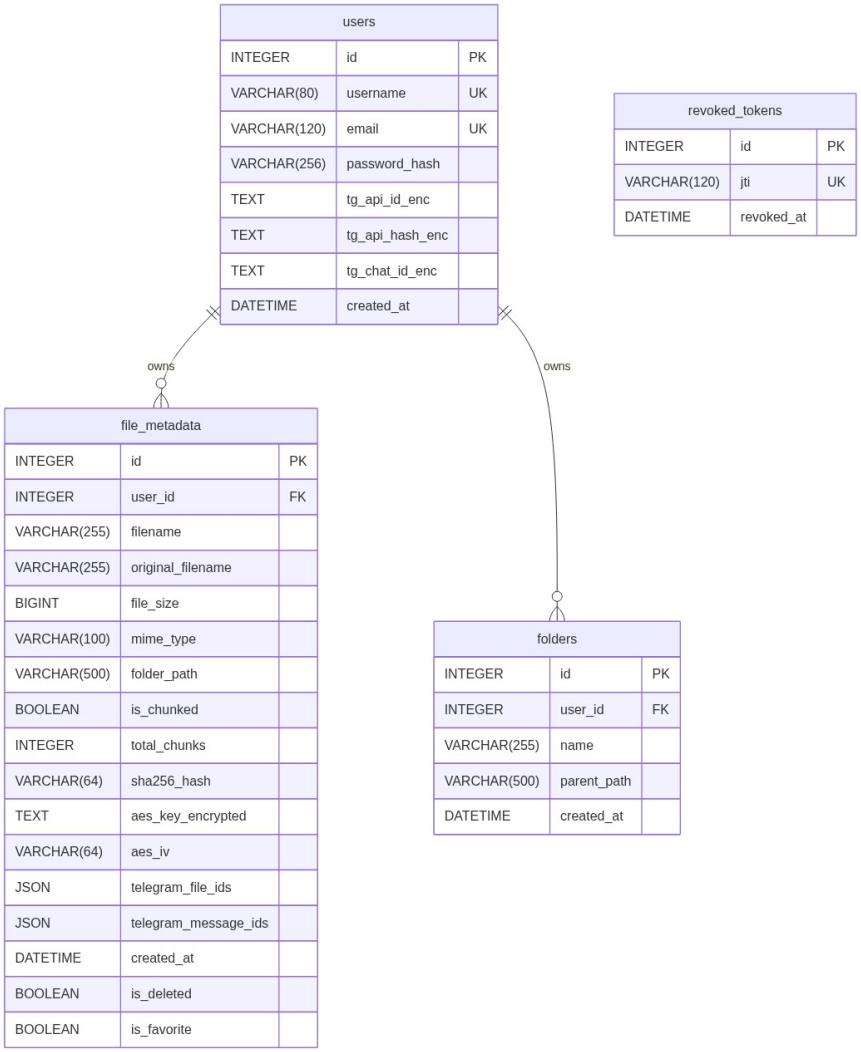
3.6.5.4 *Class Diagram*



Gambar 3. 10 *Class Diagram*

Class Diagram pada Gambar 3.9 menggambarkan struktur statis dari sistem Private *Cloud storage* yang terdiri dari beberapa kelas utama beserta atribut dan metode yang dimilikinya. Kelas User merepresentasikan entitas pengguna yang menyimpan informasi autentikasi seperti username, password_*hash* (hasil *hashing* Bcrypt), serta kredensial Telegram terenkripsi (encrypted_api_id, encrypted_api_*hash*, encrypted_chat_id). Kelas File merepresentasikan *metadata* berkas yang diunggah, mencakup nama file asli, ukuran, *hash SHA-256* untuk verifikasi integritas, serta parameter kriptografi berupa kunci AES terenkripsi (encrypted_key), Initialization Vector (IV), dan *salt* PBKDF2. Kelas File*Chunk* merepresentasikan potongan-potongan berkas hasil proses *chunking*, di mana setiap *chunk* memiliki nomor urut (*chunk*_order) dan telegram_message_id yang merujuk ke lokasi penyimpanannya di Telegram *Cloud*. Relasi antar kelas menunjukkan bahwa satu User dapat memiliki banyak File (one-to-many), dan satu File dapat terdiri dari banyak File*Chunk* (one-to-many), mencerminkan mekanisme *file chunking* yang diterapkan dalam sistem.

3.6.5.5 *Entity Relationship Diagram*



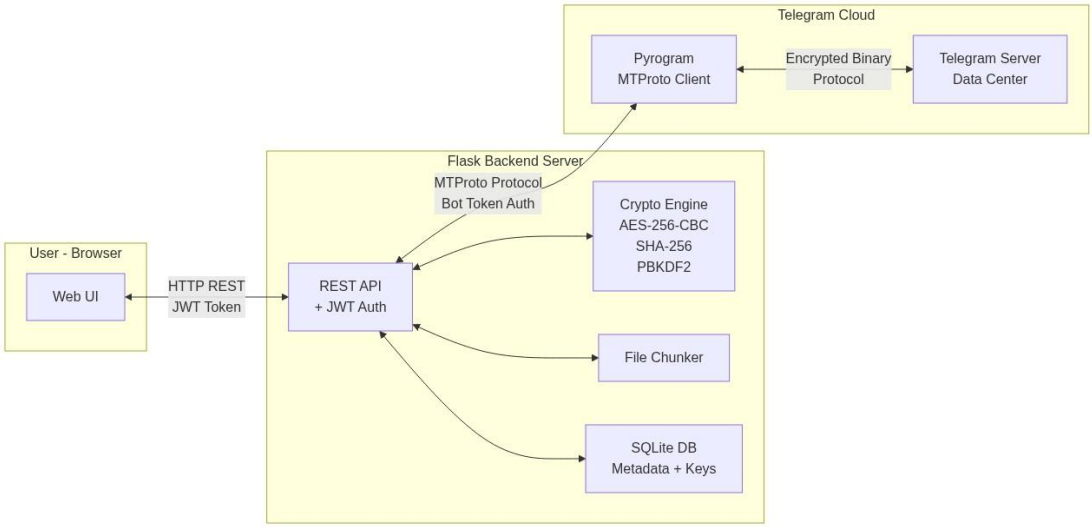
Gambar 3. 11 *Entity Relationship Diagram* (ERD) Basis Data *Metadata*

Entity Relationship Diagram pada Gambar 3.10 memodelkan struktur basis data *metadata* yang digunakan oleh sistem *Private Cloud storage*. Basis data ini dirancang untuk menyimpan seluruh informasi yang diperlukan tanpa pernah menyimpan berkas asli maupun kunci dekripsi utama secara permanen, sesuai dengan prinsip arsitektur enkripsi nir-simpan (*stateless encryption*). Entitas utama dalam ERD ini meliputi: (1) Tabel User yang menyimpan data akun pengguna beserta kredensial API Telegram terenkripsi; (2) Tabel Folder yang mencatat jalur hierarki direktori (*folder path*); serta (3) Tabel File*Metadata* yang menyimpan *metadata* berkas, parameter kriptografi (*aes_key_encrypted* dan *aes_iv*), *hash SHA-*

256, serta larik data terstruktur JSON (*telegram_file_id* dan *telegram_message_ids*) yang mencatat referensi potongan berkas (*chunk*) di *server* Telegram, serta (4) Tabel Revoked*Token* yang berfungsi mencatat *token* otentikasi JWT yang telah dibatalkan (*blocklist*) guna mencegah penyalahgunaan sesi akses pengguna yang telah melakukan *logout*. Relasi antar entitas menunjukkan bahwa satu pengguna dapat memiliki banyak folder maupun berkas (*one-to-many*). Dengan mendayagunakan kolom tipe data JSON untuk melacak informasi *chunk*, sistem tidak lagi memerlukan tabel relasional tambahan secara terpisah, sehingga skema basis data menjadi jauh lebih efisien, teroptimasi, dan meminimalisasi beban kueri antar-tabel.

3.6.5.6. Component Diagram

Component Diagram digunakan untuk menggambarkan arsitektur perangkat lunak dari sistem *Private Cloud storage* yang dibangun, serta memetakan bagaimana modul-modul program saling berkomunikasi untuk menjalankan suatu fungsi utama. Selain perancangan basis data dan diagram kelas, peneliti juga merancang arsitektur keseluruhan sistem yang menggambarkan hubungan antarkomponen utama, mulai dari antarmuka pengguna hingga *backend* penyimpanan di Telegram *Cloud*. Arsitektur sistem terdiri dari tiga lapisan utama, yaitu lapisan klien (*Web UI*), lapisan *server backend* (Flask REST API beserta modul kriptografi dan *chunking*), dan lapisan penyimpanan *cloud* (Telegram *Server* yang diakses melalui Pyrogram MTProto Client). Rancangan arsitektur ini divisualisasikan pada Gambar 3.11 berikut.



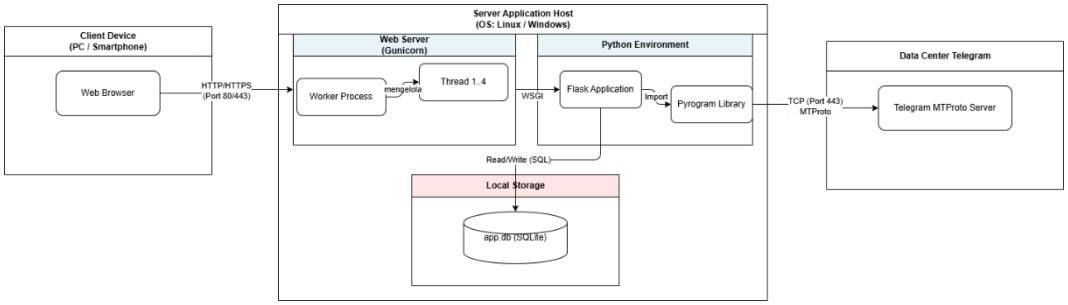
Gambar 3. 12 Arsitektur Keseluruhan Sistem Private *Cloud storage*

Alur kerja sistem dimulai dari pengguna yang berinteraksi melalui *Web UI* di *browser*. Setiap permintaan dikirimkan ke *Flask Backend Server* melalui protokol *HTTP REST* dengan menyertakan JWT *Token* sebagai kredensial autentikasi. Di sisi *backend*, file yang diunggah diproses melalui *Crypto Engine* untuk dihitung *hash SHA-256*-nya dan dienkripsi menggunakan AES-256-CBC. Jika ukuran file melebihi 1,8 GB, File *Chunker* akan memecah file menjadi beberapa segmen. File terenkripsi kemudian dikirimkan ke Telegram *Server* melalui Pyrogram MTProto Client yang berkomunikasi langsung dengan *data center* Telegram menggunakan protokol biner MTProto. *Metadata* berkas beserta kunci enkripsi yang telah di-*wrap* dengan PBKDF2 disimpan di basis data SQLite lokal.

Dengan arsitektur ini, tidak ada satu pun komponen yang menyimpan file asli (*plaintext*) secara permanen. File asli hanya ada sementara di RAM *server* saat proses enkripsi atau dekripsi berlangsung. Basis data SQLite lokal hanya menyimpan *metadata* berkas dan kunci enkripsi AES yang telah di-*wrap* menggunakan PBKDF2, sedangkan file terenkripsi (*ciphertext*) tersimpan sepenuhnya di infrastruktur Telegram *Cloud*.

3.6.5.7. *Deployment* Diagram

Deployment Diagram (Diagram Penerapan) digunakan untuk memvisualisasikan topologi jaringan fisik dan perangkat keras tempat sistem perangkat lunak ini di-*deploy* (diterapkan), termasuk rute komunikasi antarsimpul (node).

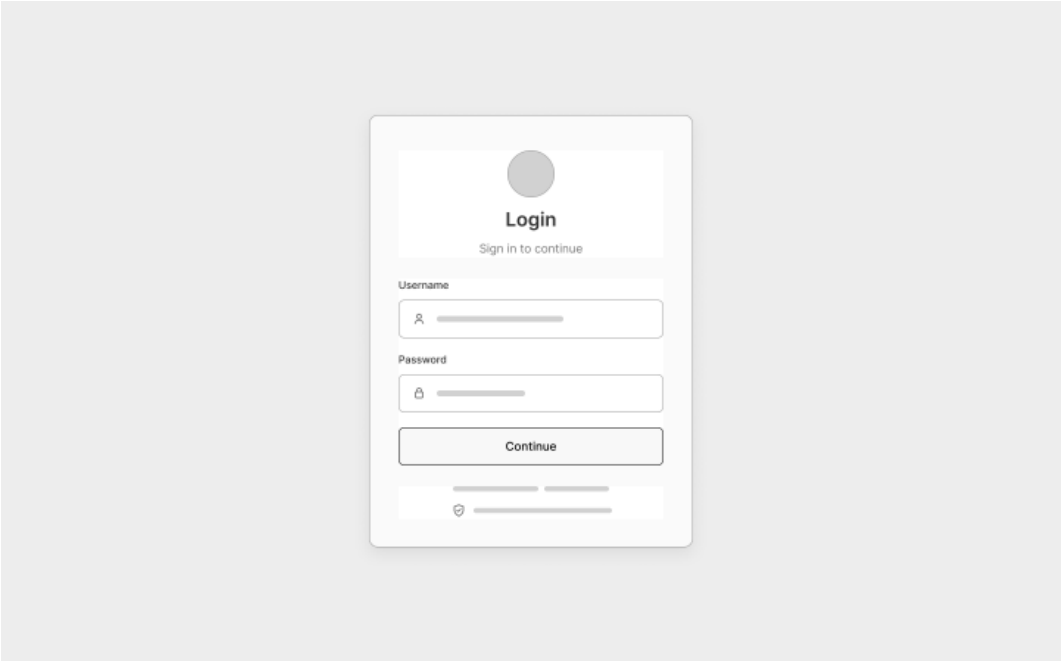


Gambar 3. 13 *Deployment* Diagram Private *Cloud storage*

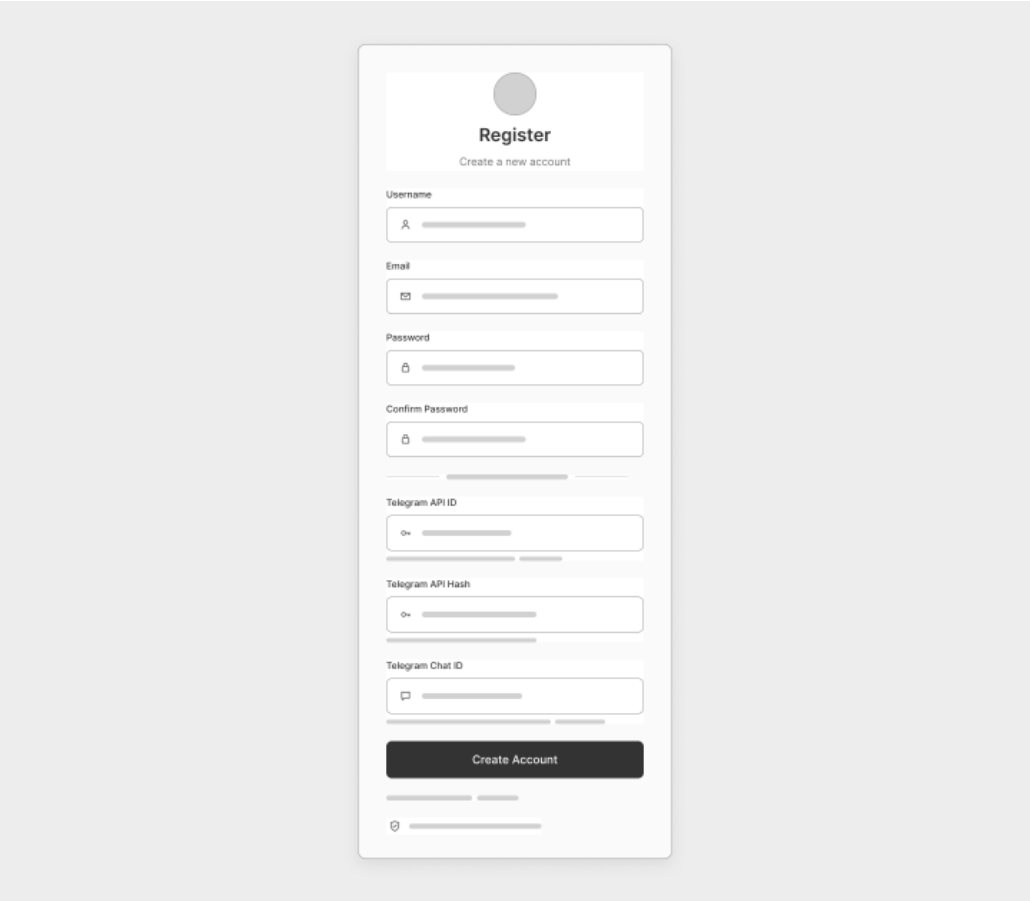
Berdasarkan Gambar 3.13, lingkungan produksi sistem terbagi menjadi tiga *node* utama. Pada *node* klien, pengguna mengakses sistem melalui peramban web (*Web Browser*). Permintaan diteruskan melalui jaringan internet ke *node Server Application Host*. Aplikasi Python ini dikonfigurasi berjalan di atas *web server Gunicorn* dengan alokasi 4-*thread* (*Worker Process*) untuk memastikan operasi I/O basis data SQLite berjalan secara asinkron tanpa memicu kendala *database is locked*. Pada tahap akhir, aplikasi di *server* akan membuka koneksi *socket* TCP langsung menuju *node Data center Telegram* menggunakan protokol biner MTProto (port 443) untuk melakukan penyimpanan dan penarikan *ciphertext*.

3.6.5.8. Perancangan Antarmuka (*Wireframe*)

Perancangan antarmuka pengguna pada sistem *Private Cloud Storage* ini digambarkan menggunakan *Wireframe*. *Wireframe* berfungsi sebagai kerangka visual dasar yang merepresentasikan tata letak tata letak elemen-elemen penyusun halaman web sebelum tahap implementasi kode dilakukan. Pendekatan ini selaras dengan fase *Quick Plan & Modeling* pada metode *Prototyping*, di mana fokus utamanya adalah fungsionalitas navigasi dan kemudahan interaksi pengguna. Berikut adalah rancangan *wireframe* untuk antarmuka utama sistem:

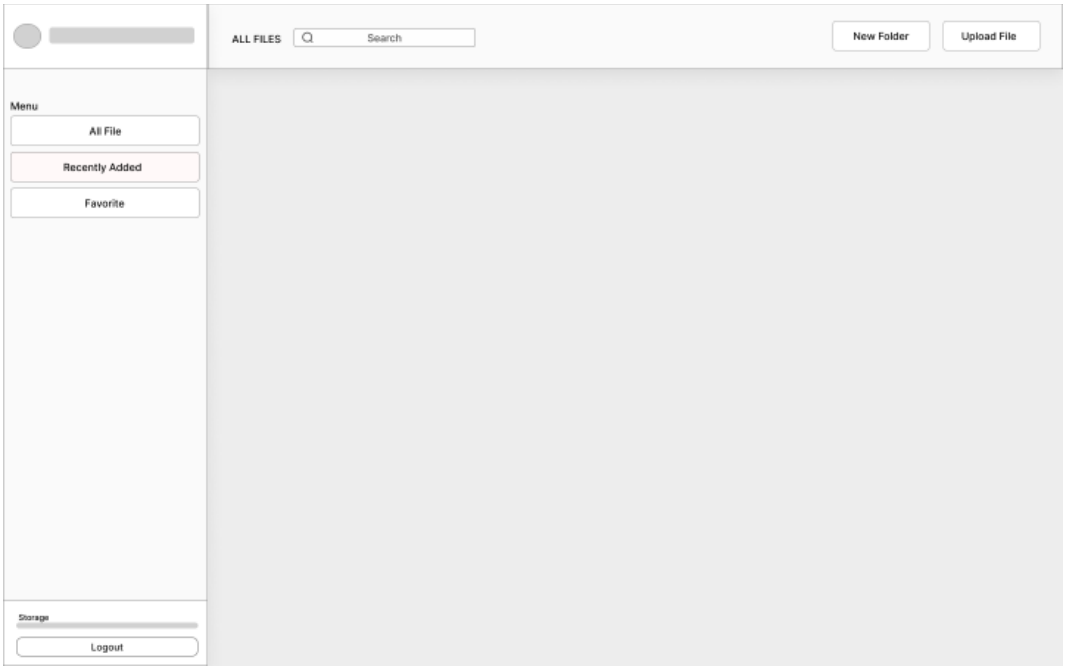


Gambar 3. 14 *Wireframe* Halaman Autentikasi Login



Gambar 3. 15 *Wireframe* Halaman Autentikasi Registrasi

Gambar di atas menunjukkan *wireframe* untuk halaman autentikasi. Halaman ini dirancang minimalis dengan menempatkan form pengisian kredensial di bagian tengah layar. Tujuannya adalah memusatkan fokus pengguna saat memasukkan kredensial API Telegram mereka yang bersifat sensitif.



Gambar 3. 16 *Wireframe* Halaman Dashboard Utama

Gambar di atas merupakan *wireframe* dari *Dashboard* utama sistem yang bertindak sebagai *virtual file manager*. Tata letak dibagi menjadi dua bagian utama: panel navigasi (*sidebar*) di sebelah kiri untuk berpindah direktori, dan area kerja utama (*workspace*) di bagian tengah-kanan yang menampilkan daftar berkas menggunakan bentuk *grid* atau *list*. Di bagian atas, disediakan area tombol aksi cepat untuk memfasilitasi proses unggah berkas berbasis *drag-and-drop*.

3.6.6. Implementasi Sistem

Prototype sistem dibangun berdasarkan rancangan yang telah dibuat. Implementasi mencakup pembangunan *Web UI* menggunakan HTML/CSS/JavaScript, pengembangan *backend* REST API menggunakan Python dengan *framework* Flask, konfigurasi koneksi Pyrogram MTProto ke *server* Telegram, dan pembangunan basis data *metadata* menggunakan SQLite.

Pada tahap ini, fungsionalitas dasar unggah dan unduh berkas melalui Telegram sudah dapat diuji secara awal.

3.6.7. Implementasi Enkripsi

Lapisan keamanan kriptografi diintegrasikan ke dalam alur pemrosesan berkas di sisi *backend*. Setiap berkas yang akan diunggah terlebih dahulu dihitung nilai *hash SHA-256*-nya sebagai referensi integritas, kemudian dienkripsi menggunakan AES-256-CBC dengan *Initialization Vector* (IV) yang diacak secara kriptografis. Kunci enkripsi dan IV disimpan secara aman pada basis data *metadata* lokal. Pada proses pengunduhan, *ciphertext* didekripsi terlebih dahulu sebelum dikirimkan kepada pengguna, dan nilai SHA-256 berkas hasil dekripsi diverifikasi untuk memastikan integritas data.

3.6.8. Implementasi *File chunking*

Mekanisme *file chunking* diimplementasikan pada lapisan *backend* untuk menangani berkas yang ukurannya melebihi 1,8 GB. Sistem secara otomatis mendeteksi ukuran berkas yang diunggah jika melampaui batas operasional tersebut, berkas dipecah menjadi sejumlah segmen (*chunk*) yang masing-masing berukuran di bawah 1,8 GB sebagaimana parameter operasional yang telah ditetapkan sejak awal penelitian. Setiap *chunk* kemudian dienkripsi secara independen menggunakan kunci dan IV unik, lalu diunggah secara sekuensial ke *server Telegram* melalui Pyrogram MTProto. *Metadata* setiap *chunk* (urutan, ID pesan Telegram, IV enkripsi) disimpan di basis data lokal untuk keperluan *reassembly* saat pengunduhan.

3.6.9. Pengujian Sistem

Sistem yang telah diimplementasikan diuji menggunakan skenario pengujian terstruktur yang mencakup *Black-Box testing* untuk seluruh fungsionalitas, pengujian performa *upload/download*, pengukuran waktu enkripsi/dekripsi, verifikasi integritas SHA-256, serta pengujian *file chunking* pada berkas berukuran lebih dari 1,8 GB. Rincian metode pengujian dijelaskan pada sub-bab 3.7.

3.6.10. Evaluasi *Prototype*

Hasil pengujian dievaluasi secara kritis oleh peneliti untuk mengidentifikasi komponen yang belum berfungsi sesuai ekspektasi, kinerja yang di bawah standar, atau antarmuka yang kurang intuitif. Evaluasi juga mencakup umpan balik dari sejumlah pengguna yang mencoba sistem secara langsung. Jika ditemukan kekurangan yang signifikan, proses dikembalikan ke tahap perancangan untuk dilakukan revisi *prototype*.

Selain itu, pengujian sistem pada tahap ini juga menggunakan metode *User Acceptance Test* untuk menilai tingkat kebergunaan (*usability*) dan penerimaan pengguna akhir terhadap sistem. Pengujian dilakukan melalui penyebaran kuesioner kepada 5 orang responden. Kuesioner dirancang menggunakan skala Likert (1-5) yang mengukur tiga aspek utama: (1) Fungsi sistem, (2) Performa sistem, dan (3) Kemudahan Akses antarmuka.

3.6.11. Penyempurnaan Sistem

Berdasarkan hasil evaluasi, sistem disempurnakan meliputi perbaikan bug, optimasi performa, peningkatan kualitas antarmuka, dan penguatan mekanisme keamanan yang ditemukan kurang optimal pada siklus sebelumnya. Proses ini berlangsung secara iteratif sesuai siklus metode Prototyping hingga seluruh kriteria keberhasilan terpenuhi.

3.6.12. Penyusunan Laporan Akhir

Tahap akhir adalah penyusunan laporan tugas akhir secara lengkap yang mendokumentasikan seluruh proses penelitian dari identifikasi masalah hingga kesimpulan. Laporan disusun mengikuti sistematika penulisan yang telah ditetapkan oleh Program Studi Teknik Informatika Universitas Islam Nusantara Bandung.

3.7. Pengolahan dan Analisis Data

Analisis data dilakukan dengan menilai keberhasilan fungsi sistem, performa transfer file, serta kesesuaian layanan terhadap karakteristik *cloud computing* seperti akses berbasis jaringan, ketersediaan layanan, dan efisiensi sumber daya.

Evaluasi ini mengacu pada konsep layanan komputasi awan yang dijelaskan oleh standar NIST (G. Wijaya, 2022).

Tabel 3. 8 Metode Pengolahan dan Analisis Data

No	**Metode Pengujian**	**Prosedur**	**Kriteria Keberhasilan**	**Jenis Analisis**
1	Uji Performa *Upload*/*Download*	Mengukur kecepatan dan waktu yang dibutuhkan untuk mengunggah dan mengunduh berkas dengan variasi ukuran (10 MB, 100 MB, 1 GB, >1,8 GB).	Waktu transfer yang dapat diterima pada kondisi jaringan normal	Kuantitatif /Benchmark
2	Uji Enkripsi & Dekripsi AES-256	Mengukur waktu yang dibutuhkan untuk mengenkripsi dan mendekripsi berkas dengan berbagai ukuran menggunakan AES-256-CBC.	Waktu enkripsi/dekripsi proporsional terhadap ukuran berkas	Kuantitatif / Benchmark

No	Metode Pengujian	Prosedur	Kriteria Keberhasilan	Jenis Analisis
3	Uji Integritas SHA-256	Memverifikasi bahwa nilai *hash SHA-256* dari berkas yang diunduh identik dengan *hash* berkas asli sebelum diunggah.	*Hash SHA-256* sebelum dan sesudah transfer identik (100%)	Verifikasi *Hash*
4	Uji *File chunking* (>1,8 GB)	Menguji proses pemecahan berkas berukuran lebih dari 1,8 GB menjadi *chunk*, pengunggahan sekuensial, dan penyusunan kembali (*reassembly*) saat diunduh.	Berkas yang diunduh identik dengan berkas asli tanpa kerusakan data	Fungsional & Integritas
5	Uji Keamanan Data di *Server*	Memverifikasi bahwa berkas yang tersimpan di infrastruktur Telegram berada dalam	*Ciphertext* tidak dapat diinterpretasikan tanpa kunci yang sah	Verifikasi Manual

No	Metode Pengujian	Prosedur	Kriteria Keberhasilan	Jenis Analisis
		kondisi terenkripsi dan tidak dapat dibaca tanpa kunci dekripsi.		
6	Evaluasi *Usability*	Meminta sejumlah responden pengguna untuk mengoperasikan sistem dan memberikan penilaian terhadap kemudahan penggunaan antarmuka *Web UI*.	dianalisis untuk mengidentifikasi elemen antarmuka yang perlu diperbaiki	Observasi

Hasil setiap pengujian dicatat secara sistematis dalam bentuk tabel data terukur (untuk pengujian kuantitatif) dan *checklist* (untuk *Black-Box testing*). Data kuantitatif seperti waktu enkripsi, kecepatan transfer, dan hasil verifikasi *hash* kemudian dianalisis secara deskriptif untuk menentukan apakah performa sistem berada dalam rentang yang dapat diterima. Hasil evaluasi *usability* dianalisis untuk mengidentifikasi elemen antarmuka yang perlu diperbaiki pada iterasi berikutnya.

.