BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Desain penelitian ini menggunakan metode *Design and Development Research (DDR)* yang dibagi menjadi tiga tahap utama sesuai dengan milik Mariappan, P dkk. (2022) yaitu *Need Analysis, Design & Development,* dan *Testing & Evaluation* seperti pada Tabel 3.1.

Tabel 3.1. *Design and Development Research* (DDR)

Tahapan	Tujuan
Need Analysis	Mengidentifikasi masalah utama dan kebutuhan penelitian terkait pengamanan data dalam komunikasi digital.
Design & Development	Merancang dan mengembangkan aplikasi website berbasis Laravel yang mengintegrasikan kriptografi *AES-256* dan steganografi pada media PDF.
Testing & Evaluation	Mengevaluasi kinerja sistem berdasarkan keamanan, fungsionalitas, dan performa website.

Tabel 3.1 menguraikan tiga tahapan utama dalam proses penelitian dan pengembangan. Tahapan pertama adalah *Need Analysis* (Analisis Kebutuhan), yang bertujuan untuk mengidentifikasi masalah utama dan kebutuhan penelitian terkait pengamanan data dalam komunikasi digital. Dalam melaksanakan tahapan *Design & Development* (Perancangan & Pengembangan), proyek ini akan mengadopsi model pengembangan perangkat lunak *Waterfall* (Wahid, 2020) khusus untuk pembangunan *website* berbasis Laravel. Model *Waterfall* dipilih karena sifatnya yang linier dan sekuensial, di mana setiap fase harus diselesaikan sepenuhnya sebelum melangkah ke fase berikutnya. Pendekatan ini sangat cocok untuk proyek dengan persyaratan antarmuka pengguna dan alur kerja yang telah didefinisikan dengan jelas di awal. Model ini akan membantu memastikan bahwa perancangan dan pengembangan aplikasi *website* Laravel dilakukan secara terstruktur dan sistematis, guna mengintegrasikan kriptografi AES-256 dan steganografi pada media PDF secara efektif. Terakhir, tahapan ketiga adalah *Testing & Evaluation*

(Pengujian & Evaluasi), di mana kinerja sistem akan dievaluasi berdasarkan aspek keamanan, fungsionalitas, dan performa *website*.

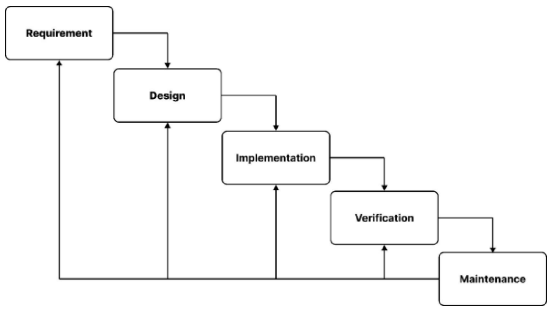
3.2 *Need Analysis*

Tahapan ini dimulai dengan analisis terhadap permasalahan yang dihadapi dalam pengamanan data digital. Permasalahan utama yang dibahas adalah bagaimana memastikan keamanan pesan rahasia menggunakan kombinasi teknik kriptografi dan steganografi, memanfaatkan dokumen PDF sebagai media cover untuk menyimpan data tanpa merusak struktur visual, serta mengembangkan aplikasi web untuk memudahkan pengguna dalam menggunakan sistem pengamanan data. Kemudian, menjaga keseimbangan antara tingkat keamanan, kapasitas penyisipan data, dan performa aplikasi website.

Untuk memperjelas permasalahan, dilakukan kajian literatur terhadap penelitian sebelumnya terkait steganografi pada media PDF, kriptografi *AES-256*, dan pengembangan aplikasi web berbasis Laravel. Dari analisis tersebut, didapatkan tujuan penelitian untuk mengembangkan aplikasi website yang mengintegrasikan kriptografi *AES-256* dan steganografi pada media PDF untuk keamanan data.

3.3 *Design and Development*

Tahapan kedua ini berfokus pada perancangan dan pengembangan aplikasi *website* berbasis Laravel yang mengintegrasikan kriptografi AES-256 dan steganografi pada media PDF. Mengikuti tahapan metode *waterfall* secara rinci milik Wahid (2020) untuk pembangunan *website* Laravel, proses pengembangan ini terbagi menjadi beberapa fase utama seperti pada Gambar 3.1.



Gambar 3.1. Desain Metode *Waterfall* untuk Pengembangan Sistem

3.3.1 *Requirement*

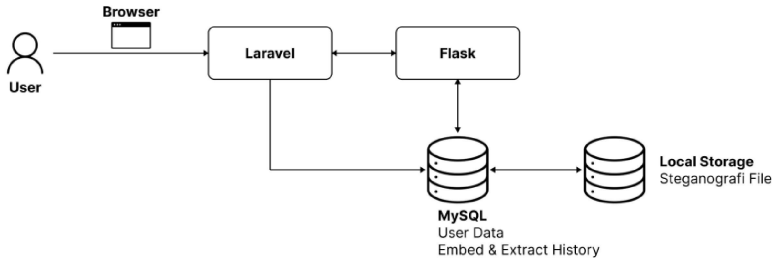
Fase Kebutuhan (*Requirement*) dalam pembangunan *website* Laravel berpusat pada pengumpulan dan analisis mendalam terhadap persyaratan antarmuka pengguna (*user interface*) dan pengalaman pengguna (*user experience*), serta bagaimana alur interaksi akan berjalan di dalam *website*. Tahap ini mencakup perincian spesifikasi fungsionalitas *frontend*, seperti sistem autentikasi pengguna (*login* dan *register*), fitur manajemen *file*, serta fungsi-fungsi pada halaman embed dan ekstrak untuk setiap metode steganografi. Selain itu, pada fase ini juga didefinisikan secara jelas bagaimana *website* akan berkomunikasi dengan *backend* (Flask API).

3.3.2 *Design*

Setelah persyaratan *website* Laravel sepenuhnya dipahami, Fase Desain (*Design*) dimulai dengan tujuan merencanakan struktur dan tampilan aplikasi web secara terperinci. Fase ini merupakan bagian krusial dari proses desain keseluruhan, di mana semua aspek teknis dan fungsional *website* direncanakan secara mendalam. Perencanaan ini meliputi Desain Arsitektur Sistem, yang merancang bagaimana *frontend* Laravel akan berinteraksi dengan *backend* Flask API dan sistem penyimpanan data, sekaligus membantu menentukan persyaratan perangkat keras dan sistem secara umum. Kemudian, Perancangan *Use Case Diagram* dilakukan untuk memvisualisasikan interaksi pengguna dengan *website* melalui berbagai fungsionalitas yang disediakan Laravel. Perancangan Basis Data fokus pada pemodelan skema *database* untuk pengelolaan pengguna dan data operasional *website* seperti riwayat embed dan ekstrak. Terakhir, Perancangan Proses Sistem merinci alur kerja setiap proses utama yang diinisiasi dari *website* Laravel, termasuk *input* pengguna untuk enkripsi AES-256 serta penyisipan dan ekstraksi data dengan metode *Metadata* dan *Hidden Stream*.

Diagram arsitektur sistem digunakan untuk menggambarkan hubungan antar komponen utama yang terlibat dalam proses penyisipan dan ekstraksi data rahasia pada dokumen PDF. Diagram ini memberikan gambaran secara keseluruhan mengenai peran masing-masing lapisan dalam sistem, mulai dari antarmuka pengguna di sisi *client*, pemrosesan logika di sisi *backend*, hingga penyimpanan

berkas di *local storage*. Berikut merupakan desain diagram arsitektur sistem pada Gambar 3.2.

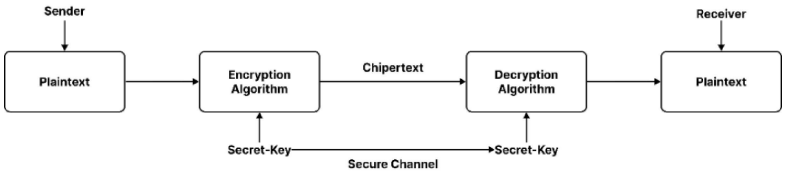


Gambar 3.2. Diagram Arsitektur Sistem.

Diagram Arsitektur Sistem pada Gambar 3.2 menunjukkan tiga lapisan utama, yaitu *Client Side*, *Backend Flask*, dan *Local File Storage*, yang saling terhubung dalam proses penyisipan dan ekstraksi data pada PDF. *Client Side* berfungsi sebagai antarmuka pengguna untuk mengunggah file PDF, memasukkan pesan atau file rahasia, memilih metode steganografi, serta mengambil hasil ekstraksi. Seluruh permintaan dari *Client Side* dikirim ke *Backend Flask* melalui API, di mana backend bertugas melakukan validasi input, proses enkripsi *AES-256*, dan penyisipan data ke PDF menggunakan metode *Metadata* atau *Hidden Stream* sesuai pilihan pengguna, serta ekstraksi dan dekripsi data jika diperlukan. Backend juga berinteraksi dengan *Local File Storage* untuk menyimpan dan mengambil file sumber maupun hasil secara terstruktur. Alur komunikasi berjalan dua arah antara *Client Side* mengirim data ke *Backend Flask* serta backend memproses dan mengakses *Local File Storage* jika diperlukan, kemudian mengembalikan hasil akhir berupa file stego PDF atau data rahasia ke *Client Side*.

Implementasi algoritma kriptografi AES-256 dalam sistem ini menggunakan *library PyCryptodome* pada Flask API. Algoritma tersebut dijalankan dengan mode CBC (*Cipher Block Chaining*) untuk memastikan setiap blok *ciphertext* bergantung pada blok sebelumnya, sehingga meningkatkan keamanan. Proses enkripsi memanfaatkan random *initialization vector* (IV) yang berbeda pada setiap proses enkripsi. Penggunaan IV acak ini krusial untuk mencegah serangan pola dan memastikan *ciphertext* yang unik meskipun *plaintext* dan kunci yang sama digunakan berulang kali. Kata sandi yang dimasukkan pengguna akan di-*hash* terlebih dahulu menggunakan SHA-256 untuk menghasilkan *key* AES-256 yang

konsisten, memastikan bahwa kunci yang digunakan memiliki panjang yang kuat dan tahan terhadap serangan *brute-force* yang langsung pada *password* asli. Proses enkripsi dilakukan sebelum data disisipkan ke dalam *file* PDF, sehingga memberikan lapisan keamanan tambahan di luar teknik steganografi. Ini berarti, bahkan jika pesan tersembunyi berhasil ditemukan dalam PDF, isinya akan tetap terenkripsi dan tidak dapat dibaca tanpa kunci yang benar. Berikut pada Gambar 3.3 merupakan gambaran arsitektur untuk kriptografi AES-256.



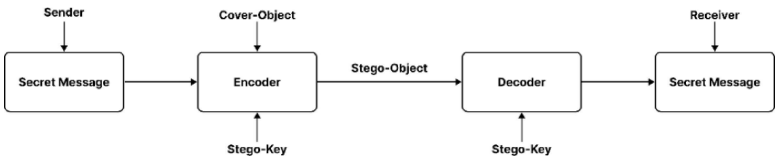
Gambar 3.3. Diagram Arsitektur Kriptografi AES-256.

Gambar 3.3 menampilkan arsitektur kriptografi yang diterapkan dalam sistem ini, menjelaskan bagaimana data diamankan melalui proses enkripsi dan dekripsi menggunakan AES-256. Proses dimulai dari *Sender* yang memiliki *Plaintext* (pesan asli yang belum dienkripsi). *Plaintext* ini kemudian dimasukkan ke dalam *Encryption Algorithm* (Algoritma Enkripsi). Pada tahap ini, *Secret-Key* (kunci rahasia) juga diberikan ke algoritma enkripsi. *Secret-Key* ini merupakan kunci simetris yang vital untuk proses enkripsi dan dekripsi. Hasil dari *Encryption Algorithm* adalah *Ciphertext* (pesan terenkripsi), yaitu *Plaintext* yang telah diubah ke format tidak terbaca.

Ciphertext ini kemudian dapat dikirim melalui Secure Channel kepada Receiver. Di sisi *Receiver*, *Ciphertext* diproses oleh *Decryption Algorithm* (Algoritma Dekripsi). Pentingnya *Secret-Key* kembali terlihat di sini, karena *Decryption Algorithm* memerlukan *Secret-Key* yang sama persis dengan yang digunakan untuk enkripsi. Jika kunci yang benar diberikan, *Decryption Algorithm* akan mengembalikan *Ciphertext* ke bentuk *Plaintext* aslinya, yang kemudian dapat diakses oleh *Receiver*.

Arsitektur steganografi dalam sistem ini berfokus pada mekanisme untuk mengintegrasikan data rahasia yang telah dienkripsi ke dalam dokumen PDF. Steganografi pada media PDF memanfaatkan sifat fleksibel dan kompleks dari

format PDF itu sendiri untuk menyembunyikan informasi. Secara umum, arsitektur ini melibatkan identifikasi area-area dalam struktur internal file PDF yang dapat dimanipulasi tanpa menyebabkan perubahan visual yang signifikan atau merusak integritas dokumen. Area-area tersebut dapat mencakup ruang-ruang tersembunyi seperti *metadata* dokumen dan penambahan *stream* data baru yang tidak ter-*render*. Tujuan utamanya adalah untuk memastikan bahwa *ciphertext* yang disisipkan tidak terdeteksi oleh inspeksi biasa dan hanya dapat diekstrak dengan pengetahuan tentang algoritma steganografi yang spesifik. Arsitektur ini dirancang untuk bekerja secara harmonis dengan proses enkripsi AES-256, menciptakan lapisan keamanan ganda di mana pesan tidak hanya dirahasiakan tetapi juga disamarkan keberadaannya. Berikut merupakan gambaran arsitektur Steganografi pada Gambar 3.4.



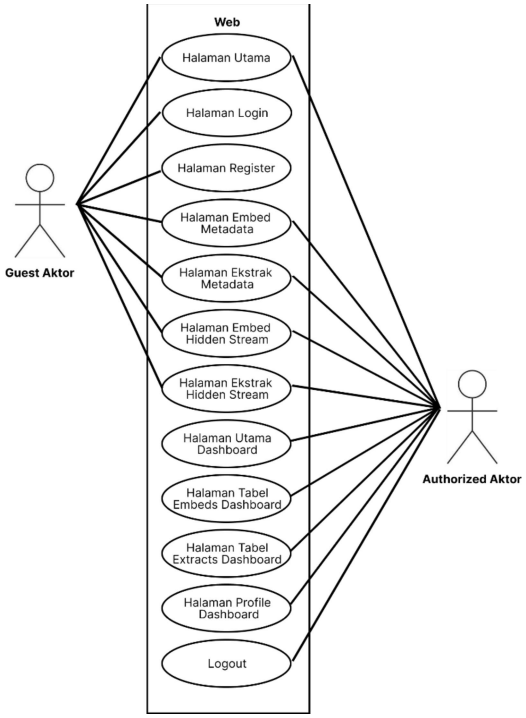
Gambar 3.4. Diagram Arsitektur Steganografi.

Gambar 3.4 menunjukkan model umum arsitektur steganografi yang digunakan dalam sistem ini, menggambarkan alur kerja penyisipan (*embedding*) dan ekstraksi (*extraction*) data rahasia. Proses dimulai dari *Sender* yang memiliki Secret Message (pesan rahasia) yang ingin disembunyikan. Pesan ini kemudian masuk ke Encoder, sebuah komponen yang bertanggung jawab untuk mengintegrasikan *Secret Message* ke dalam *Cover-Object* (objek penutup), yang dalam konteks proyek ini adalah dokumen PDF. Selama proses *encoding*, *Stego-Key* (kunci stego) juga dimasukkan. Kunci ini penting untuk menentukan bagaimana dan di mana pesan disisipkan, serta untuk proses dekripsi nanti. Hasil dari proses *encoding* adalah *Stego-Object*, yaitu *Cover-Object* yang telah disisipi *Secret Message* namun secara visual tidak menunjukkan adanya perubahan yang mencurigakan. *Stego-Object* ini kemudian dapat dikirimkan kepada Receiver.

Di sisi *Receiver*, *Stego-Object* akan diproses oleh Decoder. *Decoder* ini menggunakan Stego-Key yang sama dengan yang digunakan pada fase *encoding* untuk mengekstraksi *Secret Message* dari *Stego-Object*. Hasil ekstraksi oleh

Decoder adalah Secret Message asli, yang kemudian dapat diakses oleh *Receiver*. Arsitektur ini menekankan pentingnya *Stego-Key* sebagai elemen kunci untuk keberhasilan penyisipan dan ekstraksi, serta peran *Encoder* dan *Decoder* dalam mengelola kerahasiaan dan keberadaan pesan.

Kemudian perancangan *Use Case Diagram* dilakukan untuk menggambarkan interaksi antara pengguna (aktor) dengan sistem yang dibangun. Diagram ini memvisualisasikan fungsi-fungsi utama yang dapat diakses pengguna melalui antarmuka web, serta bagaimana sistem merespons setiap aksi yang dilakukan. Dengan adanya use case diagram, hubungan antara aktor dan fitur yang tersedia menjadi lebih jelas, sehingga mempermudah proses analisis kebutuhan dan perancangan sistem secara keseluruhan. Berikut diagram *Use Case* sistem yang akan ditampilkan pada Gambar 3.5.

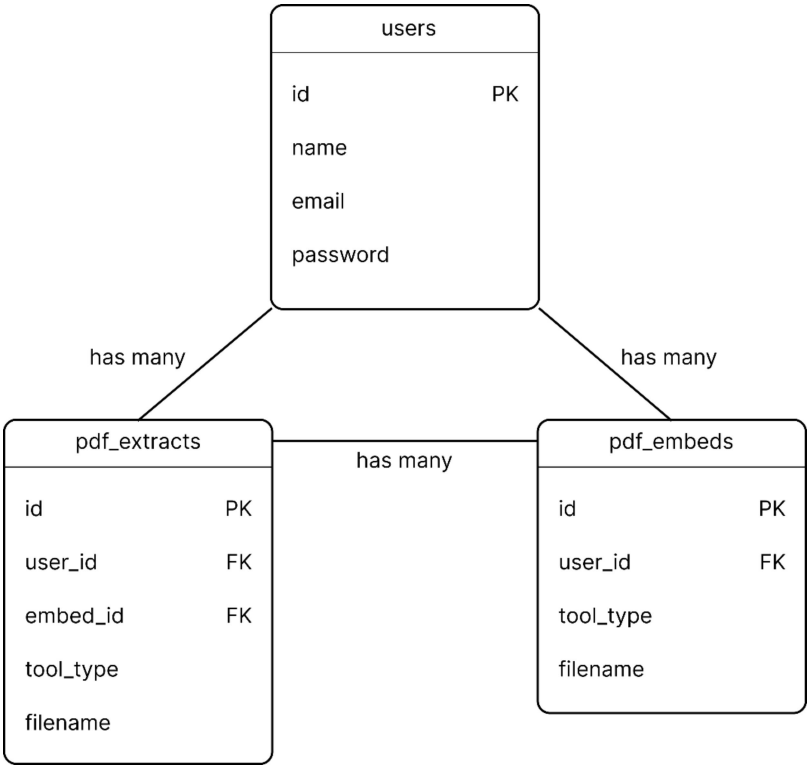


Gambar 3.5. Diagram *Use Case*.

Dalam Gambar 3.5 terdapat seorang Aktor yang merepresentasikan pengguna sistem. Aktor ini berinteraksi langsung dengan komponen Web untuk mengakses berbagai fungsionalitas, termasuk Auth (autentikasi), Dashboard, File Management (manajemen file), dan Steganografi. Sementara itu, Flask API berperan sebagai

backend yang mendukung operasi pada Dashboard, File Management, dan Steganografi di sisi *web*.

Selanjutnya, perancangan basis data merupakan tahapan krusial dalam pengembangan sistem, yang bertujuan untuk memodelkan struktur penyimpanan data dan hubungannya antar entitas. Dalam sistem ini, basis data dirancang untuk mengelola informasi pengguna serta data terkait proses penyisipan dan ekstraksi file PDF. Tabel utama pada sistem ini adalah *users*, *pdf_embeds*, dan *pdf_extracts*, yang saling berinteraksi untuk mendukung fungsionalitas utama aplikasi. Berikut pada Gambar 3.6 terdapat desain *Entity Relationship Diagram (ERD)* yang akan dipakai sistem.

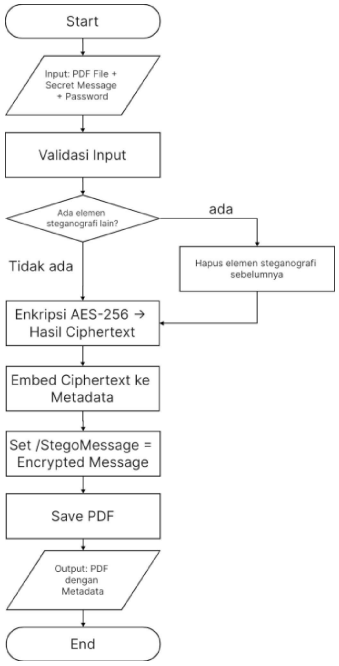


Gambar 3.6. *Entity Relationship Diagram (ERD)*.

Gambar 3.6 menggambarkan hubungan antara pengguna, data penyisipan, dan data ekstraksi dalam sistem. Entitas utama User menyimpan informasi pengguna dengan id sebagai *Primary Key*. Entitas ini memiliki hubungan "*has many*" dengan dua tabel terkait yaitu *pdf_embeds* dan *pdf_extracts*. Tabel *pdf_embeds* merepresentasikan data file PDF yang telah disisipi (*embed*), dengan

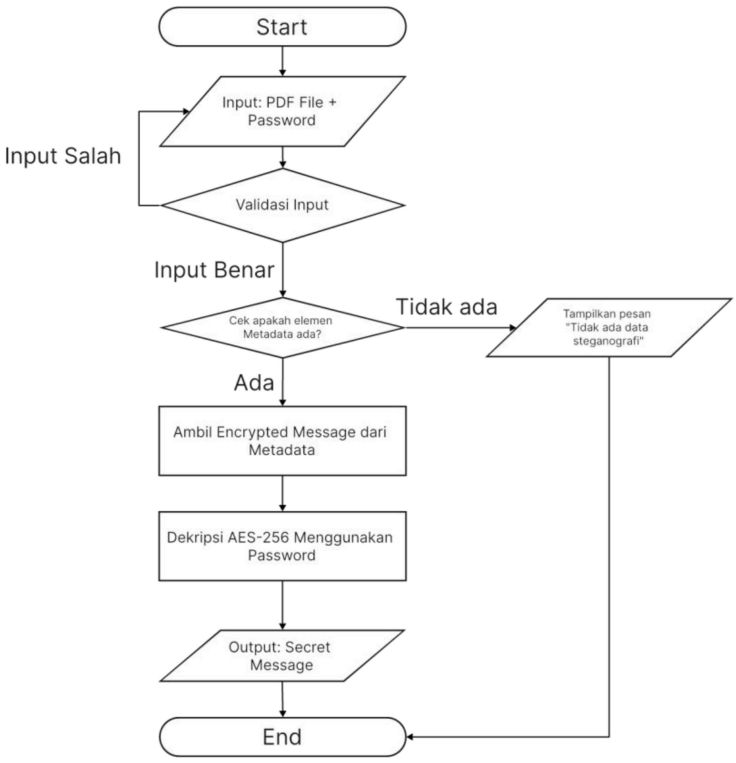
atribut seperti *id* (*Primary Key*), *user_id* (*Foreign Key*) yang mengacu pada *id* di tabel *users*, *tool_type* dan *filename*. Sementara itu, tabel *pdf_extracts* menyimpan informasi file PDF hasil ekstraksi, memiliki atribut id (*Primary Key*), user_id (*Foreign Key*) yang juga mengacu pada id di tabel User, embed_id (*Foreign Key*) yang mengacu pada *id* di tabel *pdf_embeds*, *tool_type*, dan *filename*. Hubungan "*has many*" antara *pdf_embeds* dan *pdf_extracts* menunjukkan bahwa satu file PDF yang disisipi dapat memiliki banyak hasil ekstraksi.

Setelah itu, terdapat perancangan proses pada sistem ini yang mencakup empat alur utama, yaitu proses *embed* dan *ekstrak* pada steganografi, serta proses enkripsi dan dekripsi pada algoritma AES-256. Masing-masing alur divisualisasikan melalui diagram alir yang menggambarkan langkah-langkah detail mulai dari penerimaan input, pemrosesan, hingga menghasilkan output. Diagram alir proses embed dan ekstrak steganografi menunjukkan mekanisme penyisipan maupun pengambilan data rahasia dari file PDF menggunakan metode Metadata dan Hidden Stream. Sementara itu, diagram alir proses enkripsi dan dekripsi AES-256 menjelaskan alur kerja pengamanan data rahasia menggunakan kriptografi dengan kunci yang dihasilkan dari kata sandi pengguna. Berikut merupakan salah satu diagram alir sistem pada Gambar 3.7 yang menggambarkan salah satu alur steganografi pada media PDF metode *Metadata*.



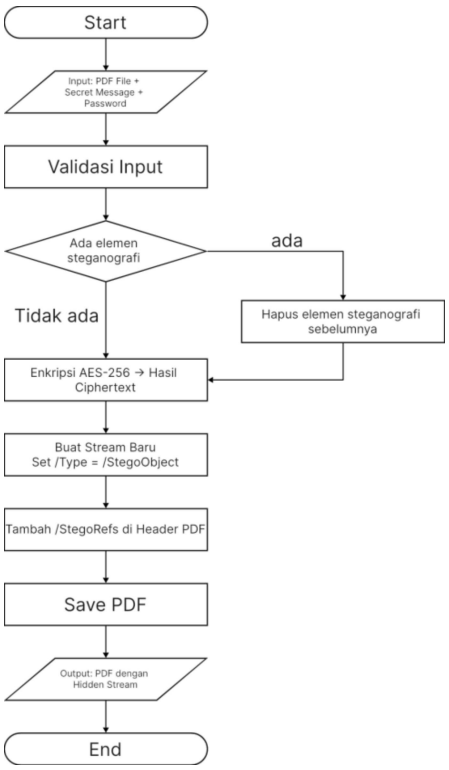
Gambar 3.7. Diagram Alir Proses Embed Metode Metadata.

Diagram alir yang ditampilkan pada Gambar 3.7 menggambarkan Proses Penyisipan Metadata Steganografi ke dalam file PDF. Proses ini dimulai dari *Start* dengan pengguna memasukkan file PDF, pesan rahasia, dan *password*. Setelah input divalidasi, sistem akan memeriksa apakah ada elemen steganografi yang sudah ada pada file tersebut. Jika ada (*"Ya"*), elemen steganografi yang lama akan dihapus. Jika tidak ada (*"Tidak ada"*) atau setelah penghapusan, pesan rahasia akan dienkripsi menggunakan algoritma *AES-256* untuk menghasilkan *ciphertext*. *Ciphertext* ini kemudian disisipkan ke dalam metadata PDF, dan *StegoMessage* akan diatur sebagai *Encrypted Message*. Proses ini diakhiri dengan penyimpanan file PDF yang telah dimodifikasi, yang kemudian menghasilkan Output berupa PDF dengan metadata yang tersembunyi, sebelum proses berakhir pada *End*. Selanjutnya terdapat diagram alir untuk proses ekstrak pada steganografi pada media PDF metode Metadata yang ditampilkan pada Gambar 3.8.



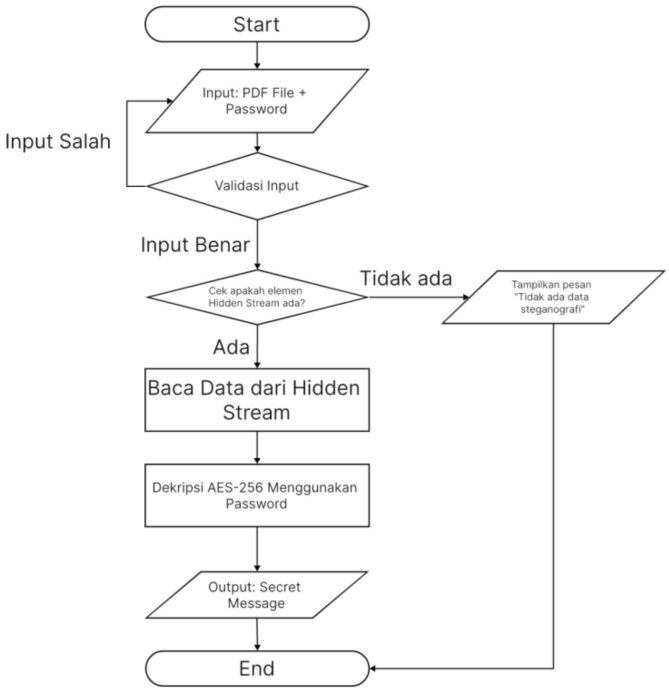
Gambar 3.8. Diagram Alir Proses Ekstrak Metode Metadata.

Proses ekstraksi steganografi pada media PDF metode metadata diawali dengan memasukkan file PDF yang akan diperiksa beserta password yang sesuai. Sistem kemudian melakukan validasi terhadap input untuk memastikan file PDF dapat diproses dan password benar. Setelah itu, dilakukan pengecekan keberadaan elemen steganografi pada metadata PDF. Jika elemen metadata yang mengandung pesan rahasia tidak ditemukan, sistem akan memberikan notifikasi bahwa tidak ada data steganografi yang tersimpan, dan proses dihentikan. Namun, jika elemen metadata ditemukan, sistem akan mengambil *encrypted message* dari metadata tersebut. Pesan terenkripsi ini kemudian didekripsi menggunakan algoritma AES-256 dengan password yang telah dimasukkan pengguna. Hasil dekripsi berupa pesan rahasia asli (*secret message*) yang kemudian ditampilkan sebagai output, sehingga proses ekstraksi dinyatakan selesai. Gambar 3.9 yang menggambarkan alur proses embed steganografi pada media PDF metode *Hidden Stream*.



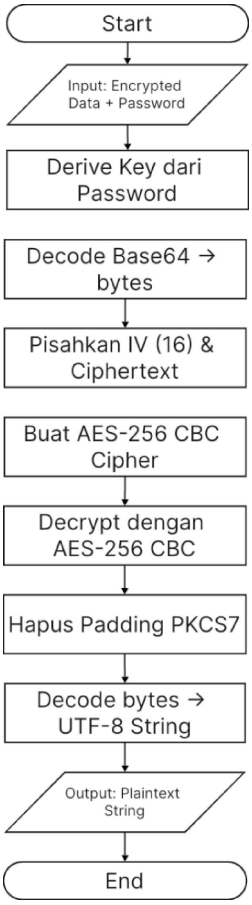
Gambar 3.9. Diagram Alir Proses Embed Metode Hidden Stream.

Gambar 3.9 menjelaskan proses penyisipan steganografi pada media PDF metode *Hidden Stream* ke dalam file PDF. Proses dimulai dari *Start* dengan pengguna memasukkan file PDF, pesan rahasia, dan *password*. Setelah input divalidasi, sistem akan memeriksa apakah ada elemen steganografi yang sudah ada pada file tersebut dan jika ya, elemen lama akan dihapus. Selanjutnya, pesan rahasia dienkripsi menggunakan *AES-256* untuk menghasilkan *ciphertext*. Kemudian, *stream* baru dibuat dengan properti *Set /Type = /StegoObject*, dan *StegoFile* ditambahkan di *header* PDF. Proses diakhiri dengan penyimpanan file PDF yang telah dimodifikasi, menghasilkan Output PDF dengan *hidden stream*, sebelum proses selesai. Berikutnya merupakan proses ekstrak steganografi pada media PDF metode Hidden Stream. Selanjutnya terdapat diagram alir untuk proses ekstrak pada steganografi pada media PDF metode *Hidden Stream* yang ditampilkan pada Gambar 3.10.



Gambar 3.10. Diagram Alir Proses Ekstrak Metode *Hidden Stream*.

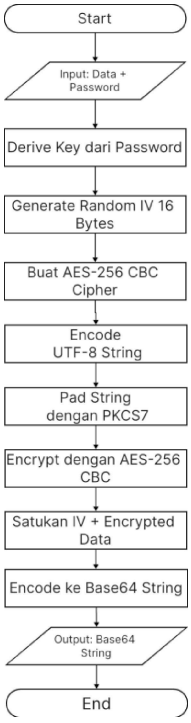
Dari Gambar 3.10 proses ekstraksi *Hidden Stream* dimulai dengan tahap *input* berupa file PDF dan kata sandi yang digunakan sebagai kunci dekripsi. Sistem terlebih dahulu melakukan validasi terhadap *input* untuk memastikan format file sesuai dan kata sandi yang diberikan benar. Setelah itu, dilakukan pengecekan apakah file PDF mengandung elemen *Hidden Stream*. Jika elemen tersebut tidak ditemukan, proses dihentikan dan sistem menampilkan pesan bahwa data tidak ditemukan. Namun, jika elemen *Hidden Stream* ada, sistem akan membaca data yang tersimpan di dalamnya, kemudian melakukan dekripsi menggunakan algoritma AES-256 untuk mengubah *ciphertext* menjadi *plaintext*. Hasil dekripsi berupa *secret message* kemudian ditampilkan sebagai keluaran, dan proses ekstraksi diakhiri. Berikut merupakan desain diagram alir untuk proses enkripsi *AES-256* pada Gambar 3.11.



Gambar 3.11. Diagram Alir Proses Enkripsi *AES-256*

Diagram alir untuk *AES-256* pada Gambar 3.11 dirancang untuk menangani dua jenis *input* secara terpisah yaitu data *string* dan data *byte*, dengan *output* yang sesuai dengan jenis *input*-nya. Proses dimulai dengan penerimaan data dan *password* dari pengguna, yang kemudian digunakan untuk menurunkan kunci enkripsi dan menghasilkan *Initialisation Vector* (IV) acak berukuran 16-*byte* sebagai bagian dari persiapan *AES-256 CBC Cipher*. Alur proses kemudian bercabang. Jika *input* adalah data *string*, *input* akan di-*encode* ke UTF-8 dan di-*pad* dengan PKCS7 sebelum dienkripsi dan sebaliknya, jika *input* adalah data *byte*, *input* akan langsung di-*pad* dengan PKCS7 sebelum enkripsi. Setelah enkripsi utama dengan *AES-256* CBC selesai untuk masing-masing jenis *input*, IV akan digabungkan dengan data terenkripsi. Untuk *input string*, *output* akan berupa *Base64 string*, sedangkan untuk *input byte*, *output* akan tetap berupa *raw bytes*,

sebelum proses enkripsi keseluruhan berakhir. Selanjutnya merupakan proses Dekripsi dari *AES-256* yang di tampilkan pada Gambar 3.12.

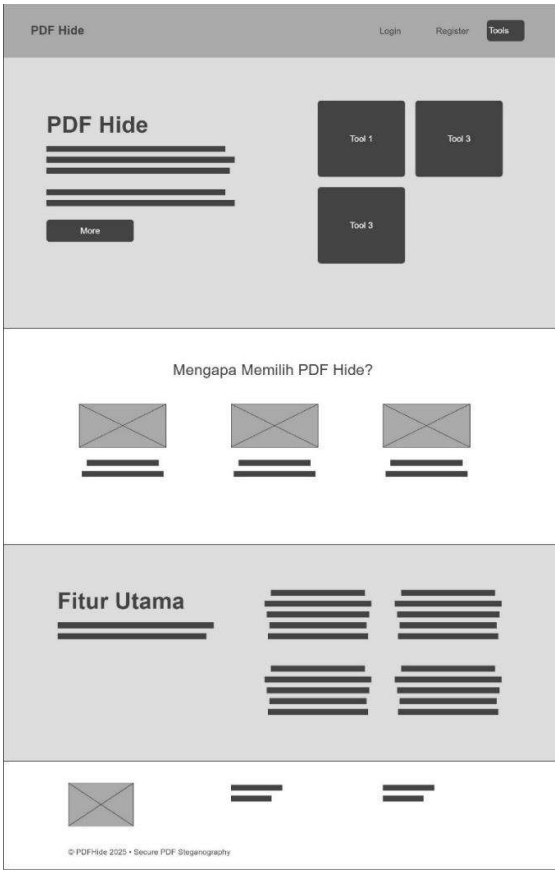


Gambar 3.12. Diagram Alir Proses Dekripsi *AES-256*

Proses dekripsi AES-256 pada Gambar 3.12 dimulai dengan menerima masukan berupa data terenkripsi dan kata sandi yang digunakan untuk menghasilkan kunci enkripsi. Sistem terlebih dahulu melakukan derivasi kunci dari kata sandi menggunakan algoritma hash SHA-256 agar menghasilkan *key* 256-bit yang konsisten. Selanjutnya, sistem memeriksa jenis masukan, apakah berupa *string* Base64 atau data *bytes* mentah. Jika berupa *string* Base64, data tersebut diubah terlebih dahulu menjadi *bytes*. Setelah itu, baik pada jalur *string* maupun *bytes*, dilakukan pemisahan *Initialization Vector* (IV) sepanjang 16-bytes dari data terenkripsi. Langkah berikutnya adalah membuat *AES-256 CBC cipher* dengan parameter *key* dan IV yang sama persis seperti pada saat enkripsi, sehingga proses dekripsi dapat dilakukan secara benar. Mesin cipher ini kemudian digunakan untuk memproses *ciphertext* menjadi data mentah hasil dekripsi. Data tersebut masih mengandung padding, sehingga dilakukan penghapusan padding PKCS7 untuk mengembalikan ukuran data ke bentuk aslinya. Pada jalur *string*, hasil dekripsi

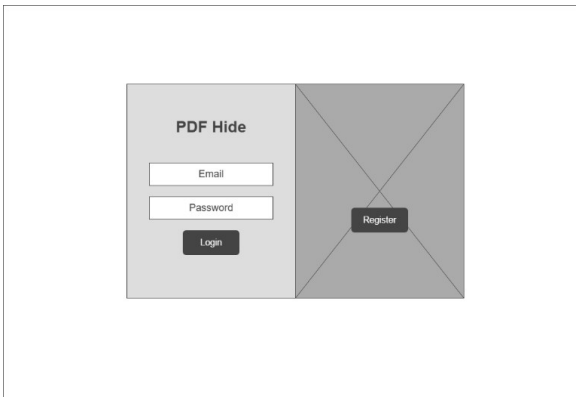
berupa *bytes* selanjutnya dikonversi ke teks menggunakan encoding UTF-8 untuk memperoleh *plaintext* asli, sedangkan pada jalur *bytes*, hasil dekripsi langsung berupa data biner yang siap digunakan.

Terakhir merupakan perancangan antarmuka pengguna yang berfokus pada perancangan tata letak visual dan interaksi *user interface* (*UI*) dari *website* sebelum implementasi kode. *Wireframe* digunakan sebagai kerangka dasar untuk memvisualisasikan struktur halaman, penempatan elemen-elemen UI (tombol, *input field*, area tampilan), dan alur navigasi tanpa detail estetika. Ini bertujuan untuk memastikan fungsionalitas dan pengalaman pengguna yang optimal sebelum pengembangan kode dimulai, serta memvalidasi kesesuaian dengan persyaratan fungsional dan non-fungsional yang telah didefinisikan. Berikut Gambar 3.14 merupakan desain halaman utama website.



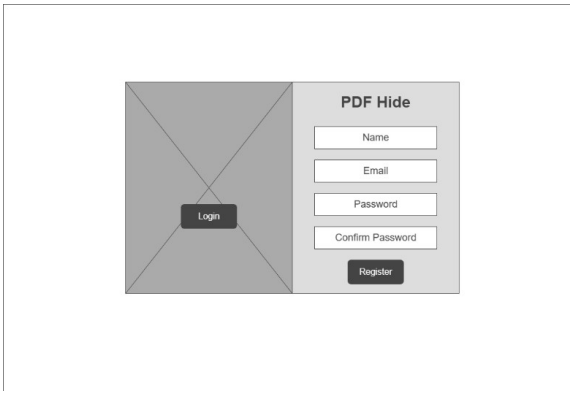
Gambar 3.14. *Wireframe* Halaman Utama

Halaman *Homepage* pada Gambar 3.14 dirancang sebagai pintu masuk utama aplikasi PDF Hide. Halaman ini menampilkan deskripsi singkat sistem, alasan memilih PDF Hide, serta fitur utama yang disediakan. Selain itu, terdapat navigasi ke bagian *Login*, *Register*, serta akses menuju halaman *Tools* untuk memulai proses embed maupun ekstrak. Selanjutnya terdapat desain halaman login pada Gambar 3.15.



Gambar 3.15. *Wireframe* Halaman *Login*

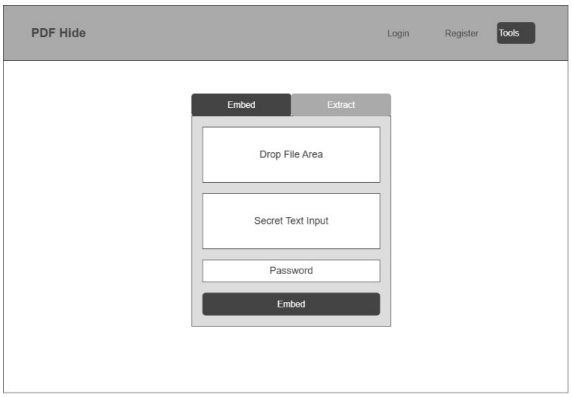
Gambar 3.15 yang menampilkan halaman *Login* digunakan oleh pengguna yang sudah memiliki akun untuk masuk ke dalam sistem. Pada halaman ini tersedia input berupa email dan password. Setelah berhasil login, pengguna akan diarahkan menuju dashboard untuk mengakses seluruh fitur yang tersedia. Setelah halaman login, selanjutnya terdapat desain halaman register pada Gambar 3.16.



Gambar 3.16. *Wireframe* Halaman *Register*

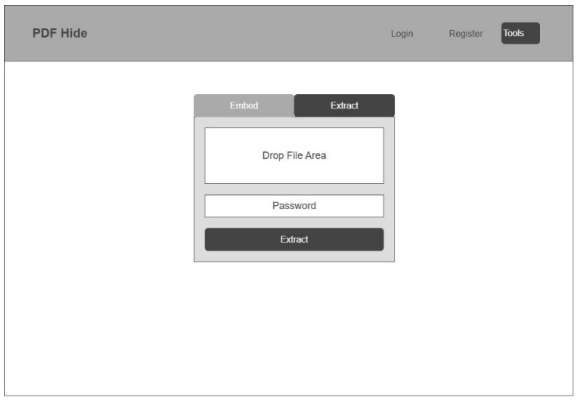
Halaman *Register* pada Gambar 3.16 ditujukan bagi pengguna baru yang ingin membuat akun. Formulir pendaftaran mencakup input nama, *email*, *password*,

dan konfirmasi *password*. Dengan adanya halaman ini, sistem memastikan bahwa hanya pengguna terdaftar yang dapat memanfaatkan fitur *Dashboard*. *Wireframe* yang dirancang untuk halaman *Embed* ditunjukkan pada Gambar 3.17.



Gambar 3.17. *Wireframe* Halaman *Embed*

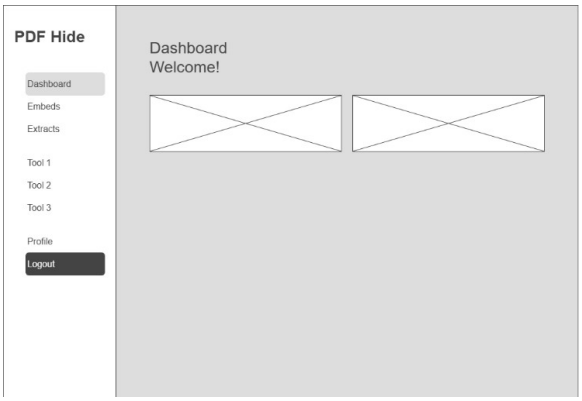
Pada Gambar 3.17 halaman *Embed* dirancang agar pengguna dapat menyisipkan pesan atau file rahasia ke dalam PDF. Pengguna cukup melakukan unggah file, memasukkan pesan atau file yang ingin disembunyikan, serta memberikan *password* untuk proses enkripsi. Setelah itu sistem akan menghasilkan file PDF baru yang sudah mengandung data tersembunyi. Berikut merupakan tampilan *wireframe* untuk halaman *Extract* dapat dilihat pada Gambar 3.18.



Gambar 3.18. *Wireframe* Halaman *Extract*

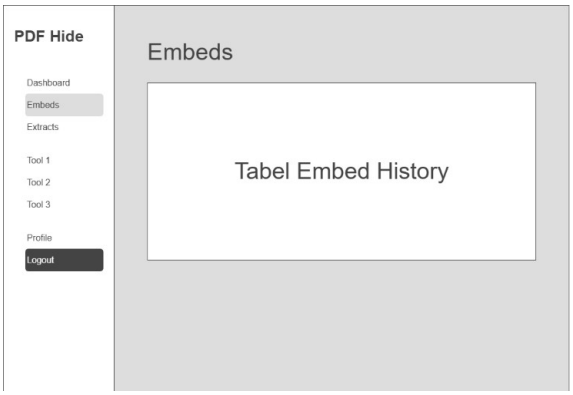
Pada Gambar 3.18 yang menampilkan halaman *Extract* memungkinkan pengguna untuk mengekstrak pesan atau file rahasia dari PDF yang telah diproses sebelumnya. Pengguna perlu mengunggah file PDF stego dan memasukkan *password* untuk melakukan dekripsi. Hasil ekstraksi kemudian ditampilkan atau

dapat diunduh kembali sesuai isi rahasia yang telah disisipkan sebelumnya. *Wireframe* yang dirancang untuk halaman utama *Dashboard* ditunjukkan pada Gambar 3.19.



Gambar 3.19. *Wireframe* Halaman Utama *Dashboard*

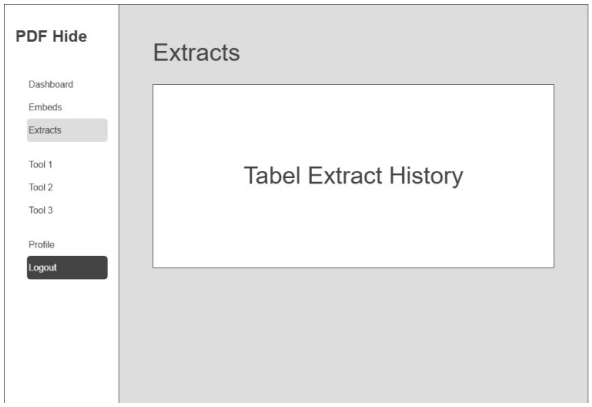
Halaman utama *Dashboard* pada Gambar 3.19 menampilkan menu navigasi pada sisi kiri yang berisi pilihan *Embeds*, *Extracts*, *Tools*, dan *Profile*. Bagian tengah halaman berfungsi sebagai area utama yang menampilkan ringkasan aktivitas serta akses cepat ke fitur-fitur inti. Berikut pada Gambar 3.20 merupakan desain *wireframe* untuk halaman *Embed Dashboard.*



Gambar 3.20. *Wireframe* Halaman *Embed Dashboard*

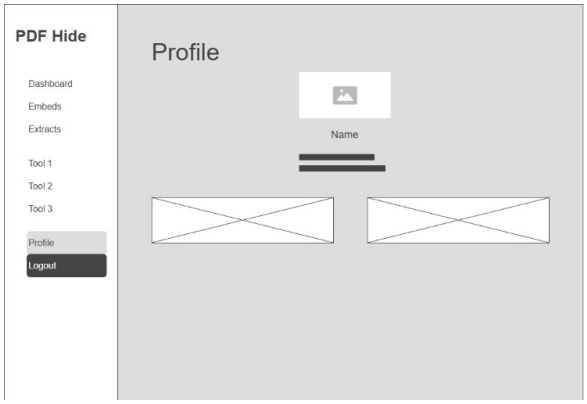
Gambar 3.20 merupakan desain untuk halaman *Embed* pada dashboard yang digunakan untuk menampilkan daftar proses penyisipan (*embedding*) yang telah dilakukan oleh pengguna. Informasi ditampilkan dalam bentuk tabel, sehingga memudahkan pengguna untuk memantau riwayat proses penyembunyian data di

dalam file PDF. Selanjutnya terdapat desain halaman *Extract Dashboard* yang ditampilkan pada Gambar 3.21.



Gambar 3.21. *Wireframe* Halaman *Extract Dashboard*

Gambar 3.21 merupakan halaman *Extract* menampilkan daftar proses ekstraksi data dari file PDF yang telah dilakukan. Sama seperti pada halaman Embed, data ditampilkan dalam bentuk tabel yang memudahkan pengguna dalam memantau aktivitas pengambilan kembali informasi tersembunyi. Berikut pada Gambar 3.22 merupakan desain *wireframe* untuk halaman *Profile.*



Gambar 3.22. *Wireframe* Halaman *Profile Dashboard*

Halaman *Profile* pada Gambar 3.22 menampilkan informasi pengguna seperti nama, alamat email, serta data profil lain yang relevan. Selain itu, halaman ini juga menyediakan akses untuk melihat riwayat aktivitas pengguna terkait *embed* dan *extract*.

3.3.3 *Implementation*

Setelah desain *website* Laravel selesai dan terdokumentasi dengan baik, Fase Implementasi (*Implementation*) dimulai. Pada tahap ini, seluruh rancangan yang telah dibuat diubah menjadi kode program yang fungsional. Proses ini melibatkan pengembangan *frontend* aplikasi *website* menggunakan *framework* Laravel, yang mencakup implementasi tampilan (*views*), logika pengontrol (*controllers*) untuk menangani permintaan pengguna, sistem autentikasi, serta integrasi dengan *backend* Flask API melalui *HTTP requests*. Selain itu, fase ini juga mencakup pembuatan struktur *database* di Laravel melalui migrasi dan *seeder* serta pengembangan model untuk berinteraksi dengan basis data. Penting juga untuk mengembangkan *user interface* yang responsif dan interaktif sesuai dengan desain yang telah disepakati, termasuk implementasi validasi *form* dan penanganan *feedback* kepada pengguna. Setiap bagian dari *website*, seperti halaman *login*, *dashboard*, dan berbagai *tool* steganografi, dikembangkan dan diuji fungsionalitasnya secara terpisah (*unit testing*) sebelum diintegrasikan menjadi satu kesatuan.

3.3.4 *Verification*

Setelah fase implementasi, Fase Verifikasi (*Verification*) menjadi fokus utama. Tahap ini bertujuan untuk melakukan pengujian menyeluruh pada *website* Laravel yang telah dibangun, guna memastikan bahwa ia memenuhi semua persyaratan yang telah ditetapkan di awal. Proses verifikasi ini mencakup dua jenis pengujian. Pertama, *Web Testing* dilakukan untuk menguji setiap fungsi Laravel, misalnya memastikan fungsi autentikasi atau validasi *input* bekerja dengan benar. Kedua, *System Testing* dilakukan untuk mengevaluasi bagaimana *website* bereaksi dan berfungsi ketika semua fungsi *frontend* dan integrasinya dengan *backend* API bekerja bersama. Pengujian fungsionalitas dilakukan melalui pengujian *blackbox* yang mencakup pengujian semua fitur aplikasi termasuk embed, ekstrak, dan manajemen file. Berikut pada Tabel 3.2 merupakan tabel pengujian *blackbox* yang akan diujikan pada website.

Tabel 3.2. Tabel Pengujian *Blackbox*

Pengujian	Input	Hasil yang Diharapkan
Fungsionalitas halaman utama	URL: /	Seluruh fitur pada halaman utama berjalan dengan baik
Fungsionalitas halaman login dan register	URL: /login dan /register	Seluruh fitur pada halaman login dan register berjalan dengan baik
Fungsionalitas halaman tool steganografi metode Hidden Stream	URL: /hidden-stream	Fitur embed dan ekstrak berjalan dengan baik
Fungsionalitas halaman tool steganografi metode Metadata	URL: /metadata	Fitur embed dan ekstrak berjalan dengan baik
Fungsionalitas halaman dashboard	URL: /dashboard	Seluruh fitur dan halaman yang ada pada dashboard berjalan dengan baik

3.3.5 *Maintenance*

Sebagai fase terakhir dari model *Waterfall*, Fase Pemeliharaan (*Maintenance*) berfokus pada dukungan berkelanjutan untuk *website* Laravel. Meskipun proyek ini utamanya berpusat pada pengembangan dan pengujian awal, dalam konteks siklus *Waterfall* yang lengkap, fase pemeliharaan sangat penting. Aktivitas pada fase ini mencakup perbaikan kesalahan (*bug*) atau masalah tampilan yang mungkin ditemukan setelah *website* digunakan secara riil. Selain itu, pembaruan *framework* Laravel atau *library* yang digunakan secara berkala juga diperlukan untuk menjaga keamanan dan kompatibilitas sistem. Fase ini juga melibatkan adaptasi terhadap perubahan kebutuhan pengguna di masa mendatang atau penambahan fitur baru pada *website* untuk memastikan relevansi dan fungsionalitas jangka panjang.

3.4 *Testing and Evaluation*

Testing dan *Evaluation* dilakukan untuk mengukur hasil kerja sistem berdasarkan empat aspek utama yaitu implementasi sistem untuk pengujian sistem secara keseluruhan. Pengujian korelasi *pearson* untuk menguji algoritma

kriptografi *AES-256*. Pengujian performa sistem untuk mengukur seberapa cepat sistem menjalankan *request* dari pengguna dan mengukur perubahan ukuran file dokumen PDF. Terakhir merupakan pengujian steganografi yang dilakukan dengan cara menyebarkan kuesioner yang berisikan pertanyaan-pertanyaan untuk mengukur tingkat kecurigaan orang-orang terhadap file PDF sampel yang berisikan data rahasia yang sudah disisipkan.

Pengujian hasil implementasi sistem meliputi tiga bagian utama, yaitu algoritma kriptografi AES-256, metode steganografi pada media PDF, dan website berbasis Laravel. Pengujian dilakukan untuk memastikan bahwa masing-masing komponen dapat berjalan sesuai rancangan, dengan memperlihatkan hasil enkripsi-dekripsi, proses penyisipan dan ekstraksi data, serta antarmuka sistem yang telah dibangun.

Pengujian korelasi *pearson* dilakukan untuk menilai tingkat hubungan antara *plaintext* dan *ciphertext* hasil enkripsi AES-256. Dengan menghitung nilai koefisien korelasi *pearson*, dapat dibuktikan bahwa *ciphertext* yang dihasilkan memiliki pola acak dan tidak berkorelasi dengan *plaintext*, sehingga menjamin aspek kerahasiaan data. Berikut pada Tabel 3.3 merupakan data yang akan digunakan dalam pengujian korelasi *pearson*.

Tabel 3.3. Tabel Data Pengujian Korelasi *Pearson.*

No	Plaintext	Nilai Korelasi
1.	Rayhan Fatih Abdurrahman	-1 - 1
2.	Hello World, apa kabar?	
3.	Anak-anak bermain di taman dengan gembira hari ini.	
4.	Kucing hitam duduk di atas pagar sambil menatap burung yang terbang.	
5.	Rumah nomor 123 di jalan utama memiliki taman yang indah!	

No	Plaintext	Nilai Korelasi
6.	Matahari terbit di pagi hari memberikan cahaya hangat ke bumi.	-1 - 1
7.	Email: rayhan@email.com | Telepon: 0812-3456-7890 | Alamat: Jl. Merdeka No. 45	
8.	Burung merpati terbang tinggi di langit biru sambil mencari makanan untuk anak-anaknya yang menunggu di sarang.	
9.	Petani bekerja keras di sawah sejak pagi hari untuk menanam padi yang akan dipanen dalam beberapa bulan mendatang.	
10	Bunga bunga bunga bunga bunga bunga bunga bunga bunga bunga.	

Kemudian, pengujian performa dilakukan dengan mengukur efisiensi sistem dalam memproses *request* pengguna, yang meliputi waktu eksekusi pada proses embed dan ekstraksi. Selain itu, dilakukan pengukuran perbandingan antara ukuran file sebelum dan sesudah penyisipan data untuk menilai efisiensi penggunaan ruang penyimpanan. Untuk data yang akan rahasia yang akan digunakan berukuran 74 sampai 671.180 *bytes* dengan tipe data teks dan file *cover* yang berukuran 49.672 sampai 2.248.567 *bytes*.

Terakhir, pengujian steganografi mencakup tiga aspek utama, yaitu imperceptibility, kapasitas, dan *robustness*. *Imperceptibility* diuji menggunakan metode survei terhadap responden dengan skala *Likert* untuk menilai tampilan dokumen PDF yang telah disisipkan data. Kapasitas diukur berdasarkan jumlah teks yang dapat disembunyikan tanpa merusak dokumen, sedangkan robustness diuji dengan menilai ketahanan data tersembunyi ketika terjadi modifikasi pada dokumen PDF. Berikut pada Tabel 3.4 merupakan pertanyaan-pertanyaan yang akan digunakan pada kuesioner.

Tabel 3.4. Tabel Pertanyaan Kuesioner.

No	Pertanyaan	Skala Penilaian (Likert)
1.	Tampilan visual (tata letak dan spasi) pada dokumen PDF ini terlihat wajar dan tidak ada sesuatu yang mencurigakan.	1 = Sangat Tidak Setuju, 2 = Tidak Setuju, 3 = Netral, 4 = Setuju, 5 = Sangat Setuju
2.	Dokumen PDF ini dapat dibuka dan dibaca dengan normal tanpa adanya masalah teknis atau keterlambatan.	
3.	Semua teks dan elemen visual dapat disorot, disalin, dan berfungsi dengan baik seperti pada dokumen PDF umumnya.	
4.	Ukuran file dokumen ini terasa wajar dan sebanding dengan jumlah konten yang terlihat di dalamnya.	
5.	Secara keseluruhan, dokumen ini tidak menimbulkan sedikitpun kecurigaan bahwa ada data tersembunyi di dalamnya.	

Hasil kuesioner yang diperoleh dari responden diolah menggunakan skala Likert untuk mengukur tingkat imperceptibility dokumen PDF. Setiap jawaban responden diberikan bobot skor dengan rentang nilai 1 sampai 5, yaitu Sangat Tidak Setuju untuk 1, Tidak Setuju untuk 2, Netral untuk 3, Setuju untuk 4, dan Sangat Setuju untuk 5. Skor total dihitung dengan menjumlahkan seluruh jawaban responden untuk setiap pernyataan, kemudian dibandingkan dengan skor maksimal yang mungkin diperoleh. dengan skor maksimal diperoleh dari perkalian jumlah responden, jumlah pertanyaan, dan skor tertinggi. Persentase yang didapat kemudian dikategorikan ke dalam lima tingkatan, yaitu 0–20% (Sangat Tidak Setuju), 21–40% (Tidak Setuju), 41–60% (Netral), 61–80% (Setuju), dan 81–100% (Sangat Setuju). Dengan cara ini, hasil kuesioner dapat menunjukkan tingkat

kesetujuan responden terhadap tampilan dokumen PDF yang sudah disisipkan data rahasia.