Karapan Network
Journal

Vol.02, No.02, Mei 2026
Available online at: https://ejournal.omahtabing.com/index.php/knj/id

Implementasi Private Cloud Storage Berbasis Arsitektur Client-Server Menggunakan Kontainerisasi Docker

Moh. Imam Hidayatullah[1)*] , Akmal [2)]

[1) 2)] Universitas Madura, Pamekasan, Indonesia

[1)] imamhidayatullah629@gmail.com, [2)] akmalzakariya1@gmail.com

Abstrak

Perkembangan teknologi digital saat ini memicu lonjakan volume data yang sangat signifikan dalam aktivitas sehari-hari. Dalam pengelolaan aset digital tersebut, ketergantungan pada media penyimpanan fisik konvensional serta layanan *public cloud* kerap menimbulkan sejumlah hambatan. Risiko kerusakan perangkat, mahalnya biaya langganan untuk kapasitas yang besar, hingga timbulnya kekhawatiran terkait privasi informasi menjadi kendala yang cukup sering dihadapi. Situasi tersebut pada akhirnya memengaruhi efisiensi pengelolaan berkas dan membatasi fleksibilitas pengguna dalam mengakses data secara cepat. Penelitian ini difokuskan pada pengembangan sebuah sistem *private cloud storage* yang dirancang untuk menyediakan alternatif ruang penyimpanan lokal yang aman, efisien secara sumber daya, dan mudah diakses melalui jaringan. Pengembangan infrastruktur ini dimulai dari kegiatan studi literatur, analisis kebutuhan perangkat keras, hingga perancangan topologi jaringan dan pemodelan menggunakan *Unified Modeling Language* (UML). Pada tahap implementasi, sistem operasi Ubuntu Server digunakan sebagai fondasi utama, sementara teknologi kontainerisasi Docker dimanfaatkan untuk menjalankan layanan aplikasi *cloud* secara terisolasi agar kinerja komputer tetap ringan. Hasil pengembangan menunjukkan bahwa sistem ini berhasil menyediakan berbagai fitur operasional penting, seperti autentikasi keamanan (*login*), manajemen direktori (unggah, unduh, dan hapus berkas), akses perangkat secara serentak, serta pemantauan sisa kapasitas penyimpanan secara *real-time*. Kehadiran sistem *private cloud* ini membantu mempercepat proses transfer data antar-perangkat di jaringan lokal, meminimalisir risiko kebocoran privasi, dan memberikan kendali penuh kepada pengguna. Dengan demikian, infrastruktur *cloud* mandiri berbasis Docker ini menjadi solusi teknologi yang sangat layak untuk menggantikan ketergantungan pada layanan komputasi awan komersial, sehingga manajemen aset digital dapat berjalan dengan lebih aman, cepat, dan ekonomis.

Keywords: *Private Cloud Storage*, Kontainerisasi Docker, Arsitektur *Client-Server*, Ubuntu Server, Manajemen Berkas Lokal.

Article history: Received 5 April 2026, first decision 22 April 2026, accepted 22 August 2026, available online 28 October 2026

`

I. PENDAHULUAN

Perkembangan teknologi yang begitu pesat telah membawa perubahan besar dalam berbagai aspek kehidupan manusia. Dampaknya tidak hanya terasa pada aktivitas sosial sehari-hari, tetapi juga sangat memengaruhi bagaimana penyimpanan aset digital seperti foto, file, dan lainnya, beroperasi. Inovasi digital yang semakin pesat dan besar tidak luput dari efek sampingnya yaitu ukuran file untuk membuat program seperti aplikasi tersebut juga semakin mengalami peningkatan kapasitas secara signifikan . [1] [2] [3]. Pada zaman sekarang di dunia industri sekalipun masalah ini juga menjadi masalah yang tak terhindarkan ukuran file seperti aset file dalam industry game, aset codingan, atau bahkan dalam industri perfilm an seperti aset video dengan resolusi 4K tentunya juga membutuhkan ruang penyimpanan yang besar karena file-file tersebut pada era digital saat ini sudah semakin mengalami peningkatan kapasitas secara signifikan dan sudah tidak kecil lagi[4] [5] [6]. Dalam situasi seperti ini kebutuhan tempat penyimpananyang besar sudah menjadi sebagai kebutuhan primer terutama pada sector industri yang sangat mengandalkan tekonologi, kebutuhan ini pada era digital saat ini bukan lagi hal yang diperuntukkan hanya sekedar trend saja.[7] [8] [9].

Walaupun pada era digital saat ini jasa teknologi cloud storage atau penyimpanan digital sudah banyak beredar dan bertebaran, dan orang orang pada era digital saat ini sudah banyak menggunakan nya seperti Google Drive, Dropbox dan sebagainya itu tidak menutup kemungkinan jasa tersebut tidak memiliki kekurangan. [10] [11] [12]. Salah satu kekurangan terbesar yang menjadi masalah pada saat ini yaitu biaya jasa atau langganan yang semakin mahal, untuk menggunakan atau menyewa penyimpanan digital atau cloud storage tersebut biayanya di patok cukup tinggi terutama jika kita membutuhkan kapasitas penyimpanan yang besar, kemudian salah satu yang menjadi pertimbangan juga yaitu isu privasi, ketika kita menaruh data pribadi ke cloud storage tersebut atau bahkan data dan

[*] Muhafiz Khairi

ISSN 2443-2xxx (online) 2598-6xxx (print)
doi: http://dx.doi.org/10.20473/KNJ.X.X.pp-pp

file perusahaan tidak menutup kemungkinan bahwasanya file atau data tersebut bisa saja diakses oleh pihak yang tidak berwenang dan di lihat oleh orang lain atau bocor ke pihak yang tidak kita inginkan.

Kemudian hal yang juga menjadi problem pada era digital saat ini terutama hal yang mengenai penyimpanan atau storage yaitu keterbatasan storage fisik konvensional, penyimpanan fisik seperti flashdisk atau hardisk eksternal itu juga memiliki beberapa keterbatasan, salah satunya yaitu penyimpanan fisik seperti flashdisk ataupun hardisk itu rawan hilang, rawan rusak, dan memiliki keterbatasan harus di membutuhkan pemasangan fisik secara manual setiap kali kita mau memindahkan data dari laptop ke komputer atau bahkan ke smartphone kita, dan hal itu tentunya juga membuat kita merasa kurang efisien terutama buat beberapa orang atau perusahaan yang membutuhkan mobilitas tinggi. Maka dari private cloud sebagai solusi dan jalan tengah dari kedua masalah tersebut, membuat atau merakit penyimpanan cloud yang dapat di gunakan secara private, yaitu hanya dapat di gunakan oleh kita sendiri dan orang orang tertentu yang sudah di berikan akses oleh kita. Membuat server cloud storage private disini tentunya menyelesaikan beberapa masalah seperti masalah privasi ini tentunya jauh lebih aman di karenakan kita sendiri yang memegang kendali penuh atas server ini, data-data penting seperti data perusahaan data industri pribadi tentunya juga lebih aman dan tidak akan mudah bocor, kemudian dari segi biaya ini jauh lebih murah karena kita bisa membangun server dengan kapasistas yang kita inginkan dan menyesuaikan dengan kebutuhan kita ataupun perusahan jika itu menyangkut perusahaan dan industri.

[13] [14] [15].

Hal yang perlu di tekankan yaitu untuk membuat atau proses instalasi server tradisional itu sangat lah tidak mudah, beberapa hal yang sering terjadi pada saat proses instalasi yaitu sering terjadinya konflik library serta proses instalasi yang memakan banyak resource storage itu sendiri, maka dari itu kita perlu mencari beberapa cara untuk memecahkan masalah tersebut yaitu dengan kita menggunakan docker. [16] [17] [18] [19]. Lalu apa itu Docker, Docker sendiri adalah platform open-source yang memungkkinkan pengembang untuk mengembangkan, menguji dan menjalankan aplikasi dalam lingkungan yang terisolasi yang di sebut sebagai container. Container ini mengemas aplikasi beserta semua depensinya seperti kode, library dan pengaturan system menjadi satu unit standar, sehingga aplikasi dapat berjalan secara konsisten di berbagai lingkungan komputasi, hal ini tentunya dapat membantu kita dalam instalasi server storage private kita karena Docker sendiri sangat ringan serta instalasinya yang konsisten tidak akan membuat sistem utama yaitu operating sistem server itu sendiri menjadi kotor oleh sampah-sampah instalasi, Docker sendiri juga sangat cocok untuk mengoptimasi storage seperti HDD maupun SSD yang terbatas

[20] [21] [22].

Secara umum, server storage private yang dirancang dan bangun disini berjalan di jaringan local yang tidak memerlukan kecepatan internet, itu tentunya akan mempermudah kita dalam mengakses data dari berbagai perangkat seperti smarthphone, laptop, dan tablet secara serentak tanpa memikirkan kecepatan internet yang terkendala masalah latensi jaringan.[23] [24] [25]. Berdasarkan pemaparan permasalahan mengenai keterbatasan kapasitas penyimpanan fisik, tingginya biaya langganan layanan *public cloud*, serta pentingnya menjaga privasi data pengguna, maka diperlukan sebuah solusi penyimpanan alternatif yang efisien dan dapat dikelola secara mandiri. Pemanfaatan perangkat keras yang sudah ada dengan didukung oleh teknologi kontainerisasi diharapkan mampu memberikan solusi sistem penyimpanan data lokal yang andal, ringan, dan mudah diakses dari berbagai perangkat. Oleh karena itu, penelitian ini mengusulkan sebuah pengembangan sistem yang dituangkan dalam judul penelitian **"Rancang Bangun dan Implementasi Private Cloud Storage Berbasis Arsitektur Client-Server Menggunakan Kontainerisasi Docker"**.

II. KAJIAN PUSTAKA

Perkembangan ekosistem digital saat ini mendorong peningkatan volume data yang sangat signifikan di berbagai sektor kehidupan [26]. Kebutuhan terhadap ruang penyimpanan yang aman dan efesien kini menjadi prioritas utama, terutama bagi pengguna personal maupun instansi skala kecil yang membutuhkan manajemen yang praktis, fleksibel dan mudah diakses dari berbagai perangkat[27] [28] [29]. Cloud storage atau komputasi awan adalah teknologi yang menggabungkan perangkat keras, perangkat lunak, dan jaringan untuk mengelola, memproses, serta menyimpan informasi guna mendukung produktivitas pengguna. [30] [31]. Penerapan infrastruktur penyimpanan yang tepat sangat krusial untuk memastikan ketersediaan data secara cepat dan sekaligus menekan biaya operasional. [32] [33].

Dalam pengelolaan aset digital, *private cloud storage* memiliki peran vital dalam memastikan privasi dan keamanan arus informasi [34]. Sistem ini merupakan infrastruktur komputasi awan yang dioperasikan secara eksklusif untuk satu entitas atau pengguna lokal, sehingga meminimalisir risiko kebocoran data yang kerap terjadi pada layanan *public cloud*. Oleh karena itu, pengimplementasian *private cloud* secara mandiri membantu pengguna memiliki kendali penuh atas data mereka, mempercepat proses transfer *file* di jaringan lokal, serta menghindari tingginya biaya langganan bulanan.

Beberapa kajian terdahulu juga menunjukkan bahwa pemanfaatan server lokal menggunakan perangkat keras standar mampu menghasilkan sistem penyimpanan yang andal. Pratama dan Wijaya, misalnya, merancang sistem cloud lokal berbasis open-source dan menemukan bahwa sistem tersebut dapat mempercepat proses sinkronisasi data antar perangkat sekaligus meminimalisir ketergantungan pada koneksi internet eksternal [35]. Selain itu, Nugroho turut membuktikan bahwa penggunaan arsitektur *client-server* pada jaringan lokal (LAN) mampu memberikan akses data secara *real-time* dengan tingkat latensi yang jauh lebih rendah dibandingkan layanan *cloud* komersial [36] [37].

Dari aspek teknologi infrastruktur, kombinasi Ubuntu Server dan teknologi kontainerisasi banyak digunakan dalam pengembangan peladen (server) modern. Keduanya bersifat open-source, sangat stabil, dan tidak memakan banyak sumber daya komputasi. Menurut Santoso. [38].

sistem operasi berbasis Linux seperti Ubuntu sangat ideal digunakan sebagai *host server* karena kemampuannya dalam mengelola memori secara efisien. Sementara itu, Docker berperan penting dalam menciptakan lingkungan yang terisolasi (*container*) untuk menjalankan layanan *cloud*. Kombinasi keduanya memungkinkan terciptanya peladen penyimpanan yang ringan, tangguh, dan sangat mudah untuk dideploy ulang [40][41].

Berdasarkan hasil telaah literatur dan temuan penelitian terdahulu, dapat disimpulkan bahwa rancang bangun *private cloud storage* menggunakan kontainerisasi Docker pada peladen Ubuntu memiliki dasar teoritis sekaligus teknis yang sangat kokoh. Berbagai referensi menunjukkan bahwa lokalisasi penyimpanan data terbukti mampu memberikan privasi maksimal, mengurangi latensi perpindahan *file*, serta mengefisiensikan penggunaan perangkat keras yang sudah ada (*legacy hardware*).

Selain itu, konsep penggunaan arsitektur *client-server* yang diintegrasikan dengan Docker memberikan nilai tambah yang strategis. Pendekatan ini memungkinkan instalasi sistem berjalan tanpa mengotori *library* sistem operasi utama, isolasi layanan yang lebih baik, serta manajemen beban kerja yang lebih ringan. Hal ini sangat selaras dengan prinsip efisiensi komputasi modern, yang memudahkan pengguna untuk mengelola *server* secara stabil meskipun dijalankan pada spesifikasi perangkat keras menengah ke bawah.

Dengan demikian, implementasi *private cloud* ini tidak hanya menjadi alternatif penyelesaian masalah biaya langganan digital, tetapi juga berperan sebagai media untuk meningkatkan kemandirian dalam tata kelola data. Pembangunan sistem seperti ini sangat relevan pada era sekarang, di mana privasi, aksesibilitas lokal yang cepat, dan efisiensi arsitektur perangkat lunak menjadi kunci utama dalam manajemen aset informasi personal maupun kelompok

.

III. METODE

Penelitian ini menggunakan metode *Software Development Life Cycle* (SDLC) dengan pendekatan model *Waterfall*. Pendekatan ini dipilih karena memiliki tahapan pengembangan yang sistematis dan bergerak secara berurutan mulai dari analisis kebutuhan, perancangan sistem, implementasi, hingga pengujian. Alur yang terstruktur seperti ini sangat sesuai untuk pengembangan sistem yang kebutuhan dan spesifikasinya telah ditetapkan dengan jelas sejak awal. Dengan demikian, setiap tahap dapat diselesaikan secara fokus dan terkontrol, meminimalkan perubahan mendadak, serta memastikan bahwa hasil akhir sistem sesuai dengan rancangan yang telah ditentukan [42]. Model *Waterfall* menawarkan kerangka kerja yang terstruktur dan sistematis, dimulai dari analisis kebutuhan hingga tahap pengujian. Alur yang berurutan ini membantu meminimalkan potensi kesalahan pada saat implementasi, karena setiap tahap harus diselesaikan dan divalidasi sebelum berlanjut ke tahap berikutnya. [43] [44].

Proses rancang bangun *private cloud storage* berbasis Docker ini dilaksanakan melalui beberapa tahap utama, dengan rincian tahap pertama sebagai berikut:

1) Amalisis Kebutuhan

Tahapan ini difokuskan untuk mengidentifikasi kebutuhan teknis serta permasalahan mendasar terkait penyimpanan data digital yang dialami pengguna. Berdasarkan identifikasi masalah pada rutinitas pengelolaan data berskala besar, ditemukan bahwa penggunaan media penyimpanan fisik konvensional dinilai kurang praktis, membatasi mobilitas, dan rentan terhadap kerusakan. Di sisi lain, ketergantungan pada layanan *public cloud* menimbulkan kendala terkait tingginya biaya langganan untuk kapasitas besar serta adanya kekhawatiran mengenai privasi data. Dari analisis kondisi tersebut, ditarik kesimpulan bahwa diperlukan sebuah peladen penyimpanan mandiri di jaringan lokal. Berikut adalah rincian spesifikasi kebutuhan sistem yang dirancang:

Kebutuhan Fungsional :

1. Fitur login dan autentikasi pengguna.
   Sistem wajib menyediakan pintu masuk (*login screen*) dengan kredensial kata sandi yang aman, sehingga direktori penyimpanan hanya bisa diakses oleh pengguna yang memiliki otorisasi.
2. Pengolahan berkas (File Management).
   Pengguna dapat melakukan operasi pengelolaan data secara penuh, termasuk mengunggah (*upload*), mengunduh (*download*), menghapus (*delete*), serta mengelompokkan data ke dalam folder.Proses penjualan otomatis dan penyesuaian stok.
3. Akses multi-perangkat (Client-server).
   Sistem harus mampu melayani permintaan akses data dari berbagai perangkat *client* yang berbeda, seperti *smartphone* dan laptop, secara bersamaan selama berada di dalam satu Jaringan Lokal (LAN).
4. Monitoring kapasitas penyimpanan.
   Aplikasi *cloud* harus menampilkan informasi indikator sisa ruang pada media penyimpanan (SSD/HDD) agar pengguna dapat memantau kuota data secara *real-time*.

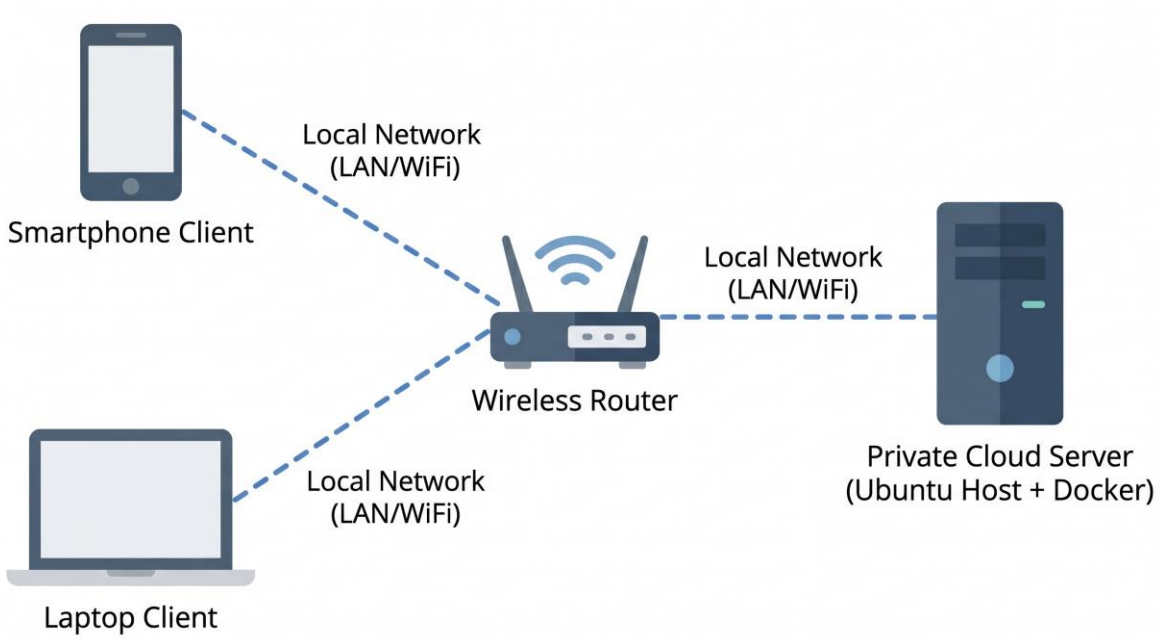
Kebutuhan Non-Fungsional :

1. Usability.
   Antarmuka pengguna (UI) harus bersifat responsif, sederhana, dan dapat dioperasikan langsung melalui *web browser* tanpa mengharuskan pengguna menginstal aplikasi tambahan di sisi *client*.
2. Performance.
   Sistem harus mampu memberikan kecepatan transfer data yang stabil dengan tingkat latensi yang sangat rendah, memanfaatkan kapabilitas *bandwidth* maksimal dari *router* Jaringan Lokal.
3. RecourceEffenciency.
   Sistem peladen harus mengonsumsi sumber daya komputasi (RAM dan *Storage*) seefisien mungkin. Oleh karena itu, penerapan teknologi kontainerisasi sangat dibutuhkan agar proses operasional *server* tetap ringan.
4. Maintainability.
   Struktur direktori penyimpanan dan konfigurasi peladen harus mudah diperbarui (*update*) atau dicadangkan (*backup*) di masa mendatang tanpa merusak sistem operasi utama.

Analisis kebutuhan perangkat keras dan perangkat lunak ini menjadi fondasi dasar dalam merancang topologi jaringan dan arsitektur sistem *private cloud*, guna memastikan implementasi berjalan sesuai dengan spesifikasi komputer yang digunakan.

2) Desain Sistem

Pada tahap desain sistem, digunakan *Unified Modeling Language* (UML) sebagai alat pemodelan untuk merepresentasikan arsitektur jaringan dan alur kerja *private cloud* secara visual. UML dipilih karena mampu menggambarkan interaksi antara perangkat *client* dengan peladen (*server*) serta urutan aktivitas akses data dengan lebih terstandarisasi. Dalam penelitian ini, perancangan difokuskan pada penggambaran topologi jaringan dan interaksi pengguna terhadap layanan penyimpanan melalui beberapa diagram utama.

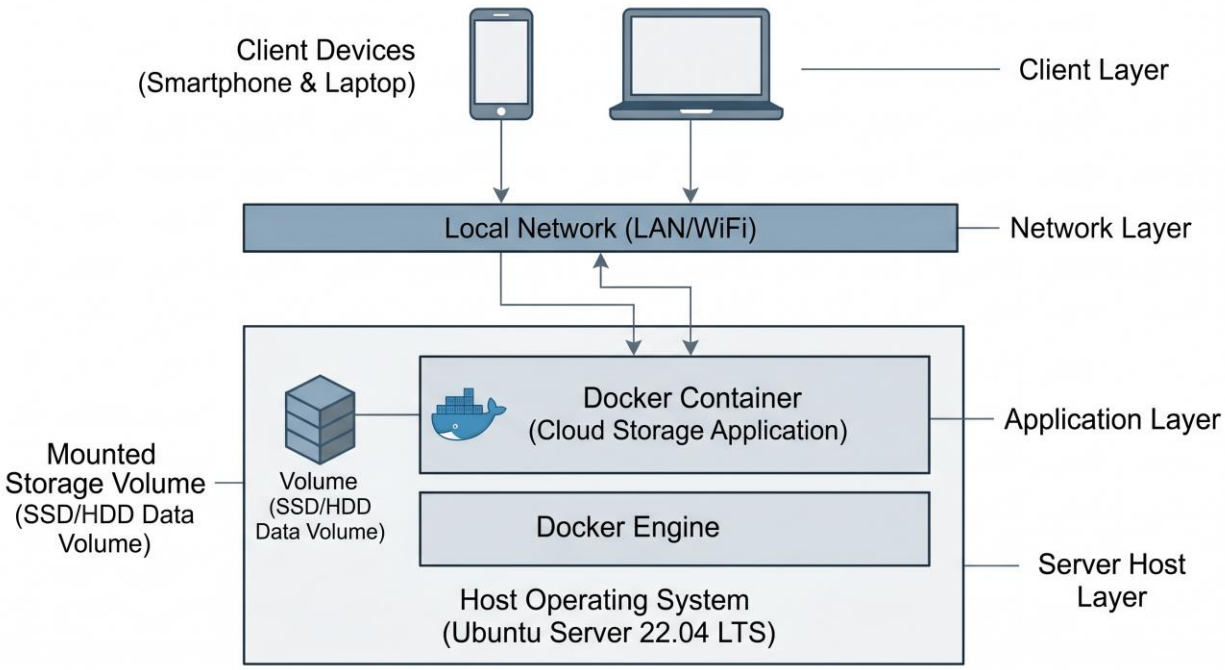


Gambar. 1. Topologi Jaringan Private Cloud Storage

Use Case Diagram digunakan untuk mendeskripsikan hak akses pengguna terhadap sistem penyimpanan. Dalam rancangan ini, terdapat dua aktor utama dengan peran yang berbeda:

1. **Administrator:** Memiliki hak akses penuh untuk melakukan konfigurasi peladen, mengelola kontainer Docker, memantau kapasitas total penyimpanan, serta mengatur akun pengguna.
2. **User:** Memiliki akses terbatas yang mencakup aktivitas manajemen berkas pribadi, seperti mengunggah (*upload*), mengunduh (*download*), dan mengorganisir folder di dalam *cloud*.

Selanjutnya, *Activity Diagram* memvisualiskan alur proses utama, dimulai dari tahapan autentikasi pengguna pada halaman *login*, proses validasi kredensial oleh sistem, hingga pengguna dapat melakukan manipulasi data pada direktori penyimpanan. Alur ini dirancang sedemikian rupa agar proses pertukaran data antara perangkat pengguna dan peladen Ubuntu berjalan secara sinkron tanpa mengalami kendala latensi yang berarti.



Gambar. 2. Arsitektur Client-Server Berbasis Docker

Antarmuka sistem dikembangkan dengan prinsip responsivitas menggunakan kombinasi HTML, CSS, dan framework Bootstrap. Penggunaan Bootstrap bertujuan agar halaman manajemen berkas dapat diakses secara optimal melalui berbagai perangkat, baik komputer maupun ponsel pintar (*smartphone*), dengan tampilan yang tetap konsisten. Selain itu, perancangan penyimpanan data dilakukan dengan metode *mounting* direktori pada sistem operasi Ubuntu Server guna memastikan data tersimpan pada media fisik yang tepat (SSD/HDD) dan tetap terjaga integritasnya meskipun kontainer Docker mengalami *restart* [17]. Seluruh konfigurasi jaringan diarahkan pada alamat IP lokal yang statis untuk memudahkan akses perangkat *client* di dalam lingkup Jaringan Lokal (LAN).

3) Implementasi

Tahapan implementasi merupakan proses menerjemahkan rancangan arsitektur jaringan dan sistem ke dalam lingkungan peladen (*server*) yang nyata. Pada tahap ini, Ubuntu Server digunakan sebagai sistem operasi utama (*host*) karena kestabilannya dalam mengelola lalu lintas jaringan. Sementara itu, Docker difungsikan sebagai mesin kontainerisasi untuk menjalankan aplikasi *cloud storage*. Kombinasi kedua teknologi ini dipilih karena mampu memisahkan lingkungan aplikasi dari sistem utama, sehingga pengelolaan sumber daya komputer menjadi jauh lebih efisien, ringan, dan tentunya bersifat *open-source* [18]. Penggunaan pendekatan ini memastikan peladen dapat berjalan secara optimal tanpa membebani perangkat keras secara berlebihan. [47].
Langkah implementasi meliputi:

1. **Persiapan Lingkungan Peladen (Host OS)**
   Pada tahap awal, sistem operasi Ubuntu diinstal dan dikonfigurasi. Langkah ini mencakup pengaturan alamat IP statis pada jaringan lokal agar peladen mudah dikenali oleh perangkat *client* tanpa mengalami perubahan alamat saat perangkat dihidupkan ulang.
2. **Instalasi Mesin Docker dan Dependensi**
   Proses selanjutnya adalah memasang *Docker Engine* beserta *Docker Compose* pada peladen. Komponen ini sangat krusial karena akan bertugas sebagai wadah (kontainer) terisolasi yang menjalankan seluruh layanan *cloud storage* secara mandiri.
3. **Konfigurasi Ruang Penyimpanan (Mounting Volume)**
   Untuk menjaga kapasitas SSD sistem utama agar tidak cepat penuh, dilakukan proses *mounting* direktori. Ruang penyimpanan utama untuk data pengguna diarahkan ke media penyimpanan sekunder (seperti HDD/SSD eksternal) agar manajemen berkas lebih aman dan kapasitasnya lebih leluasa.
4. **Deployment Aplikasi Cloud Storage**
   Aplikasi penyimpanan berbasis *web* di-*deploy* ke dalam kontainer Docker. Proses ini mencakup pengaturan *port forwarding* agar antarmuka aplikasi dapat diakses dengan lancar melalui *web browser* dari perangkat klien yang terhubung di jaringan LAN yang sama.
5. **Pengaturan Akun dan Autentikasi**
   Langkah terakhir adalah membuat akun administrator utama dan mengatur batasan hak akses. Mekanisme ini diterapkan untuk memastikan bahwa hanya pengguna yang memiliki kredensial sah yang diizinkan untuk mengelola berkas di dalam sistem *private cloud*.

4) Pengujian

Tahapan pengujian dilaksanakan untuk mengevaluasi apakah setiap fitur dan konfigurasi peladen telah berfungsi sesuai dengan spesifikasi yang dirancang. Evaluasi ini menggunakan metode *Blackbox Testing*, yaitu pendekatan pengujian yang berfokus pada kesesuaian antara perintah yang dimasukkan pengguna (input) dengan hasil yang ditampilkan oleh sistem (output), tanpa perlu membongkar struktur kode di baliknya [19] [20]. Setiap fungsi diuji berdasarkan respons aplikasi *cloud* terhadap berbagai skenario aktivitas pengguna sehari-hari. [48] [49]. Selain pengujian secara fungsional, dilakukan juga pengujian langsung menggunakan beberapa perangkat *client* (seperti laptop dan ponsel pintar) untuk memastikan antarmuka sistem mudah dioperasikan dan responsif. Pengujian ini bertujuan untuk mengukur tingkat kenyamanan pengguna saat mengakses data secara nirkabel di lingkungan lokal. Beberapa skenario pengujian yang dilakukan antara lain:

1. Proses autentikasi (login) berhasil mendeteksi kredensial pengguna yang valid dan menolak kata sandi yang salah.
2. Proses unggah (*upload*) berbagai format berkas ke dalam peladen berjalan lancar tanpa mengalami *corrupt* data.
3. Proses unduh (*download*) berkas dari peladen ke perangkat *client* berfungsi dengan baik dan utuh.

4. Manajemen direktori, seperti penambahan, pengubahan nama, dan penghapusan folder, langsung tersinkronisasi di sistem.
5. Sistem dapat diakses secara bersamaan dari dua perangkat berbeda tanpa menyebabkan gangguan koneksi (*crash*).

Hasil pengujian menunjukkan bahwa infrastruktur *private cloud* beroperasi dengan stabil, responsif, dan tidak ditemukan gangguan fungsional yang berarti. Sistem dinyatakan siap untuk digunakan sebagai alternatif penyimpanan data lokal.

5) Pemeliharaan

Tahap terakhir dalam siklus pengembangan ini adalah pemeliharaan (*maintenance*). Pemeliharaan merupakan proses krusial untuk memastikan peladen tetap beroperasi secara optimal dalam jangka waktu yang panjang. Kegiatan ini meliputi pengecekan suhu perangkat keras, pemantauan penggunaan memori (RAM), serta pembersihan *cache* atau data sementara yang menumpuk di dalam sistem [50]. Dalam konteks *private cloud storage*, pemeliharaan juga sangat berfokus pada manajemen kapasitas *storage*. Administrator perlu secara rutin memantau sisa ruang kosong pada SSD/HDD agar sistem tidak mengalami kegagalan proses unggah saat kapasitas penuh. Selain itu, pembaruan (*update*) versi *image* Docker juga perlu dilakukan secara berkala untuk menambal celah keamanan terbaru dan meningkatkan performa sistem di masa mendatang.

IV. HASIL

Tahapan ini memaparkan hasil dari implementasi sistem peladen *private cloud storage* yang telah dibangun sesuai dengan rancangan arsitektur jaringan pada tahap sebelumnya. Sistem ini diterapkan pada lingkungan jaringan lokal (LAN/WiFi) dengan memanfaatkan spesifikasi perangkat keras komputer *desktop* (PC) sebagai *host server* Ubuntu. Pengembangan infrastruktur penyimpanan ini ditujukan untuk mempermudah manajemen aset digital, meningkatkan efisiensi ruang penyimpanan fisik pada perangkat pengguna, serta menjamin privasi data secara mandiri. Aplikasi penyimpanan ini dikonfigurasi melalui lingkungan kontainerisasi Docker dengan antarmuka berbasis *web* yang responsif. Kombinasi teknologi ini memberikan fleksibilitas tinggi, memungkinkan pengguna untuk mengakses, mengelola, dan mencadangkan data mereka menggunakan berbagai perangkat secara praktis dan minim latensi.

1. Autentikasi Pengguna (Login)
   Sebagai gerbang keamanan utama, pengguna diwajibkan untuk memasukkan nama pengguna (username) dan kata sandi (password) yang valid. Sistem akan memvalidasi kredensial tersebut sebelum memberikan hak akses ke dalam dasbor penyimpanan. Jika data sesuai, pengguna akan langsung diarahkan ke halaman utama.
2. Validasi Kredensial Tidak Sah
   Fitur ini berfungsi sebagai perlindungan dasar sistem. Ketika ada upaya masuk menggunakan kata sandi atau nama pengguna yang salah, sistem akan langsung menolak akses dan menampilkan pesan peringatan. Hal ini memastikan bahwa berkas privasi di dalam peladen aman dari akses pihak luar.
3. Tampilan Dasbor Direktori (File Manager)
   Setelah berhasil masuk, pengguna disajikan halaman utama yang menampilkan struktur direktori server secara lengkap. Data berupa folder, dokumen, dan media ditampilkan secara terorganisir, sehingga pengguna dapat dengan cepat memantau aset digital yang tersimpan di dalam basis data cloud.
4. Fitur Unggah (Upload) Berkas
   Pengguna dapat menambahkan file baru dari memori perangkat (laptop atau ponsel pintar) ke dalam peladen. Karena proses transfer berjalan di jaringan lokal, kecepatan unggah data berukuran besar menjadi jauh lebih stabil dan efisien tanpa menguras kuota internet eksternal.
5. Manajemen Berkas (Ubah Nama dan Hapus)
   Sistem menyediakan fitur operasional dasar untuk merapikan data. Pengguna dapat mengubah nama dokumen atau menghapus file yang sudah usang. Saat sebuah file dihapus dari antarmuka, sistem akan langsung menghapusnya dari memori fisik (SSD/HDD) peladen sehingga kapasitas ruang penyimpanan kembali lega.
6. Fitur Unduh (Download) Data
   Fitur ini memungkinkan pengguna untuk menarik file dari cloud kembali ke penyimpanan lokal perangkat mereka. Fungsi ini sangat berguna ketika pengguna membutuhkan akses data secara luring (offline) di luar jaringan server.
7. Pembuatan dan Organisasi Folder

Untuk menjaga kerapian penyimpanan, sistem mengizinkan pengguna untuk membuat direktori (folder) baru. Fitur ini mempermudah kategorisasi berkas, misalnya memisahkan antara folder tugas akademik, dokumen pekerjaan, dan media hiburan.

8. Monitoring Kapasitas Penyimpanan
   Dasbor dilengkapi dengan indikator visual yang memberikan informasi kapasitas penyimpanan secara real-time. Administrator dapat melihat total kapasitas yang terpakai dan sisa ruang kosong pada perangkat keras, sehingga proses pengelolaan atau pencadangan tambahan dapat diantisipasi sebelum disk penuh.
9. Akses Multi-Perangkat Serenta
   Sistem dikonfigurasi agar mampu melayani permintaan (request) dari beberapa client sekaligus. Berkat isolasi lingkungan oleh Docker, server dapat menangani proses unggah dari satu smartphone bersamaan dengan proses unduh dari laptop tanpa menyebabkan gangguan performa (crash).
10. Pengakhiran Sesi (Logout)
    Fitur ini digunakan untuk menutup sesi pengguna secara aman setelah selesai beraktivitas. Sangat esensial untuk mencegah penyalahgunaan wewenang akses jika perangkat client sedang digunakan oleh pihak lain..

Pengujian fungsionalitas sistem dilakukan menggunakan metode *Black Box Testing* yang dipadukan dengan pendekatan *User Acceptance Test* (UAT). Evaluasi ini berfokus pada kesesuaian respons sistem terhadap interaksi pengguna, tanpa keharusan untuk meninjau struktur kode program di baliknya. Melalui pengujian ini, kelayakan setiap fungsi krusial, seperti transfer data dan navigasi direktori, dapat dipastikan berjalan sesuai standar kebutuhan operasional harian. Berikut adalah rekapitulasi hasil pengujian terhadap fitur utama:

No.	Fitur yang Diuji	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
1	Autentikasi Pengguna (Login)	Sistem menerima akses dengan *username* dan *password* yang valid.	Login berhasil dan pengguna masuk ke dasbor.	Berhasil
2	Validasi login salah	Sistem menolak masuk jika kata sandi salah.	Muncul peringatan akses ditolak.	Berhasil
3	Tampilan Direktori Utama	Sistem menampilkan seluruh daftar folder dan *file* yang tersimpan.	Berkas muncul terstruktur pada halaman utama.	Berhasil
4	Proses Unggah Data	Pengguna dapat mengirim *file* dari perangkat ke peladen.	Berkas tersimpan dengan utuh tanpa *corrupt*.	Berhasil
5	Ubah Nama & Hapus Berkas	Pengguna dapat mengganti nama atau menghapus data permanen.	Perubahan langsung tersinkronisasi di peladen.	Berhasil
6	Proses Unduh Data	Sistem memproses penarikan data ke memori perangkat *client*.	Berkas terunduh secara otomatis dengan lancar.	Berhasil
7	Pembuatan Folder Baru	Sistem membuat direktori kosong baru sesuai nama yang diinput.	Sistem membuat direktori kosong baru sesuai nama yang diinput.	Berhasil
8	Monitor Ruang Penyimpanan	Sistem menunjukkan pemakaian kapasitas *disk* secara akurat.	Data sisa kapasitas sesuai dengan memori fisik PC.	Berhasil
9	Akses Multi-Perangkat	Sistem mampu diakses dari laptop dan ponsel pintar secara bersamaan.	Tidak ada gangguan koneksi atau putus jaringan.	Berhasil
1	Logout	Sistem memutuskan akses dan	Sesi terhenti dengan aman dan	Berhasil

0	kembali ke halaman *login*.	normal.

V.PEMBAHASAN

Sebelum sistem penyimpanan berbasis cloud lokal ini diterapkan, manajemen data pribadi masih sangat bergantung pada penggunaan media fisik (seperti flashdisk dan hardisk eksternal) serta layanan public cloud. Penggunaan media fisik sering kali merepotkan karena membutuhkan bongkar-pasang perangkat, rentan hilang, dan berisiko terkena virus saat dipindahkan antar komputer. Di sisi lain, penggunaan public cloud sering terhambat oleh lambatnya kecepatan internet dan terbatasnya kuota penyimpanan gratis. Setelah implementasi private cloud storage ini berjalan, seluruh kendala tersebut dapat diminimalisir. Waktu yang dibutuhkan untuk mengunggah atau mengunduh file berukuran besar (seperti aset desain, video, maupun arsip coding) menjadi jauh lebih singkat karena transfer data sepenuhnya memanfaatkan kecepatan bandwidth jaringan lokal (LAN/WiFi), tanpa harus bergantung pada latensi penyedia layanan internet (ISP). Dari sisi keamanan dan pengelolaan, sistem ini membantu pengguna memusatkan seluruh informasi di satu peladen mandiri. Hal ini sangat menurunkan risiko kehilangan data apabila perangkat client (seperti laptop atau ponsel) mengalami kerusakan fisik. Keberadaan dasbor utama yang dilengkapi fitur monitoring kapasitas storage secara real-time juga memberikan manfaat yang signifikan. Pengguna dapat dengan mudah memantau sisa ruang kosong pada SSD/HDD secara instan melalui browser, sehingga tindakan antisipasi atau pencadangan tambahan dapat dilakukan jauh sebelum kapasitas peladen penuh. Secara keseluruhan, implementasi sistem ini membuktikan bahwa arsitektur perangkat lunak berbasis Docker mampu menghadirkan solusi infrastruktur IT yang efisien dengan memanfaatkan perangkat keras yang sudah ada (legacy hardware). Pemanfaatan teknologi open-source ini memperkuat temuan dari berbagai penelitian sebelumnya [masukkan sitasi jurnal 1] dan [masukkan sitasi jurnal 2], yang menyatakan bahwa peladen lokal mampu memberikan akses data yang lebih aman, cepat, dan lebih terjangkau dibandingkan solusi komersial.

VI. KESIMPULAN

Berdasarkan hasil perancangan dan implementasi yang telah dipaparkan pada bab-bab sebelumnya, dapat ditarik kesimpulan bahwa pengembangan private cloud storage menggunakan kontainerisasi Docker pada Ubuntu Server telah berhasil diwujudkan sesuai dengan analisis kebutuhan pengguna. Penerapan metode pengembangan System Development Life Cycle (SDLC) model Waterfall terbukti efektif dalam menjaga alur pengerjaan tetap terstruktur, mulai dari fase identifikasi masalah, perancangan topologi jaringan, hingga tahap pengujian akhir. Sistem yang dibangun mampu menjalankan fungsinya sebagai wadah penyimpanan terpusat, di mana pengguna dapat melakukan manajemen berkas (unggah, unduh, hapus, dan pengelompokan folder) dengan mudah melalui antarmuka web. Isolasi lingkungan menggunakan mesin Docker memastikan aplikasi cloud dapat berjalan dengan sangat stabil dan ringan tanpa membebani sistem operasi utama peladen. Selain itu, fitur pemantauan kapasitas penyimpanan secara aktual (real-time) sangat membantu administrator dalam mengelola memori fisik dengan lebih terukur dan mencegah terjadinya penumpukan data yang dapat merusak sistem. Hasil evaluasi yang dilakukan melalui pendekatan Black Box Testing menunjukkan bahwa seluruh fitur inti, mulai dari layar autentikasi (login) pengguna hingga proses transfer berkas antar-perangkat, berjalan dengan responsif tanpa ditemukan adanya malfungsi atau celah kegagalan proses. Secara keseluruhan, penerapan private cloud ini memberikan dampak penyelesaian masalah yang sangat nyata. Kendala terkait tingginya biaya langganan layanan pihak ketiga dan kekhawatiran mengenai privasi data digital berhasil diatasi. Sistem ini tidak hanya menjadi alternatif penyimpan data yang aman dan gratis secara jangka panjang, tetapi juga membuktikan bahwa implementasi teknologi peladen rumahan dapat diakses dan dioperasikan secara efisien untuk mendukung produktivitas sehari-hari.

Kontribusi Penulis: Akmal: Melakukan konseptualisasi penelitian, perancangan metodologi, analisis kebutuhan sistem, implementasi aplikasi, serta penulisan draf awal naskah penelitian.
Moh. Imam Hidayatullah: Berperan dalam proses validasi sistem, pengujian fungsional aplikasi, visualisasi data hasil uji coba, serta penyuntingan naskah akhir untuk keperluan publikasi.
Seluruh Penulis telah membaca dan menyetujui versi naskah yang telah diterbitkan.
Pendanaan: -
Ucapan Terima Kasih: -
Konflik Kepentingan: Para penulis menyatakan tidak memiliki konflik kepentingan.
Ketersediaan Data: -

Persetujuan Berdasarkan Informasi ORCID: Tidak tersedia.
Penulis Pertama: https: -
Penulis Kedua: https: -
Penulis Ketiga: -

DAFTAR PUSTAKA

[1] W. Eckerson, W. Stallings, Network Security Essentials: Applications and Standards, 6th ed., Pearson, 2020.

[2] T. H. Davenport and J. G. Harris, Competing on Analytics: The New Science of Winning, Boston, MA, USA: Harvard Business School Press, 2020.

[3] A. S. Tanenbaum and D. J. Wetherall, Computer Networks, 6th ed., Pearson, 2021.

[4] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” NIST Special Publication 800-145, 2020.

[5] J. Kurose and K. Ross, Computer Networking: A Top-Down Approach, 8th ed., Pearson, 2021.

[6] Canonical Ltd., “Ubuntu Server Guide,” 2023.

[7] R. Oppliger, SSL and TLS: Theory and Practice, Artech House, 2020.

[8] E. Rescorla, SSL and TLS: Designing and Building Secure Systems, Addison-Wesley, 2021.

[9] M. Scarfone and P. Mell, “Guide to Intrusion Detection and Prevention Systems,” NIST, 2020.

[10] N. Provos and T. Holz, Virtual Honeypots: From Botnet Tracking to Intrusion Detection, Addison-Wesley, 2020.

[11] S. Garfinkel and G. Spafford, Practical UNIX and Internet Security, O’Reilly, 2021.

[12] T. Bradley, “Essential Linux Security,” Packt Publishing, 2022.

[13] F. P. E. Putra, R. W. Efendi, A. B. Tamam and W. A. Pramadi , "TREN DAN PRAKTIK TERBAIK DALAM PENGEMBANGAN WEB BERBASIS API : KAJIAN LITERATUR TERHADAP FRAMEWORK LARAVEL DAN REACT," *NFOMATEK: Jurnal Informatika, Manajemen dan Teknologi,* vol. 27, no. 1, pp. 165-178, 2025.

[14] F. P. E. Putra, R. W. Efendi , A. B. Tamam and W. A. Pramadi, "Trends and Best Practices in API-Based Web Development Using Laravel and React," *Brilliance,* vol. 5, no. 1, pp. 223-233, 2025.

[15] F. P. E. Putra, O. F. Kusuma, M. Mursidi and A. Hamzah , "Comparative Analysis of Laravel and Symfony in PHP-Based Web Application Developmen," *Brilliance,* vol. 5, no. 1, pp. 272-280, 2025.

[16] F. P. E. Putra, Ubaidi, A. Hamzah , W. A. Pramadi and A. Nuraini , "Systematic Literature Review: Security Gap Detection On Websites Using Owasp Zap," *Brilliance,* vol. 4, no. 1, pp. 348-355, 2024.

[17] F. P. E. Putra, S. Katsir , M. U. Mansyur and K. Z. Imam, "OPTIMALISASI PENGEMBANGAN SISTEM INFORMASI LABORATORIUM TERINTEGERASI SISTEM AKADEMIK MENGGUNAKAN METODE SCRUMB," *Jurnal Informatika,* vol. 23, no. 2, pp. 183-198, 2023.

[18] F. P. E. Putra, Ubaidi, R. N. Saputra , F. M. Haris and S. N. R. Barokah, "Application of Internet of Things Technology in Monitoring Water Quality in Fishponds," *Brilliance,* vol. 4, no. 1, pp. 356-361, 2024.

[19] F. P. E. Putra, M. N. Arifin, K. Z. Imam and E. Saputra, "Pengembangan Sistem Informasi Laboratorium Terintegerasi Sistem Akademik Menggunakan Agile Scrum," *Jurnal Informasi dan Teknologi,* vol. 5, no. 2, pp. 109-119, 2023.

[20] F. P. E. Putra, L. Fitriyah, Z. Naimah and S. A. Rofika, "Evaluasi Kinerja Aplikasi Wireshark dalam Monitoring Jaringan Kecil dengan Topologi Star dan Bus," *Jurnal Ilmiah ILKOMINFO-Jurnal Ilmu Komputer dan Informatika,* vol. 8, no. 2, pp. 164-176, 2025.

[21] F. P. E. Putra, M. A. Mahmud and I. S. Maqom, "Pengembangan Sistem Pemantauan Lingkungan Berbasis Internet of Things (IoT) di Kampus," *Digital Transformation Technology,* vol. 3, no. 2, pp. 996-1001, 2023.

[22] S. Arifin , N. P. Dewi, Ubaidi, M. N. Arifin and F. P. E. Putra, "APLIKASI PENGOLAHAN DATA MAHASISWA KKN PADA UNIVERSITAS MADURA," *InsandComtech,* vol. 8, no. 2, p. 24, 2023.

[23] F. P. E. Putra, S. R. Sutarsih, Sofiyulloh, P. Pernama and M. U. Mansyur, "OPTIMALISASI

PERANCANGAN APLIKASI MANAJEMEN DATA KOLOMAN, DI DESA PULAU MANDANGIN SAMPANG – MADURA BERBASIS WEBSITE," *RABIT : Jurnal Teknologi dan Sistem Informasi Univrab,* vol. 9, no. 2, pp. 285-294, 2024.

[24] N. H. Hari, F. P. E. Putra, M. N. Arifin, M. Y. Zain and I. Dermawan, "Desain dan Perancangan Smart Campus berbasis ZigBee Wireless Sensor Network," *Jurnal Inovasi Teknik dan Edukasi Teknologi,* vol. 1, no. 11, pp. 842-850, 2021.

[25] F. P. E. Putra, M. Riski, Riyan, Y. R. Febriani and M. U. Mansyur, "Optimization of Web Based Academic Information System Design to Increase Efficiency in Junior High Schools," *Jurnal Informasi dan Teknologi,* vol. 6, no. 2, pp. 150-158, 2024.

[26] A. Nemeth et al., *UNIX and Linux System Administration Handbook*, 5th ed., Pearson, 2021.

[27] Nextcloud GmbH, "Nextcloud Administration Manual," 2023.

[28] OWASP Foundation, "Top 10 Web Application Security Risks," 2023.

[29] S. Singh and N. Sharma, "Security Challenges in Cloud Computing," *International Journal of Computer Applications*, vol. 182, no. 12, pp. 20–27, 2021.

[30] R. Buyya et al., *Cloud Computing: Principles and Paradigms*, Wiley, 2020.

[31] D. Bernstein, "Cloud Computing Security," *IEEE Computer Society*, vol. 43, no. 4, pp. 40–47, 2020.

[32] M. Bishop, *Computer Security: Art and Science*, 2nd ed., Addison-Wesley, 2021.

[33] K. Behl and S. Behl, *Cybersecurity and Cyberwar: What Everyone Needs to Know*, Oxford University Press, 2020.

[34] J. Andress, *The Basics of Information Security*, 3rd ed., Syngress, 2021.

[35] M. Burhanuddin and D. Hermanto, "Pengembangan Aplikasi Kasir Berbasis Web Dalam Pengelolaan Transaksi Keuangan," *Jatekom,* vol. 1, no. 1, pp. 1-12, 2025.

[36] M. S. Chandra and A. Ichwani, "Perancangan dan Implementasi Aplikasi Kasir Berbasis Web dengan Spring Boot (Studi Kasus: CV Mulia Tetap Jaya)," *Innovative : Journal of Social Science Research,* vol. 5, no. 1, pp. 6758-6775, 2025.

[37] M. Z. Akbar, M. A. Nur, M. F. Sabana and T. Tanjung , "Perancangan Aplikasi Kasir Berbasis Website Pada Toko Sembako Menggunakan Metode Waterfall," *OKTAL: Jurnal Ilmu Komputer dan Sains,* vol. 1, no. 8, pp. 1274-1281, 2022.

[38] N. P. N. K. Dewi, E. M. Dharma and A. I. I. Paramitha, "Analisa Dan Perancangan Sistem Informasi Penjualan Dan Pengelolaan Persediaan Pada PT Alfajores Bali Enak Menggunakan Metode Waterfall," *SMART TECHNO,* vol. 5, no. 1, pp. 59-64, 2023.

[39] S. Sotnik, V. Manakov and V. Lyashenko, "Overview: PHP and MySQL Features for Creating Modern Web Projects," *IJAISR Open Archive.,* 2023.

[40] E. C. Palma, M. P. Ortega and W. D. Z. Romero, "Bootstrap as a Tool for Web Development and Graphic Optimization on Mobile Devices," *Artificial Intelligence, Computer and Software Engineering Advances, Proceedings of the CIT 2020,* vol. 1, pp. 290-302, 2021.

[41] D. Lestari, "Sistem Monitoring Stok Barang Berbasis Web untuk Usaha Mikro dan Menengah," *Jurnal Mantik,* vol. 7, no. 3, pp. 1220-1229, 2023.

[42] H. Nur, "Penggunaan Metode Waterfall Dalam Rancang Bangun Sistem Informasi Penjualan," *Generation Journal,* vol. 3, no. 1, p. 1, 2019.

[43] P. A. Sunarya, U. Rahardja, N. P. L. Santoso , Mulyati, K. I. Mustofa and D. Bennet, "Pengaruh Metode Waterfall dalam Penyempurnaan Proses Pengembangan Sistem Informasi Akademik secara Sistematis: Impact of Waterfall Method on Systematic Academic Information System Development," *Technomedia Journal,* vol. 9, no. 3, pp. 360-373, 2025.

[44] M. Tabrani, "PENERAPAN METODE WATERFALL PADA SISTEM INFORMASI INVENTORI PT. PANGAN SEHAT SEJAHTERA," *Jurnal Inkofar,* vol. 1, no. 2, p. 12, 2018.

[45] C. Perdana, Maharani and M. A. Wijaya, "Implementasi Framework Bootstrap 5 Pada Perancangan Front-End Website MC BRO di PT X," *JURNAL SISTEM INFORMASI GALUH,* vol. 2, no. 1, pp. 30-43, 2024.

[46] R. Yanuar and Fersellia, "Implementasi E-commerce Penjualan Kaos Distro Berbasis Website pada Toko Art n Soul Purbalingga," *INSOLOGI: Jurnal Sains dan Teknologi,* vol. 4, no. 3, pp. 379-394, 2025.

[47] J. V. Tamah, I. Kanedi and R. Zulfiandry, "Sistem Pemesanan Berbasis Web Pada Usaha Bouquetqu.AD Menggunakan PhP Dan MySQL," *Jurnal Media Infotama*, vol. 21, no. 2, pp. 488-497, 2025.

[48] R. S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill Education, 2015.

[49] S. R. Wicaksono, Blackbox Testing Teori dan Studi Kasus, Malang: Seribu Bintang, 2022.

[50] I. Sommerville, Software Engineering (9th ed.), Addison-Wesley, 2011.