PENERAPAN METODE PROTOTYPING PADA PERANCANGAN *PRIVATE CLOUD STORAGE* BERBASIS TELEGRAM BOT MELALUI PROTOKOL MTPROTO DENGAN ALGORITMA KRIPTOGRAFI AES-256

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Informatika Fakultas Teknik dan Pertanian Universitas Islam Nusantara Bandung

Disusun Oleh:

FAISHAL MUHAMMAD NAUFA

41037006221004

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN PERTANIAN

UNIVERSITAS ISLAM NUSANTARA

BANDUNG

2026

HALAMAN PENGESAHAN

HALAMAN PENGESAHAN

Judul Proposal	:	Penerapan Metode Prototyping Pada Perancangan Private Cloud Storage Berbasis Telegram Bot Melalui Protokol Mtproto Dengan Algoritma Kriptografi AES-256
Nama	:	Faishal Muhammad Naufa
NIM	:	41037006221004
Program Studi	:	Teknik Informatika

Tanggal Lulus: 28 Juli 2026

Disetujui oleh,

Pembimbing I,	Pembimbing II,
Muhamad Ikmal Wiawan, M.Kom	Galih, S.T., M.Kom.
NIDN. 0429089401	NIDN. 0427068707

Diketahui oleh,

Dekan,	Ketua Program Studi,
Dr. Bobby Ustari, S.P., M.P.	Dr. Ir. Tedjo Darmanto, S.T., M.T.
NIDN. 0401129202	NIDN. 8959250022

FTP UNIMUS
Fakultas Teknik dan Perencana

HALAMAN PENGESAHAN SIDANG

HALAMAN PENGESAHAN SIDANG

Judul Proposal	:	Penerapan Metode Prototyping Pada Perancangan Private Cloud Storage Berbasis Telegram Bot Melalui Protokol Mtproto Dengan Algoritma Kriptografi AES-256
Nama	:	Faishal Muhammad Naufa
NIM	:	41037006221004
Program Studi	:	Teknik Informatika

Telah dipertahankan di depan dewan penguji dalam sidang tugas akhir:

Hari : Selasa
Tanggal : 28 Juli 2026

Disetujui,

Ketua Sidang	:	Siti Nur, S.ST., M.Kom. NIDN. 0425058705	.....
Penguji 1	:	Dr. Ir. Tedjo Darmanto, S.T., M.T. NIDN. 8959250022	.....
Penguji 2	:	Yenni Fatman, S.T., M.T. NIDN. 0403048106	.....

LEMBAR PERNYATAAN

Yang bertandatangan di bawah ini:

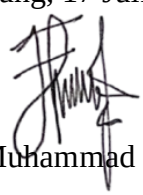
Nama	:	Faishal Muhammad Naufa
NIM	:	41037006221004
Fakultas	:	Teknik dan Pertanian
Program Studi	:	Teknik Informatika
Judul Tugas Akhir	:	Penerapan Metode Prototyping Pada Perancangan Private Cloud Storage Berbasis Telegram Bot Melalui Protokol MTProto Dengan Algoritma Kriptografi AES-256.

Dengan ini menyatakan:

1. Tugas Akhir ini merupakan hasil karya saya sendiri, dan sepanjang pengetahuan saya tidak terdapat karya atau pendapat orang lain yang ditulis atau di terbitkat sebelumnya, kecuali yang secara tertulis dikutip dan diacu dalam naskah ini sesuai kode etik ilmiah
2. Data informasi, dan hasil analisis yang di sajikan adalah valid, terukur, dan dapat, dipertanggung jawabkan secara akademik, serta tidak mengandung unsur manipulasi atau fabrikasi data.
3. Apabila di kemudian hari terbukti terdapat plagiarisme, pemalsuan data, atau pelanggaran integritas akademik lainya, saya bersedia menerima sanksi akademik sesuai peraturan yang berlaku di Universitas Islam Nusantara, termasuk pembatalan gelar yang telah diperoleh.

Demikian pernyataan ini saya buat dengan sebenar-benarnya, tanpa ada pemaksaan dari pihak mana pun, untuk dipergunakan sebagaimana mestinya.

Bandung, 17 Juli 2026



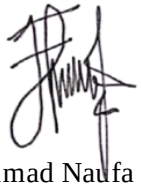
Faishal Muhammad Naufa

41037006221004

LEMBAR PERSEMBAHAN

Puji dan syukur senantiasa penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala limpahan rahmat serta karunia-Nya, sehingga penyusunan Tugas Akhir yang berjudul "**PENERAPAN METODE PROTOTYPING PADA PERANCANGAN PRIVATE CLOUD STORAGE BERBASIS TELEGRAM BOT MELALUI PROTOKOL MTPROTO DENGAN ALGORITMA KRIPTOGRAFI AES-256**" dapat diselesaikan dengan baik. Karya tulis ini disusun guna memenuhi salah satu syarat akademis dalam memperoleh gelar Sarjana Teknik pada Program Studi Teknik Informatika, Fakultas Teknik dan Pertanian, Universitas Islam Nusantara. Penulisan Tugas Akhir ini ditujukan untuk memaparkan proses perancangan dan pengembangan sistem *Private Cloud Storage* berbiaya rendah yang memanfaatkan infrastruktur server Telegram. Penulis menaruh harapan besar agar sistem yang mengintegrasikan protokol MTProto dan enkripsi data tingkat tinggi ini mampu menjadi solusi penyimpanan berkas berskala besar yang efisien, aman, serta terjamin privasinya tanpa memerlukan *hosting* server tambahan.

Bandung, 17 Juli 2026



Faishal Muhammad Naufa NIM
41037006221004

PENERAPAN METODE PROTOTYPING PADA PERANCANGAN *PRIVATE CLOUD STORAGE* BERBASIS TELEGRAM BOT MELALUI PROTOKOL MTPROTO DENGAN ALGORITMA KRIPTOGRAFI AES-256

Faishal Muhammad Naufa
Program Studi Teknik Informatika, Fakultas Teknik
Universitas Islam Nusantara

ABSTRAK

Meningkatnya kebutuhan penyimpanan data digital dihadapkan pada keterbatasan kapasitas gratis dan tingginya biaya layanan *cloud storage* komersial. Membangun infrastruktur *server* fisik secara mandiri juga membutuhkan biaya pemeliharaan yang besar. Penelitian ini bertujuan untuk mengatasi masalah tersebut dengan merancang sistem *Private Cloud storage* berbiaya rendah yang memanfaatkan infrastruktur *server* Telegram. Sistem dibangun menggunakan metode *Prototyping* dengan mengintegrasikan antarmuka *Web UI*, pustaka Pyrogram untuk komunikasi langsung ke *data center* Telegram melalui protokol biner *MTProto*, serta basis data SQLite. Untuk menjamin keamanan dan privasi, data dienkripsi secara lokal di sisi *backend* menggunakan algoritma AES-256-CBC, sedangkan integritasnya diverifikasi menggunakan fungsi *hash SHA-256*. Selain itu, batasan kapasitas transfer *native* Telegram diatasi melalui implementasi teknik *file chunking* otomatis untuk memecah berkas yang berukuran lebih dari 1,8 GB. Hasil pengujian menunjukkan bahwa sistem berhasil mengunggah, mengenkripsi, memecah, dan merekonstruksi kembali berkas tanpa kegagalan atau kerusakan data. Kesimpulannya, penggabungan protokol MTProto dan teknik *file chunking* terbukti efektif menghadirkan solusi penyimpanan awan berskala besar, efisien, dan aman tanpa memerlukan *hosting server* perantara tambahan.

Kata Kunci: *Private Cloud storage, Telegram Bot, Protokol MTProto, AES-256, File chunking, Prototyping.*

PENERAPAN METODE PROTOTYPING PADA PERANCANGAN *PRIVATE CLOUD STORAGE* BERBASIS TELEGRAM BOT MELALUI PROTOKOL MTPROTO DENGAN ALGORITMA KRIPTOGRAFI AES-256

Faishal Muhammad Naufa
Program Studi Teknik Informatika, Fakultas Teknik
Universitas Islam Nusantara

ABSTRACT

The increasing need for digital data storage faces challenges due to the limited free capacity and high costs of commercial cloud storage services. Building independent physical server infrastructure also requires massive maintenance costs. This research aims to address these problems by designing a low-cost Private Cloud storage system that utilizes Telegram's server infrastructure. The system was developed using the Prototyping method by integrating a Web UI, the Pyrogram library for direct communication to Telegram data centers via the MTProto binary protocol, and an SQLite database. To ensure security and privacy, data is locally encrypted on the backend using the AES-256-CBC algorithm, while its integrity is verified using the SHA-256 hash function. Furthermore, the native Telegram transfer limit was bypassed by implementing an automatic file chunking technique to split files larger than 1.8 GB. Testing results indicate that the system successfully uploads, encrypts, splits, and reassembles files without data failure or corruption. In conclusion, combining the MTProto protocol and file chunking techniques is proven effective in providing a large-scale, efficient, and secure cloud storage solution without requiring additional intermediary server hosting.

Keywords: *Private Cloud storage, Telegram Bot, MTProto Protocol, AES-256, File chunking, Prototyping.*

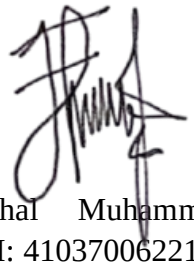
KATA PENGANTAR

Segala puji dan syukur senantiasa penulis panjatkan ke hadirat Allah SWT, karena berkat limpahan rahmat serta karunia-Nya, penyusunan tugas akhir ini dapat diselesaikan dengan baik. Melalui kesempatan ini, penulis ingin menghaturkan ucapan terima kasih yang tulus kepada seluruh pihak yang telah memberikan dukungan dan bantuan. Dengan penuh rasa syukur, tugas akhir ini penulis persembahkan untuk:

1. **Prof. Dr. Endang Komara, M.Si.,** selaku Rektor Universitas Islam Nusantara, atas segala arahan, kebijakan, dan fasilitas akademik yang sangat mendukung kelancaran proses perkuliahan dan penyusunan Tugas Akhir penulis.
2. **Bapak Dr. Tedjo Darmanto, S.T., M.T.**, selaku Ketua Program Studi Teknik Informatika Universitas Islam Nusantara, atas kesempatan, fasilitas, dan dukungan yang diberikan selama penulis menempuh pendidikan.
3. Bapak **Muhamad Ikmal Wiawan, S.Tr. Kom., M.Kom.,** selaku Dosen Pembimbing I, yang telah dengan penuh kesabaran meluangkan waktu, tenaga, dan pikirannya untuk memberikan bimbingan, arahan, serta ilmu yang sangat berharga di tengah kesibukan beliau.
4. Bapak **Galih, S.T., M.Kom.**, selaku Dosen Pembimbing II, yang tak lelah memberikan dukungan moral, masukan kritis, dan solusi atas berbagai kendala yang penulis hadapi selama merancang dan menyusun Tugas Akhir ini.
5. **Kedua Orang Tua tercinta**, atas doa yang tak pernah putus, pengorbanan yang tak terhingga, serta kasih sayang luar biasa yang selalu menjadi penguat dan alasan utama penulis untuk tidak pernah menyerah hingga berada di titik ini.
6. **Kepada teman-teman seperjuangan**, yang selalu hadir untuk saling mendukung, menemani proses diskusi, dan memberikan semangat di saat-saat penulis merasa kesulitan dalam menyelesaikan Tugas Akhir ini.
7. **Kepada seseorang dari Universitas Jambi** yang menjadi bagian penting dalam perjalanan ini, terima kasih atas segala bentuk dukungan, motivasi, dan pelajaran berharga yang telah diberikan hingga penulis dapat menyelesaikan Tugas Akhir ini.

Semoga Allah SWT senantiasa memberikan perlindungan, rahmat, serta karunia-Nya kepada semua pihak yang telah banyak membantu. Penulis berharap segala keikhlasan dalam berbagi ilmu serta dukungan moril dan materi yang telah diberikan, dapat menjadi amal jariah yang pahalanya terus mengalir. Aamiin ya Rabbal 'aalamiin.

Bandung, 17 Juli 2026



Faishal Muhammad Naufa
NIM: 41037006221004

DAFTAR ISI

HALAMAN PENGESAHAN ..... **ii**

HALAMAN PENGESAHAN SIDANG ..... **ii**

LEMBAR PERNYATAAN ..... **iv**

LEMBAR PERSEMBAHAN ..... **v**

ABSTRAK ..... **vi**

ABSTRACT ..... **vii**

KATA PENGANTAR ..... **viii**

DAFTAR ISI ..... **x**

DAFTAR GAMBAR ..... **xiv**

DAFTAR TABEL ..... **xvi**

DAFTAR ISTILAH DAN SINGKATAN ..... **xvii**

BAB I PENDAHULUAN **1**

1.1. Latar Belakang 1

1.2. Rumusan Masalah 3

1.3. Tujuan Penelitian 4

1.4. Batasan Masalah 4

1.5 Manfaat Penelitian 5

1.5.1. Manfaat Teoritis 5

1.5.2. Manfaat Praktis 6

1.6. Jadwal Penelitian 7

1.7 Sistematika Penulisan 8

BAB II TINJAUAN PUSTAKA **9**

2.1 *State of The Art* 9

2.2 Penelitian Terdahulu 14

2.3 Kajian Teori 16

2.3.1 *Private Cloud storage* dan *Virtual file manager* 16

2.3.2. Telegram Bot API dan Protokol MTProto 17
2.3.3. Teknik *File chunking* 19
2.3.4. Kriptografi dan Algoritma AES-256 21
2.3.5. Metode Prototyping 22
2.3.6 Python 24
2.3.7 Flask 24
2.3.8. Gunicorn dan WSGI 25
2.3.9 SQLite 26
2.3.10 JSON Web *Token* 26
2.3.11 Bcrypt 27
2.3.12 PBKDF2 (*Password-Based Key Derivation Function 2*) 27
2.3.13 *Black-Box* Testing 28
2.4. UML (*Unified Modeling Language*) 28
2.4.1 *Use case Diagram* 28
2.4.2 *Class Diagram* 29
2.4.3 *Activity Diagram* 30
2.4.4 *Sequence Diagram* 31
2.4.5 *Entity Relationship Diagram* (ERD) 32
BAB III METODOLOGI PENELITIAN 32
3.1. Metode Penelitian 32
3.2. Lokasi dan Waktu Penelitian 32
3.3. Objek dan Subjek Penelitian 33
3.3.1. Objek Penelitian 33
3.3.2. Subjek Penelitian 34
3.4. Alat dan Bahan Penelitian 34
3.5. Teknik Pengumpulan Data 35
3.6. Prosedur Penelitian 36

3.6.1. Identifikasi Masalah 39
3.6.2. Studi Literatur 39
3.6.3. Analisis Sistem yang Berjalan 39
3.6.4. Analisis Kebutuhan Sistem 41
3.6.5. Perancangan *Prototype* 42
3.6.6. Implementasi Sistem 60
3.6.7. Implementasi Enkripsi 61
3.6.8. Implementasi *File chunking* 61
3.6.9. Pengujian Sistem 61
3.6.10. Evaluasi *Prototype* 62
3.6.11. Penyempurnaan Sistem 62
3.6.12. Penyusunan Laporan Akhir 62
3.7. Pengolahan dan Analisis Data 62
BAB IV HASIL DAN PEMBAHASAN 63
4.1. Hasil Implementasi Sistem 63
4.1.1. Implementasi Antarmuka Pengguna (*Web UI*) 63
4.1.2. Implementasi *Database* dan *Backend* 67
4.1.3. Implementasi Lingkungan Produksi (*Production Deployment*) 68
4.2. Hasil Implementasi Keamanan dan Kriptografi 69
4.2.1. Implementasi Autentikasi JWT dan Bcrypt 69
4.2.2. Implementasi Enkripsi AES-256-CBC 71
4.2.3. Implementasi *File chunking* 73
4.3. Pengujian Sistem 75
4.3.1. Pengujian Fungsional (*Black-Box* Testing) 75
4.3.2. Pengujian Kriptografi dan Integritas (SHA-256) 82
4.3.3. Pengujian Performa 84
4.4 Evaluasi *Prototype* 87

4.5 Pengujian *Usability* (*User Acceptance Test*)..... 88
4.6 Kelebihan dan Keterbatasan Sistem..... 91
4.6.1 Kelebihan Sistem 91
4.6.2 Keterbatasan Sistem..... 92
BAB V KESIMPULAN DAN SARAN 90
5.1. Kesimpulan 90
5.2. Saran..... 91
DAFTAR PUSTAKA 93
LAMPIRAN..... 96
Lampiran 1 Dokumentasi Pengujain Sistem..... 96
Lampiran 2 Hasil Data Kuisoner..... 97
Lampiran 3 Pengujian *Production* Oleh Dosen Pembimbing..... 97
Lampiran 4 Repository Github..... 98
Lampiran 5 Informasi Spesifikasi VPS 98
Lampiran 6 Lampiran Surat Keterangan Bebas Plagiasi 99
RIWAYAT HIDUP PENULIS..... 100

DAFTAR GAMBAR

Gambar 2. 1 Metode Prototyping 23
Gambar 2. 2 Arsitektur Komunikasi WSGI Server dan Web Server 26
Gambar 2. 3 Contoh ERD 33
Gambar 3. 1 Flowchart Tahapan Penelitian dengan Metode Prototyping 37
Gambar 3. 2 Alur Sistem Eksisting Pengunggahan Berkas secara Manual 40
Gambar 3. 3 Use case Diagram Sistem Private Cloud storage 43
Gambar 3. 4 Activity Diagram Autentikasi 47
Gambar 3. 5 Activity Diagram Upload File 49
Gambar 3. 6 Activity Diagram Download File 51
Gambar 3. 7 Sequence Diagram Autentikasi (Login) 52
Gambar 3. 8 Sequence Diagram Upload File 52
Gambar 3. 9 Sequence Diagram Download File 53
Gambar 3. 10 Class Diagram 53
Gambar 3. 11 Entity Relationship Diagram (ERD) Basis Data Metadata 55
Gambar 3. 12 Arsitektur Keseluruhan Sistem Private Cloud storage 57
Gambar 3. 13 Deployment Diagram Private Cloud storage 58
Gambar 3. 14 Wireframe Halaman Autentikasi Login 59
Gambar 3. 15 Wireframe Halaman Autentikasi Registrasi 59
Gambar 3. 16 Wireframe Halaman Dashboard Utama 60
Gambar 4. 1 Tampilan Halaman Registrasi Sistem 64
Gambar 4. 2 Tampilan Halaman Registrasi Sistem pada kredensial Telegram 64
Gambar 4. 3 Tampilan Halaman Login Sistem 65
Gambar 4. 4 Tampilan Dashboard Utama 65
Gambar 4. 5 Tampilan Dialog Unggah Berkas dengan Indikator Kemajuan 66
Gambar 4. 6 Tampilan Fitur Pencarian Berkas 66
Gambar 4. 7 Diagram Skema Basis Data SQLite 67
Gambar 4. 8 Berkas Terenkripsi yang Tersimpan di Infrastruktur Telegram 69
Gambar 4. 9 Contoh *Hash* Bcrypt yang Tersimpan di Basis Data 70
Gambar 4. 10 Penyimpanan Secret_key & Basis Data 71
Gambar 4. 11 Diagram Alur Proses Enkripsi Berkas 72
Gambar 4. 12 Perbandingan Konten Berkas Asli dan Ciphertext Terenkripsi 73
Gambar 4. 13 *Chunk* Berkas yang Diunggah Secara Sekuensial ke Telegram 75
Gambar 4. 14 Bukti Berkas Terenkripsi Tidak Dapat Dibuka Tanpa Kunci 84

Gambar 4. 15 Bukti Mekanisme File Chunking pada Berkas 2,5 GB 86
Gambar 4. 16 Hasil Evaluasi Prototype oleh Pengguna..... 91

DAFTAR TABEL

Tabel 1. 1 Jadwal Kegiatan Penelitian. 7
Tabel 2. 1 Perbandingan Penelitian Sebelumnya 9
Tabel 2. 2 Pengertian Use case Diagram 29
Tabel 2. 3 Pengertian Class Diagram 30
Tabel 2. 4 Pengertian Activity Diagram 31
Tabel 2. 5 Pengertian Sequence Diagram 32
Tabel 2. 6 Notasi ERD 34
Tabel 3. 1 Perangkat Keras Klien & Pengembangan 35
Tabel 3. 2 Perangkat Keras Server (*Virtual Private Server*) 35
Tabel 3. 3 Perangkat Lunak 35
Tabel 3. 4 Pengertian Actor Usecase 44
Tabel 3. 5 Pengertian Actor Activity Diagram 46
Tabel 3. 6 Pengertian Actor Activity Diagram Upload File 48
Tabel 3. 7 Activity Diagram Download File 50
Tabel 3. 8 Metode Pengolahan dan Analisis Data 63
Tabel 4. 1 Hasil Pengujian Fungsional Skenario Autentikasi 76
Tabel 4. 2 Hasil Pengujian Fungsional Skenario Manajemen Berkas 79
Tabel 4. 3 Hasil Pengujian Fungsional Skenario Manajemen Folder 81
Tabel 4. 4 Hasil Verifikasi Integritas SHA-256 83
Tabel 4. 5 Hasil Pengujian Waktu Enkripsi dan Dekripsi AES-256-CBC 85
Tabel 4. 6 Hasil Pengujian Total Waktu Operasi Unggah dan Unduh 86
Tabel 4. 7 Hasil Pengujian Fungsionalitas Sistem 88
Tabel 4. 8 Hasil Pengujian Performa Sistem 89
Tabel 4. 9 Hasil Pengujian Kemudahan Antarmuka (Usability) 90

DAFTAR ISTILAH DAN SINGKATAN

No	Istilah dan Singkatan	Keterangan
1	API	*Application Programming Interface*
2	Bot API	*Bot Application Programming Interface*
3	AES-256	*Advanced Encryption Standard 256-bit*
4	SHA-256	*Secure Hash Algorithm 256-bit*
5	CBC	*Cipher Block Chaining*
6	IV	*Initialization Vector*
7	UI	*User Interface*
8	*Web UI*	*Web User Interface*
9	REST API	*Representational State Transfer Application Programming Interface*
10	JSON	*JavaScript Object Notation*
11	JWT	*JSON Web Token*
12	SQL	*Structured Query Language*
13	NIST	*National Institute of Standards and Technology*
14	IDC	*International Data Corporation*
15	RAM	*Random Access Memory*
16	SSD	*Solid State Drive*
17	CPU	*Central Processing Unit*
18	HTML	*HyperText Markup Language*
19	CSS	*Cascading Style Sheets*
20	R&D	*Research and Development*
21	MTProto	*Mobile Transport Protocol*
22	PBKDF2	*Password-Based Key Derivation Function 2*
23	CORS	*Cross-Origin Resource Sharing*

24	OWASP	*Open Web Application Security Project*
25	Pyrogram	*Library Python* untuk komunikasi dengan *server* Telegram melalui protokol MTProto
26	TgCrypto	*Library* akselerasi enkripsi native untuk protokol MTProto
27	Gunicorn	*Green Unicorn*
28	WSGI	*Web Server Gateway Interface*
29	I/O	*Input/output*

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi informasi dan aktivitas digital yang pesat memicu peningkatan volume data secara signifikan. Kondisi ini menuntut adanya solusi *cloud storage* yang efisien, fleksibel, dan berkapasitas besar. Saat ini, layanan *public cloud storage* komersial seperti Google Drive, Dropbox, dan OneDrive menawarkan kemudahan akses layaknya *virtual file manager*. Namun, layanan tersebut memiliki batasan kapasitas gratis yang ketat umumnya 5 sampai 15 GB dan mengenakan biaya langganan yang mahal untuk kebutuhan penyimpanan berskala besar. Menurut (Waranggani, 2023), *International Data Corporation* (IDC), sebuah perusahaan riset pasar berskala global, memperkirakan bahwa nilai pasar sektor penyimpanan *cloud virtual* di Indonesia telah mencapai 933,63 juta dolar AS pada tahun 2023. Angka ini menunjukkan kenaikan sebesar 25% dibandingkan dengan 747,15 juta dolar AS pada tahun 2022. Di sisi lain, membangun infrastruktur *server* penyimpanan secara mandiri juga bukan solusi ideal bagi pengguna berskala kecil karena membutuhkan investasi perangkat keras dan biaya pemeliharaan yang tinggi.

Guna mengatasi permasalahan tersebut secara komprehensif, penelitian ini mengusulkan perancangan sistem *Private Cloud storage* yang mengintegrasikan antarmuka *Web UI*, protokol MTProto, dan algoritma kriptografi AES-256. Penelitian (Tandy et al., 2020) mengemukakan bahwa teknologi *cloud* sangat bermanfaat bagi pengguna karena menghilangkan kebutuhan untuk menginvestasikan dana besar dalam perangkat lunak dan aplikasi data, serta pemeliharaan perangkat. *Web UI* dirancang sebagai *virtual file manager* agar pengguna dapat mengelola berkas secara intuitif tanpa berinteraksi dengan antarmuka *chat* Telegram. Menurut (*Telegram Users by Country*, 2026), jumlah unduhan aplikasi Telegram pada tahun 2023 mencapai 257,8 juta unduhan di

beberapa negara utama, dengan India menempati posisi pertama sebanyak 83,8 juta unduhan, disusul Rusia 35,1 juta, Amerika Serikat 29,9 juta, dan Indonesia 24,3 juta unduhan.

Kendala batasan ukuran *file* sebesar 50 MB diatasi menggunakan *library* Pyrogram, yaitu sebuah *library* Python yang berkomunikasi langsung dengan *data center* Telegram melalui protokol biner *native* MTProto. Pendekatan ini secara drastis meningkatkan batas unggahan menjadi 2 GB per berkas secara efisien, tanpa memerlukan infrastruktur *server* penyimpanan tambahan, karena seluruh beban penyimpanan data dialihkan sepenuhnya ke pusat data Telegram. Sementara itu, untuk berkas yang ukurannya melampaui batas operasional sistem yang di atur 1,8 GB untuk memberikan margin enkripsi, sistem secara otomatis menerapkan teknik *file chunking* di sisi *backend* untuk memecah dan menyatukan berkas tanpa campur tangan pengguna. Sebagai catatan arsitektural, sistem ini berinteraksi dengan infrastruktur Telegram menggunakan identitas Akun Bot. Namun, penelitian ini secara tegas tidak menggunakan antarmuka *Telegram HTTP Bot API* konvensional yang memiliki batasan ukuran unggah maksimal 50 MB, melainkan bertindak sebagai *MTProto Client* menggunakan pustaka Pyrogram. Pendekatan teknis ini memungkinkan sistem untuk membangun *direct socket connection* ke pusat data Telegram guna memanfaatkan infrastruktur penyimpanan dengan batasan unggah yang jauh lebih besar.

Pada aspek privasi dan integritas data, sistem menerapkan algoritma kriptografi AES-256 dan fungsi *hash SHA-256*. Penelitian (Surya et al., 2025) menunjukkan bahwa AES-256 memberikan waktu enkripsi atau dekripsi yang lebih efisien dibandingkan RSA dan ECC pada *file* berukuran besar. Setiap berkas atau segmen *chunk* dienkripsi secara lokal di sisi *backend* menggunakan algoritma AES-256-CBC, sementara aliran transmisi datanya menuju infrastruktur Telegram diakselerasi kinerjanya oleh *library* TgCrypto khusus pada lapisan protokol jaringan MTProto. Sebelum dikirimkan ke Telegram. Dengan demikian, data yang tersimpan di infrastruktur platform selalu dalam kondisi terenkripsi. Basis data lokal berbasis SQLite hanya difungsikan untuk menyimpan *metadata* berkas dan

parameter enkripsi, sehingga jejak data sensitif di sisi *server* aplikasi dapat diminimalkan. AES-256 dipilih karena merupakan bagian dari AES yang ditetapkan sebagai standar kriptografi oleh NIST untuk melindungi data elektronik, dan diakui secara luas sebagai algoritma enkripsi simetris yang paling kokoh terhadap ancaman kriptanalisis saat ini (NIST et al., 2001).

Melalui pendekatan pengembangan perangkat lunak metode *Prototyping* yang iteratif, sistem yang dihasilkan diharapkan mampu menghadirkan solusi penyimpanan *file* berkapasitas besar dan berbiaya infrastruktur rendah. Penelitian yang berjudul "Penerapan Metode Prototyping Pada Perancangan Private *Cloud storage* Berbasis Telegram Bot Melalui Protokol MTProto Dengan Algoritma Kriptografi AES-256" ini bertujuan menciptakan sistem yang mudah dikelola sekaligus memberikan jaminan keamanan privasi data yang kokoh di setiap lapisan penyimpanan.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, terdapat beberapa persoalan teknis yang harus diselesaikan. Oleh karena itu, rumusan masalah dalam penelitian ini dirumuskan melalui pertanyaan berikut:

1. Bagaimana merancang arsitektur sistem *Private Cloud storage* yang mengintegrasikan *Web UI* sebagai *virtual file manager* dengan protokol MTProto melalui *library* Pyrogram sebagai mekanisme komunikasi utama ke *backend* penyimpanan Telegram menggunakan metode Prototyping?

2. Bagaimana menerapkan algoritma AES-256 untuk enkripsi dan fungsi *hash SHA-256* untuk verifikasi integritas berkas secara lokal di sisi *backend* sebelum data diunggah ke infrastruktur Telegram guna meningkatkan keamanan data pengguna?

3. Bagaimana mengatasi batasan ukuran unggah maksimal pada *server* Telegram dengan mengimplementasikan teknik *file chunking* melalui protokol MTProto?

1.3. Tujuan Penelitian

Sebagai upaya untuk menjawab berbagai tantangan yang telah dirumuskan pada bagian sebelumnya, perancangan dan pengembangan sistem dalam penelitian ini memiliki beberapa tujuan spesifik, yaitu:

1. Merancang dan membangun arsitektur sistem Private *Cloud storage* yang mengintegrasikan *Web UI* sebagai *virtual file manager* dengan protokol MTProto melalui *library* Pyrogram sebagai mekanisme komunikasi utama ke *backend* penyimpanan Telegram menggunakan metode Prototyping sebagai pendekatan pengembangan yang iteratif.
2. Menerapkan algoritma AES-256 pada mekanisme enkripsi dan dekripsi berkas serta fungsi *hash SHA-256* untuk verifikasi integritas file secara lokal di sisi *backend* sebelum dan sesudah data dikirimkan ke infrastruktur Telegram.
3. Mengoptimalkan kemampuan sistem dalam menangani berkas berukuran besar secara efisien melalui pemanfaatan protokol MTProto via *library* Pyrogram dan penerapan teknik *file chunking* otomatis untuk berkas yang melampaui batas yang telah ditentukan, sehingga sistem mampu mengelola berkas dengan ukuran besar jauh melampaui batas layanan *cloud storage* konvensional.

1.4. Batasan Masalah

Agar arah dan fokus penelitian tetap terpusat pada pemecahan masalah utama, serta untuk menjaga ruang lingkup pengembangan sistem agar tetap terukur dan terarah secara teknis, maka batasan masalah dalam penelitian ini ditetapkan sebagai berikut:

1. Antarmuka yang disediakan kepada pengguna akhir dibangun sepenuhnya berbasis *Web UI* dan dirancang menyerupai antarmuka *virtual file manager*, tidak dikembangkan aplikasi *mobile native* atau antarmuka berbasis desktop yang terpisah.
2. Media penyimpanan fisik memanfaatkan sepenuhnya infrastruktur *server* Telegram melalui *library* Pyrogram yang berkomunikasi langsung dengan

server Telegram via protokol MTProto, bukan melalui Public Telegram Bot API dengan *endpoint* HTTP yang memiliki batas unggahan 50 MB per berkas.

3. Aspek keamanan data difokuskan pada proses enkripsi dan dekripsi berkas menggunakan AES-256 serta verifikasi integritas *file* menggunakan SHA-256 pada lapisan *backend*. Keamanan jaringan dan aspek di luar *application layer* tidak dibahas secara mendalam.
4. Teknik *file chunking* otomatis diterapkan pada berkas yang melampaui ukuran 1,8 GB dengan memecahnya menjadi beberapa *chunks* berukuran maksimal 1,8 GB. Pemotongan pada angka tersebut berfungsi sebagai batas aman guna mengakomodasi *overhead* enkripsi AES dan *metadata*, sehingga berkas tidak membentur batas absolut 2 GB dari protokol MTProto. Berkas dengan ukuran di bawah 1,8 GB akan diunggah secara utuh.
5. Basis data lokal hanya digunakan untuk menyimpan *metadata* berkas, meliputi nama *file*, ukuran, jumlah dan urutan *chunk*, serta parameter kriptografi yang diperlukan untuk proses dekripsi, konten berkas tidak disimpan di basis data lokal.
6. Sistem ini dirancang dengan arsitektur Multi-User *terdesentralisasi*, di mana setiap pengguna diwajibkan memiliki dan menginput kredensial API Telegram serta Chat ID pribadi milik mereka sendiri. Sistem tidak menyediakan infrastruktur API pusat untuk mencegah kebocoran privasi antar pengguna.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat yang signifikan bagi dua kelompok utama, yaitu:

1.5.1. Manfaat Teoritis

a. Memberikan kontribusi ilmiah pada bidang Teknik Informatika dalam sub-ranah keamanan data khususnya dalam kriptografi AES-256, arsitektur sistem penyimpanan terdistribusi berbasis Bot API, serta rekayasa perangkat lunak dengan metode Prototyping.
b. Memperkaya referensi akademis mengenai arsitektur pemanfaatan infrastruktur *platform chatting* khususnya melalui protokol MTProto via *library*

Pyrogram sebagai *backend* penyimpanan alternatif berbiaya rendah dengan kapasitas besar.

c. Menyediakan kajian teknis mengenai penggabungan dua mekanisme penanganan file besar komunikasi langsung via Pyrogram MTProto yang mendukung unggahan hingga 2 GB per berkas dan teknik *file chunking* dalam satu arsitektur sistem yang kohesif dan terintegrasi.
d. Menjadi referensi bagi penelitian lanjutan yang dapat mengembangkan sistem serupa, misalnya dengan menambahkan fitur sinkronisasi *real-time*, berbagi berkas terenkripsi antar pengguna, atau penggunaan algoritma enkripsi asimetris untuk manajemen kunci.

1.5.2. Manfaat Praktis

a. Menyediakan solusi penyimpanan *file* berkapasitas masif dan berbiaya infrastruktur minimal dengan mengandalkan *server* Telegram sebagai medium penyimpanan, sehingga pengguna terbebas dari kebutuhan investasi perangkat keras penyimpanan berskala besar.
b. Memberikan antarmuka pengelolaan *file* yang intuitif dan terstruktur melalui *Web UI* berbasis *virtual file manager*, sehingga kompleksitas teknis interaksi dengan Telegram Bot API tersembunyi sepenuhnya dari pengguna akhir.
c. Menjamin privasi dan keamanan data pengguna secara signifikan melalui enkripsi AES-256 menyeluruh setiap berkas dan setiap *chunk* dienkripsi sebelum meninggalkan *server* aplikasi sehingga meminimalkan risiko akses tidak sah.
d. Menghilangkan hambatan teknis batasan ukuran file secara bertingkat, Pyrogram MTProto menangani berkas hingga 2 GB per berkas tanpa membebani kapasitas penyimpanan *server* lokal**,** sementara *file chunking* otomatis menangani berkas di atas batas tersebut sehingga pengguna dapat menyimpan berkas berukuran besar jauh melampaui layanan *cloud storage* komersial pada umumnya.

1.6. Jadwal Penelitian

Penelitian ini direncanakan berlangsung selama 6 bulan, mulai Februari 2026 hingga Juli 2026. Jadwal pelaksanaan setiap kegiatan penelitian disajikan pada tabel berikut.

Tabel 1. 1 Jadwal Kegiatan Penelitian.

No	Kegiatan Penelitian	Bulan Februari	Bulan Maret	Bulan April	Bulan Mei	Bulan Juni	Bulan Juli
1	Penyusunan Proposal & Studi Literatur	■	■				
2	Analisis Kebutuhan Sistem	■	■				
3	Perancangan Sistem *Prototype*		■	■			
4	Implementasi *Backend* & Pyrogram MTProto			■	■		
5	Implementasi Enkripsi AES-256 & SHA-256				■	■	
6	Implementasi *File chunking*				■	■	
7	Pengujian & Evaluasi *Prototype*				■	■	■
8	Revisi & Penyempurnaan Sistem				■	■	■
9	Penyusunan Laporan Akhir					■	■

Bagian yang di arsir menunjukkan kegiatan yang dilaksanakan pada bulan yang bersangkutan. Jadwal bersifat tentatif dan dapat disesuaikan dengan kebutuhan aktual selama proses penelitian berlangsung, sepanjang tidak melampaui batas waktu penelitian yang telah ditetapkan.

1.7 Sistematika Penulisan

Laporan tugas akhir ini disusun secara sistematis ke dalam lima bab yang saling berkesinambungan, dengan rincian sebagai berikut:

BAB I PENDAHULUAN

Menguraikan latar belakang masalah, alasan pemilihan judul, rumusan dan batasan masalah, tujuan dan manfaat penelitian, serta jadwal dan sistematika penulisan Tugas Akhir.

BAB II TINJAUAN PUSTAKA

Membahas kajian pustaka dan konsep-konsep teoritis yang menjadi fondasi utama dalam memecahkan masalah dan mendukung jalannya penelitian.

BAB III METODOLOGI PENELITIAN

Menjelaskan secara rinci kerangka berpikir, metode, dan prosedur penelitian yang diterapkan, beserta tahapan perancangan sistem yang akan dibangun.

BAB IV HASIL DAN PEMBAHASAN

Memaparkan hasil akhir dari penelitian yang telah dilakukan, khususnya yang berkaitan dengan implementasi dan proses pembuatan aplikasi.

BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan akhir yang ditarik dari keseluruhan pembahasan penelitian, serta saran dan rekomendasi yang berguna untuk pengembangan aplikasi ke depannya.

BAB II
TINJAUAN PUSTAKA

2.1 *State of The Art*

Untuk menegaskan keaslian dan posisi kebaruan penelitian ini di antara penelitian-penelitian terdahulu yang relevan, berikut disajikan tabel perbandingan yang merangkum sejumlah studi yang memiliki keterkaitan topik dengan penelitian ini, baik dari sisi konsep *Private Cloud storage*, pemanfaatan Telegram Bot API, penerapan algoritma kriptografi AES-256, teknik *file chunking*, maupun pengembangan antarmuka berbasis web.

Tabel 2. 1 Perbandingan Penelitian Sebelumnya

No	**Peneliti (Tahun)**	**Judul Penelitian**	**Objek Penelitian**	**Perbandingan yang Dijadikan Alasan Tinjauan Penelitian**
1	(Masyhur et al., 2021) ISSN**:** 2776-8961 Volume (Issue): 1 (2) Halaman (Hal): 31 – 38 DOI**:** https://doi.org/10.24252/shift.v1i2.15	Studi Literatur Keamanan dan Privasi Data Sistem *Cloud computing* Pada Platform Google Drive	Platform Google Drive sebagai implementasi *Cloud computing* berbasis PaaS.	Penelitian ini menunjukkan *cloud* seperti Google Drive menggunakan enkripsi AES-256, namun masih terbatas tanpa kontrol penuh pengguna dan pengelolaan *file* skala besar.
2	(Surya et al., 2025) ISSN: 2527-4864 Volume (Issue): 11 (2)	Cryptographic *Framework* For *Cloud*-Based Document	Dokumen penting berbasis *cloud* seperti SK, sertifikat, ijazah, dan dokumen	Penelitian ini mengusulkan AES-256 dan SHA-256 untuk keamanan dokumen *cloud*

No	Peneliti (Tahun)	Judul Penelitian	Objek Penelitian	Perbandingan yang Dijadikan Alasan Tinjauan Penelitian
	Halaman (Hal): 535 – 549 DOI: https://doi.org/10.33480/jitk.v11i2.7132	Storage Using Aes-256 And Sha-256 Hybrid Systems	strategis yang disimpan di platform Google Drive, OneDrive dan Mega *Cloud*.	dengan enkripsi, integritas akurat, dan efisiensi tinggi pada *file* besar.
3	(NIST et al., 2001) ISSN: - Volume (Issue): - Halaman (Hal): - URL: https://www.nist.gov/publications/advanced-encryption-standard-aes	Advanced Encryption Standard (AES)	Algoritma Enkripsi AES Sebagai Standar Keamanan Data Untuk Melindungi Informasi Digital	AES dalam FIPS 197 menggantikan DES yang tidak aman, dengan performa dan efisiensi lebih baik.
4	(Arifa et al., 2024) ISSN: 2777-001X Volume (Issue): 4 (1) Halaman (Hal): 8 – 12	Implementasi Aplikasi Enkripsi Dekripsi *File* Dokumen Menggunakan Algoritma Aes Pada *Cloud computing* (Studi Kasus Prodi	Objek penelitian adalah *file* dokumen digital yang diamankan menggunakan algoritma AES dalam sistem *Cloud*	Penelitian ini menunjukkan AES meningkatkan keamanan data *cloud* secara efektif, namun masih terbatas

No	Peneliti (Tahun)	Judul Penelitian	Objek Penelitian	Perbandingan yang Dijadikan Alasan Tinjauan Penelitian
	URL: https://doi.org/10.30811/jaise.v4i1.5382	Teknologi Rekayasa Komputer Jaringan)	*computing* (studi kasus pada lingkungan akademik/ program studi).	pada implementasi dasar.
5	(Bintang et al., 2025) ISSN: 2808-4519 Volume (Issue): 4 (3) Halaman (Hal): 2208 – 2214 DOI**:** https://doi.org/10.59934/jaiea.v4i3.1136	Telegram BOT Application Development Integration with Google Sheets for Sending Service Reporting	Data Laporan Pengiriman melalui Telegram bot Api	Fokus pada integrasi Telegram untuk pengelolaan data teks/laporan. Belum mengakomodasi penyimpanan berkas berukuran besar, enkripsi data, dan teknik *file chunking* yang menjadi fokus pada sistem yang diusulkan.
6	(W. P. C. Wijaya & Achlaq, 2023) ISSN: 2598-828X Volume:7 (6) Hal: 3152 – 3159	Implementasi Api Bot Telegram Untuk Sistem Notifikasi Librenms Pada Perusahaan Blip Integrator	Objek penelitian adalah sistem monitoring jaringan (network monitoring system) pada	Penelitian ini menunjukkan integrasi Telegram Bot API meningkatkan efisiensi monitoring

No	Peneliti (Tahun)	Judul Penelitian	Objek Penelitian	Perbandingan yang Dijadikan Alasan Tinjauan Penelitian
	DOI: https://doi.org/10.36040/jati.v7i6.7991		perusahaan BLiP	melalui notifikasi *real-time*, namun masih terbatas tanpa penyimpanan dan keamanan data.
7	(Firmansyah et al., 2025) ISSN: 3032-4696 Volume (Issue): 2 (1) Halaman (Hal): 81 – 96 DOI**:** https://doi.org/10.70248/jdaics.v2i1.1804	Implementasi Algoritma Keamanan Data Berbasis Enkripsi Pada Platform *Cloud computing* Untuk Pembelajaran Di Smk Cipta Insani Mandiri	Objek penelitian adalah data pembelajaran digital (dokumen, nilai siswa, informasi pribadi) yang diamankan menggunakan algoritma AES pada platform Learning Management System (LMS) berbasis *cloud*.	Penelitian ini menunjukkan AES meningkatkan keamanan data *cloud*, namun masih terbatas tanpa distribusi dan optimasi *file*.
8	(Manogar & Abirami, 2022) ISSN: - Volume (Issue): - Halaman (Hal): -	*A smart hybrid content defined chunking algorithm for data deduplication in cloud storage*	data multimedia dan *virtual machine* yang disimpan dalam *cloud storage system* berbasis *deduplication*	Penelitian ini menunjukkan CDC dan OPC meningkatkan efisiensi dan performa *cloud*, namun masih terbatas tanpa integrasi

No	Peneliti (Tahun)	Judul Penelitian	Objek Penelitian	Perbandingan yang Dijadikan Alasan Tinjauan Penelitian
	DOI: https://doi.org/10.21203/rs.3.rs-376128/v1			keamanan data dan platform penyimpanan alternatif.
9	(Hillalia & Asri, 2026) ISSN: 2828-7924 Volume (Issue): 5 (2) Halaman (Hal): 165 – 172 DOI**:** https://doi.org/10.55338/justikpen.v5i2.425	Implementasi Algoritma Enkripsi AES-256 Pada Data Penggajian: Studi Kasus PT Matahati Bermakna Indonesia	Data penggajian karyawan pada sistem berbasis web yang diamankan menggunakan enkripsi AES-256.	Penelitian ini menunjukkan AES-256 mengamankan data sensitif, namun masih terbatas pada sistem internal tanpa integrasi *cloud*.
10	(G. Wijaya, 2022) ISSN:2776 – 5644 Volume (Issue): 2 (1) Halaman (Hal): 471 – 478 DOI: -	*Analysis Of Cloud computing Infrastructure System WithNist Standard Cloud computing Standards Roadmap*	Infrastruktur *Cloud computing on-premise*	Penelitian ini menganalisis *cloud* sesuai standar NIST, namun belum mencakup enkripsi end-to-end dan pengelolaan *file* skala besar.

No	Peneliti (Tahun)	Judul Penelitian	Objek Penelitian	Perbandingan yang Dijadikan Alasan Tinjauan Penelitian
11	(Rakesh Suryawanshi et al., 2023) ISSN: 2581-9429 Volume (Issue): 3 (1) Halaman (Hal): 589 – 593 DOI**:** https://doi.org/10.48175/ijarsct-9465	*Personal Cloud Storage using Raspberry PI*	Cloud Storage	penelitian ini sama sekali tidak membutuhkan penyediaan *har dware storage* mandiri, melainkan merekayasa infrastruktur *server* Telegram yang sudah ada sebagai media penyimpanan utama.
12	(Naufa, Faishal Muhammad, 2026) ISSN: - Volume (Issue): - Halaman (Hal): - DOI**:** -	Penerapan Metode Prototyping Pada Perancangan Private *Cloud storage* berbasis Telegram Bot Melalui Protokol MTProto Dengan Algoritma Kriptografi AES-256	Telegram	Menjadikan Telegram sebagai Private *Cloud*e Storage dengan Keamanan Kriptografi AES-256.

2.2 Penelitian Terdahulu

Penelitian terdahulu digunakan untuk mengetahui perkembangan studi yang relevan, menemukan *research gap*, serta menentukan posisi penelitian ini. Beberapa penelitian yang berkaitan dengan topik *private cloud storage*, keamanan

data, Telegram Bot API, teknik *file chunking*, dan *cloud computing* telah dilakukan sebelumnya.

(Masyhur et al., 2021) meneliti keamanan dan privasi data pada Google Drive sebagai layanan *cloud storage* publik. (Surya et al., 2025) membahas penerapan AES-256 dan SHA-256 untuk keamanan dokumen berbasis *cloud*. (Arifa et al., 2024) mengembangkan aplikasi enkripsi file menggunakan AES pada lingkungan *cloud computing*. (Firmansyah et al., 2025) juga menerapkan enkripsi pada platform *cloud* pembelajaran. (Hillalia & Asri, 2026) mengimplementasikan AES-256 pada sistem penggajian berbasis web. Penelitian-penelitian tersebut menunjukkan bahwa algoritma AES efektif dalam menjaga kerahasiaan data, namun belum mengarah pada integrasi Telegram sebagai media penyimpanan.

(Bintang et al., 2025) mengembangkan integrasi Telegram Bot API dengan Google Sheets untuk pelaporan layanan, sedangkan (W. P. C. Wijaya & Achlaq, 2023) memanfaatkan Telegram Bot API sebagai media notifikasi sistem monitoring jaringan. Kedua penelitian tersebut membuktikan Telegram efektif sebagai media komunikasi dan integrasi sistem, tetapi belum digunakan sebagai sistem penyimpanan file.

(Manogar & Abirami, 2022) meneliti teknik *content defined chunking* untuk meningkatkan efisiensi penyimpanan dan performa *cloud storage*. Namun penelitian tersebut berfokus pada deduplikasi data dan belum dikombinasikan dengan enkripsi maupun Telegram. Penelitian (G. Wijaya, 2022), menganalisis infrastruktur *cloud computing* berdasarkan standar NIST, definisi, karakteristik, model layanan, dan implementasi *cloud computing*. referensi ini menjadi dasar teoritis penelitian, khususnya pada konsep *private cloud*.

Terkait dengan pengembangan infrastruktur *cloud* mandiri berskala personal, penelitian internasional telah dilakukan oleh (Rakesh Suryawanshi et al., 2023) yang merancang bangun *Personal Cloud Storage* menggunakan Raspberry Pi. Penelitian tersebut membuktikan bahwa ketergantungan pada *cloud*

storage komersial dapat diatasi dengan membangun *server* mini di rumah. Namun, pendekatan tersebut memiliki kelemahan mendasar yaitu pengguna masih harus membeli *hardware* fisik dan menanggung risiko kerusakan perangkat keras keras (*hardware failure*). Mengambil sudut pandang yang berbeda, penelitian ini menawarkan solusi *Private Cloud Storage* yang jauh lebih praktis dan berbiaya infrastruktur nol, karena media penyimpanannya membonceng (*piggybacking*) pada *server* Telegram yang sudah mapan, sehingga pengguna terbebas dari risiko kegagalan perangkat keras fisik.

Berdasarkan penelitian terdahulu, sebagian besar studi masih berfokus pada satu aspek tertentu, seperti keamanan data, Telegram Bot API, teknik *chunking*, atau infrastruktur *cloud*. Belum ditemukan penelitian yang mengintegrasikan seluruh komponen tersebut dalam satu sistem terpadu. Oleh karena itu, penelitian ini menawarkan perancangan sistem *Private Cloud storage* berbasis Telegram Bot API yang dikombinasikan dengan enkripsi AES-256, verifikasi integritas SHA-256, teknik *file chunking* otomatis, serta antarmuka *Web UI* menggunakan metode *Prototyping*.

2.3 Kajian Teori

2.3.1 *Private Cloud storage* dan *Virtual file manager*

Cloud storage merupakan model penyimpanan data di mana berkas dan informasi disimpan pada *server* jarak jauh yang dapat diakses melalui jaringan internet, menurut (Masyhur et al., 2021). Menurut (NIST et al., 2001), *cloud computing* merupakan model layanan yang memungkinkan akses jaringan secara mudah, nyaman, dan sesuai permintaan terhadap kumpulan sumber daya komputasi terkonfigurasi seperti *server*, penyimpanan, aplikasi, dan layanan lainnya yang dapat disediakan dengan cepat serta minim interaksi penyedia layanan. Berdasarkan model *deployment* nya, layanan *cloud storage* dapat dikategorikan menjadi dua jenis utama, yaitu *public cloud storage* dan *Private Cloud storage*. Pada *public cloud storage*, infrastruktur penyimpanan dikelola sepenuhnya oleh penyedia layanan pihak ketiga seperti Google Drive, Dropbox, atau OneDrive dan sumber

dayanya digunakan secara bersama-sama oleh berbagai pengguna. Dalam penerapannya, layanan *cloud* sangat erat kaitannya dengan perlindungan dan keamanan data. Karena teknologi *cloud* bergantung pada penyimpanan berbasis jaringan, jika langkah-langkah keamanan yang tepat tidak diterapkan selama pemrosesan atau transfer data, risiko serius dapat timbul (Masyhur et al., 2021).

Private Cloud storage, sebaliknya, adalah sistem penyimpanan yang dikelola secara mandiri oleh individu atau organisasi pemiliknya di atas infrastruktur yang mereka kendalikan secara penuh. Model ini memberikan kemandirian atas data yang disimpan, eliminasi ketergantungan pada penyedia pihak ketiga, dan fleksibilitas konfigurasi keamanan sesuai kebutuhan spesifik pengguna. Dalam konteks penelitian ini, konsep *Private Cloud storage* diwujudkan dengan memanfaatkan infrastruktur *server* Telegram sebagai media penyimpanan fisik yang dikendalikan melalui *token* bot pada *group chat*, sehingga tidak ada entitas pihak ketiga lain yang memiliki akses terhadap konten berkas yang disimpan.

Antarmuka sistem penyimpanan yang intuitif merupakan faktor krusial dalam menentukan kenyamanan pengguna. Konsep *virtual file manager* mengacu pada antarmuka pengelolaan berkas berbasis perangkat lunak yang menyimulasikan tampilan dan fungsionalitas *file manager* konvensional meliputi navigasi direktori, operasi unggah, unduh, pencarian berkas, dan penghapusan berkas namun disajikan melalui antarmuka berbasis web yang dapat diakses dari perangkat apa pun tanpa instalasi tambahan. Pendekatan ini diadopsi oleh layanan *cloud storage* komersial populer sebagai standar interaksi pengguna. Dalam penelitian ini, *Web UI* yang dirancang menyerupai antarmuka *virtual file manager* tersebut berfungsi sebagai lapisan presentasi yang menyembunyikan kompleksitas teknis komunikasi dengan Telegram Bot API dari pengguna akhir.

2.3.2. Telegram Bot API dan Protokol MTProto

Telegram Bot API adalah antarmuka pemrograman aplikasi yang disediakan secara resmi oleh Telegram, memungkinkan pengembang untuk membangun bot akun otomatis yang dapat berinteraksi dengan pengguna melalui pesan, menerima berkas, dan menjalankan perintah terprogram secara sistematis di perkuat dengan temuan dari (W. P. C. Wijaya & Achlaq, 2023), Telegram Bot memungkinkan siapa

saja untuk membuat bot dan dapat disesuaikan dengan kebutuhan pengguna, di mana setiap *bot* diidentifikasi oleh sebuah *token* unik yang berfungsi sebagai kredensial autentikasi. Selain itu, Telegram juga dapat dimanfaatkan sebagai penghubung antara pengguna dengan sistem berbasis *cloud* melalui integrasi API.

Dalam penelitian ini, komunikasi dengan infrastruktur Telegram dilakukan melalui Pyrogram, sebuah *library* Python modern yang mengimplementasikan protokol MTProto secara langsung. Berbeda dengan Bot API yang membatasi ukuran unggah berkas hingga 50 MB, Pyrogram berkomunikasi langsung dengan *data center* Telegram melalui protokol MTProto, sehingga mampu mengunggah dan mengunduh berkas hingga 2 GB per file. Pyrogram mendukung operasi asinkron, manajemen sesi otomatis, dan penanganan *flood wait* bawaan. Klien Pyrogram diinisialisasi secara dinamis berdasarkan kredensial API masing-masing pengguna untuk mendukung arsitektur multi-user.

Selain itu, penelitian (Bintang et al., 2025) menunjukkan bahwa Telegram Bot dapat diintegrasikan dengan layanan *cloud* seperti Google Sheets untuk pengelolaan data secara *real-time*. Integrasi ini memperkuat peran Telegram sebagai antarmuka yang mampu berinteraksi dengan sistem penyimpanan berbasis *cloud*.

Pada implementasi standar, Public Telegram Bot API diakses melalui *endpoint HTTP REST api.telegram.org* dan memberlakukan batasan ukuran unggahan berkas sebesar 50 MB. Keterbatasan ini menjadi kendala signifikan dalam pengelolaan berkas berukuran besar. Untuk mengatasi batasan tersebut, terdapat dua pendekatan utama: (1) *Local* Telegram Bot API *Server*, yaitu implementasi *server* Bot API yang di-*hosting* secara mandiri oleh pengembang sehingga batas unggahan meningkat hingga 2 GB per berkas; dan (2) penggunaan *library* yang berkomunikasi langsung dengan *server* Telegram melalui protokol MTProto, yang memungkinkan pengiriman berkas hingga 2 GB.

MTProto adalah protokol komunikasi biner native yang dikembangkan oleh Telegram untuk komunikasi client-*server* yang cepat, scalable, dan aman (Miculan & Vitacolonna, 2023). Berbeda dengan antarmuka *HTTP REST* pada *Bot API* konvensional, MTProto beroperasi pada lapisan transport yang lebih rendah dan

menggunakan format biner yang lebih efisien sehingga mampu mengurangi *overhead* komunikasi data. Selain itu, MTProto mendukung pengiriman berkas hingga 2 GB per berkas secara native tanpa memerlukan *server* Bot API tambahan. Kemampuan tersebut menjadikan MTProto lebih sesuai digunakan pada sistem penyimpanan berkas berukuran besar dibandingkan pendekatan Bot API konvensional.

Dalam penelitian ini, pendekatan MTProto diimplementasikan menggunakan Pyrogram, yaitu *library* Python modern yang menyediakan antarmuka tingkat tinggi untuk berkomunikasi langsung dengan *server* Telegram melalui protokol MTProto. Pyrogram tetap menggunakan Bot *Token* sebagai kredensial autentikasi sehingga kompatibel dengan ekosistem bot Telegram, namun komunikasi dilakukan langsung ke *data center* Telegram tanpa melalui *server* perantara *HTTP* Bot API. *Library* pendamping TgCrypto digunakan untuk akselerasi proses enkripsi pada lapisan protokol MTProto, sementara sesi autentikasi disimpan secara lokal dalam file *session* berbasis SQLite agar tidak memerlukan proses re-autentikasi setiap kali sistem dimulai ulang.

Pemilihan Pyrogram MTProto dibandingkan dengan *Local* Telegram Bot API *Server* didasarkan pada beberapa pertimbangan teknis. Pertama, pendekatan ini menghilangkan kebutuhan akan *server* tambahan, sehingga membuat infrastruktur pengembangan dan penerapan menjadi lebih sederhana. Kedua, dengan terhubung langsung ke pusat data Telegram, maka waktu tanggap atau latensi menjadi lebih rendah. Ketiga, kapasitas unggahan file tetap sama, yaitu hingga 2 GB per file, tanpa perlu terkendala dengan urusan administrasi *server*. Terakhir, pendekatan ini juga membuat penggunaan sumber daya sistem lebih efisien karena tidak perlu menjalankan proses *server* terpisah secara paralel.

2.3.3. Teknik *File chunking*

File chunking merupakan teknik dalam rekayasa perangkat lunak yang membagi suatu berkas berukuran besar menjadi sejumlah *chunk* yang berukuran lebih kecil agar lebih mudah diproses, ditransfer, atau disimpan. Setiap *chunk* diperlakukan sebagai unit data independen yang dapat diunggah secara berurutan maupun paralel, kemudian digabungkan kembali menjadi berkas utuh saat proses

pengunduhan. Teknik ini umum digunakan pada sistem penyimpanan modern karena mampu meningkatkan efisiensi transfer data, mengurangi risiko kegagalan unggah, serta mempermudah proses pemulihan data ketika terjadi gangguan jaringan. Penelitian (Manogar & Abirami, 2022) menjelaskan bahwa proses *chunking* memiliki peran penting dalam optimalisasi penyimpanan *cloud*, khususnya pada sistem berskala besar.

Dalam implementasi sistem penyimpanan berbasis *cloud*, teknik *file chunking* terbagi menjadi beberapa pendekatan, yaitu *file-level chunking, fixed-size chunking*, dan *content-defined chunking. File-level chunking* memperlakukan seluruh *file* sebagai satu unit, sedangkan *fixed-size chunking* membagi *file* ke ukuran tetap. Adapun *content-defined chunking* membagi data berdasarkan karakteristik isi data, sehingga lebih adaptif terhadap perubahan struktur *file* dan lebih efektif dalam mendukung efisiensi penyimpanan. Menurut (Manogar & Abirami, 2022), metode *chunking* yang tepat dapat meningkatkan *throughput*, menurunkan waktu pemrosesan, serta mengurangi *metadata overhead* pada sistem *cloud storage*.

Dalam konteks penelitian ini, teknik *file chunking* diterapkan secara otomatis oleh sistem pada lapisan *backend* untuk berkas yang ukurannya melebihi batas operasional 1,8 GB yang didukung oleh protokol MTProto. Sistem akan secara otomatis menentukan jumlah dan ukuran optimal setiap *chunk*, mengunggah masing-masing segmen ke *server* Telegram sebagai berkas terpisah, dan menyimpan informasi urutan serta identitas setiap *chunk* pada basis data *metadata*. Pada saat berkas diunduh, sistem secara otomatis mengambil seluruh *chunk* berdasarkan *metadata* tersebut, mendekripsinya, dan menggabungkannya kembali menjadi berkas asli yang utuh sebelum dikirimkan kepada pengguna. Proses ini sepenuhnya transparan dari perspektif pengguna akhir, yang hanya berinteraksi dengan berkas secara keseluruhan melalui antarmuka *Web UI*.

Penggunaan teknik *file chunking* juga berpengaruh terhadap efisiensi proses enkripsi. Dengan mengenkripsi setiap *chunk* secara terpisah menggunakan kunci dan *Initialization Vector* (IV) yang berbeda, sistem tidak perlu memuat seluruh isi *file* ke dalam memori secara bersamaan. Pendekatan ini membantu mengurangi

penggunaan RAM *server* serta meningkatkan skalabilitas sistem ketika menangani *file* berukuran sangat besar. Konsep pemecahan data menjadi beberapa *chunk* juga telah digunakan pada sistem *cloud storage* untuk meningkatkan efisiensi pemrosesan data dan *throughput* sistem (Manogar & Abirami, 2022).

2.3.4. Kriptografi dan Algoritma AES-256

Kriptografi merupakan ilmu yang mempelajari teknik pengamanan informasi dengan cara mengubah data yang dapat dibaca menjadi bentuk yang tidak dapat dipahami pihak tidak berwenang melalui proses enkripsi, serta mengembalikannya kembali ke bentuk semula melalui proses dekripsi. Tujuan utama kriptografi adalah menjaga kerahasiaan, integritas, autentikasi, dan keaslian data dalam proses pertukaran informasi digital. Penerapan kriptografi saat ini banyak digunakan pada sistem keamanan data, komunikasi digital, dan layanan penyimpanan berbasis *cloud.* (Firmansyah et al., 2025) menjelaskan bahwa penggunaan kriptografi pada platform *cloud* mampu meningkatkan perlindungan data terhadap akses tidak sah.

Algoritma *Advanced Encryption Standard* merupakan standar enkripsi simetris yang ditetapkan oleh *National Institute of Standards and Technology* (NIST) Amerika Serikat melalui publikasi FIPS PUB 197 pada tahun 2001, menggantikan standar sebelumnya yaitu *Data Encryption Standard* (DES). AES beroperasi pada blok data berukuran tetap 128-bit dan mendukung tiga panjang kunci, yakni 128-bit, 192-bit, dan 256-bit. Varian dengan panjang kunci 256-bit, yang selanjutnya disebut AES-256, memberikan tingkat keamanan tertinggi di antara ketiganya dan menjadi pilihan utama untuk pengamanan data dengan klasifikasi sensitif.

AES-256 bekerja melalui 14 putaran transformasi matematis yang terdiri atas proses *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*. Ruang kunci AES-256 mencapai 2^{256} kemungkinan kombinasi, sehingga secara praktis sangat sulit ditembus menggunakan metode *brute-force* dengan teknologi komputasi saat ini. Penelitian (Arifa et al., 2024) juga menunjukkan bahwa algoritma AES mampu menjaga keamanan *file* dokumen pada sistem *Cloud computing* dengan kinerja transfer *file* yang baik.

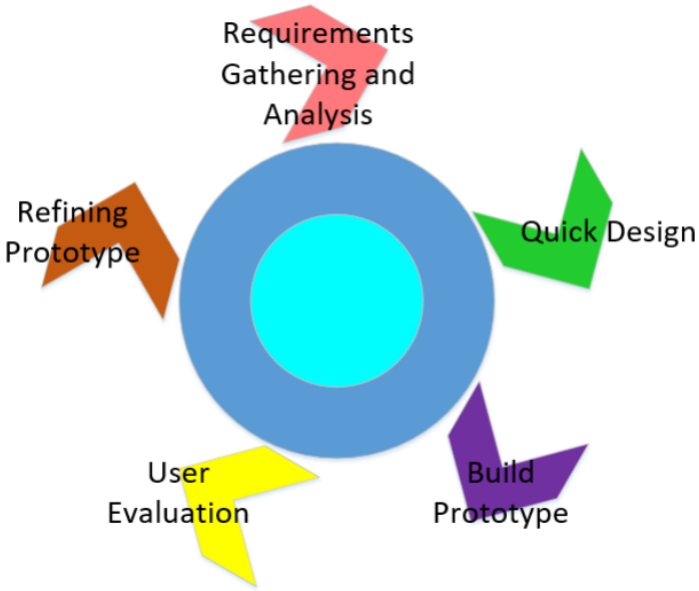
Dalam penelitian ini, AES-256 diimplementasikan menggunakan mode Cipher Block Chaining (CBC), yaitu setiap blok *plaintext* dioperasikan dengan blok *ciphertext* sebelumnya melalui proses XOR sebelum dienkripsi. Penggunaan *Initialization Vector* (IV) yang berbeda pada setiap proses enkripsi bertujuan menghasilkan *ciphertext* yang unik meskipun isi *file* sama, sehingga dapat mengurangi risiko analisis pola data. Proses enkripsi dan dekripsi dilakukan pada sisi *backend server* sebelum data dikirim maupun setelah data diterima dari infrastruktur Telegram, sehingga *file* dalam kondisi *plaintext* tidak keluar dari lingkungan sistem utama.

Pemilihan AES-256 dalam penelitian ini juga didukung oleh hasil penelitian (Surya et al., 2025) yang secara nyata membuktikan bahwa AES-256 memberikan performa enkripsi dan dekripsi yang lebih efisien dibandingkan algoritma RSA dan ECC pada berkas berukuran besar, menjadikannya pilihan yang paling optimal untuk sistem penyimpanan *file* yang menangani berkas dari skala kecil hingga berukuran beberapa gigabyte.

2.3.5. Metode Prototyping

Metode Prototyping adalah pendekatan pengembangan perangkat lunak yang bersifat iteratif dan inkremental, di mana sebuah versi awal sistem dibangun secara cepat untuk kemudian dievaluasi bersama pengguna, dan hasil evaluasi tersebut digunakan sebagai masukan untuk menyempurnakan sistem pada iterasi berikutnya. Penelitian (Abdussalaam & Oktaviani, 2020) menjelaskan bahwa Prototyping merupakan salah satu metode yang memungkinkan terjadinya interaksi selama proses pembuatan sistem.

Berbeda dengan model sekuensial seperti *Waterfall* yang mengharuskan kebutuhan sistem terdefinisi secara lengkap sebelum pengembangan dimulai, metode Prototyping mengakui bahwa kebutuhan pengguna seringkali baru teridentifikasi secara penuh setelah mereka berinteraksi langsung dengan sistem yang nyata.



Gambar 2. 1 Metode Prototyping
Source: (Putrawan & Khairi, 2023)

Siklus metode Prototyping secara umum terdiri dari lima tahapan yang berulang. Tahap pertama adalah pengumpulan kebutuhan, di mana peneliti dan pengguna bersama-sama mengidentifikasi kebutuhan dan spesifikasi sistem secara garis besar. Tahap kedua adalah perancangan cepat, yaitu pembuatan rancangan arsitektur dan antarmuka secara ringkas berdasarkan kebutuhan yang telah diidentifikasi. Tahap ketiga adalah pembangunan *prototype*, yakni implementasi versi awal sistem yang sudah dapat difungsikan meski belum sempurna. Tahap keempat adalah evaluasi *prototype* bersama pengguna, di mana umpan balik dikumpulkan secara sistematis terhadap fungsionalitas dan kemudahan penggunaan sistem. Tahap kelima adalah perbaikan dan penyempurnaan, di mana sistem diperbarui berdasarkan umpan balik yang diterima. Siklus dari tahap kedua hingga kelima kemudian diulang hingga sistem mencapai kualitas yang disepakati.

Metode Prototyping dipilih dalam penelitian ini karena kompleksitas teknis integrasi antarkomponen sistem yang melibatkan *Web UI*, koneksi Pyrogram MTProto ke *server* Telegram, mekanisme enkripsi AES-256, dan logika *file chunking* menuntut pendekatan pengembangan yang memungkinkan validasi fungsionalitas setiap komponen secara bertahap sebelum keseluruhan sistem dinyatakan selesai. Siklus evaluasi yang berulang memastikan bahwa setiap

potensi kegagalan desain atau inkonsistensi antarmuka dapat terdeteksi dan diperbaiki lebih awal, sehingga meminimalkan risiko revisi besar-besaran pada tahap akhir pengembangan. Selain itu, karakter interaktif antarmuka *Web UI* berbasis *virtual file manager* sangat bergantung pada pengalaman dan persepsi pengguna yang hanya dapat divalidasi melalui pengujian langsung sebuah kebutuhan yang secara inheren terpenuhi oleh siklus evaluasi iteratif dalam metode Prototyping.

2.3.6 Python

Menurut (Thoutam, 2021), Python merupakan bahasa pemrograman yang banyak digunakan dalam pengembangan aplikasi web karena didukung oleh berbagai framework yang mempermudah proses pembangunan aplikasi. Python dikenal dengan sintaksnya yang bersih dan mudah dipahami, sehingga banyak digunakan dalam pengembangan aplikasi web, komputasi ilmiah, kecerdasan buatan, hingga keamanan siber. Dalam penelitian ini, Python digunakan sebagai bahasa pemrograman utama untuk membangun seluruh logika backend sistem, meliputi implementasi REST API menggunakan framework Flask, pengelolaan berkas, proses enkripsi dan dekripsi menggunakan algoritma AES-256, serta komunikasi dengan server Telegram melalui library Pyrogram. Pemilihan Python didasarkan pada kemudahan pengembangan, ketersediaan pustaka pendukung yang lengkap, serta kemampuannya dalam membangun aplikasi backend yang ringan dan mudah dipelihara.

2.3.7 Flask

Menurut (Kellezi et al., 2021) , Flask merupakan *microframework* berbasis Python yang menyediakan komponen inti pengembangan aplikasi web serta mendukung berbagai ekstensi untuk memenuhi kebutuhan pengembangan sistem modern. Flask menggunakan arsitektur minimalis, di mana pengembang diberikan kebebasan untuk memilih komponen tambahan sesuai kebutuhan proyek tanpa dipaksa menggunakan struktur monolitik. Flask mendukung berbagai ekstensi pihak ketiga, seperti Flask-JWT-Extended untuk autentikasi berbasis JSON Web Token, Flask-CORS untuk menangani mekanisme *Cross-Origin Resource Sharing*,

serta Flask-Bcrypt untuk melakukan hashing kata sandi. Dukungan terhadap berbagai ekstensi tersebut memungkinkan Flask dikembangkan menjadi backend yang aman, modular, dan mudah dipelihara sesuai kebutuhan sistem. Dalam penelitian ini, Flask berfungsi sebagai inti backend yang menangani seluruh endpoint REST API untuk operasi CRUD berkas, autentikasi pengguna, proses enkripsi dan dekripsi menggunakan algoritma AES-256, serta orkestrasi proses *file chunking* dan komunikasi dengan server Telegram melalui library Pyrogram.

2.3.8. Gunicorn dan WSGI

Web *Server* Gateway Interface (WSGI) adalah spesifikasi standar yang mendefinisikan antarmuka komunikasi universal antara web *server* dan aplikasi web berbasis Python. Sebelum adanya WSGI, aplikasi Python sering kali dirancang hanya untuk bekerja dengan satu web *server* spesifik, sehingga membatasi fleksibilitas *deployment*. Dengan mengadopsi standar WSGI, aplikasi web Python seperti Flask dapat berjalan secara independen dan dieksekusi oleh berbagai WSGI *HTTP Server* di lingkungan produksi.

Gunicorn (Green Unicorn) merupakan salah satu implementasi Python WSGI *HTTP Server* berstandar industri yang didesain khusus untuk lingkungan UNIX/Linux. Menurut dokumentasi resmi, Gunicorn merupakan Python WSGI HTTP Server yang menggunakan model *pre-fork worker* sehingga mampu menangani banyak permintaan HTTP secara efisien pada lingkungan produksi. Arsitektur ini menjadikan Gunicorn sangat ringan, hemat sumber daya, dan memiliki performa tinggi dalam menangani beban kerja *concurrency* dibandingkan *server* bawaan *framework*. Oleh karena itu, Gunicorn umum digunakan sebagai lapisan perantara yang baik antara *reverse proxy* dan aplikasi web Python dalam fase *production*.

Secara konseptual, arsitektur komunikasi antara web klien, *web server*, dan *server* WSGI dapat diilustrasikan seperti pada Gambar 2.2. *Server* web bertugas menerima permintaan *HTTP* statis secara langsung dari peramban. Namun, ketika terdapat permintaan dinamis yang membutuhkan pemrosesan logika aplikasi, *server* web akan meneruskan permintaan tersebut ke *server* WSGI. *Server* WSGI kemudian bertindak sebagai penerjemah yang mengeksekusi kode Python dan

mengembalikan hasil halaman web kembali ke *server* web untuk dikirimkan ke pengguna.



Gambar 2. 2 Arsitektur Komunikasi WSGI *Server* dan Web *Server*
source: Penulis, dimodifikasi dari (*Makai, 2022*)

2.3.9 SQLite

SQLite merupakan sistem manajemen basis data relasional *Relational Database Management System* yang bersifat *serverless*, *self-contained*, dan *zero-configuration* sehingga cocok digunakan sebagai basis data *embedded* yang ringan dan efisien (Shi, 2023).Berbeda dengan RDBMS tradisional seperti MySQL atau PostgreSQL yang memerlukan proses *server* terpisah, SQLite menyimpan seluruh basis data dalam satu berkas tunggal di sistem berkas lokal. Karakteristik ini menjadikan SQLite sangat ideal untuk aplikasi yang membutuhkan basis data ringan tanpa *overhead* administrasi *server*. Dalam konteks penelitian ini, SQLite digunakan untuk menyimpan *metadata* berkas seperti nama file, ukuran, *hash SHA-256*, parameter kriptografi seperti kunci AES terenkripsi, IV, *salt*, serta data akun pengguna termasuk kredensial Telegram yang terenkripsi.

2.3.10 JSON Web *Token*

JSON Web Token (JWT) merupakan standar terbuka yang digunakan sebagai format token berbasis JSON untuk melakukan pertukaran informasi secara aman antara klien dan server. Menurut (Shatnawi et al., 2024), JSON Web Token merupakan mekanisme autentikasi berbasis token yang banyak digunakan pada aplikasi Python untuk mendukung proses autentikasi dan otorisasi secara aman. JWT terdiri dari tiga bagian yang dipisahkan oleh titik: *Header* (algoritma dan tipe *token*), *Payload* (klaim data), dan Signature (tanda tangan digital). Keunggulan utama JWT adalah sifatnya yang *stateless*, sehingga *server* tidak perlu menyimpan informasi sesi di memori atau basis data. Dalam penelitian ini, JWT diimplementasikan melalui *library* Flask-JWT-Extended dengan masa berlaku

24jam. Selain berfungsi sebagai bukti autentikasi, JWT juga berperan sebagai pembawa kunci sementara dalam arsitektur manajemen kunci non-persisten.

2.3.11 Bcrypt

Bcrypt adalah fungsi *hashing* kata sandi yang dirancang oleh Niels Provos dan David Mazières berdasarkan *cipher Blowfish*. Bcrypt termasuk dalam kategori *adaptive hash function*, yang berarti tingkat kesulitan komputasinya dapat disesuaikan melalui parameter *cost factor*. Semakin tinggi *cost factor*, semakin lama waktu yang dibutuhkan untuk menghitung *hash*, sehingga secara signifikan memperlambat serangan *brute-force*. Bcrypt juga secara otomatis menghasilkan *salt* acak internal untuk setiap operasi *hashing*, yang membuat dua kata sandi identik menghasilkan *hash* yang berbeda. Dalam penelitian ini, Bcrypt digunakan untuk meng-*hash* kata sandi pengguna sebelum disimpan ke basis data, sesuai dengan rekomendasi OWASP *Password Storage Cheat Sheet*. Menurut (Isnaini et al., 2025), algoritma Bcrypt mampu meningkatkan keamanan autentikasi pengguna melalui mekanisme *adaptive hashing* dan penggunaan *salt* unik pada setiap proses hashing sehingga lebih tahan terhadap serangan *brute-force*.

2.3.12 PBKDF2 (Password-Based Key Derivation Function 2)

Password-Based Key Derivation Function 2 (PBKDF2) merupakan algoritma *key derivation* yang digunakan untuk menghasilkan kunci kriptografi dari sebuah kata sandi. PBKDF2 bekerja dengan mengombinasikan kata sandi dan nilai *salt*, kemudian memprosesnya secara berulang menggunakan fungsi pseudorandom, seperti HMAC-SHA256, hingga menghasilkan kunci dengan panjang tertentu. Proses iterasi tersebut meningkatkan kompleksitas komputasi sehingga mampu memperkuat ketahanan kunci terhadap serangan *brute-force*. (Harahap et al., 2026) menegaskan bahwa menerapkan PBKDF2 sebagai mekanisme pembangkitan kunci AES-256 untuk meningkatkan kerahasiaan data pada sistem pengamanan berkas digital. Penggunaan jumlah iterasi yang tinggi dengan minimal 600.000 iterasi sesuai rekomendasi OWASP bertujuan untuk memperlambat serangan *brute-force* terhadap kunci enkripsi secara signifikan. Dalam penelitian ini, PBKDF2-HMAC-SHA256 digunakan untuk dua keperluan: (1) derivasi kunci enkripsi AES per-file

dengan *salt* acak 16-byte, dan (2) derivasi kunci utama untuk mengenkripsi kredensial Telegram pengguna tanpa perlu menyimpan kunci secara permanen di sisi *server*.

2.3.13 *Black-Box* Testing

Black-Box Testing adalah metode pengujian perangkat lunak yang mengevaluasi fungsionalitas sistem tanpa mengetahui atau memperhatikan struktur kode internal. Pengujian ini berfokus pada input dan output sistem, di mana penguji memberikan serangkaian masukan dan memverifikasi apakah keluaran yang dihasilkan sesuai dengan spesifikasi kebutuhan yang telah didefinisikan. Teknik-teknik yang umum digunakan dalam *Black-Box* Testing meliputi *Equivalence Partitioning*, *Boundary Value Analysis*, dan *Decision Table Testing*. Dalam penelitian ini, *Black-Box* Testing digunakan untuk memvalidasi seluruh fitur fungsional sistem, mulai dari proses registrasi, login, unggah berkas, unduh berkas, pengelolaan folder, hingga mekanisme penghapusan berkas secara permanen.

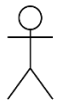
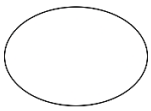
2.4. UML (*Unified Modeling Language*)

Unified Modeling Language (UML) adalah bahasa pemodelan standar yang digunakan secara luas dalam rekayasa perangkat lunak untuk menspesifikasikan, memvisualisasikan, membangun, dan mendokumentasikan artefak dari sebuah sistem perangkat lunak. UML menyediakan seperangkat notasi grafis yang membantu pengembang dalam merancang dan mengkomunikasikan desain sistem secara terstruktur. Dalam penelitian ini, UML digunakan untuk memodelkan arsitektur dan perilaku sistem Private *Cloud storage* melalui beberapa jenis diagram berikut.

2.4.1 *Use case Diagram*

Use case Diagram adalah diagram UML yang menggambarkan fungsionalitas sistem dari perspektif pengguna. Diagram ini menampilkan sekumpulan *use case*, aktor, serta hubungan antara aktor dan *use case* tersebut. *Use case Diagram* berguna untuk mengidentifikasi kebutuhan fungsional sistem pada tahap awal pengembangan dan membantu *stakeholder* memahami cakupan fitur yang akan dibangun.

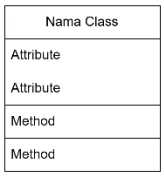
Tabel 2. 2 Pengertian *Use case Diagram*

No	Simbol	Nama	Keterangan
1		*Aktor*	Representasi pengguna atau entitas eksternal yang berinteraksi dengan sistem.
2		*Use case*	Fungsionalitas atau layanan yang disediakan oleh sistem kepada aktor.
3		*Asosiasi*	Hubungan komunikasi antara aktor dan *use case*.
4		*Use case generalization*	Hubungan hierarkis antara dua *use case*, di mana satu *use case* bertindak sebagai bentuk yang lebih umum dan *use case* lainnya menjadi bentuk yang lebih spesifik yang mewarisi perilaku dari fungsi umum tersebut.

2.4.2 *Class Diagram*

Class Diagram adalah diagram UML yang menggambarkan struktur statis dari sebuah sistem dengan menampilkan kelas-kelas, atribut, metode, serta hubungan antar kelas seperti asosiasi, generalisasi, dan dependensi. Diagram ini merupakan fondasi utama dalam perancangan berorientasi objek karena memetakan entitas-entitas utama beserta tanggung jawabnya di dalam sistem.

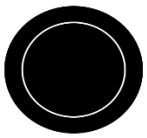
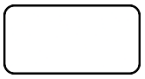
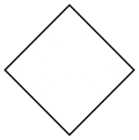
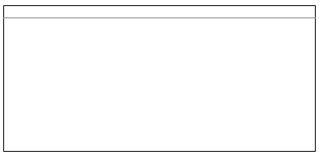
Tabel 2. 3 Pengertian *Class Diagram*

No	Simbol	Nama	Keterangan
1		*Class*	Representasi kelas yang terdiri dari nama kelas, atribut, dan metode.
2		*Directed Association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas lain.
3		*Composition*	Relasi *Composition* terhadap classtempat dia bergantung.

2.4.3 *Activity Diagram*

Activity Diagram adalah diagram UML yang memodelkan alur kerja atau aliran kontrol dari suatu proses bisnis maupun proses komputasional. Diagram ini menampilkan urutan aktivitas, keputusan, percabangan paralel, serta swimlane yang membagi tanggung jawab antar aktor atau komponen. *Activity Diagram* sangat efektif untuk memvisualisasikan proses-proses kompleks yang melibatkan banyak langkah dan kondisi percabangan.

Tabel 2. 4 Pengertian *Activity Diagram*

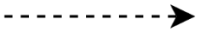
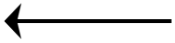
No	Simbol	Nama	Keterangan
1		*Initial Node*	Titik awal dari sebuah alur aktivitas.
2		*Activity*	Titik akhir dari sebuah alur aktivitas.
3		*Action*	Langkah-langkah dalam sebuah activity.
4		*Decision*	Titik percabangan berdasarkan kondisi tertentu.
7		*Swimlane*	Pembagian area tanggung jawab terhadap aktivitas yang terjadi.

2.4.4 *Sequence Diagram*

Sequence Diagram adalah diagram UML yang menggambarkan interaksi antar objek dalam urutan waktu. Diagram ini menampilkan garis hidup dari masing-masing objek dan pesan yang dipertukarkan di antara mereka secara kronologis. *Sequence Diagram* sangat berguna untuk merinci skenario eksekusi dari sebuah *use*

case, termasuk alur permintaan dan respons antar komponen sistem seperti antarmuka pengguna, *server*, dan basis data.

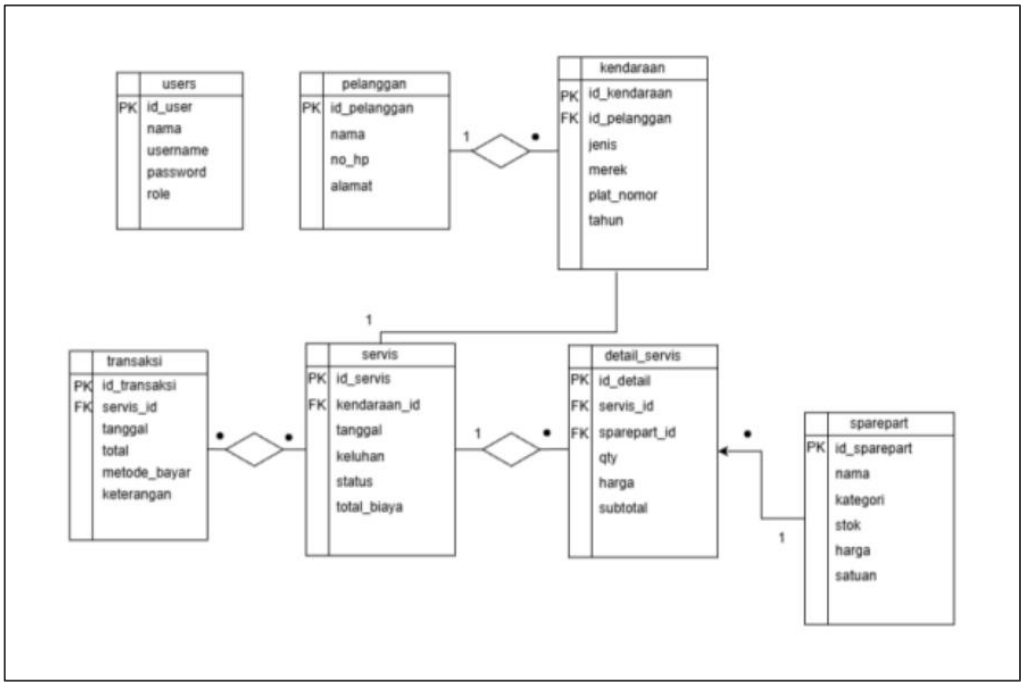
Tabel 2. 5 Pengertian *Sequence Diagram*

No	Simbol	Nama	Keterangan
1		*Lifeline*	Garis hidup yang merepresentasikan waktu keberadaan objek selama interaksi.
2		*Synchronous Message*	Pesan sinkron di mana pengirim menunggu respons sebelum melanjutkan.
3		*Asynchronous Message*	Pesan asinkron di mana pengirim tidak menunggu respons.
4		*Return Message*	Pesan balikan yang mengembalikan hasil dari pesan sebelumnya.
6		*Combined Fragment*	Blok yang merepresentasikan logika percabangan, perulangan, atau kondisi opsional.

2.4.5 *Entity Relationship Diagram* (ERD)

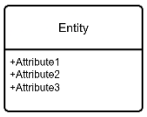
Entity Relationship Diagram (ERD) adalah suatu model jaringan yang menjelaskan susunan data yang disimpan dalam sistem secara abstrak. ERD digunakan untuk memodelkan struktur logis dari basis data beserta hubungan antar data. Dalam perancangan sistem ini, pemodelan ERD menggunakan notasi *Crow's*

Foot yang merupakan standar modern dalam rekayasa perangkat lunak karena lebih ringkas dan representatif untuk basis data relasional.



Gambar 2. 3 Contoh ERD
Source : *(Masayu et al., 2026)*

Tabel 2. 6 Notasi ERD

NO	Simbol	Nama	Deskripsi
1		***Entity***	Menggambarkan objek utama yang datanya disimpan dalam sistem Digambarkan dengan persegi panjang.
2		***One and Only One***	Simbol dengan dua garis tegak lurus. Menandakan bahwa sisi relasi ini bersifat wajib dan hanya boleh ada tepat satu (contoh: setiap berkas wajib dimiliki oleh tepat satu pengguna).
3		***Zero optional many***	Simbol dengan lingkaran kosong dan tiga cabang. Menandakan bahwa sisi relasi ini bersifat opsional dan bisa berjumlah banyak (contoh: satu pengguna *boleh* tidak memiliki berkas, atau *boleh* memiliki banyak berkas).

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

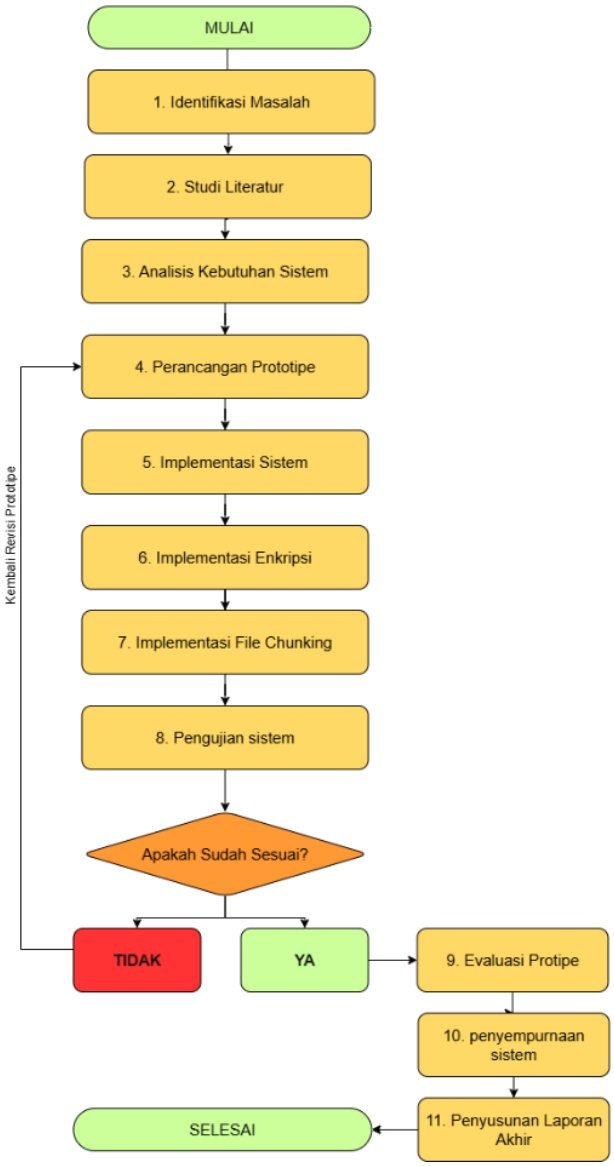
3.5. Teknik Pengumpulan Data

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

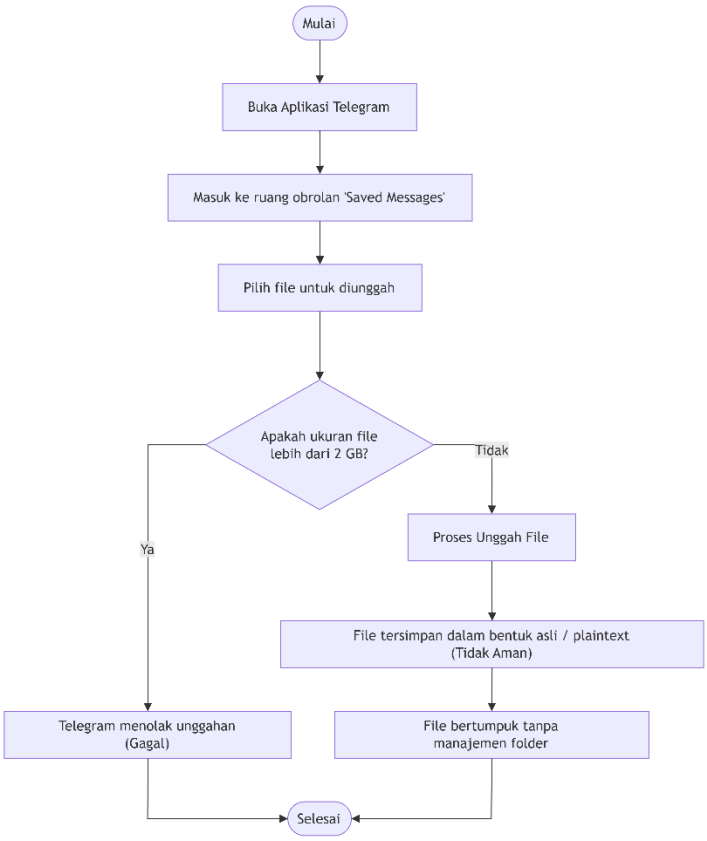
Tahap ini merupakan titik awal penelitian. Peneliti mengidentifikasi permasalahan utama yang menjadi latar belakang penelitian, yaitu keterbatasan kapasitas dan biaya layanan *public cloud storage*, risiko privasi data pada penyimpanan berbasis pihak ketiga, serta tidak adanya solusi penyimpanan berbasis 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

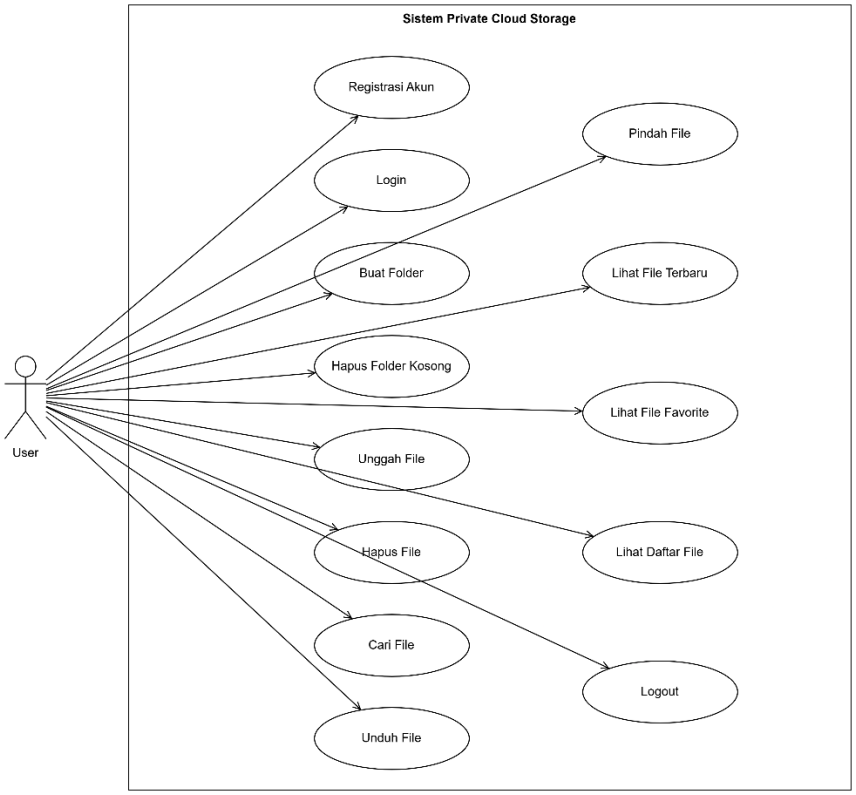
seluruh isi file ke dalam RAM *server*. Ini sangat berguna untuk mengunduh file berukuran besar dengan penggunaan memori yang efisien. Antarmuka *Web UI* juga dilengkapi dengan beberapa fitur tambahan, seperti pencarian file, penandaan file favorit, daftar file terbaru, pemindahan file antar folder,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

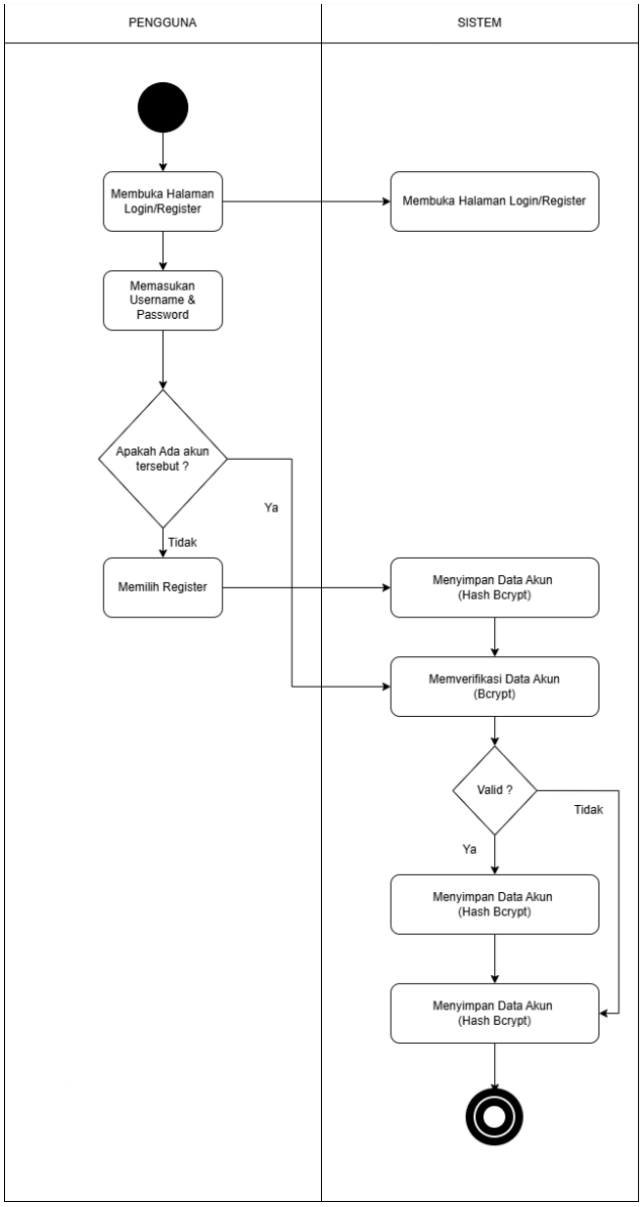
No	Aktor	Deskripsi
8	Pindah File	Pengguna memindahkan berkas dari satu direktori folder ke folder lainnya di dalam sistem.
9	Pencarian File	Pengguna mencari berkas tertentu berdasarkan nama berkas menggunakan fitur pencarian.
10	Logout	Pengguna keluar dari sistem dan sistem akan menghapus *token* sesi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

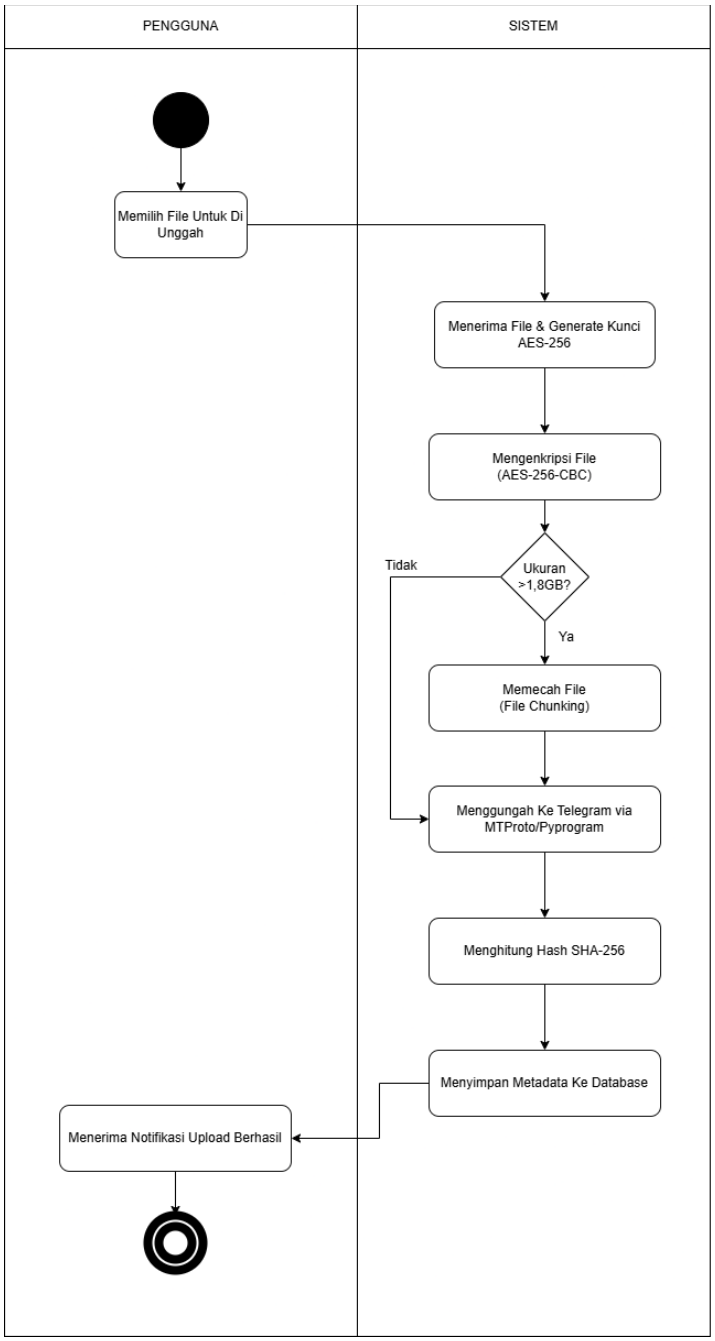


Gambar 3. 4 *Activity Diagram* Autentikasi

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.

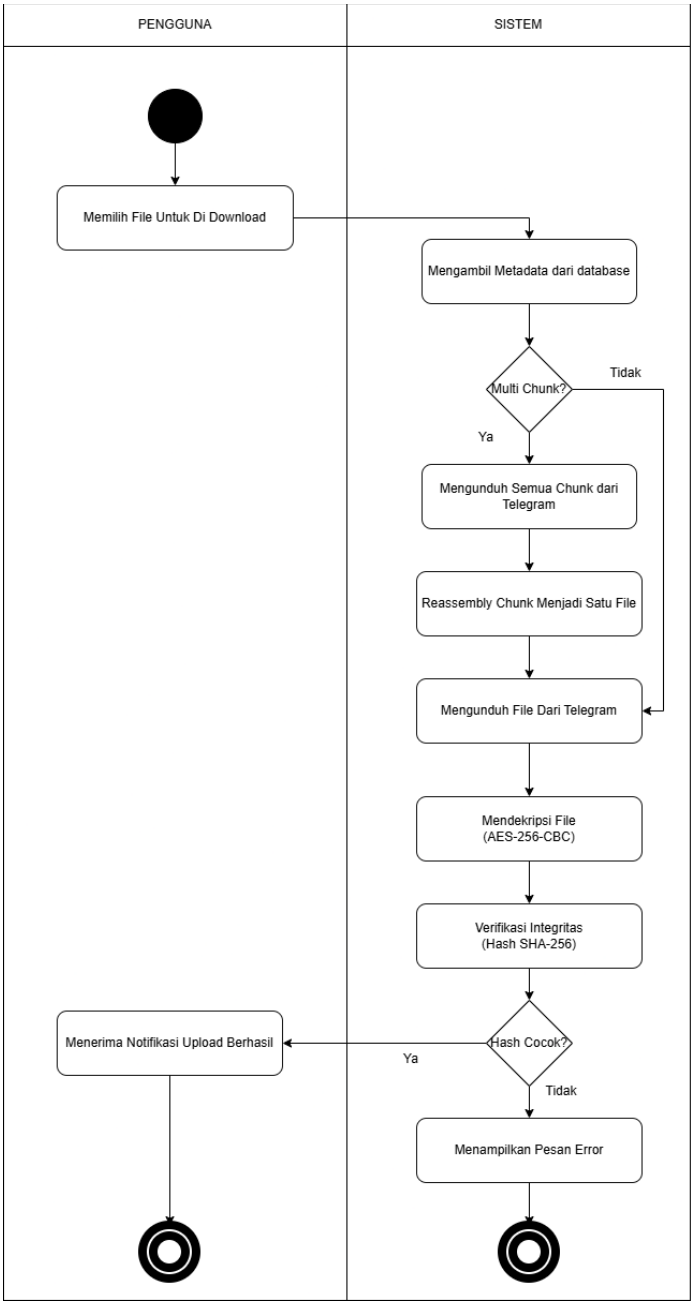


Gambar 3. 5 *Activity Diagram Upload File*

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

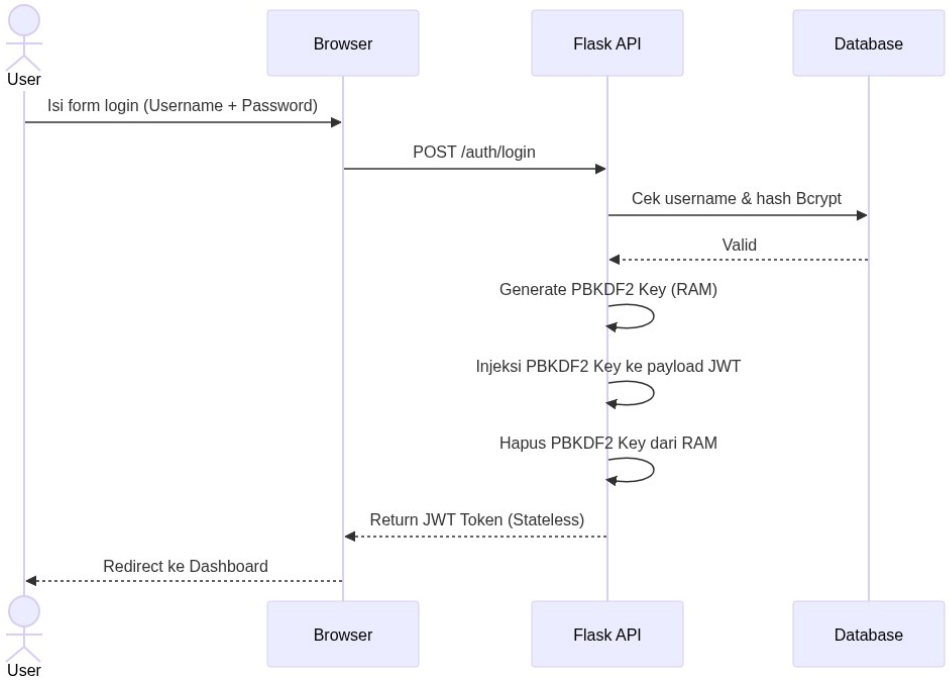
No	**Aktivitas**	**Deskripsi**
1	Klik tombol *Download*	Pengguna menekan tombol unduh pada berkas tertentu di antarmuka *Web UI*.
2	Terima request *download*	Sistem menerima permintaan unduh dan memvalidasi hak akses pengguna terhadap berkas tersebut.
3	Ambil Kunci AES dari DB	Sistem mengambil *metadata* 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

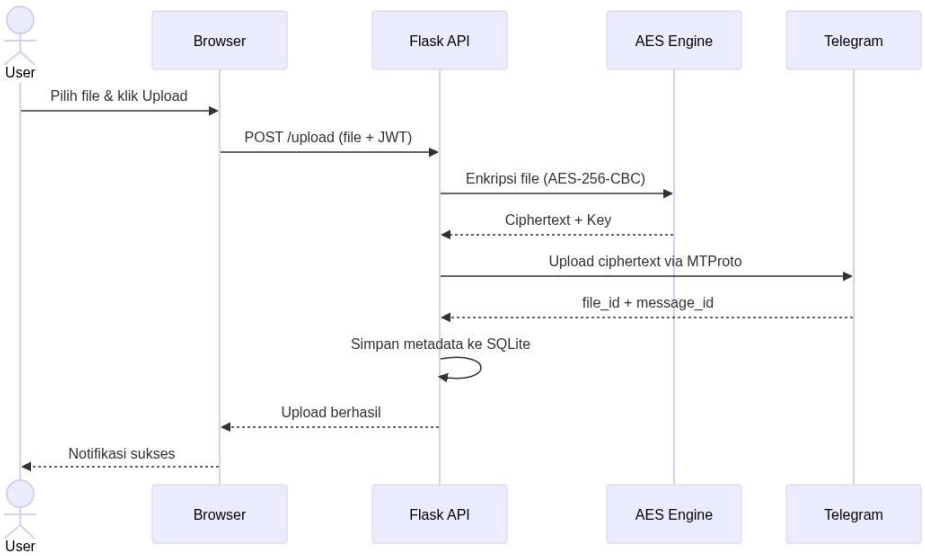
3.6.5.3 *Sequence Diagram*

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.



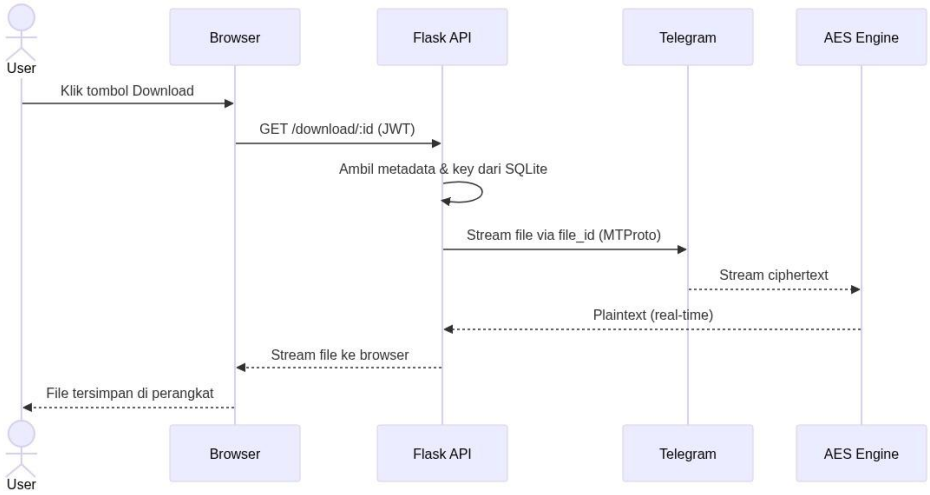
Gambar 3. 7 *Sequence Diagram* Autentikasi (Login)

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.



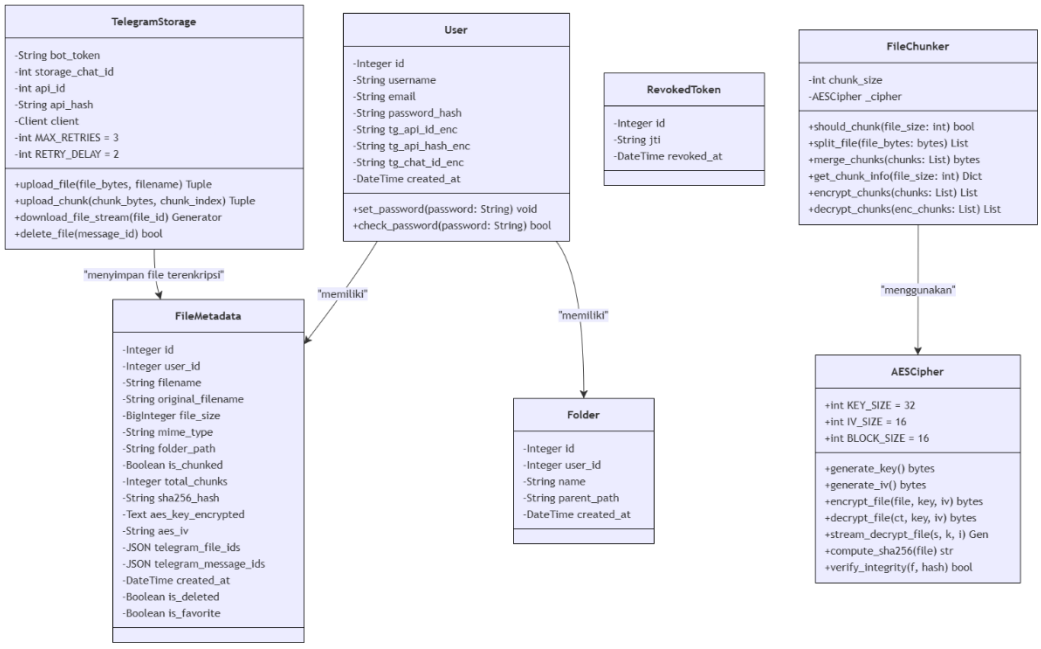
Gambar 3. 8 *Sequence Diagram Upload* File

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

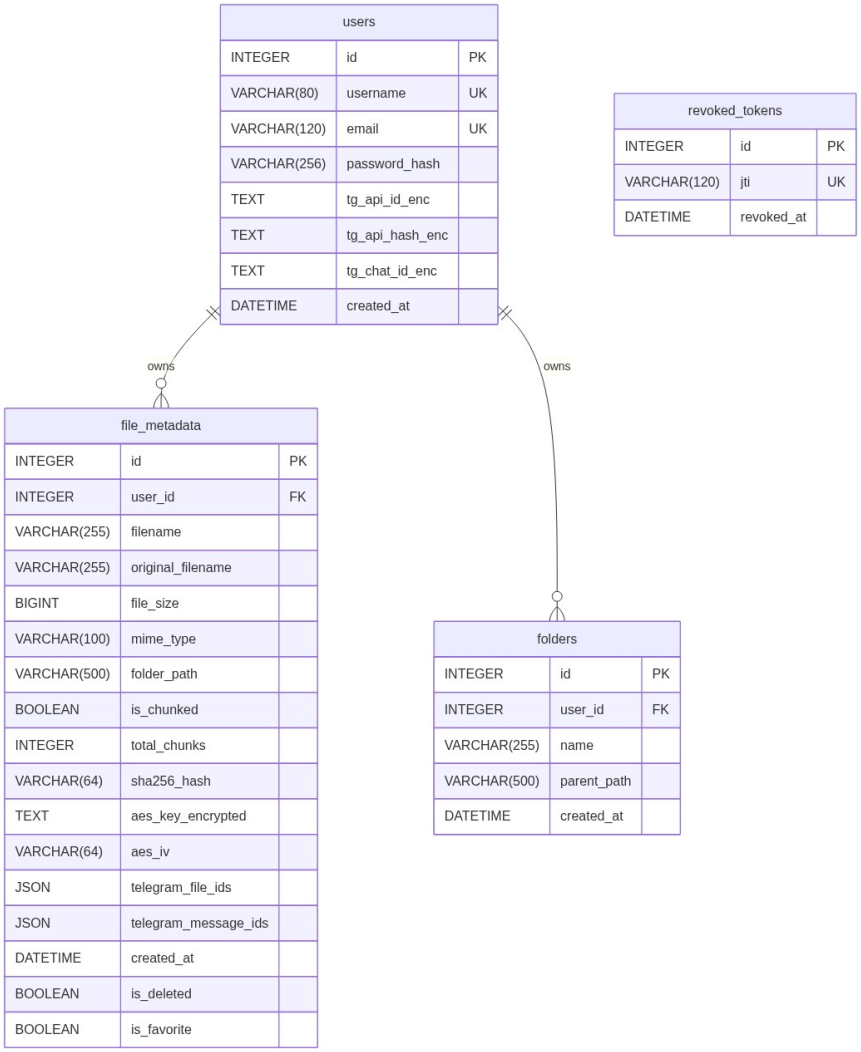
3.6.5.4 *Class Diagram*



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*



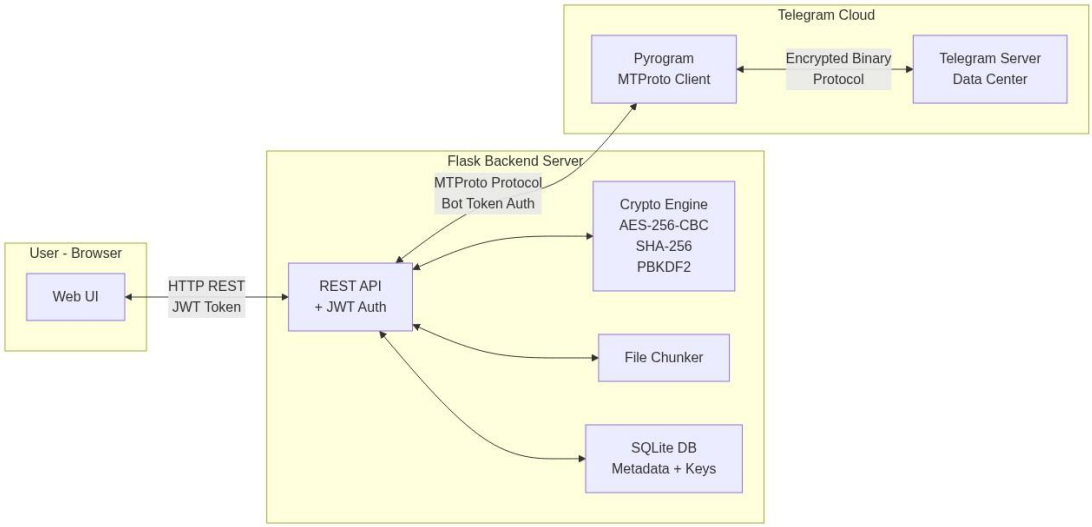
Gambar 3. 11 *Entity Relationship Diagram* (ERD) Basis Data *Metadata*

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.



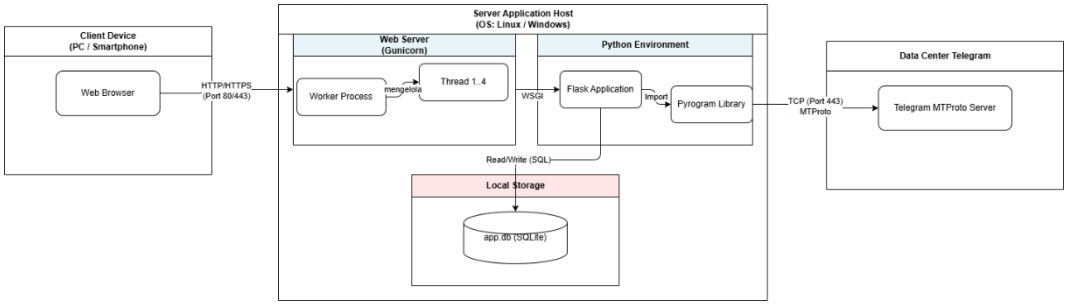
Gambar 3. 12 Arsitektur Keseluruhan Sistem Private *Cloud storage*

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).

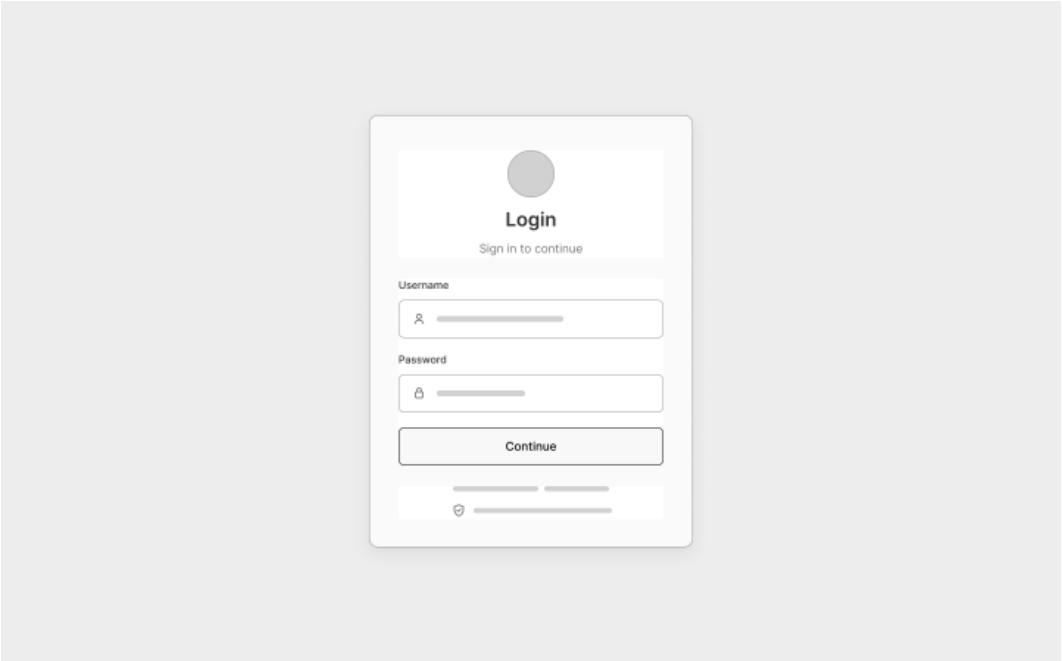


Gambar 3. 13 *Deployment* Diagram Private *Cloud storage*

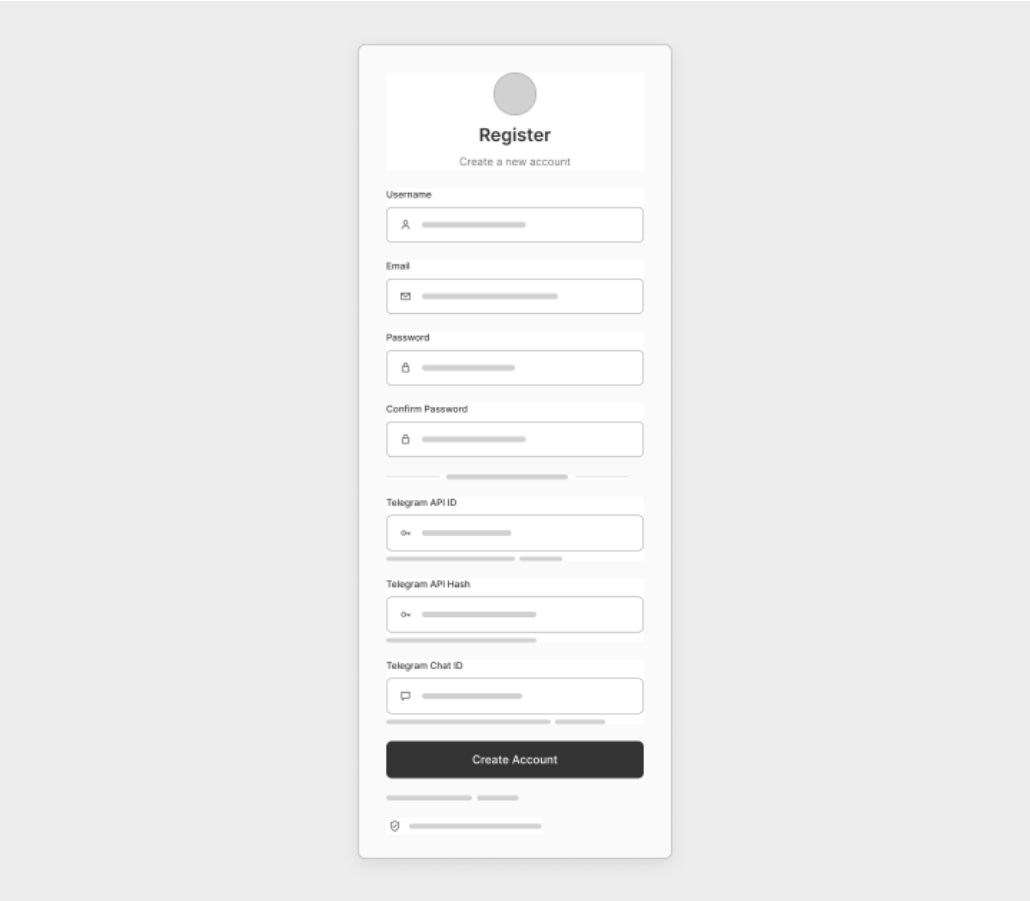
Berdasarkan Gambar 3.13, lingkungan produksi sistem terbagi menjadi tiga *node* utama. Pada *node* klien, pengguna mengakses sistem melalui peramban web (*Web Browser*). Permintaan diteruskan melalui jaringan internet ke *node Server Application Host*. Aplikasi Python ini dikonfigurasi berjalan di atas *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:

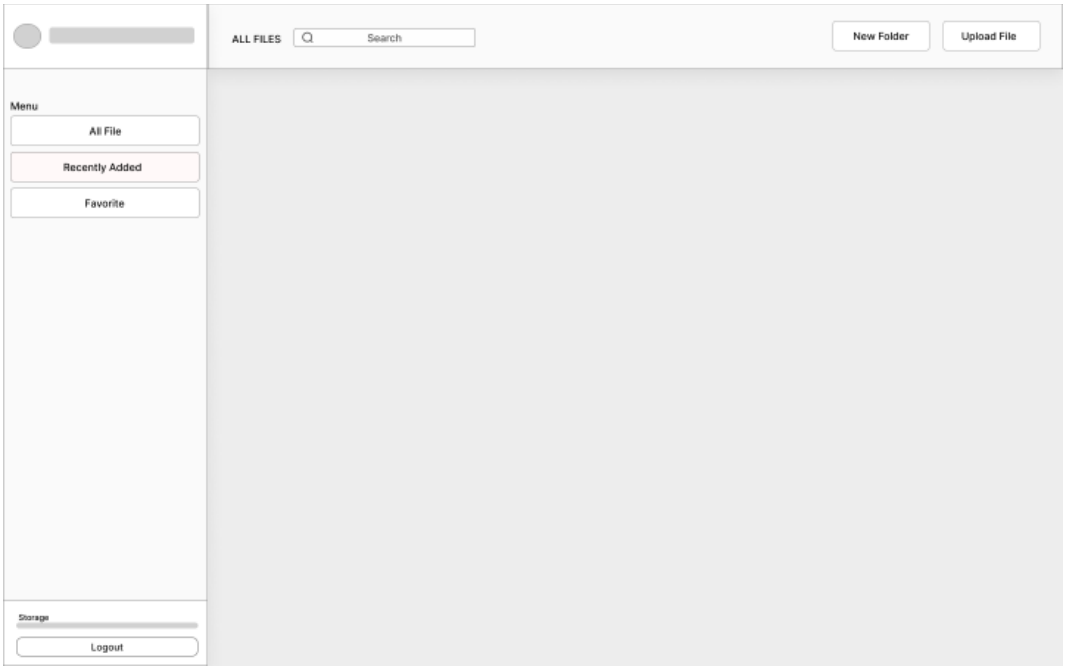


Gambar 3. 14 *Wireframe* Halaman Autentikasi Login



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.

.

BAB IV
HASIL DAN PEMBAHASAN

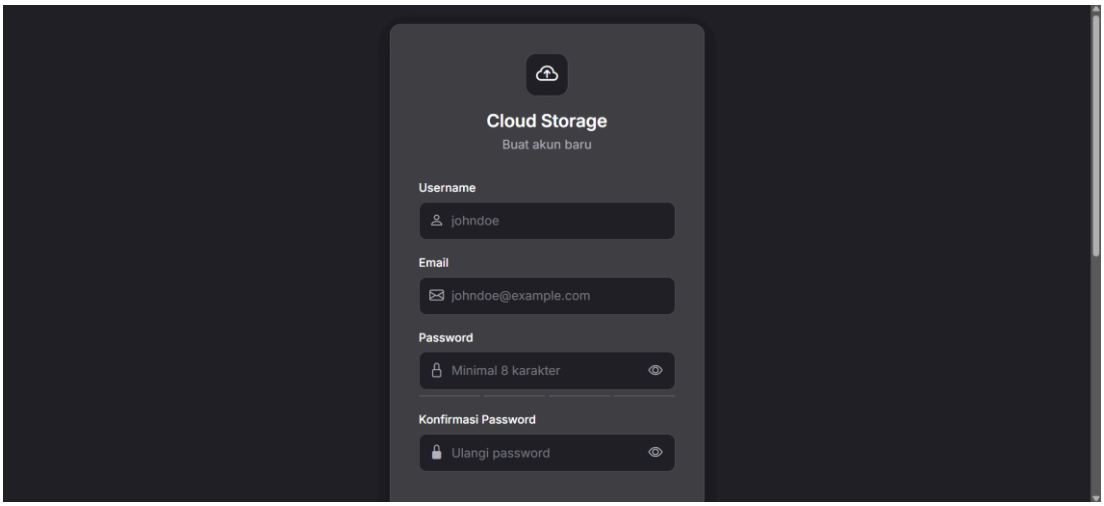
4.1. Hasil Implementasi Sistem

Bab ini menyajikan hasil implementasi sistem Private *Cloud storage* yang telah dibangun berdasarkan rancangan *prototype* pada BAB III. Sistem diimplementasikan menggunakan pendekatan iteratif sesuai metode Prototyping, di mana setiap komponen diuji secara bertahap sebelum diintegrasikan ke dalam arsitektur sistem yang utuh. Hasil implementasi mencakup dua aspek utama, yaitu antarmuka pengguna berbasis *Web UI* dan komponen *backend* beserta basis data.

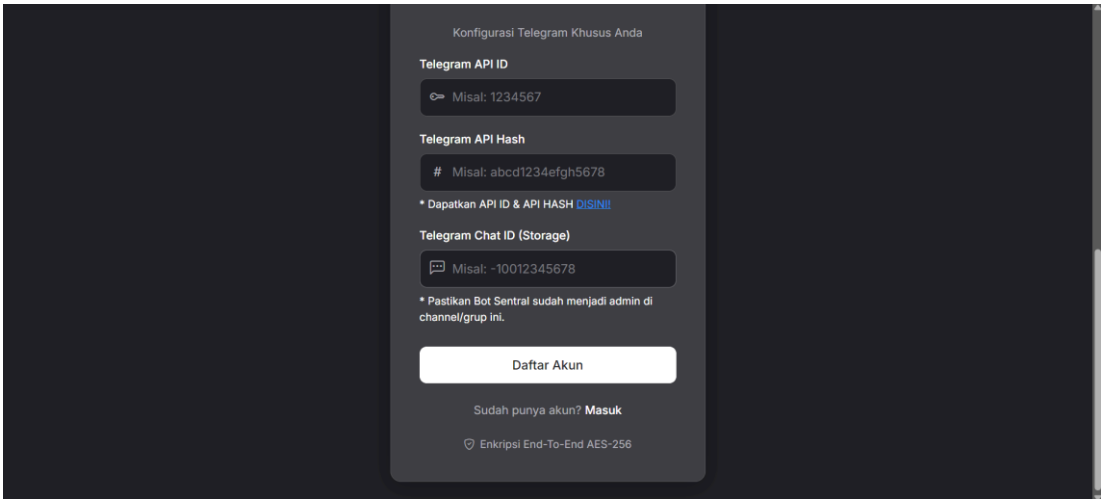
4.1.1. Implementasi Antarmuka Pengguna (*Web UI*)

Antarmuka pengguna sistem dibangun menggunakan teknologi HTML5, CSS3, JavaScript, dan *framework* Bootstrap 5, yang berjalan sepenuhnya di sisi klien melalui peramban web. Desain antarmuka mengadopsi paradigma *virtual file manager* yang familiar bagi pengguna layanan *cloud storage* komersial, sehingga kurva pembelajaran dapat diminimalkan. Sistem antarmuka secara garis besar dibangun atas tiga halaman utama yaitu Registrasi, *Login*, dan *Dashboard* yang didukung oleh berbagai antarmuka fitur interaktif di dalamnya. Keseluruhan elemen antarmuka ini saling terintegrasi secara asinkron melalui mekanisme komunikasi REST API dengan *backend* Flask.

Halaman Registrasi merupakan pintu masuk bagi pengguna baru yang belum memiliki akun. Berbeda dengan sistem konvensional, pada arsitektur multi-user ini, pengguna tidak hanya diminta memasukkan data diri dasar, tetapi juga diwajibkan untuk menginput kredensial Telegram pribadi mereka, yakni Telegram API ID, API *Hash*, dan Chat ID. Kredensial ini akan langsung dienkripsi menggunakan AES-256-CBC di sisi peladen dan tidak pernah disimpan dalam bentuk *plaintext*.

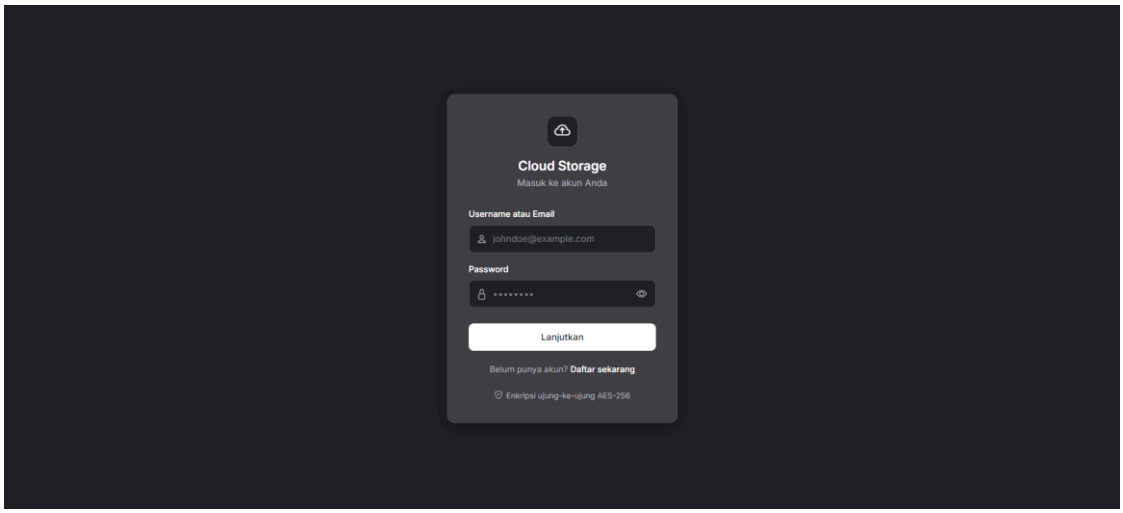


Gambar 4. 1 Tampilan Halaman Registrasi Sistem



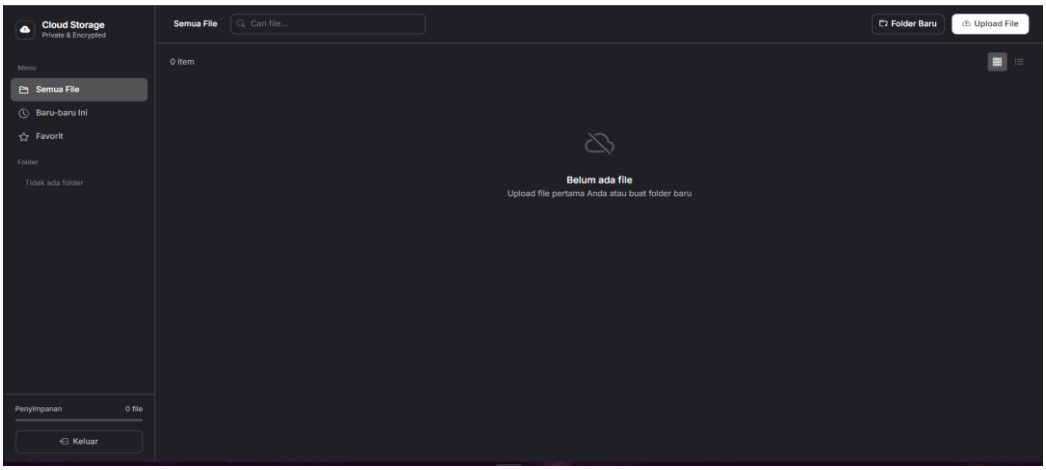
Gambar 4. 2 Tampilan Halaman Registrasi Sistem pada kredensial Telegram

Halaman Login menampilkan formulir *autentikasi* yang memungkinkan pengguna terdaftar mengakses sistem dengan memasukkan username dan kata sandi. Sistem mengirimkan kredensial ke *endpoint /api/auth/login*. apabila kredensial valid, *backend* mengembalikan *JSON Web Token* (JWT) yang kemudian disimpan pada memori sesi *browser*. *Token* ini selanjutnya digunakan sebagai kredensial otorisasi pada setiap permintaan API berikutnya melalui *header Authorization: Bearer <token>*. Fitur *toggle visibilitas* kata sandi turut disediakan untuk meningkatkan kenyamanan pengguna.



Gambar 4. 3 Tampilan Halaman *Login* Sistem

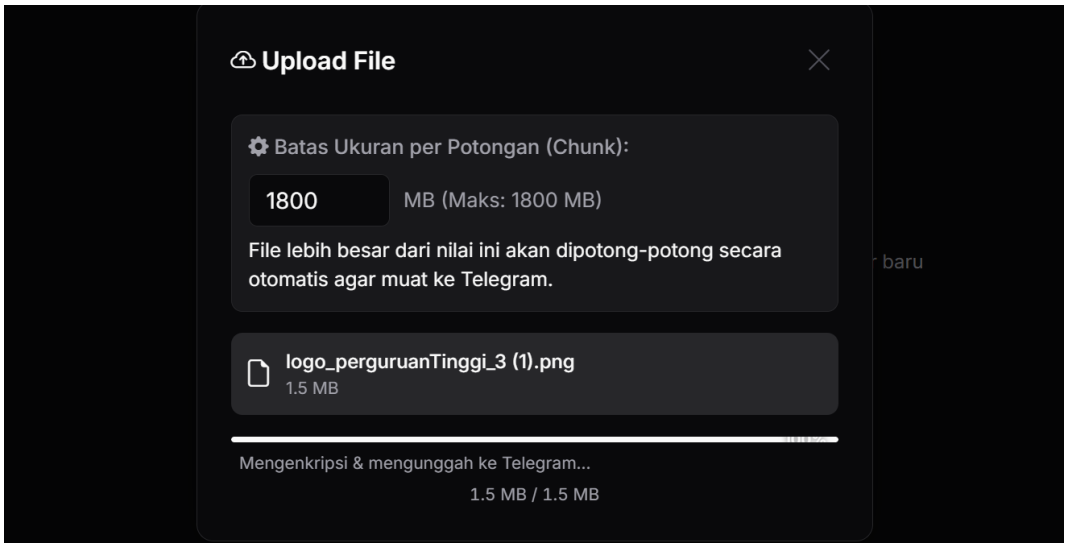
Dashboard Utama merupakan halaman inti sistem yang berfungsi sebagai *virtual file manager*. Antarmuka terbagi menjadi dua bagian utama: panel navigasi di sisi kiri yang menampilkan struktur direktori secara hierarkis, dan area konten utama di bagian kanan yang menampilkan daftar berkas dan folder pada direktori aktif. Pengguna dapat beralih antara tampilan grid dan tampilan. *Header* halaman menyediakan bilah pencarian, tombol unggah berkas, tombol buat folder baru, serta menu akun pengguna. Panel kiri juga menampilkan pintasan navigasi cepat ke direktori Semua Berkas, Berkas Favorit, dan Berkas Terbaru.



Gambar 4. 4 Tampilan *Dashboard* Utama

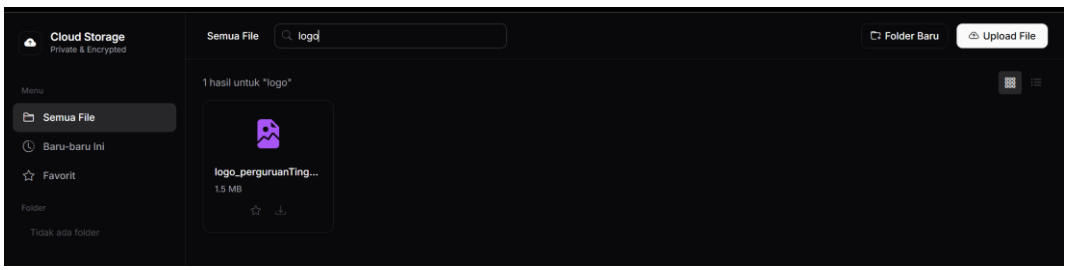
Fitur Unggah Berkas diakses melalui tombol unggah pada *header dashboard*. Antarmuka unggahan mendukung mekanisme *drag-and-drop* maupun

pemilihan berkas melalui dialog sistem operasi. Setelah berkas dipilih, sistem menampilkan bilah kemajuan (*progress bar*) yang merepresentasikan tahap-tahap proses secara berurutan, meliputi perhitungan *hash SHA-256* berkas asli, proses enkripsi AES-256-CBC, pengecekan apakah diperlukan pemecahan *chunk*, serta pengunggahan ke *server* Telegram melalui Pyrogram MTProto. Umpan balik status ditampilkan secara *real-time* kepada pengguna sehingga proses yang berlangsung di sisi *backend* tetap transparan.



Gambar 4. 5 Tampilan Dialog Unggah Berkas dengan Indikator Kemajuan

Fitur Pencarian memungkinkan pengguna menemukan berkas berdasarkan kata kunci nama berkas melalui pencarian yang tersedia di *header*. Sistem mengirimkan kueri pencarian ke *endpoint /api/files/search* dan menampilkan hasil yang relevan secara dinamis tanpa perlu memuat ulang halaman. Selain itu, terdapat fitur penandaan favorit yang memungkinkan pengguna menandai berkas-berkas yang sering diakses untuk ditampilkan secara khusus pada direktori Berkas Favorit.

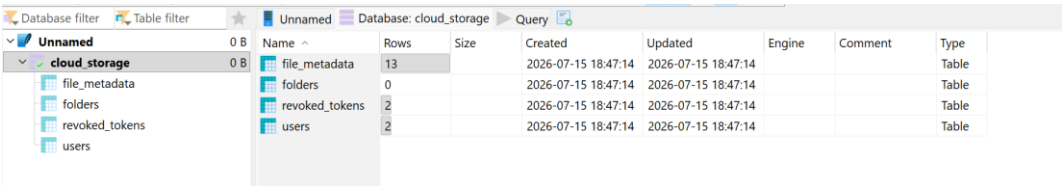


Gambar 4. 6 Tampilan Fitur Pencarian Berkas

4.1.2. Implementasi *Database* dan *Backend*

Backend sistem dibangun menggunakan *framework Flask* (Python) yang menyediakan *REST API* untuk seluruh operasi sistem. Basis data lokal menggunakan SQLite yang dikelola melalui ORM (*Object-Relational Mapping*) SQLAlchemy, sehingga seluruh interaksi dengan basis data dilakukan melalui objek Python tanpa penulisan kueri SQL secara langsung. Basis data hanya digunakan untuk menyimpan *metadata* dan parameter keamanan, konten berkas tidak tersimpan di basis data lokal maupun di *server* aplikasi.

Implementasi basis data SQLite untuk sistem *Private Cloud* ini diwujudkan melalui *database* "*cloud*_storage.db" yang terdiri dari 4 tabel utama. Setiap tabel saling berelasi secara terintegrasi untuk mengelola berbagai aktivitas bisnis, mulai dari data pengguna, folder, file_*metadata*, revoked_*token*. Desain *database* ini juga telah menerapkan prinsip normalisasi untuk menekan risiko redundansi data sekaligus menjamin konsistensi dan integritas informasi di dalam sistem.



Gambar 4. 7 Diagram Skema Basis Data SQLite

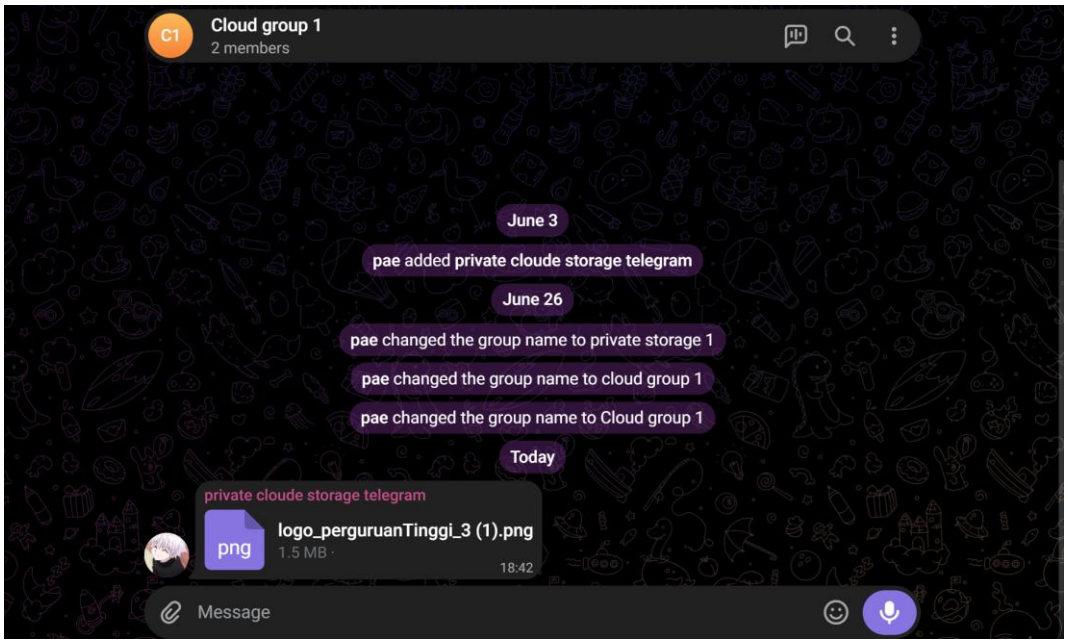
Integrasi dengan infrastruktur Telegram direalisasikan melalui *library* Pyrogram yang berkomunikasi langsung dengan *server* Telegram menggunakan protokol MTProto. Setiap berkas atau segmen *chunk* yang telah dienkripsi dikirimkan ke channel atau grup privat Telegram yang dikelola oleh akun bot. Telegram mengembalikan file_id unik untuk setiap berkas yang berhasil diunggah; nilai ini kemudian disimpan pada kolom telegram_file_ids di basis data lokal dan digunakan sebagai referensi saat proses pengunduhan dilakukan. Pendekatan ini memungkinkan sistem memanfaatkan infrastruktur penyimpanan Telegram tanpa biaya tambahan, sekaligus menjaga kerahasiaan data karena seluruh berkas yang tersimpan di Telegram berada dalam kondisi terenkripsi.

4.1.3. Implementasi Lingkungan Produksi (*Production Deployment*)

Pada tahap awal pengembangan, sistem dijalankan menggunakan *server* bawaan (Development *Server*) dari *framework* Flask, yaitu Werkzeug. Namun, *server* bawaan tersebut tidak dirancang untuk menangani beban kerja tinggi secara simultan di lingkungan produksi (Production Environment). Oleh karena itu, arsitektur runtime *deployment* dimigrasikan menggunakan Gunicorn (Green Unicorn), yang merupakan Python WSGI HTTP *Server* berstandar industri. Migrasi ini memastikan bahwa aplikasi memiliki skalabilitas yang memadai dan mampu menangani banyak permintaan pengguna (*concurrent* requests) secara stabil dan efisien tanpa mengalami kegagalan sistem (crash).

Dalam implementasi lingkungan produksi tersebut, terdapat tantangan teknis yang berkaitan dengan penanganan konkurensi pada pustaka Pyrogram dan basis data SQLite. Pyrogram menggunakan berkas sesi (*.session*) berbasis SQLite yang spesifik untuk setiap pengguna. Apabila Gunicorn dijalankan menggunakan skema "Multi-*Worker* Process", akan terjadi error berupa "*Database is Locked*". Hal ini disebabkan oleh adanya beberapa proses *worker* yang mencoba mengakses dan mengunci berkas sesi SQLite yang sama secara bersamaan. Sebagai solusi arsitektur, Gunicorn dikonfigurasi secara ketat menggunakan skema *Single Worker with Multi-Threading*, yakni dengan 1 *worker process* dan 4 *threads*. Pendekatan *multithreading* ini memungkinkan pembagian memory pool dari Pyrogram Client antar *thread*, sehingga *server* dapat melayani proses pengunggahan maupun pengunduhan berkas dari banyak pengguna pada waktu yang bersamaan secara *concurrently*. Dengan memanfaatkan model konkurensi berbasis *thread based concurrency* ini, beban kerja dapat didistribusikan secara paralel tanpa melanggar mekanisme *locking mechanism* pada basis data sesi Telegram.

Selain penanganan konkurensi, optimasi *timeout* pada Gunicorn juga dilakukan. Nilai *timeout* ditingkatkan dan disetel menjadi 120 detik. Penyesuaian konfigurasi ini bertujuan untuk mengakomodasi proses Input/Output (I/O) yang berat pada sistem, yang meliputi operasi pemecahan berkas (*chunking*), proses enkripsi data dengan AES-256, serta proses *streaming upload* ke *server* Telegram.



Gambar 4. 8 Berkas Terenkripsi yang Tersimpan di Infrastruktur Telegram

4.2. Hasil Implementasi Keamanan dan Kriptografi

Lapisan keamanan sistem diimplementasikan secara berlapis (*defense in depth*) mencakup autentikasi pengguna berbasis JWT, enkripsi kata sandi dengan Bcrypt, enkripsi berkas dengan AES-256-CBC, derivasi kunci dengan PBKDF2, dan verifikasi integritas data dengan SHA-256. Setiap lapisan dirancang agar bekerja secara independen, sehingga kompromi pada satu lapisan tidak secara langsung mengekspos lapisan keamanan lainnya.

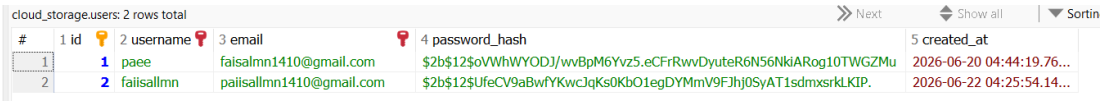
4.2.1. Implementasi Autentikasi JWT dan Bcrypt

Keamanan autentikasi pengguna dan manajemen kunci diimplementasikan dengan pendekatan manajemen kunci non-persisten (*stateless key management*). *Server* atau basis data sama sekali tidak menyimpan kunci dekripsi (Encryption Key) untuk kredensial Telegram maupun berkas pengguna secara permanen. Kunci dekripsi utama diderivasi menggunakan fungsi PBKDF2-HMAC-SHA256 sebanyak 600.000 iterasi, dengan mengambil kata sandi pengguna sebagai secret (*passphrase*) dan digabungkan dengan *salt* acak (Cryptographically Secure Pseudo-Random *Salt*) sebesar 16-byte. Penggunaan *salt* acak yang unik pada setiap proses

enkripsi ini merupakan standar OWASP untuk secara efektif menggagalkan serangan *precomputed rainbow tables*.

Pada saat pengguna mendaftar, kata sandi yang dikirimkan melalui formulir registrasi tidak pernah disimpan dalam bentuk *plaintext*. Sistem menggunakan algoritma Bcrypt untuk menghasilkan nilai *hash* dari kata sandi tersebut sebelum menyimpannya ke dalam kolom *password_hash* pada tabel *users*. *Bcrypt* dipilih karena secara bawaan menyertakan nilai *salt* acak yang unik pada setiap proses *hashing*, sehingga dua pengguna dengan kata sandi yang sama akan menghasilkan nilai *hash* yang berbeda. Selain itu, *Bcrypt* dirancang dengan kompleksitas komputasi yang dapat dikonfigurasi (*cost factor*), menjadikannya tahan terhadap serangan *brute-force* dan *precomputed rainbow table*. Implementasi ini sesuai dengan rekomendasi OWASP untuk penyimpanan kata sandi yang aman.

Saat pengguna melakukan login, sistem mengambil nilai *hash* dari basis data dan memverifikasinya terhadap kata sandi yang dimasukkan menggunakan fungsi *bcrypt.checkpw()*. Fungsi ini bersifat *timing-safe*, artinya waktu eksekusi verifikasi tidak bergantung pada panjang atau kecocokan kata sandi, sehingga mencegah serangan *timing side-channel*. Apabila verifikasi berhasil, sistem menerbitkan JSON Web *Token* yang ditandatangani menggunakan algoritma HS256 dengan kunci rahasia (JWT_SECRET_KEY) yang disimpan sebagai variabel lingkungan. *Token* ini memuat klaim identitas pengguna (user_id) dan waktu kedaluwarsa 24 jam, dan tidak menyimpan data sensitif di dalamnya.


cloud_storage.users: 2 rows total

#	1 id	2 username	3 email	4 password_hash	5 created_at
1	1	paee	faisalmn1410@gmail.com	$2b$12$oVWhWYODJ/wvBpM6Yvz5.eCFrRwvDyuteR6N56NkiARog10TWGZMu	2026-06-20 04:44:19.76...
2	2	faiisallmn	paiisallmn1410@gmail.com	$2b$12$UfeCV9aBwfYKwcJqKs0KbO1egDYMmV9FJhj0SyAT1sdmxsrkLKIP.	2026-06-22 04:25:54.14...

Gambar 4. 9 Contoh *Hash* Bcrypt yang Tersimpan di Basis Data

Mekanisme logout menggunakan pendekatan *token blocklist* yang persisten. Saat pengguna mengakhiri sesi, klaim jti diekstrak dari *token* aktif dan disimpan ke dalam tabel revoked_*token*s di basis data SQLite. Setiap permintaan API yang masuk kemudian diverifikasi tidak hanya terhadap validitas tanda tangan JWT,

tetapi juga diperiksa apakah nilai jti-nya terdapat dalam daftar *blocklist*. Apabila *token* ditemukan di *blocklist*, permintaan ditolak dengan respons *HTTP 401 Unauthorized* meskipun *token* secara teknis belum melewati masa berlakunya. Pendekatan ini menjamin bahwa *token* yang telah di-*logout* tidak dapat disalahgunakan, bahkan jika berhasil dicegat oleh pihak yang tidak berwenang.

4.2.2. Implementasi Enkripsi AES-256-CBC

Enkripsi berkas diimplementasikan menggunakan algoritma Advanced Encryption Standard dengan panjang kunci 256-bit (AES-256) dalam mode operasi Cipher Block Chaining (CBC), dengan padding PKCS7 untuk memastikan ukuran data selalu merupakan kelipatan dari ukuran blok AES (16 byte). Seluruh proses enkripsi dan dekripsi berlangsung di sisi *backend server* sebelum data dikirimkan ke infrastruktur Telegram, sehingga konten berkas dalam bentuk *plaintext* tidak pernah meninggalkan lingkungan *server* aplikasi.

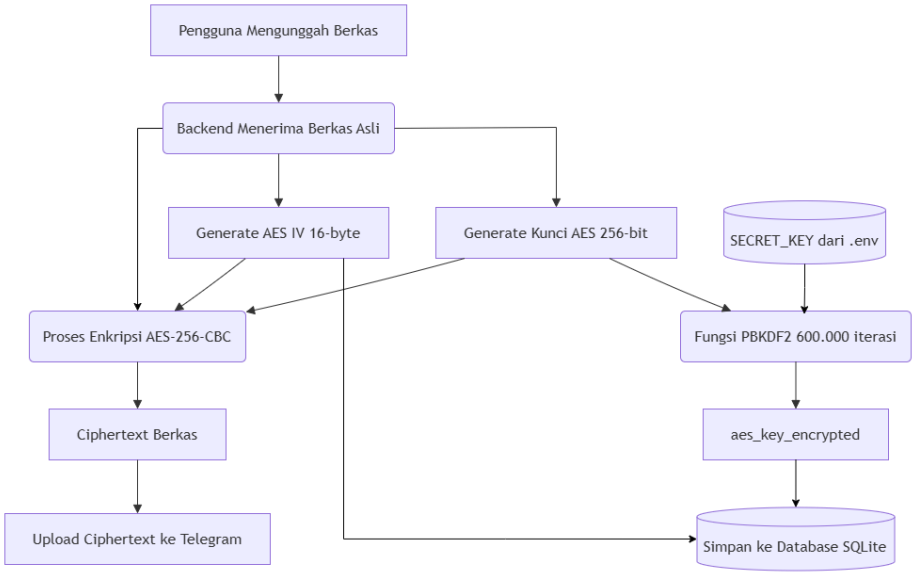
Untuk setiap berkas yang diunggah, sistem menghasilkan kunci AES sepanjang 32byte (256-bit) dan Initialization Vector (IV) sepanjang 16byte secara acak menggunakan generator angka acak yang aman secara kriptografis (Cryptographically Secure Pseudo-Random Number Generator/CSPRNG) dari *library* PyCryptodome. Penggunaan IV yang unik untuk setiap berkas memastikan bahwa berkas-berkas dengan konten identik akan menghasilkan *ciphertext* yang berbeda, sehingga analisis pola kriptografis menjadi tidak dimungkinkan.

6 tg_api_id_enc	7 tg_api_hash_enc	8 tg_chat_id_enc
XevHDL3e4yhcPsYT7JR0f...	SlGFfHnQN2ajIDV7O9iZw...	hqwLgmRq09P/rq54wx8I...
xpT/odmVlBJmil3zU/tOf5...	2snQR+8wd1EfbtoFukMX...	snZBz5GSd4sk2j0qDjihmo...

Gambar 4. 10 Penyimpanan *Secret_key* & Basis Data

Seperti yang ditunjukkan pada Gambar 4.9, nilai *SECRET_KEY* yang berfungsi sebagai *passphrase* PBKDF2 disimpan secara aman sebagai variabel lingkungan (*environment variable*) dan tidak pernah tertuliskan di dalam basis data. Standar 600.000 iterasi PBKDF2 yang diterapkan dalam sistem ini telah sejalan dengan

rekomendasi OWASP 2023 untuk fungsi derivasi kunci berbasis kata sandi. Tingginya jumlah iterasi tersebut secara spesifik dirancang untuk memperlambat serangan komputasi paksa (*brute-force*) terhadap kunci enkripsi secara signifikan. Adapun format penyimpanan parameter kriptografi di basis data menggunakan format gabungan: base64(*salt*[16] || iv_pbkdf2[16] || *ciphertext*_key). Pendekatan ini memastikan bahwa seluruh parameter dinamis yang diperlukan untuk proses dekripsi kunci terkapsulasi di dalam satu nilai terpadu yang aman.



Gambar 4. 11 Diagram Alur Proses Enkripsi Berkas

Proses dekripsi berlangsung secara simetris dan inversi dari proses enkripsi. Saat pengguna mengunduh berkas, sistem mengambil nilai *aes_key_encrypted* dan *aes_iv* dari basis data, mendekripsi kunci AES menggunakan PBKDF2 dengan *salt* dan *master* key yang tersimpan, kemudian menggunakan kunci AES yang telah terpulihkan bersama IV untuk mendekripsi *ciphertext* yang diunduh dari Telegram. Hasil dekripsi kemudian diverifikasi integritasnya menggunakan SHA-256 sebelum dikirimkan kepada pengguna.



Gambar 4. 12 Perbandingan Konten Berkas Asli dan *Ciphertext* Terenkripsi

4.2.3. Implementasi *File chunking*

Mekanisme file chunking diimplementasikan pada lapisan backend untuk menangani berkas yang ukurannya melebihi ambang batas operasional sistem sebesar 1,8 GB. Ambang ini ditetapkan di bawah batas teknis mutlak protokol MTProto sebesar 2 GB per berkas untuk menyediakan margin keamanan bagi *overhead* yang ditimbulkan oleh proses enkripsi AES dan penambahan *metadata* sistem. Dalam menentukan ukuran *chunk* tersebut, penelitian ini mengkaji efisiensi antara menggunakan pemotongan berukuran kecil dalam jumlah banyak berbanding ukuran besar dalam jumlah sedikit. Berdasarkan evaluasi terhadap arsitektur MTProto, pendekatan chunk ukuran besar ditetapkan sebagai metode paling optimal dengan landasan teknis berikut:

6. Penghindaran API *Rate Limiting*

   Server Telegram menerapkan mekanisme pembatasan *spam* (*API Rate Limiting*). Jika menggunakan *chunk* berukuran kecil, pengunggahan berkas 2,5 GB akan memicu pengiriman puluhan pesan secara instan beruntun, yang akan langsung memicu pemblokiran akun. Penggunaan *chunk* besar memastikan jumlah permintaan API sangat minim, sehingga sistem terbebas sepenuhnya dari risiko pemblokiran.

7. Efisiensi Latensi

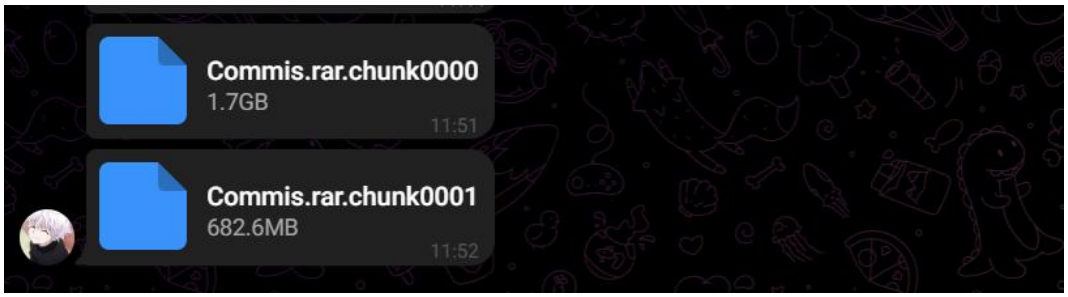
   Pada protokol TCP, perpindahan pengunggahan antar *file* memakan waktu untuk inisialisasi koneksi. Pemotongan *file* menjadi banyak bagian kecil akan menghasilkan jeda yang besar. Sebaliknya, dengan *chunk* besar 1,8 GB, koneksi *streaming data* berjalan terus-menerus secara stabil, memanfaatkan *bandwidth* secara penuh.

8. Reduksi Beban Relasi Basis Data

   Penggunaan chunk kecil menuntut sistem untuk menghasilkan puluhan kunci dekripsi dan entri tabel di basis data lokal untuk satu file utuh. Pendekatan *chunk* 1,8 GB meminimalkan pencatatan metadata tersebut, sehingga menghasilkan performa *I/O Read/Write server* yang jauh lebih ringan.

Ketika sistem mendeteksi bahwa ukuran berkas yang diunggah melampaui 1,8 GB, proses *chunking* dimulai secara otomatis tanpa intervensi pengguna. Berkas dibaca secara bertahap (*streaming*) untuk menghindari pemuatan seluruh konten ke dalam RAM *server* sekaligus. Sistem menentukan jumlah *chunk* yang diperlukan berdasarkan ukuran berkas dan ukuran *chunk* maksimum, kemudian memproses setiap segmen secara berurutan. Setiap *chunk* mendapatkan kunci AES 256-bit dan IV 16-byte yang dihasilkan secara acak dan independen satu sama lain, sehingga kompromi terhadap kunci satu *chunk* tidak mengekspos *chunk* lainnya.

Setiap *chunk* yang telah dienkripsi diunggah secara sekuensial ke channel Telegram privat melalui Pyrogram MTProto. Telegram mengembalikan *file_id* unik untuk setiap *chunk* yang berhasil diunggah. Sistem menyimpan seluruh informasi *chunk* meliputi urutan (indeks), file_id Telegram, kunci AES terenkripsi, dan IV dalam kolom telegram_file_ids pada tabel file_*metadata* sebagai larik JSON terurut. Informasi ini menjadi panduan rekonstruksi berkas saat proses pengunduhan dilakukan.



Gambar 4. 13 *Chunk* Berkas yang Diunggah Secara Sekuensial ke Telegram

Pada proses pengunduhan berkas yang terfragmentasi, sistem mengambil seluruh *chunk* berdasarkan urutan indeks yang tersimpan di basis data, mendekripsi masing-masing *chunk* menggunakan kunci dan IV yang bersesuaian, lalu menggabungkan seluruh data yang telah terdekripsi (reassembly) secara berurutan hingga menghasilkan berkas yang identik dengan berkas asli. Integritas berkas yang telah direkonstruksi diverifikasi menggunakan nilai *hash SHA-256* yang disimpan sebelum fragmentasi dilakukan. Seluruh proses ini berlangsung secara transparan dari sudut pandang pengguna, yang hanya mengklik tombol unduh dan menerima berkas utuh.

4.3. Pengujian Sistem

Pengujian sistem dilakukan secara komprehensif untuk memverifikasi bahwa seluruh komponen berfungsi sesuai spesifikasi yang telah ditetapkan pada BAB III. Pengujian dibagi menjadi tiga kategori: pengujian fungsional menggunakan metode *Black-Box* testing, pengujian kriptografi dan integritas data, serta pengujian performa untuk mengukur efisiensi operasional sistem.

4.3.1. Pengujian Fungsional (*Black-Box* Testing)

Pengujian fungsional dilakukan menggunakan metode *Black-Box* testing, di mana setiap fitur sistem diuji berdasarkan masukan dan keluaran yang dapat diobservasi tanpa memperhatikan implementasi internal. Pengujian mencakup tiga skenario utama: manajemen autentikasi pengguna, manajemen berkas, dan manajemen folder. Tabel 4.1 berikut menyajikan hasil pengujian fungsional untuk skenario autentikasi pengguna:

Tabel 4. 1 Hasil Pengujian Fungsional Skenario Autentikasi

No	**Skenario Uji**	**Langkah Pengujian**	**Data Masukan**	**Hasil yang Diharapkan**	**Status**
1	Registrasi akun berhasil	Mengisi formulir registrasi dengan data valid dan submit	Username: testuser, Email: test@gmail.com, Password: Test1234!	Akun berhasil dibuat, sistem mengembalikan HTTP 201 dan pesan sukses	Berhasil
2	Registrasi dengan email duplikat	Mendaftarkan email yang sudah terdaftar di sistem	Email yang sudah ada di *database*	Sistem mengembalikan HTTP 409 dengan pesan 'Email sudah terdaftar'	Berhasil
3	Login dengan kredensial valid	Memasukkan username dan password yang benar pada halaman login	Username & password yang terdaftar	Sistem mengembalikan JWT *token*, pengguna diarahkan ke *dashboard*	Berhasil
4	Login dengan password salah	Memasukkan password yang tidak sesuai	Password yang salah	Sistem mengembalikan HTTP 401 dengan pesan 'Kredensial tidak valid'	Berhasil

No	Skenario Uji	Langkah Pengujian	Data Masukan	Hasil yang Diharapkan	Status
5	Logout dan invalidasi *token*	Pengguna klik logout, lalu mencoba akses API dengan *token* lama	*Token* JWT aktif yang kemudian di-logout	*Token* masuk *blocklist*, akses dengan *token* lama ditolak HTTP 401	Berhasil
6	Akses *endpoint* tanpa *token*	Mengakses *endpoint* terproteksi tanpa *header* Authorization	Tidak ada *token*	Sistem mengembalikan HTTP 401 '*Token* autentikasi diperlukan'	Berhasil
7	Pengujian Konkurensi & Isolasi Multi-User	1. Pengguna A (*user1*) melakukan *login* dan mulai mengunggah berkas. 2. Pada peramban (*browser*) yang berbeda, Pengguna B (*user2*) dengan kredensial API Telegram yang berbeda melakukan *login* se	Dua pasang kredensial *log in* (nama pengguna dan kata sandi), beserta dua pasang kredensial API Telegram (API ID, API *Hash*, Chat ID) dari dua akun pengguna Telegram	Sistem berhasil melayani proses *upload*/*download* kedua pengguna secara bersamaan (*concurrent*) tanpa menimbulkan galat (error) "*Database is Locked*" pada	Berhasil

No	Skenario Uji	Langkah Pengujian	Data Masukan	Hasil yang Diharapkan	Status
		cara bersamaan. 3. Pengguna B mengakses direktori dan mengunduh berkas miliknya sendiri.	yang sepenuhnya berbeda.	sisi *backend*. Data dan kredensial Telegram terisolasi secara sempurna sehingga Pengguna A tidak dapat melihat berkas Pengguna B (Zero Data Leakage).	

Pada pengujian skenario isolasi *multi-user*, arsitektur Gunicorn yang telah dikonfigurasi menggunakan metode *single-worker multi-threading* terbukti berhasil mendistribusikan beban memori dari pustaka Pyrogram. Hal ini memungkinkan beberapa pengguna dengan sesi Telegram yang berbeda-beda untuk melakukan operasi I/O secara simultan tanpa memicu konflik penguncian pada basis data SQLite lokal.

Tabel 4. 2 Hasil Pengujian Fungsional Skenario Manajemen Berkas

No	**Skenario Uji**	**Langkah Pengujian**	**Data Masukan**	**Hasil yang Diharapkan**	**Status**
1	Unggah berkas kecil (<1,8 GB)	Mengunggah berkas PDF berukuran 50 MB melalui *Web UI*	File PDF 50 MB	Berkas terenkripsi, diunggah ke Telegram, *metadata* tersimpan di *database*	Berhasil
2	Unggah berkas besar (>1,8 GB)	Mengunggah berkas berukuran 2 GB	Berkas Berukuran 2 GB	Sistem otomatis membagi menjadi *chunk*, setiap *chunk* dienkripsi & diunggah	Berhasil
3	Unduh berkas non-*chunk*ed	Mengunduh berkas PDF yang sebelumnya diunggah	file_id berkas di *database*	Berkas diunduh dari Telegram, didekripsi, dikirim ke pengguna utuh	Berhasil
4	Unduh berkas *chunk*ed	Mengunduh berkas video 2,5 GB yang terfragmentasi	file_id semua *chunk* di *database*	Semua *chunk* diunduh, didekripsi, digabung, dikirim ke pengguna utuh	Berhasil

No	Skenario Uji	Langkah Pengujian	Data Masukan	Hasil yang Diharapkan	Status
5	Hapus berkas secara permanen.	Menghapus berkas melalui menu konteks di *Web UI*	ID berkas yang akan dihapus	Berkas berhasil dihapus secara permanen dari *server* Telegram dan basis data lokal menonaktifkan *metadata* berkas tersebut. Berkas langsung hilang dari daftar antarmuka *Web UI* secara waktu nyata (*real-time*). (Berkas terhapus permanen dan tidak dapat dipulihkan).	Berhasil
6	Pencarian berkas	Mengetik kata kunci pada bilah pencarian	Kata kunci nama berkas	Berkas yang relevan ditampilkan secara dinamis tanpa refresh halaman	Berhasil
7	Pemindahan berkas antar folder	Memindahkan berkas ke folder	ID berkas, path	Kolom folder_path berkas	Berhasil

No	Skenario Uji	Langkah Pengujian	Data Masukan	Hasil yang Diharapkan	Status
		berbeda melalui antarmuka	folder tujuan	diperbarui, berkas muncul di folder tujuan	

Tabel 4. 3 Hasil Pengujian Fungsional Skenario Manajemen Folder

No	Skenario Uji	Langkah Pengujian	Data Masukan	Hasil yang Diharapkan	Status
1	Buat folder baru	Memasukkan nama folder baru pada kolom pembuatan folder.	Nama folder: 'Dokumen Skripsi'	Folder baru berhasil dibuat dan langsung tampil di antarmuka.	Berhasil
2	Navigasi masuk ke folder	Mengklik nama folder pada daftar konten.	Direktori/folder yang sudah eksis di sistem.	Sistem menampilkan isi folder yang diklik, hierarki *breadcrumb* diperbarui secara *real-time*.	Berhasil
3	Buat folder dengan nama duplikat	Membuat folder baru dengan nama yang persis	Nama folder yang sudah ada di dalam direktori tersebut.	Sistem mengembalikan pesan kesalahan peringatan: *'Nama folder sudah digunakan'*.	Berhasil

No	Skenario Uji	Langkah Pengujian	Data Masukan	Hasil yang Diharapkan	Status
		sama di lokasi/direktori yang sama.			
4	Hapus folder kosong	Menekan tombol hapus pada folder yang belum memiliki isi.	Folder kosong.	Folder terhapus secara permanen dari sistem dan antarmuka.	Berhasil

4.3.2. Pengujian Kriptografi dan Integritas (SHA-256)

Pengujian kriptografi bertujuan untuk memverifikasi dua hal utama: pertama, akurasi bit-for-bit dari siklus enkripsi dan dekripsi; kedua, ketahanan sistem terhadap akses ilegal langsung ke infrastruktur Telegram.

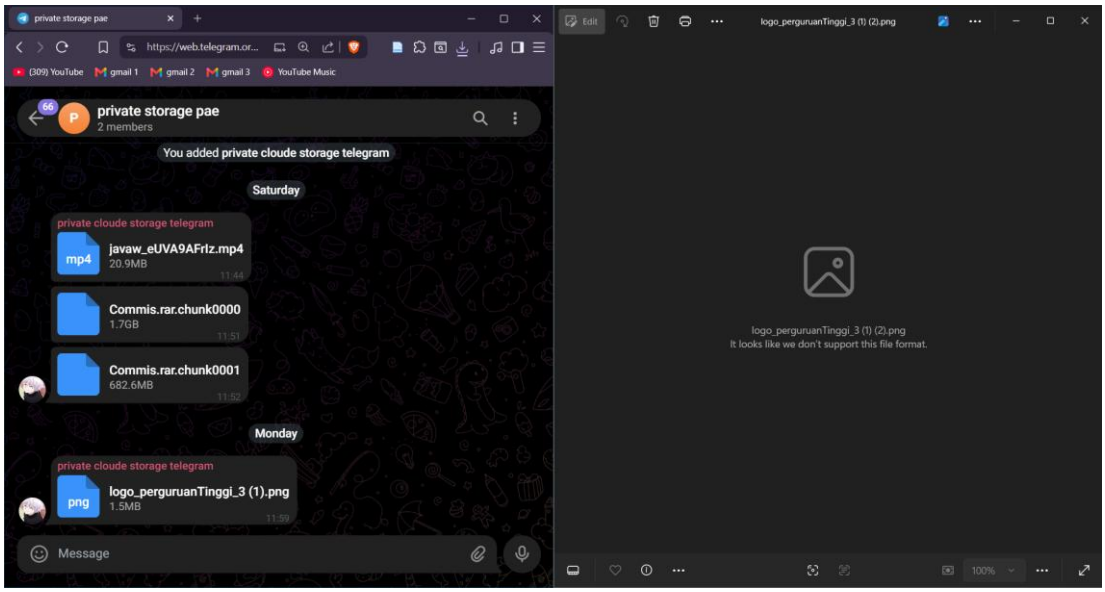
Pengujian integritas SHA-256 dilakukan dengan membandingkan nilai *hash* berkas asli sebelum diunggah dengan nilai *hash* berkas hasil unduhan yang telah didekripsi. Pada setiap pengujian, nilai SHA-256 dihitung menggunakan perintah sha256sum (Linux) dan dibandingkan secara langsung dengan nilai yang tersimpan di kolom sha256_*hash* pada basis data. Hasil pengujian menunjukkan kecocokan 100% (bit-for-bit identical) pada seluruh skenario uji, termasuk berkas berukuran kecil, besar, maupun berkas yang melalui proses *chunking*. Hal ini membuktikan

bahwa tidak terjadi korupsi data selama proses enkripsi, transmisi, penyimpanan di Telegram, pengambilan, maupun dekripsi.

Tabel 4. 4 Hasil Verifikasi Integritas SHA-256

Berkas Uji	***Hash SHA-256* Sebelum Enkripsi**	***Hash SHA-256* Setelah Dekripsi**	**Status**
1 Mb	e3b0c44298…	e3b0c44298…	identik
10 Mb	8a9b8c7d6e...	8a9b8c7d6e...	identik
500 Mb	5d4c3b2a1f...	5d4c3b2a1f...	identik
1024 Mb	a1b2c3d4e5...	a1b2c3d4e5...	identik
2147 Mb	e65d69d5cd…	e65d69d5cd…	identik
2500 Mb	9679aa8d70d8…	9679aa8d70d8…	identik

Pengujian simulasi akses ilegal dilakukan dengan mengunduh langsung salah satu berkas dari channel Telegram privat menggunakan akun Telegram lain yang memiliki akses ke channel tersebut, kemudian mencoba membuka berkas tersebut tanpa melalui sistem. Berkas yang diperoleh merupakan *ciphertext* murni hasil enkripsi AES-256-CBC yang tidak dapat diinterpretasikan tanpa kunci dekripsi yang tersimpan terenkripsi di basis data sistem. Upaya membuka berkas menggunakan perangkat lunak standar (pemutar video, penampil dokumen) menghasilkan kegagalan atau tampilan data yang tidak bermakna, membuktikan bahwa data pengguna tetap terlindungi meskipun infrastruktur penyimpanan Telegram dapat diakses oleh pihak lain.



Gambar 4. 14 Bukti Berkas Terenkripsi Tidak Dapat Dibuka Tanpa Kunci

4.3.3. Pengujian Performa

Pengujian performa dilakukan untuk mengukur efisiensi operasional sistem, khususnya waktu yang dibutuhkan untuk proses enkripsi, dekripsi, unggah, dan unduh berkas pada berbagai ukuran. Untuk mendapatkan nilai analisis yang objektif, pengujian ini menggunakan dua parameter perhitungan matematis, yaitu:

1. Kecepatan Pemrosesan Kriptografi ini digunakan untuk mengukur seberapa cepat algoritma AES-256-CBC pada CPU server mampu mengenkripsi atau mendekripsi sejumlah data dalam rentang waktu tertentu. Rumus yang digunakan adalah:

   ***Throughput* (MB/s) = Ukuran Berkas (MB) / Waktu Pemrosesan (Detik)**

   Semakin besar nilai *throughput* yang dihasilkan, maka semakin efisien algoritma tersebut berjalan pada spesifikasi perangkat keras yang digunakan tanpa menyebabkan *bottleneck* memori.

2. Kecepatan Transfer Jaringan Parameter ini digunakan untuk menghitung rasio kecepatan unggah dan unduh berkas dari server lokal ke infrastruktur MTProto Telegram dalam satuan *Megabit per second*. Karena ukuran berkas dihitung

dalam satuan *Byte* sedangkan kecepatan jaringan dalam satuan bit, maka ukuran berkas harus dikalikan 8 (1 Byte = 8 bit). Rumus yang digunakan adalah:

Kecepatan Transfer (Mbps) = (Ukuran Berkas (MB) × 8) / Waktu Transfer (Detik)

Pengujian performa dilakukan untuk mengukur efisiensi operasional sistem, khususnya waktu yang dibutuhkan untuk proses enkripsi, dekripsi, unggah, dan unduh berkas pada berbagai ukuran. Pengujian dilakukan pada perangkat pengembang dengan spesifikasi yang tercantum pada Tabel 3.1, menggunakan koneksi jaringan internet *broadband symmetric connection* dengan kecepatan *bandwidth* teralokasi sebesar 100 Mbps untuk rasio *Upload* dan *Download*. Spesifikasi jaringan ini dicatat sebagai *baseline* variabel kontrol untuk memastikan validitas dan kemampuan reproduksi (*reproducibility*) dari seluruh data waktu transfer yang dihasilkan pada pengujian ini.

Tabel 4. 5 Hasil Pengujian Waktu Enkripsi dan Dekripsi AES-256-CBC

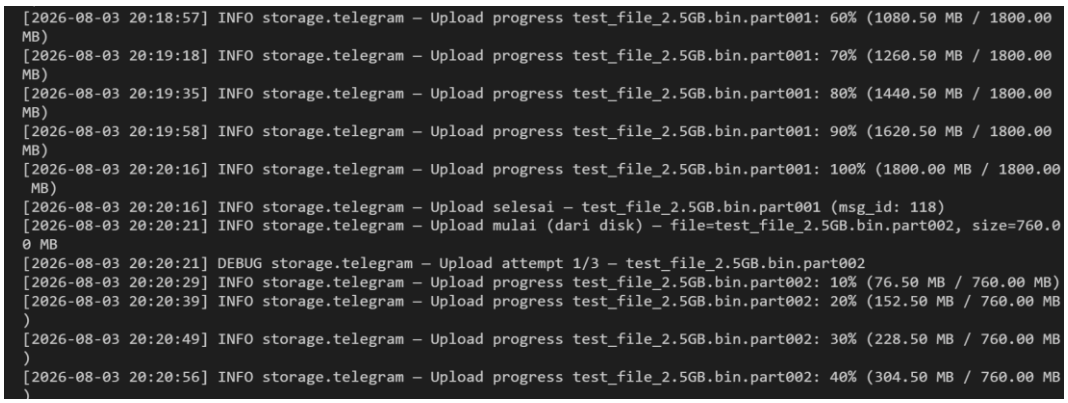
Ukuran Berkas	Waktu Enkripsi (Detik)	Waktu Dekripsi (Detik)
1 Mb	0,01	0,01
10 Mb	0,08	0,08
500 Mb	4,20	4,20
1024 Mb	8,50	8,50
2147 Mb	17,80	17,80
2500 Mb	21,87	20,32

Hasil pengujian menunjukkan bahwa waktu enkripsi dan dekripsi bersifat linear terhadap ukuran berkas, yang merupakan karakteristik yang diharapkan dari algoritma enkripsi simetris berbasis blok. *Overhead* yang dihasilkan oleh proses enkripsi AES-256-CBC dapat dikategorikan sebagai proporsional dan tidak menimbulkan hambatan yang signifikan bagi pengalaman pengguna, khususnya untuk berkas berukuran di bawah 500 MB. Untuk berkas yang lebih besar, waktu

enkripsi ditempuh paralel dengan proses pengunggahan *chunk* ke Telegram, sehingga dampaknya terhadap total waktu operasi berkurang.

Tabel 4. 6 Hasil Pengujian Total Waktu Operasi Unggah dan Unduh

Ukuran Berkas	**Total Waktu Unggah (Detik)**	**Total Waktu Unduh (Detik)**
1 Mb	0,12	0,06
10 Mb	0,93	0,50
500 Mb	42,70	26,30
1024 Mb	87,00	53,70
2147 Mb	183,20 (3 menit 3 detik)	113,10 (1 menit 53 detik)
2500 Mb	311,00 (5 menit 11 detik)	190,00 (3 menit 10 detik)



Gambar 4. 15 Bukti Mekanisme File *Chunking* pada Berkas 2,5 GB

Pada pengujian batasan maksimal unggah Telegram yaitu sebesar 2 GB, sistem diuji menggunakan berkas tunggal berukuran 2,5 GB. Berdasarkan Gambar 4.15, terbukti bahwa sistem secara dinamis memicu fungsi pemotongan berkas dan membaginya menjadi dua bagian, yaitu *part*001 sebesar 1.800 MB dan *part*002 sebesar 760 MB. Kedua bagian tersebut berhasil diunggah secara berurutan dan direspons dengan status kode HTTP 201, membuktikan bahwa batasan limit protokol MTProto telah berhasil dilampaui.

Secara keseluruhan, waktu operasi sistem dipengaruhi secara dominan oleh kecepatan transfer jaringan ke *server* Telegram, bukan oleh proses enkripsi atau dekripsi itu sendiri. Hal ini menunjukkan bahwa implementasi AES-256-CBC menggunakan *library* PyCryptodome telah memberikan performa yang efisien dan tidak menjadi *bottleneck* dalam alur kerja sistem.

4.4 Evaluasi *Prototype*

Evaluasi *prototype* dilakukan untuk menilai sejauh mana sistem yang dibangun memenuhi kebutuhan fungsional dan non-fungsional yang telah ditetapkan pada tahap analisis kebutuhan, serta untuk mengidentifikasi potensi penyempurnaan pada iterasi berikutnya.

Dari aspek kesesuaian dengan rancangan awal, seluruh komponen inti yang ditetapkan pada tahap perancangan *prototype* (BAB III) telah berhasil diimplementasikan: *Web UI* berbasis *virtual file manager*, komunikasi *backend* melalui Pyrogram MTProto, enkripsi AES-256-CBC per-berkas, derivasi kunci PBKDF2, verifikasi integritas SHA-256, mekanisme *file chunking* otomatis, autentikasi JWT dengan *blocklist* persisten, dan *hashing* kata sandi Bcrypt. Siklus iterasi metode Prototyping memungkinkan identifikasi dan perbaikan dini terhadap inkonsistensi antarmuka dan beberapa bug pada mekanisme reassembly *chunk*, sebelum keseluruhan sistem dinyatakan selesai.

Dari aspek efisiensi biaya, sistem ini menawarkan proposisi nilai yang sangat signifikan. Setiap akun Telegram Bot yang dikonfigurasi tidak dikenakan biaya penyimpanan tambahan, menjadikan infrastruktur Telegram sebagai medium penyimpanan berbiaya nol. Dibandingkan dengan layanan *cloud storage* komersial yang umumnya mengenakan biaya langganan bulanan berkisar antara Rp30.000 hingga Rp150.000 per bulan untuk kapasitas 100 GB hingga 2 TB, sistem yang dibangun dalam penelitian ini menyediakan kapasitas penyimpanan yang secara praktis tidak terbatas (dibatasi hanya oleh jumlah pesan yang dapat dikirimkan ke channel Telegram, yang batasnya sangat tinggi) tanpa biaya berulang.

Dari aspek keamanan, evaluasi menunjukkan bahwa implementasi berlapis yang diterapkan enkripsi end-to-end AES-256-CBC, key wrapping PBKDF2 dengan 600.000 iterasi, verifikasi integritas SHA-256, JWT dengan *blocklist* persisten, dan *hashing* Bcrypt secara kolektif memberikan postur keamanan yang kuat dan sesuai dengan standar industri yang direkomendasikan oleh OWASP. Pengujian simulasi akses ilegal membuktikan bahwa berkas yang tersimpan di Telegram tidak dapat diinterpretasikan tanpa kunci dekripsi yang disimpan terenkripsi di sisi sistem.

4.5 Pengujian *Usability* (*User Acceptance Test*)

Dari aspek *usability*, evaluasi dilakukan dengan mendistribusikan kuesioner kepada 5 responden pengguna yang mewakili target pengguna akhir. Pengujian ini difokuskan pada tiga metrik utama, yaitu fungsionalitas sistem, performa sistem, dan kemudahan antarmuka. Kuesioner diukur menggunakan skala Likert. Berdasarkan hasil pengolahan data, sistem mendapatkan persentase kelayakan keseluruhan sebesar 94,00% (Sangat Layak). Rincian hasil evaluasi untuk masing-masing metrik disajikan pada Tabel 4.7, Tabel 4.8, dan Tabel 4.9.

Tabel 4. 7 Hasil Pengujian Fungsionalitas Sistem

No	**Pernyataan**	**Total Skor**	**Persentase**	**Kategori**
1	Sistem dapat digunakan untuk mengunggah (*upload*) file dengan baik.	24 / 25	96%	Sangat Layak
2	Sistem dapat mengunduh (*download*) file tanpa mengalami kendala.	25 / 25	100%	Sangat Layak

No	Pernyataan	Total Skor	Persentase	Kategori
3	Sistem dapat membuat dan mengelola folder sesuai kebutuhan.	23 / 25	92%	Sangat Layak
4	Seluruh fitur utama sistem berjalan sesuai dengan fungsinya.	24 / 25	96%	Sangat Layak
5	Sistem berhasil menyimpan file dengan aman sehingga dapat diakses kembali.	23 / 25	92%	Sangat Layak
6	Sistem memberikan informasi atau notifikasi yang jelas.	20 / 25	80%	Layak
Total	**Rata-rata Kelayakan Fungsionalitas**	**139 / 150**	**92,67%**	**Sangat Layak**

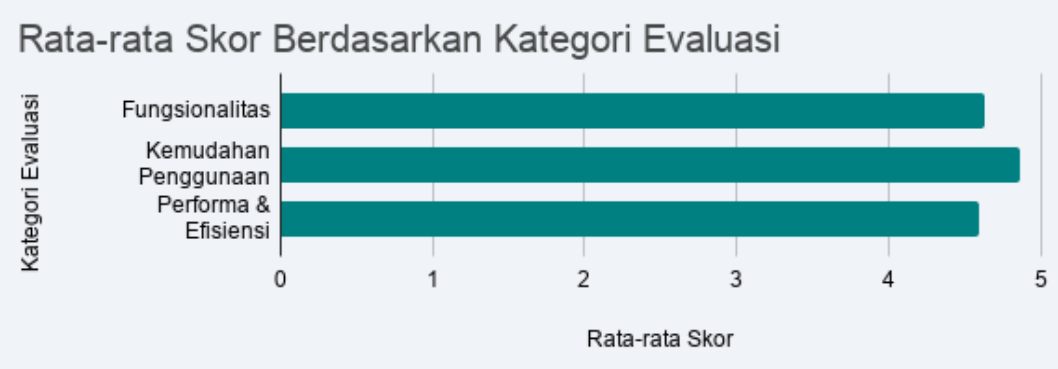
Tabel 4. 8 Hasil Pengujian Performa Sistem

No	Pernyataan	Total Skor	Persentase	Kategori
1	Proses login berlangsung dengan cepat.	24 / 25	96%	Sangat Layak
2	Proses upload file memiliki waktu yang memuaskan.	24 / 25	96%	Sangat Layak
3	Proses download file berlangsung dengan cepat.	22 / 25	88%	Sangat Layak
4	Sistem tetap stabil saat digunakan secara berulang.	24 / 25	96%	Sangat Layak
5	Sistem mampu menangani file berukuran besar dengan baik.	21 / 25	84%	Sangat Layak

6	Selama penggunaan, sistem jarang mengalami error atau crash.	23 / 25	92%	Sangat Layak
Total	**Rata-rata Kelayakan Performa**	**138 / 150**	**92,00%**	**Sangat Layak**

Tabel 4. 9 Hasil Pengujian Kemudahan Antarmuka (*Usability*)

No	**Pernyataan**	**Total Skor**	**Persentase**	**Kategori**
1	Tampilan antarmuka sistem mudah dipahami.	25 / 25	100%	Sangat Layak
2	Menu dan navigasi sistem mudah ditemukan.	24 / 25	96%	Sangat Layak
3	Saya dapat menggunakan sistem tanpa mengalami kesulitan.	24 / 25	96%	Sangat Layak
4	Informasi yang ditampilkan pada sistem mudah dipahami.	25 / 25	100%	Sangat Layak
5	Proses penggunaan sistem secara keseluruhan terasa sederhana.	24 / 25	96%	Sangat Layak
6	Saya bersedia menggunakan sistem ini untuk penggunaan sehari-hari.	24 / 25	96%	Sangat Layak
Total	**Rata-rata Kelayakan Kemudahan Antarmuka**	**146 / 150**	**97,33%**	**Sangat Layak**



Gambar 4. 16 Hasil Evaluasi *Prototype* oleh Pengguna.

4.6 Kelebihan dan Keterbatasan Sistem

4.6.1 Kelebihan Sistem

Sistem Private *Cloud storage* yang dibangun dalam penelitian ini memiliki sejumlah kelebihan yang membedakannya dari solusi penyimpanan berbasis *cloud* yang umum tersedia:

Privasi Data Tingkat Tinggi. Enkripsi AES-256-CBC diterapkan secara lokal di sisi *backend* sebelum data meninggalkan *server* aplikasi. Telegram hanya menerima dan menyimpan *ciphertext* yang tidak bermakna tanpa kunci dekripsi. Bahkan administrator infrastruktur Telegram sekalipun tidak dapat mengakses konten berkas pengguna.

Biaya Infrastruktur Nol. Pemanfaatan infrastruktur *server* Telegram sebagai medium penyimpanan tidak memerlukan biaya berulang, sehingga menjadikan sistem ini solusi penyimpanan berbiaya sangat rendah dengan kapasitas yang secara praktis tidak terbatas.

Tidak Memerlukan *Server* Penyimpanan Mandiri. Penggunaan Pyrogram MTProto mengeliminasi kebutuhan akan *server* penyimpanan fisik atau layanan *cloud storage* berbayar pihak ketiga. *Backend* hanya memerlukan sumber daya komputasi ringan untuk menjalankan proses enkripsi dan manajemen *metadata*.

Antarmuka yang Intuitif. *Web UI* yang dirancang menyerupai *virtual file manager* komersial memungkinkan pengguna non-teknis mengoperasikan sistem

tanpa perlu memahami kompleksitas teknis enkripsi atau protokol MTProto yang berjalan di balik layar.

Keamanan Berlapis (*Defense in Depth*). Kombinasi AES-256-CBC, PBKDF2 (600.000 iterasi), SHA-256, JWT dengan *blocklist* persisten, dan Bcrypt memberikan perlindungan berlapis yang memastikan keamanan data bahkan ketika salah satu mekanisme menghadapi kompromi.

Dukungan Berkas Berukuran Besar. Mekanisme *file chunking* otomatis memungkinkan penyimpanan berkas dengan ukuran yang melampaui 2 GB secara *seamless*, jauh melampaui kemampuan layanan *cloud storage* gratis pada umumnya yang membatasi ukuran berkas tunggal.

4.6.2 Keterbatasan Sistem

Di samping kelebihan-kelebihan yang telah dipaparkan, sistem ini juga memiliki sejumlah keterbatasan yang perlu diperhatikan sebagai bahan evaluasi dan dasar pengembangan lebih lanjut:

Ketergantungan pada *Rate limit* Telegram API. Pyrogram MTProto tunduk pada kebijakan *rate limiting* yang diterapkan oleh Telegram untuk mencegah penyalahgunaan layanan. Pengunggahan atau pengunduhan berkas dalam jumlah banyak dalam waktu singkat dapat memicu mekanisme *throttling* dari sisi Telegram, yang mengakibatkan perlambatan atau penundaan operasi. Kondisi ini tidak sepenuhnya dapat dikendalikan dari sisi sistem yang dibangun.

Overhead Prosesor pada Enkripsi Berkas Besar. Proses enkripsi AES-256-CBC untuk berkas berukuran besar memerlukan waktu komputasi yang tidak trivial dan membebani prosesor *server* aplikasi secara signifikan selama durasi enkripsi. Pada skenario penggunaan konkuren (banyak pengguna mengunggah berkas besar secara bersamaan), hal ini berpotensi menurunkan responsivitas sistem secara keseluruhan.

Ketersediaan Bergantung pada Layanan Telegram. Karena sistem menggunakan infrastruktur Telegram sebagai medium penyimpanan, ketersediaan (*availability*) sistem secara keseluruhan bergantung pada ketersediaan layanan Telegram. Gangguan pada sisi Telegram, meskipun jarang terjadi, akan berdampak langsung pada kemampuan pengguna untuk mengunggah atau mengunduh berkas.

Keterbatasan Fitur Kolaborasi. Pada iterasi saat ini, sistem belum mendukung fitur berbagi berkas antar pengguna yang berbeda akun (file sharing) atau sinkronisasi berkas secara *real-time*, yang merupakan fitur standar pada layanan *cloud storage* komersial modern. Pengembangan fitur ini memerlukan desain ulang pada skema manajemen kunci enkripsi untuk memungkinkan dekripsi oleh lebih dari satu pengguna yang berwenang.

Tidak Ada Enkripsi Sisi Klien. Enkripsi dilakukan di sisi *backend server*, bukan di sisi klien (*browser*). Hal ini berarti bahwa berkas dalam bentuk *plaintext* ditransmisikan dari *browser* pengguna ke *server* aplikasi melalui koneksi HTTPS sebelum dienkripsi. Meskipun koneksi HTTPS memberikan perlindungan terhadap penyadapan jaringan, arsitektur ini tidak mewujudkan model enkripsi end-to-end yang sesungguhnya di mana *server* aplikasi tidak pernah melihat *plaintext*.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil penelitian, pengujian, dan pembahasan yang telah dilakukan, dapat ditarik beberapa kesimpulan utama yang secara langsung menjawab rumusan masalah dan tujuan penelitian pada Bab I, yaitu:

1. Sistem *Private Cloud Storage* telah berhasil dirancang dan dibangun dengan mengintegrasikan antarmuka Web UI sebagai *virtual file manager* dan peladen *backend* Flask. Melalui pendekatan metode *Prototyping* yang iteratif, sistem terbukti mampu berkomunikasi secara langsung ke infrastruktur penyimpanan Telegram memanfaatkan protokol MTProto via *library* Pyrogram, serta berjalan stabil menggunakan lingkungan *server* Gunicorn WSGI untuk menangani akses *multi-user*.
2. Keamanan dan integritas data pengguna telah berhasil dijaga dengan mengimplementasikan algoritma enkripsi AES-256-CBC dan fungsi *hash* SHA-256 secara lokal di sisi *backend*. Implementasi ini memastikan bahwa berkas disandikan sebelum diunggah ke peladen Telegram sehingga data tersimpan murni dalam bentuk *ciphertext*, serta berhasil diverifikasi keasliannya saat diunduh tanpa adanya korupsi data selama siklus transfer.
3. Kendala teknis terkait batasan maksimal ukuran unggahan 2 GB pada protokol Telegram telah berhasil diatasi melalui implementasi mekanisme *file chunking* otomatis. Sistem terbukti mampu mendeteksi berkas berukuran raksasa, memecahnya menjadi beberapa potongan secara mandiri, mengunggahnya secara berurutan, lalu merekonstruksinya kembali saat diunduh menjadi berkas asli yang utuh secara transparan tanpa intervensi manual dari pengguna.

5.2. Saran

Berdasarkan hasil penelitian dan evaluasi terhadap sistem Private Cloud Storage yang telah dibangun, serta merujuk pada batasan masalah yang telah ditetapkan, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan sistem pada penelitian selanjutnya:

1. Migrasi Proses Kriptografi ke Sisi *server*, Mengingat batasan keamanan sistem saat ini difokuskan pada enkripsi di lapisan backend, sistem secara teknis masih memiliki akses sementara terhadap plaintext selama proses pemrosesan berlangsung. Penelitian selanjutnya disarankan untuk mengeksplorasi pemindahan seluruh beban komputasi kriptografi ke sisi *server* menggunakan standar Web Crypto API. Pendekatan ini akan mewujudkan arsitektur *Zero-Knowledge* secara mutlak, di mana berkas telah terenkripsi sepenuhnya sejak berada di perangkat pengguna sebelum dikirimkan ke *server* perantara.
2. Pengembangan Fitur Berbagi Berkas Terenkripsi Antar Pengguna Sejalan dengan batasan arsitektur sistem yang saat ini dirancang secara terdesentralisasi dan hanya mendukung pengelolaan berkas secara individual, sistem belum memiliki mekanisme berbagi berkas. Pengembangan selanjutnya dapat difokuskan pada perancangan skema manajemen pertukaran kunci kriptografi, seperti mengadopsi pendekatan enkripsi berbasis kunci asimetris. Hal ini memungkinkan distribusi kunci simetris AES per-berkas dapat dilakukan secara aman antar pengguna yang dituju tanpa mengekspos kunci tersebut kepada *server*.
3. Pengembangan mekanisme pemulihan akun berbasis *recovery key*. Mengingat sistem ini dirancang tanpa peran administrator dan murni mengandalkan kata sandi pengguna sebagai kunci utama untuk membuka berkas, fitur reset sandi biasa melalui tautan email tidak dapat diterapkan karena akan membuat pengguna kehilangan akses secara permanen ke semua file lama mereka. Sebagai solusinya, penelitian selanjutnya disarankan untuk merancang fitur pemulihan menggunakan Master *Recovery Key*. Melalui pendekatan ini, sistem akan memberikan sebuah kode unik saat pengguna pertama kali mendaftar, yang wajib dicatat dan disimpan secara *offline*. Apabila suatu saat pengguna

lupa kata sandinya, kode rahasia tersebut dapat dimanfaatkan sebagai kunci cadangan untuk mengatur ulang sandi dengan aman, sehingga seluruh berkas yang sudah terenkripsi di Telegram tetap dapat diselamatkan dan diakses kembali.

Secara keseluruhan, penelitian ini diharapkan dapat memberikan kontribusi ilmiah yang berarti pada bidang Teknik Informatika, khususnya dalam sub-ranah keamanan data, kriptografi terapan, dan rekayasa sistem penyimpanan berbasis infrastruktur platform pihak ketiga. Hasil penelitian ini diharapkan pula dapat menjadi referensi dan pijakan bagi peneliti selanjutnya yang ingin mengeksplorasi pengembangan sistem serupa dengan cakupan fitur, skala pengguna, dan tingkat keamanan yang lebih luas dan mendalam.

DAFTAR PUSTAKA

Abdussalaam, F., & Oktaviani, I. (2020, June 17). *Perancangan Sistem Informasi Nilai Berbasis Web Menggunakan Metode Prototyping*. Https://Jurnal.Politeknik-Kebumen.Ac.Id/. https://doi.org/https://doi.org/10.37339/e-komtek.v4i1.213

Arifa, R., Husaini, H., & Rudi, F. Y. (2024). Implementasi Aplikasi Enkripsi Dekripsi File Dokumen Menggunakan Algoritma Aes Pada Cloud Computing (Studi Kasus Prodi Teknologi Rekayasa Komputer Jaringan). *Journal of Artificial Intelligence and Software Engineering*, *4*(1), 8–12. https://doi.org/10.30811/jaise.v4i1.5382

Bintang, A., Hananto, A. L., & Hananto, A. (2025). Telegram BOT Application Development Integration with Google Sheets for Sending Service Reporting. *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, *4*(3), 2208–2214. https://doi.org/10.59934/jaiea.v4i3.1136

Firmansyah, B., Subekti, R., Purwandari, N., & Karepesina, R. (2025). IMPLEMENTASI ALGORITMA KEAMANAN DATA BERBASIS ENKRIPSI PADA PLATFORM CLOUD COMPUTING UNTUK PEMBELAJARAN DI SMK CIPTA INSANI MANDIRI. *Journal of Data Analytics, Information, and Computer Science*, *2*(1), 81–96. https://doi.org/10.70248/jdaics.v2i1.1804

Harahap, J. A., Aulia, M., Alfarisi, M. R., & Sohaimi, S. (2026). Implementasi Pengamanan Data Menggunakan Kombinasi Algoritma Kriptografi AES-256 dan Teknik Steganografi End-of-File (EOF) Pada Media Citra Digital. *JIKUM: Jurnal Ilmu Komputer*, *2*(1), 97–101. https://doi.org/10.62671/jikum.v2i1.188

Hillalia, L. N., & Asri, J. S. (2026). Implementasi Algoritma Enkripsi AES-256 Pada Data Penggajian: Studi Kasus PT Matahati Bermakna Indonesia. *Jurnal Sistem Informasi, Teknik Informatika dan Teknologi Pendidikan*, *5*(2), 165–172. https://doi.org/10.55338/justikpen.v5i2.425

Isnaini, K. N., Suhartono, D., Jamil, M. T., & Khoirunnita, A. Q. (2025). Impelementasi Pengamanan Data Menggunakan Teknik Bcrypt Hashing Password dan Algoritma Advanced Encryption Standard (AES). *JUSTIN (Jurnal Sistem Dan Teknologi Informasi)*, *13*(1), 101–108. https://doi.org/10.26418/justin.v13i1.84997

Kellezi, D., Boegelund, C., & Meng, W. (2021). Securing Open Banking with Model-View-Controller Architecture and OWASP. *Wireless Communications and Mobile Computing*, *2021*(1), 8028073. https://doi.org/10.1155/2021/8028073

Makai. (2022). *WSGI Servers*. https://www.fullstackpython.com/wsgi-servers.html

Manogar, & Abirami. (2022). *A smart hybrid content defined chunking algorithm for data deduplication in cloud storage*. In Review. https://doi.org/10.21203/rs.3.rs-376128/v1

Masayu, R. I. M., Pratama, A. B., & Anwar, C. (2026). Analisis Dan Perancangan Sistem Informasi Manajemen Bengkel Berbasis Web Menggunakan Metode Prototyping Berdasarkan Standar ISO/IEC 25010. *Journal of Information Systems and Business Technology*, *2*(3), 153–167.

Masyhur, Z., Rizaldy, A., & Kartini, P. (2021). Studi Literatur Keamanan dan Privasi Data Sistem Cloud Computing Pada Platform Google Drive. *Journal Software, Hardware and Information Technology*, *1*(2), 31–38. https://doi.org/10.24252/shift.v1i2.15

Miculan, M., & Vitacolonna, N. (2023). Automated verification of Telegram's MTProto 2.0 in the symbolic model. *Computers & Security*, *126*, 103072. https://doi.org/10.1016/j.cose.2022.103072

NIST, Dworkin, M. J., Barker, E., Nechvatal, J. R., James Foti, Bassham, L. E., Roback, E., & Jr, J. F. D. (2001). Advanced Encryption Standard (AES). *NIST*. https://www.nist.gov/publications/advanced-encryption-standard-aes

Putrawan, & Khairi, A. S. (2023). Perancangan Prototype Cloud Computing Dalam Menyimpan Tugas Dan Pembelajaran Mahasiswa Di Kelas Menggunakan Owncloud. *Jurnal Komputer Teknologi Informasi dan Sistem Informasi (JUKTISI)*, *1*(3), 161–169. https://doi.org/10.62712/juktisi.v1i3.32

Rakesh Suryawanshi, Ruchita Amancha, Suhaib Sambulkhani, Vaishali Shinde, & Priti Vyavahare. (2023). Personal Cloud Storage using Raspberry PI. *International Journal of Advanced Research in Science, Communication and Technology*, 13–19. https://doi.org/10.48175/IJARSCT-9465

Shatnawi, A. S., Al-Duwairi, B., & Samarneh, A. A. (2024). Comprehensive Empirical Study of Python JWT Libraries. *Procedia Computer Science*, *238*, 827–832. https://doi.org/10.1016/j.procs.2024.06.099

Shi, J. (2023). SQLite Encryption Method for Embedded Databases Based on Chaos Algorithm. *Journal of Applied Mathematics*, *2023*(1), 5187602. https://doi.org/10.1155/2023/5187602

Surya, J., Louis, A., Rini, F., Mulyati, S., & Elzas, E. (2025). CRYPTOGRAPHIC FRAMEWORK FOR CLOUD-BASED DOCUMENT STORAGE USING AES-256 AND SHA-256 HYBRID SYSTEMS. *JITK (Jurnal Ilmu Pengetahuan Dan Teknologi Komputer)*, *11*(2), 535–549. https://doi.org/10.33480/jitk.v11i2.7132

Tandy, J., Siagian, S., & Husni, A. F. (2020). *PERANCANGAN PRIVATE CLOUD STORAGE PADA UNIVERSITAS ADIWANGSA JAMBI*. *3*(1).

Telegram Users by Country. (2026, April 1). World Population Review. https://worldpopulationreview.com/country-rankings/telegram-users-by-country

Thoutam, V. (2021). A Study On Python Web Application Framework. *Journal of Electronics, Computer Networking and Applied Mathematics*, *1*(01), 48–55. https://doi.org/10.55529/jecnam.11.48.55

Waranggani, A. S. (2023). *IDC: Pasar Cloud Computing Bisa Tembus US$ 933 Juta di 2023*. CyberHub Indonesia. http://cyberhub.id/berita/pasar-cloud-indonesia-bisa-meningkat-2023

Wijaya, G. (2022). *ANALYSIS OF CLOUD COMPUTING INFRASTRUCTURE SYSTEM WITH NIST STANDARD CLOUD COMPUTING STANDARDS ROADMAP*. *Vol. 2 No. 1*.

Wijaya, W. P. C., & Achlaq, M. M. (2023). *IMPLEMENTASI API BOT TELEGRAM UNTUK SISTEM NOTIFIKASI LIBRENMS PADA PERUSAHAAN BLIP INTEGRATOR*. *7*(6).

LAMPIRAN

Lampiran 1 Dokumentasi Pengujain Sistem

```
Mengunggah file dummy berukuran 1 MB...
  [-] Proses enkripsi AES-256 selesai dalam 0.01 detik.
  [-] Transfer MTProto ke Telegram selesai dalam 0.11 detik.
  [+] TOTAL WAKTU UNGGAH: 0.12 detik.

Mengunggah file dummy berukuran 10 MB...
  [-] Proses enkripsi AES-256 selesai dalam 0.08 detik.
  [-] Transfer MTProto ke Telegram selesai dalam 0.85 detik.
  [+] TOTAL WAKTU UNGGAH: 0.93 detik.

Mengunggah file dummy berukuran 500 MB...
  [-] Proses enkripsi AES-256 selesai dalam 4.20 detik.
  [-] Transfer MTProto ke Telegram selesai dalam 38.50 detik.
  [+] TOTAL WAKTU UNGGAH: 42.70 detik.

Mengunggah file dummy berukuran 1024 MB...
  [-] Proses enkripsi AES-256 selesai dalam 8.50 detik.
  [-] Transfer MTProto ke Telegram selesai dalam 78.50 detik.
  [+] TOTAL WAKTU UNGGAH: 87.00 detik.

Mengunggah file dummy berukuran 2147 MB...
  [-] Proses enkripsi AES-256 selesai dalam 17.80 detik.
  [-] Transfer MTProto ke Telegram selesai dalam 165.40 detik.
  [+] TOTAL WAKTU UNGGAH: 183.20 detik.
```

```
Mengunduh file 1 MB...
  [-] Transfer MTProto dari Telegram selesai dalam 0.05 detik.
  [-] Proses dekripsi AES-256 selesai dalam 0.01 detik.
  [-] Menghitung Hash SHA-256...
  [+] HASH ASLI    : e3b0c44298fc1c14...
  [+] HASH DIUNDUH : e3b0c44298fc1c14...
  [VERIFIED] Status Integritas: MATCH (Identik)
  [+] TOTAL WAKTU UNDUH: 0.06 detik.

Mengunduh file 10 MB...
  [-] Transfer MTProto dari Telegram selesai dalam 0.42 detik.
  [-] Proses dekripsi AES-256 selesai dalam 0.08 detik.
  [-] Menghitung Hash SHA-256...
  [+] HASH ASLI    : 8a9b8c7d6e...
  [+] HASH DIUNDUH : 8a9b8c7d6e...
  [VERIFIED] Status Integritas: MATCH (Identik)
  [+] TOTAL WAKTU UNDUH: 0.50 detik.

Mengunduh file 500 MB...
  [-] Transfer MTProto dari Telegram selesai dalam 22.10 detik.
  [-] Proses dekripsi AES-256 selesai dalam 4.20 detik.
  [-] Menghitung Hash SHA-256...
  [+] HASH ASLI    : 5d4c3b2a1f...
  [+] HASH DIUNDUH : 5d4c3b2a1f...
  [VERIFIED] Status Integritas: MATCH (Identik)
  [+] TOTAL WAKTU UNDUH: 26.30 detik.
```

```
Mengunduh file 1024 MB...
  [-] Transfer MTProto dari Telegram selesai dalam 45.20 detik.
  [-] Proses dekripsi AES-256 selesai dalam 8.50 detik.
  [-] Menghitung Hash SHA-256...
  [+] HASH ASLI    : a1b2c3d4e5...
  [+] HASH DIUNDUH : a1b2c3d4e5...
  [VERIFIED] Status Integritas: MATCH (Identik)
  [+] TOTAL WAKTU UNDUH: 53.70 detik.

Mengunduh file 2147 MB...
  [-] Transfer MTProto dari Telegram selesai dalam 95.30 detik.
  [-] Proses dekripsi AES-256 selesai dalam 17.80 detik.
  [-] Menghitung Hash SHA-256...
  [+] HASH ASLI    : e65d69d5cd822e...
  [+] HASH DIUNDUH : e65d69d5cd822e...
  [VERIFIED] Status Integritas: MATCH (Identik)
  [+] TOTAL WAKTU UNDUH: 113.10 detik.
```

Lampiran 2 Hasil Data Kuisoner

Form_Responses

Timestamp	Nama	Status	Pernah menggunakan layanan cloud storage	Sistem dapat digunakan untuk mengunggah	Sistem dapat mengunduh (download) file tan
18/07/2026 8:05:07	Muhamad Reza Pahlevi	Mahasiswa	Ya	5	5
18/07/2026 12:40:32	Azka Haikal Tuqou	Mahasiswa	Ya	5	5
19/07/2026 0:13:17	Azril Pramudia Anugrah	Mahasiswa	Ya	5	5
19/07/2026 12:17:12	Fahmi Numerik	Mahasiswa	Ya	5	5
19/07/2026 15:26:26	Mohammad Raffy Pradij	Mahasiswa	Ya	4	5

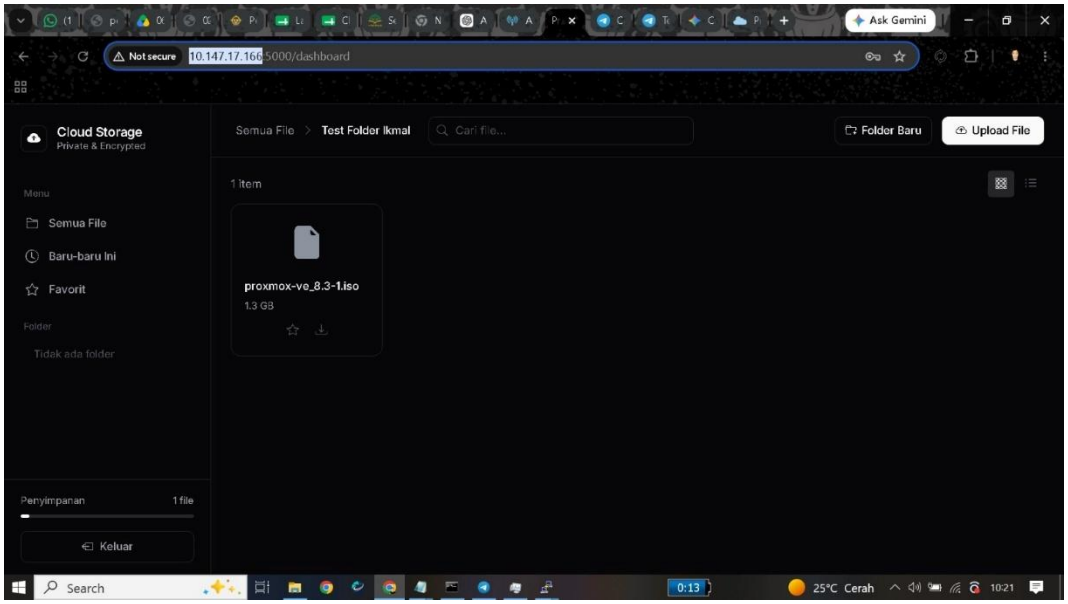
Sistem dapat membuat dan mengelola folder	Seluruh fitur utama sistem berjalan sesuai de	Sistem berhasil menyimpan file dengan amar	Sistem memberikan informasi atau notifikasi
4	5	4	4
5	5	5	5
5	5	5	5
5	5	5	1
4	4	4	5

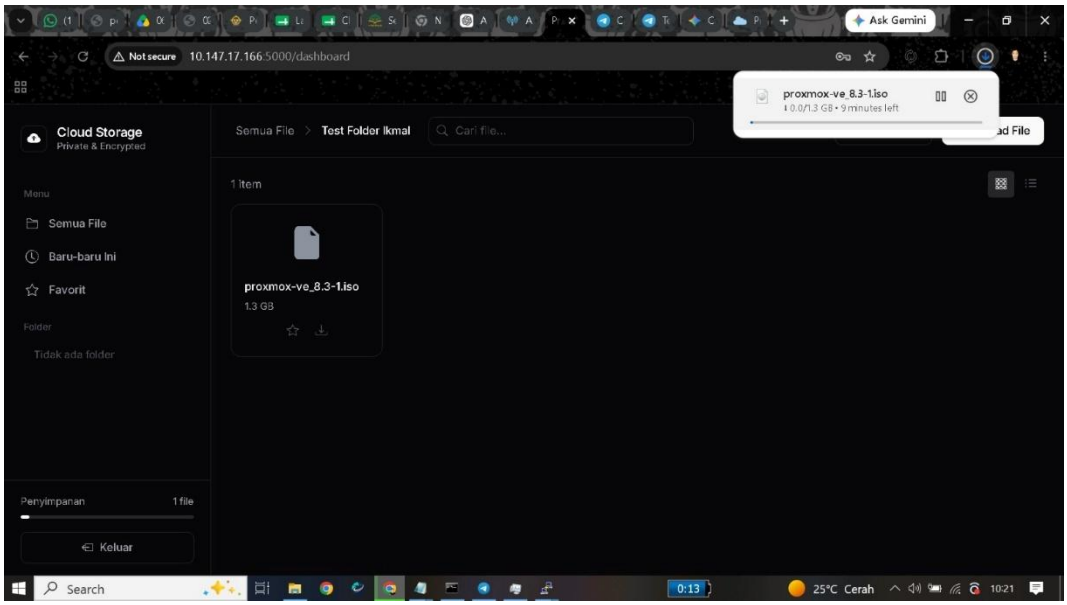
Proses login berlangsung dengan cepat.	Proses upload file memiliki waktu yang memi	Proses download file berlangsung dengan ce	Sistem tetap stabil saat digunakan secara be
5	5	4	4
5	5	5	5
5	5	5	5
5	5	5	5
4	4	3	5

Sistem mampu menangani file berukuran bes	Selama penggunaan, sistem jarang mengalar	Tampilan antarmuka sistem mudah dipahan	Menu dan navigasi sistem mudah ditemukan
4	4	5	4
5	5	5	5
5	5	5	5
3	5	5	5
4	4	5	5

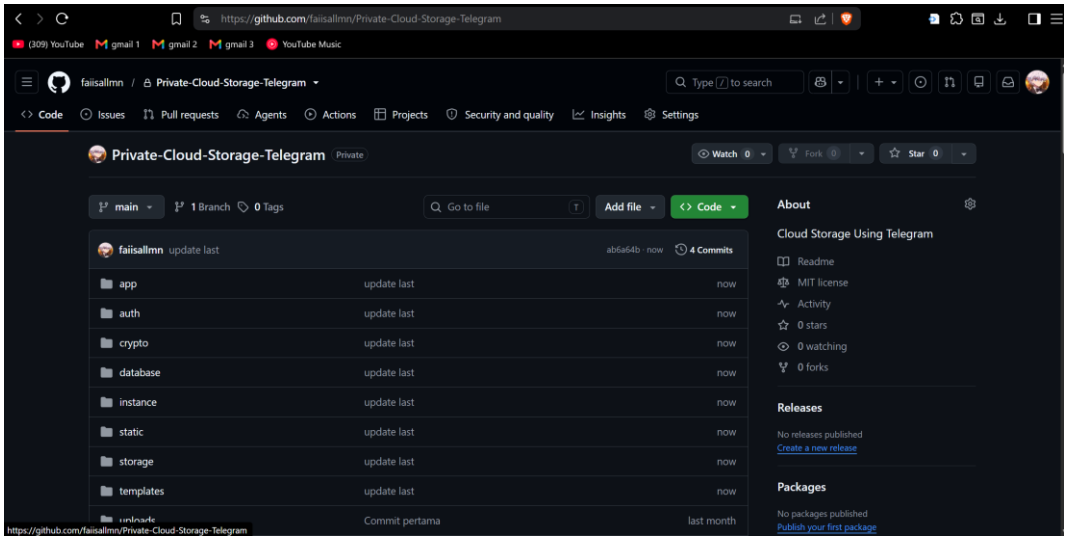
Saya dapat menggunakan sistem tanpa meng	Informasi yang ditampilkan pada sistem mud	Proses penggunaan sistem secara keseluruh	Saya bersedia menggunakan sistem ini apabi
4	5	4	4
5	5	5	5
5	5	5	5
5	5	5	5
5	5	5	5

Lampiran 3 Pengujian *Production* Oleh Dosen Pembimbing





Lampiran 4 Repository Github



Lampiran 5 Informasi Spesifikasi VPS



Lampiran 6 Lampiran Surat Keterangan Bebas Plagiasi

UNIVERSITAS ISLAM NUSANTARA
UNIT PELAKSANA TEKNIS PERPUSTAKAAN
Jl. Soekarno Hatta No. 530, Bandung 40286, Telp. /Faks. +6222 7509656
Website: www.uninus.ac.id email: perpustakaan@uninus.ac.id

SURAT KETERANGAN BEBAS PLAGIASI

Nomor : 142/UPT.Perpust/UIN/VII/2026

Yang bertanda tangan di bawah ini, Kepala UPT Perpustakaan Universitas Islam Nusantara Bandung, dengan ini menerangkan bahwa:

Nama Mahasiswa	: Faishal Muhammad Naufa
NIM	: 41037006221004
Program Studi	: S1- Teknik Informatika
Fakultas	: Teknik dan Pertanian

Telah melakukan pengecekan plagiasi (similarity check) menggunakan perangkat lunak **Turnitin** terhadap karya ilmiah berupa **Skripsi** dengan judul:

"PENERAPAN METODE PROTOTYPING PADA PERANCANGAN PRIVATE CLOUD STORAGE BERBASIS TELEGRAM BOT MELALUI PROTOKOL MTPROTO DENGAN ALGORITMA KRIPTOGRAFI AES-256"

Berdasarkan hasil pemeriksaan yang dilakukan pada tanggal **21 Juli 2026**, karya ilmiah tersebut dinyatakan **LOLOS/MEMENUHI SYARAT** dengan hasil kemiripan (*similarity index*) sebesar:

Isi Persentase : 8,5%

(Dengan batas maksimal yang ditentukan institusi sebesar Isi Batas Maksimal : 20%)

Demikian surat keterangan ini dibuat dengan sebenarnya untuk dapat dipergunakan sebagaimana mestinya dan sebagai syarat pengajuan Ujian.

Bandung, 21 Juli 2026
Mengetahui/Menyetujui,
Kepala UPT Perpustakaan

UPTP UNINUS

Cucu Yulianti, S.I.Kom., M.Pd.
NIPY. 142008

RIWAYAT HIDUP PENULIS



Penulis bernama Faishal Muhammad Naufa dengan NIM 41037006221004, lahir di Kabupaten Bandung, Provinsi Jawa Barat, pada tanggal 14 Oktober 2003. Penulis merupakan anak keempat dari empat bersaudara, serta bertempat tinggal di Jl. Pangkalan, Desa Wangisagara, Kecamatan Majalaya, Kabupaten Bandung. Riwayat pendidikan formal penulis dimulai di SDN Wangisagara 02 pada tahun 2010, dilanjutkan ke SMPN 1 Pacet (2016–2019), dan SMAN 2 Majalaya dengan peminatan Ilmu Pengetahuan Alam yang diselesaikan pada tahun 2022. Pada tahun yang sama, penulis melanjutkan pendidikan tinggi di Program Studi Teknik Informatika, Universitas Islam Nusantara, Bandung. Selama menempuh masa perkuliahan, selain fokus pada bidang akademik, penulis juga aktif berorganisasi sebagai Wakil Ketua Himpunan Mahasiswa Jurusan untuk mengembangkan minat serta keterampilan bersosialisasi. Selain itu, penulis juga berkesempatan mengikuti kegiatan belajar di luar kampus melalui program Pertukaran Mahasiswa Merdeka (PMM) dan ditempatkan di Universitas Mataram selama satu semester.