

BAB III

METODOLOGI PENELITIAN

3.1 Metode Penelitian

Metode penelitian merupakan tahapan yang digunakan sebagai pedoman dalam pelaksanaan penelitian agar proses penelitian dapat berjalan secara sistematis dan terarah. Pada penelitian ini, metode yang digunakan meliputi jenis dan pendekatan penelitian, objek dan subjek penelitian, teknik pengumpulan data, alat dan bahan penelitian, serta prosedur pengembangan sistem. Metode penelitian ini disusun untuk mendukung proses perancangan dan pembangunan sistem otomatisasi *cloud hosting* berbasis *Docker Container* menggunakan pendekatan pengembangan *Incremental*.

3.1.1 Jenis dan Pendekatan Penelitian

Penelitian ini menggunakan jenis penelitian Rancang Bangun (Research and Development / R&D). Metode R&D digunakan karena penelitian ini tidak hanya menganalisis suatu permasalahan, tetapi juga menghasilkan produk konkret berupa platform otomatisasi cloud hosting berbasis container. Menurut Waruwu (2023), metode R&D digunakan untuk menghasilkan produk tertentu dan menguji keefektifan produk tersebut. Oleh karena itu, pendekatan R&D dinilai sangat sesuai karena berorientasi pada pengembangan produk teknologi yang dapat diimplementasikan secara langsung untuk memecahkan masalah praktis manajemen server yang selama ini masih dilakukan secara manual.

Adapun pendekatan yang digunakan dalam penelitian ini adalah Pendekatan Kualitatif Deskriptif yang bersifat teknis, dengan didukung oleh Metode Pengembangan Sistem Incremental. Pendekatan ini dipilih berdasarkan beberapa pertimbangan teknis:

1. Analisis Teknis dan Kebutuhan Peneliti melakukan analisis mendalam terhadap kebutuhan fungsional sistem, konfigurasi jaringan, serta arsitektur integrasi antar-framework (Next.js dan NestJS).

2. Eksperimental Teknis, Peneliti melakukan analisis mendalam terhadap kebutuhan fungsional sistem, konfigurasi jaringan, serta arsitektur integrasi antar-framework (Next.js dan NestJS).
3. Pengembangan Sistem Bertahap (Incremental Model), Pengembangan platform menerapkan metode Incremental, di mana sistem dibangun dan dirilis secara bertahap dalam beberapa iterasi (increment). Pendekatan ini dipilih agar setiap modul utama—mulai dari autentikasi, manajemen container, monitoring infrastruktur, hingga interactive web terminal—dapat dirancang, diuji, dan dievaluasi fungsinya secara mandiri dan terukur sebelum diintegrasikan ke dalam seluruh platform CaaS.

Melalui kombinasi jenis R&D, pendekatan kualitatif teknis, dan penerapan metode Incremental ini, diharapkan penelitian dapat menghasilkan platform yang tidak hanya fungsional dan responsif secara teknis, tetapi juga memiliki landasan prosedur pengembangan yang sistematis serta dapat dipertanggungjawabkan secara akademis (Rusandi & Rusli, 2021).

3.1.2 Lokasi dan Waktu Penelitian

Bagian ini menjelaskan mengenai tempat pelaksanaan pengembangan sistem serta rincian jadwal kegiatan penelitian. Penentuan lokasi dan waktu ini disesuaikan dengan kebutuhan infrastruktur teknis peneliti serta mengikuti jadwal akademik resmi yang telah ditetapkan oleh universitas. Penelitian ini dilakukan secara mandiri dengan memanfaatkan infrastruktur perangkat keras dan jaringan lokal milik peneliti yang berlokasi di Bandung. Seluruh proses pengembangan perangkat lunak, mulai dari konfigurasi server Linux, integrasi backend controller, hingga pengujian kontainer Docker, dilakukan dalam lingkungan pengembangan terkontrol (development environment) untuk memastikan stabilitas sistem sebelum diimplementasikan pada layanan cloud publik. Pelaksanaan penelitian ini berlangsung selama Semester Genap Tahun Akademik 2025/2026, terhitung mulai bulan April 2026 hingga Agustus 2026. Rangkaian kegiatan diawali pada bulan April dengan fokus pada finalisasi bimbingan, pelaksanaan Seminar Proposal, serta

penyelesaian revisinya. Memasuki periode April hingga Juni, peneliti melaksanakan tahap inti yang meliputi pengembangan sistem menggunakan metode Incremental, pengumpulan data teknis, dan bimbingan laporan secara intensif. Selanjutnya, pada bulan Juni dilakukan pendaftaran serta pelaksanaan Seminar Hasil dengan batas waktu penyelesaian revisi hingga pertengahan Juli. Tahap akhir penelitian ini dilaksanakan pada bulan Juli hingga Agustus, yang mencakup pendaftaran dan pelaksanaan Sidang, penyelesaian naskah final, serta diakhiri dengan proses administrasi Yudisium pada bulan Agustus 2026 sebagai tanda tuntasnya seluruh rangkaian penelitian.

3.1.3 Objek dan Subjek Penelitian

Penentuan objek dan subjek penelitian merupakan langkah penting untuk menetapkan batasan operasional dan sumber data dalam pengembangan sistem. Bagian ini merinci entitas teknologi yang menjadi fokus utama penelitian serta pihak-pihak yang berperan dalam proses validasi dan pengujian sistem.

1. Objek Penelitian

Objek penelitian adalah variabel atau hal yang menjadi titik perhatian dalam suatu penelitian. Dalam penelitian ini, objek penelitian yang ditetapkan adalah:

1. Sistem Otomatisasi *Container as a service* Berupa *platform* berbasis *web* yang mengintegrasikan manajemen kontainer *Docker*, konfigurasi *reverse proxy*, dan fitur *interactive terminal*.
2. Integrasi Teknologi Fokus pada cara kerja *backend* NestJS dalam mengontrol *Docker Engine* melalui *library* SSH2 dan protokol *WebSocket* untuk komunikasi *real-time*.

2. Subjek Penelitian

Subjek penelitian merupakan pihak yang terlibat dalam memberikan data, masukan, serta melakukan pengujian terhadap fungsionalitas sistem. Dalam penelitian ini, subjek penelitian difokuskan pada Mahasiswa Teknik Informatika yang memiliki pengalaman atau pemahaman dasar menggunakan teknologi Container as a Service (CaaS).

Pihak ini bertindak sebagai subjek dalam pengujian pengguna (user testing) untuk memberikan penilaian terhadap pengalaman penggunaan (User Experience) serta kemudahan operasional sistem otomatisasi cloud hosting. Subjek ini dipilih karena memiliki pemahaman dasar mengenai pengelolaan container, sehingga mampu mengevaluasi fungsionalitas sistem secara lebih relevan sesuai dengan kebutuhan pengguna di lingkungan cloud computing.

3.1.4 Teknik Pengumpulan Data

Teknik pengumpulan data dilakukan untuk memperoleh informasi yang akurat dan relevan guna mendukung proses perancangan serta implementasi sistem. Adapun teknik yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Studi Pustaka (*Library Research*)

Peneliti melakukan pengumpulan data dengan cara mempelajari buku, jurnal ilmiah, dokumentasi resmi, dan artikel terkait untuk memperkuat landasan teori. Dalam hal ini, peneliti telah meninjau 15 penelitian terdahulu yang relevan dengan teknologi *Docker*, *cloud hosting*, dan pengembangan *web* menggunakan *framework* modern. Studi pustaka ini juga mencakup pendalaman dokumentasi teknis mengenai NestJS, Next.js, SSH2 *library*, dan protokol WebSocket.

2. Observasi Teknis

Peneliti melakukan pengamatan langsung terhadap proses manajemen kontainer secara manual melalui *Command Line Interface* (CLI) pada sistem operasi Linux

Mint XFCE. Observasi ini bertujuan untuk memetakan alur kerja (*workflow*) yang akan diotomatisasi ke dalam sistem, mulai dari proses *pull image*, pembuatan kontainer, hingga pemantauan *log* server.

3. Eksperimen Laboratorium

Data teknis diperoleh melalui serangkaian uji coba pada lingkungan pengembangan lokal. Peneliti melakukan eksperimen terhadap pengiriman perintah SSH dari *backend* ke server tujuan serta pengujian transmisi data *real-time* menggunakan Xterm.js untuk memastikan fungsionalitas *terminal* interaktif berjalan dengan baik.

3.1.5 Alat dan Bahan Penelitian

Pengembangan sistem otomatisasi *cloud hosting* ini didukung oleh penggunaan perangkat keras dan perangkat lunak yang memadai guna menunjang kebutuhan komputasi, terutama dalam menjalankan teknologi kontainerisasi. Berikut adalah rincian alat dan bahan yang digunakan dalam penelitian ini:

1) Perangkat Keras (*Hardware*)

Perangkat keras utama yang digunakan peneliti dalam merancang dan mengimplementasikan sistem ini meliputi:

1. Laptop Utama: Digunakan sebagai stasiun kerja (*workstation*) untuk penulisan kode dan manajemen sistem.
 - 1) Processor: Intel Core i5-5200U
 - 2) Memori (RAM): 16GB DDR3
 - 3) Penyimpanan: 512GB
2. Server target yang digunakan dalam penelitian ini berupa *Virtual Private Server* (VPS) yang berfungsi sebagai *host* bagi *Docker Engine* yang dikelola oleh sistem.

2) Perangkat Lunak (Software)

Perangkat lunak yang digunakan mencakup sistem operasi dan berbagai *tools* pengembangan dengan fokus pada efisiensi *Command Line Interface* (CLI):

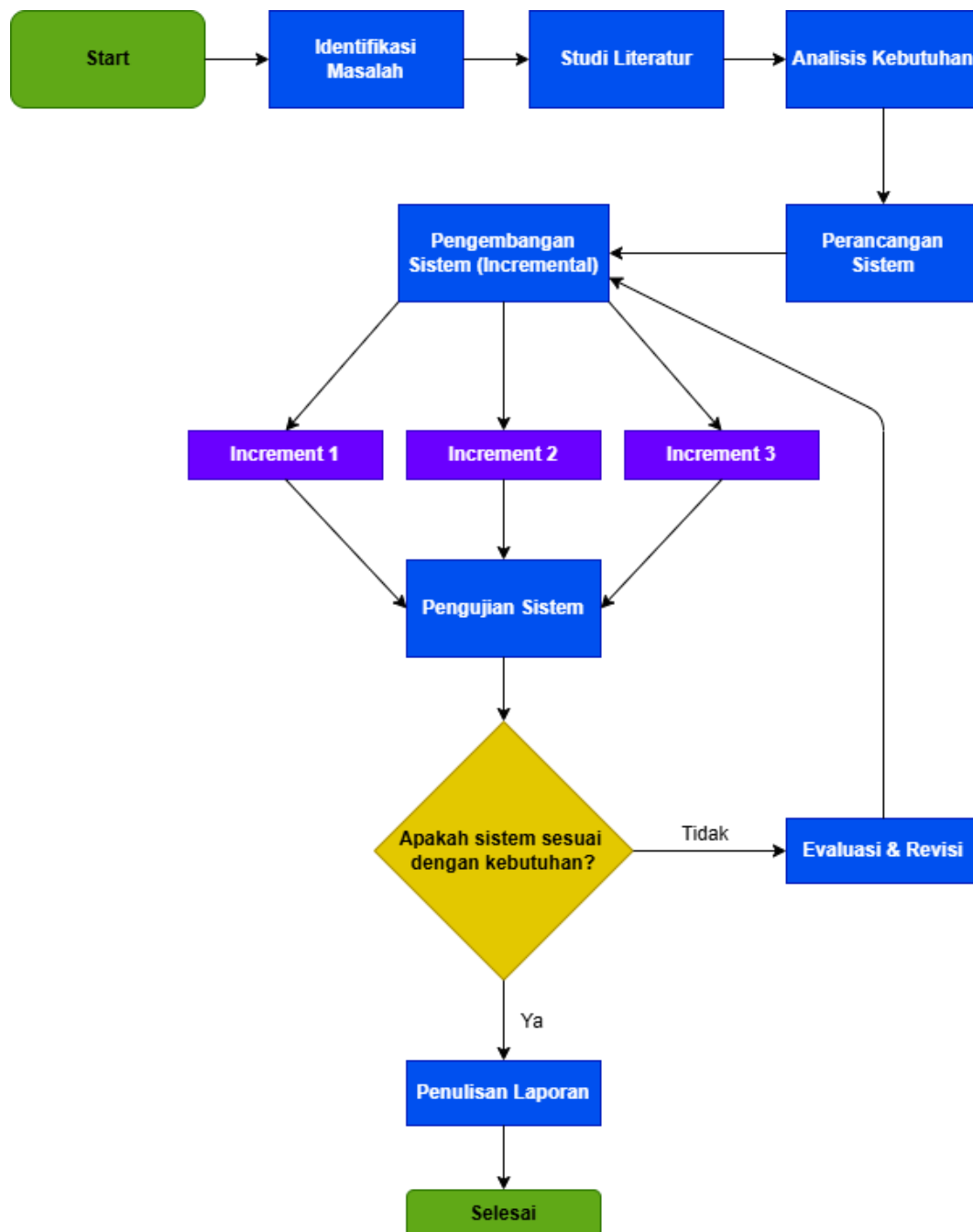
1. Sistem Operasi: Linux Mint XFCE (Lingkungan pengembangan utama).
2. Bahasa Pemrograman & Runtime: Node.js dan TypeScript.
3. *Framework Backend*: NestJS untuk membangun arsitektur server yang modular.
4. *Framework Frontend*: Next.js untuk antarmuka pengguna yang responsif dan mendukung SSR.
5. *Database & ORM*: PostgreSQL sebagai penyimpanan data dan Prisma ORM untuk pemetaan objek *database*.
6. Editor Kode: Visual Code Studio / Neovim
7. *Terminal Emulator*: Kitty yang dikombinasikan dengan Tmux untuk manajemen sesi *terminal*.
8. *Library Komunikasi*: SSH2 *Library* (untuk kontrol server) dan *WebSocket/Socket.io* (untuk komunikasi *real-time*).
9. Layanan Kontainer: *Docker Engine* dan *Docker Compose*.
10. *Version Control*: Git untuk manajemen repositori kode program.

3.2 Prosedur Penelitian

Prosedur penelitian merupakan rangkaian tahapan yang dilakukan secara sistematis untuk mencapai tujuan penelitian. Dalam pengembangan sistem ini, prosedur yang dijalankan mengacu pada tahapan dalam metode Incremental, yang memungkinkan pengembangan dilakukan secara bertahap berdasarkan modul atau fitur tertentu.

Setiap modul yang telah dikembangkan akan melalui tahap pengujian untuk memastikan bahwa sistem berjalan sesuai dengan kebutuhan yang telah ditentukan. Apabila pada tahap pengujian ditemukan bahwa sistem belum memenuhi kebutuhan, maka akan dilakukan evaluasi dan revisi sebelum kembali ke tahap

pengembangan. Proses ini dilakukan secara berulang hingga sistem dinyatakan sesuai dengan kebutuhan.



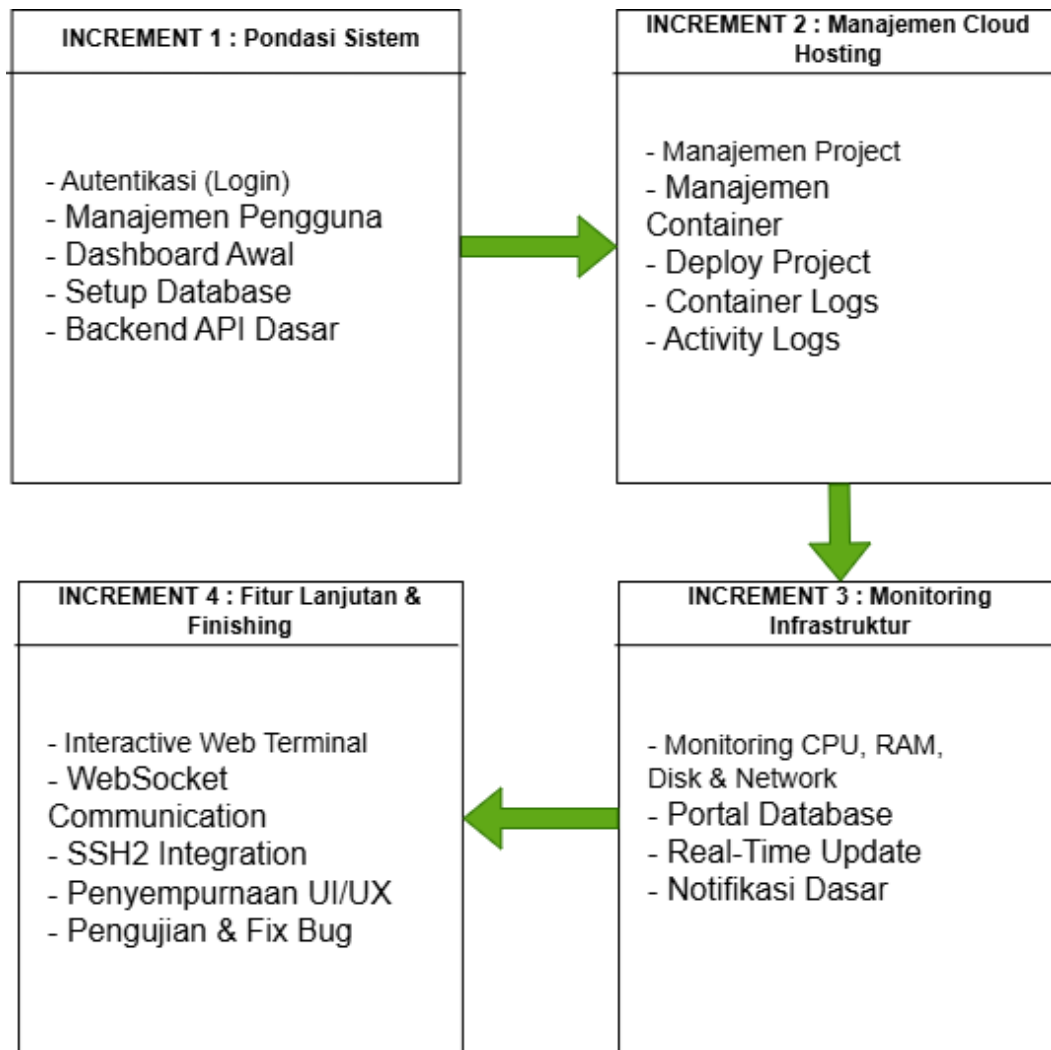
Gambar 3. 1 Flowchart Prosedur Penelitian

3.2.1 Tahap Perancangan Sistem (*Design*)

Tahap perancangan ini dilaksanakan berdasarkan hasil analisis kebutuhan sistem yang telah didefinisikan sebelumnya, serta mengacu pada fase *Design* dalam siklus pengembangan Metode *Incremental*. Pada tahap ini, peneliti merancang arsitektur sistem secara menyeluruh yang melibatkan *backend* NestJS, *frontend* Next.js, dan skema basis data menggunakan Prisma ORM. Perancangan ini juga mencakup alur komunikasi data antara sistem dengan *Docker Engine* melalui protokol *WebSocket*, serta perancangan antarmuka pengguna (*User Interface*) agar setiap modul yang dikembangkan pada tiap inkremen memiliki struktur arsitektur yang terukur dan terintegrasi dengan baik.

3.2.2 Tahap Implementasi dan Pengembangan (*Build*)

Proses pengkodean (coding) dilakukan secara bertahap (*incremental*) pada lingkungan pengembangan Linux. Pembagian tahapan pengembangan sistem menjadi beberapa modul/increment disajikan secara visual pada Gambar 3.3 berikut:



Gambar 3. 2 Tahapan Pengembangan Sistem Metode Incremental

Berdasarkan Gambar 3.2 di atas, rincian fungsionalitas dan modul yang dikembangkan pada setiap tahap incremental meliputi:

1. Incremental 1 (Pondasi Sistem & Backend Utama)

Fokus pada pembangunan inti backend (Node.js/NestJS), sistem autentikasi pengguna, setup basis data, serta konektivitas SSH2 dasar ke server.

2. Incremental 2 (Manajemen Cloud Hosting)

Fokus pada pengembangan fitur pengelolaan kontainer Docker, deployment project, pemantauan log kontainer, serta otomatisasi konfigurasi reverse proxy.

3. Incremental 3 (Monitoring & Antarmuka Real-time)

Fokus pada implementasi antarmuka frontend (Next.js), pemantauan sumber daya server (CPU, RAM, Disk, Network) secara real-time, serta sistem notifikasi.

4. Incremental 4 (Fitur Lanjutan & Penyempurnaan)

Fokus pada penyediaan fitur Interactive Web Terminal (Xterm.js), integrasi komunikasi WebSocket, penyempurnaan UI/UX, serta tahap pengujian menyeluruh dan perbaikan bug (fixing bug).

3.2.3 Tahap Pengujian dan Validasi

Setiap modul yang telah selesai dikembangkan akan melalui tahap pengujian internal untuk memastikan bahwa setiap fungsi sistem berjalan sesuai dengan kebutuhan yang telah ditentukan. Pengujian ini dilakukan oleh peneliti dengan memverifikasi fitur-fitur utama seperti manajemen kontainer, eksekusi perintah *Docker*, serta integrasi antar komponen sistem.

Setelah seluruh modul terintegrasi, dilakukan pengujian pengguna (*user testing*) secara terbatas bersama mahasiswa yang memiliki pengalaman dalam penggunaan teknologi container. Pengujian ini bertujuan untuk memperoleh masukan dari sisi pengalaman pengguna (*user experience*) serta kemudahan dalam operasional sistem.

Tahap pengujian dan validasi ini dijadwalkan selesai sebelum pendaftaran Seminar Hasil pada bulan Juli 2026.

3.2.4 Tahap Penyusunan Laporan Akhir

Langkah terakhir adalah mendokumentasikan seluruh proses penelitian dan hasil pengujian ke dalam naskah skripsi. Tahap ini diakhiri dengan pelaksanaan Sidang Tugas Akhir.

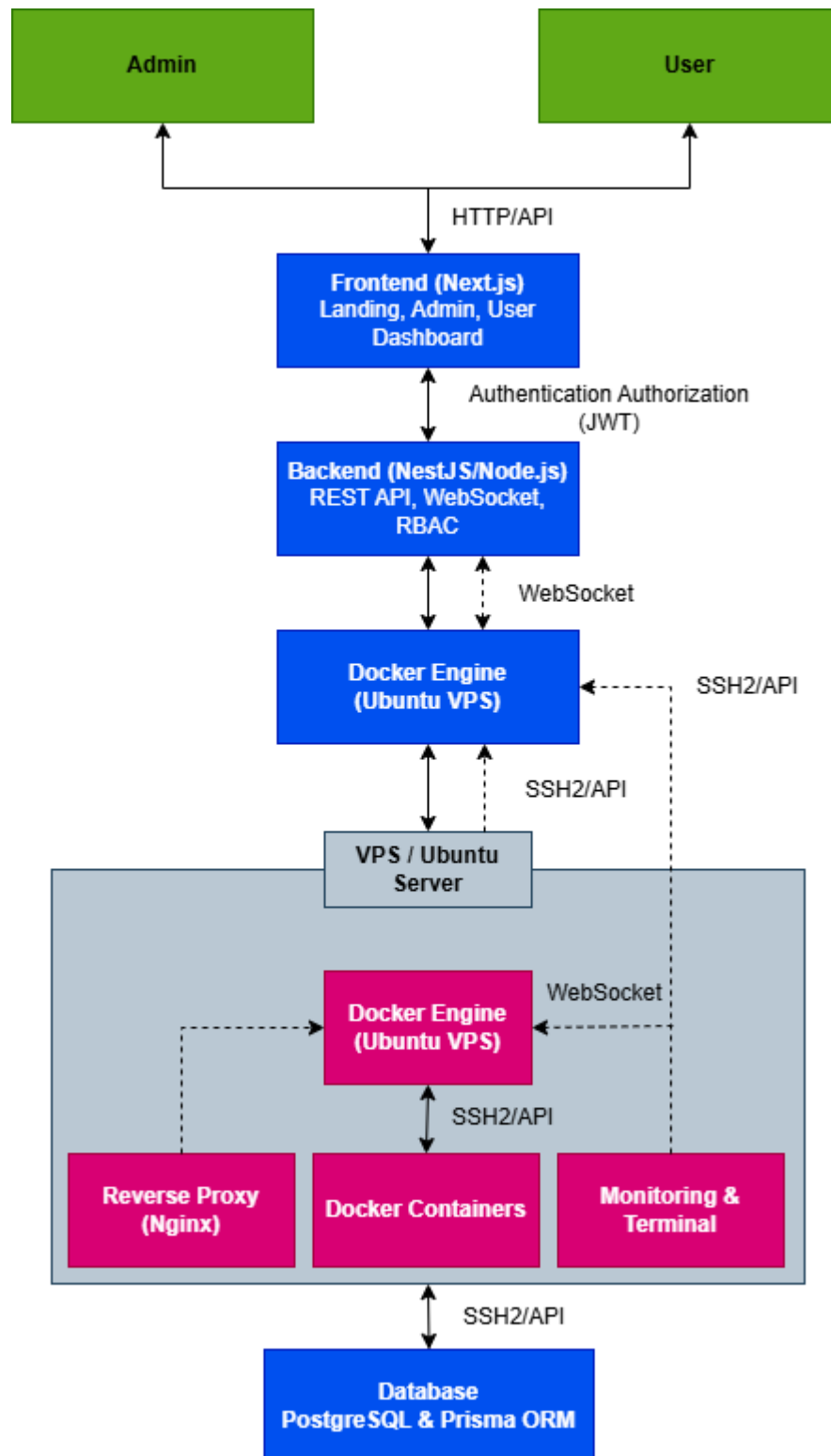
3.3 Rancangan Sistem

Rancangan sistem merupakan tahapan perencanaan yang dilakukan sebelum proses implementasi sistem. Pada tahap ini dilakukan perancangan arsitektur sistem, alur proses, struktur database, serta antarmuka pengguna yang akan digunakan dalam pengembangan sistem otomatisasi cloud hosting berbasis Docker Container. Perancangan sistem bertujuan untuk memberikan gambaran umum mengenai cara kerja sistem, hubungan antar komponen, serta mekanisme pengelolaan layanan berbasis container yang akan dibangun. Adapun perancangan dalam penelitian ini meliputi perancangan arsitektur sistem, use case diagram, activity diagram, perancangan database, perancangan antarmuka pengguna, serta rancangan infrastruktur Docker yang digunakan dalam sistem.

3.3.1 Arsitektur Sistem

Arsitektur sistem merupakan gambaran umum mengenai hubungan dan alur komunikasi antar komponen dalam sistem yang dibangun. Pada penelitian ini, sistem dirancang menggunakan arsitektur berbasis *client-server* dengan memanfaatkan teknologi *Docker Container* sebagai inti pengelolaan layanan *cloud hosting*. Pengguna mengakses sistem melalui antarmuka web berbasis Next.js yang berfungsi sebagai frontend. Seluruh permintaan pengguna akan diteruskan ke backend yang dibangun menggunakan Node.js dan NestJS untuk memproses logika sistem. Backend kemudian berkomunikasi dengan *Docker Engine* melalui *REST API*, *SSH2*, dan *WebSocket* untuk melakukan otomatisasi manajemen container, monitoring layanan, serta akses terminal secara real-time. Selain itu, sistem juga memanfaatkan database sebagai media penyimpanan data pengguna, konfigurasi container, dan informasi layanan lainnya. Reverse proxy digunakan untuk mengatur

routing domain menuju container Docker yang aktif pada server VPS. Dengan arsitektur ini, sistem mampu menyediakan layanan cloud hosting berbasis Container as a Service (CaaS) yang terintegrasi dan mudah dikelola melalui dashboard web.



Gambar 3. 3 Diagram Arsitektur Sistem

Berdasarkan Gambar 3.3 Diagram Arsitektur Sistem, alur kerja sistem dimulai ketika pengguna mengakses dashboard melalui *browser web*. Antarmuka sistem dibangun menggunakan *framework* Next.js yang berfungsi sebagai *frontend* untuk menampilkan fitur manajemen container, monitoring sistem, pengelolaan *reverse proxy*, serta akses terminal berbasis web.

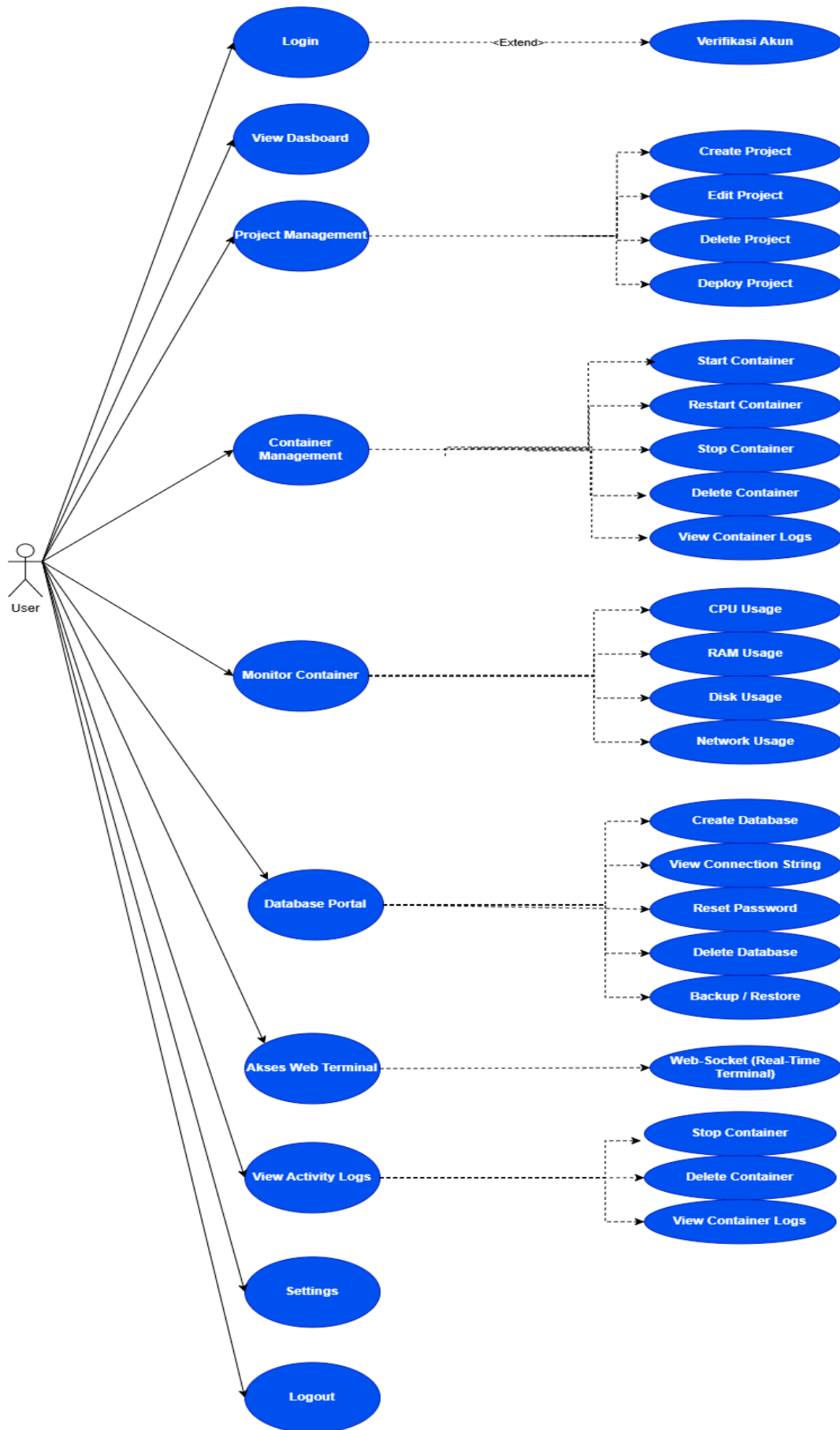
Frontend kemudian berkomunikasi dengan *backend* menggunakan protokol HTTP/API dan *WebSocket*. *Backend* sistem dibangun menggunakan Node.js dan NestJS yang berfungsi sebagai pusat pengelolaan logika sistem. *Backend* bertanggung jawab dalam memproses permintaan pengguna, mengelola autentikasi, menghubungkan sistem dengan *Docker Engine*, serta menangani komunikasi *real-time* untuk fitur monitoring dan terminal interaktif.

Backend selanjutnya melakukan komunikasi dengan *Docker Engine* yang berjalan pada server VPS berbasis Ubuntu melalui REST API, SSH2, dan *WebSocket*. *Docker Engine* berfungsi untuk menjalankan dan mengelola *Docker Container* secara otomatis, termasuk proses pembuatan, penghentian, penghapusan, serta monitoring container aplikasi pengguna.

3.3.2 Use Case Diagram

Use Case Diagram digunakan untuk menggambarkan interaksi antara aktor dengan sistem yang dikembangkan. Diagram ini menunjukkan fungsi-fungsi utama yang dapat diakses oleh setiap aktor sesuai dengan hak akses yang dimiliki. Pada sistem otomatisasi cloud hosting berbasis Docker Container yang dikembangkan, terdapat dua aktor utama yaitu **User** dan **Admin**. User merupakan pengguna yang memanfaatkan layanan cloud hosting untuk melakukan deployment dan pengelolaan aplikasi, sedangkan Admin memiliki hak akses tambahan untuk mengelola keseluruhan sistem melalui halaman Admin Panel.

1. Use Case Diagram User



Gambar 3. 4 Use Case Diagram User

Berdasarkan Gambar 3.4, aktor User berinteraksi secara langsung dengan sistem untuk melakukan pengelolaan aplikasi yang dimilikinya. Seluruh aktivitas pengguna dilakukan melalui dashboard berbasis web sehingga proses deployment dan monitoring dapat dilakukan tanpa harus mengakses server secara langsung. Adapun penjelasan masing-masing use case adalah sebagai berikut.

1). Login

Use case ini digunakan oleh User untuk melakukan autentikasi ke dalam sistem menggunakan akun yang telah terdaftar. Setelah proses autentikasi berhasil dilakukan, pengguna akan diarahkan menuju halaman dashboard.

2). Dashboard

Use case ini berfungsi untuk menampilkan ringkasan informasi mengenai project, container, deployment, serta penggunaan resource yang dimiliki oleh pengguna.

3). Project Management

Use case ini digunakan untuk mengelola project aplikasi yang akan dideploy. Pengguna dapat membuat project baru, mengubah informasi project, maupun menghapus project yang sudah tidak digunakan.

4. Deploy Application

Use case ini digunakan untuk melakukan proses deployment aplikasi melalui file ZIP maupun integrasi dengan repository GitHub. Sistem akan secara otomatis membangun Docker Image, membuat container, dan menjalankan aplikasi.

5. Container Management

Use case ini digunakan untuk mengelola container yang dimiliki oleh pengguna. Aktivitas yang dapat dilakukan meliputi menjalankan, menghentikan, me-restart, melihat informasi container, maupun menghapus container.

6. Monitoring Container

Use case ini digunakan untuk memantau kondisi container secara real-time. Informasi yang ditampilkan meliputi penggunaan CPU, RAM, disk, jaringan, serta status container yang sedang berjalan.

7. Domain Management

Use case ini digunakan untuk mengelola domain maupun subdomain yang terhubung dengan aplikasi yang telah dideploy. Pengguna dapat menambahkan, mengubah, maupun menghapus konfigurasi domain.

8. Database Portal

Use case ini digunakan untuk mengelola layanan database yang digunakan oleh aplikasi. Pengguna dapat membuat database baru, melihat informasi koneksi database, melakukan reset password, maupun menghapus database apabila sudah tidak digunakan.

9. Web Terminal

Use case ini memungkinkan pengguna mengakses terminal server secara langsung melalui browser menggunakan komunikasi berbasis WebSocket sehingga proses eksekusi perintah dapat dilakukan secara real-time.

10. Activity Logs

Use case ini digunakan untuk melihat riwayat aktivitas deployment, container, maupun aktivitas sistem yang berkaitan dengan project milik pengguna.

11. Settings

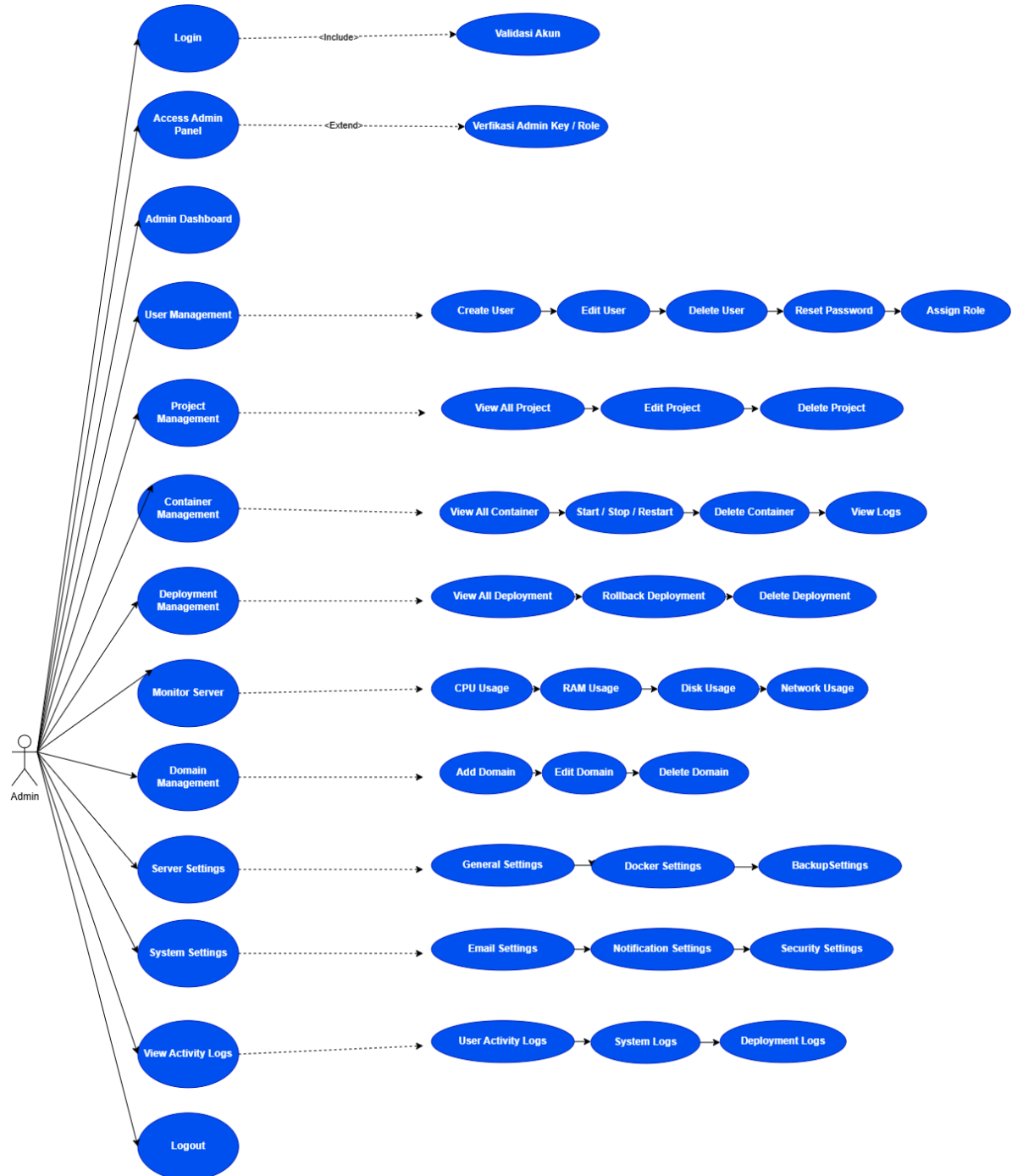
Use case ini digunakan untuk mengelola informasi akun pengguna, mengganti password, mengatur integrasi GitHub, serta melakukan konfigurasi SSH Key dan preferensi akun lainnya.

12. Logout

Use case ini digunakan untuk mengakhiri sesi penggunaan sistem dan keluar dari dashboard secara aman.

Berdasarkan Use Case Diagram User, sistem mampu menyediakan seluruh layanan yang dibutuhkan pengguna untuk melakukan deployment aplikasi berbasis Docker Container secara mandiri melalui dashboard berbasis web tanpa perlu melakukan konfigurasi server secara langsung.

2. Use Case Diagram Admin



Gambar 3. 5 Use Case Diagram Admin

Berdasarkan Gambar 3.5, aktor Admin memiliki hak akses yang lebih tinggi dibandingkan User. Selain dapat menggunakan seluruh fitur yang dimiliki oleh User, Admin juga memiliki akses ke Admin Panel untuk melakukan pengelolaan sistem secara menyeluruh. Akses ke halaman Admin hanya dapat dilakukan setelah sistem berhasil memverifikasi hak akses administrator. Adapun penjelasan masing-masing use case adalah sebagai berikut.

1. Login

Use case ini digunakan oleh Admin untuk melakukan autentikasi ke dalam sistem menggunakan akun administrator.

2. Access Admin Panel

Use case ini digunakan untuk mengakses halaman Admin Panel. Sebelum halaman ditampilkan, sistem akan melakukan verifikasi terhadap Admin Key maupun Role Administrator sehingga hanya pengguna dengan hak akses tertentu yang dapat memasuki halaman ini.

3. Dashboard Admin

Use case ini digunakan untuk menampilkan informasi keseluruhan sistem, seperti jumlah pengguna, jumlah project, jumlah container, deployment yang sedang berjalan, serta penggunaan resource server.

4. User Management

Use case ini digunakan untuk mengelola seluruh akun pengguna yang terdaftar pada sistem. Admin dapat menambah pengguna baru, mengubah data pengguna, menghapus akun, melakukan reset password, serta mengatur hak akses (role).

5. Project Management

Use case ini digunakan untuk melihat dan mengelola seluruh project milik pengguna yang tersimpan pada sistem. Admin dapat melakukan perubahan maupun penghapusan project apabila diperlukan

6. Container Management

Use case ini digunakan untuk mengelola seluruh Docker Container yang berjalan pada server. Admin dapat menjalankan, menghentikan, me-restart, menghapus container, serta melihat log container.

7. Deployment Management

Use case ini digunakan untuk memantau seluruh proses deployment aplikasi. Admin dapat melihat riwayat deployment, melakukan rollback deployment apabila terjadi kegagalan, maupun menghapus data deployment.

8. Monitor Server

Use case ini digunakan untuk memantau kondisi server secara real-time. Informasi yang ditampilkan meliputi penggunaan CPU, RAM, penyimpanan, serta aktivitas jaringan server.

9. Domain Management

Use case ini digunakan untuk mengelola seluruh konfigurasi domain dan subdomain yang digunakan oleh aplikasi pengguna, termasuk konfigurasi reverse proxy menggunakan Nginx.

10. Server Settings

Use case ini digunakan untuk melakukan konfigurasi server, Docker Engine, serta pengaturan backup sistem.

11. System Settings

Use case ini digunakan untuk mengatur konfigurasi global sistem, seperti pengaturan email, notifikasi, keamanan, serta parameter sistem lainnya.

12. Activity Logs

Use case ini digunakan untuk melihat seluruh aktivitas yang terjadi pada sistem, meliputi aktivitas pengguna, deployment aplikasi, maupun aktivitas server sebagai bahan monitoring dan audit.

13. Logout

Use case ini digunakan untuk mengakhiri sesi administrator dan keluar dari sistem secara aman.

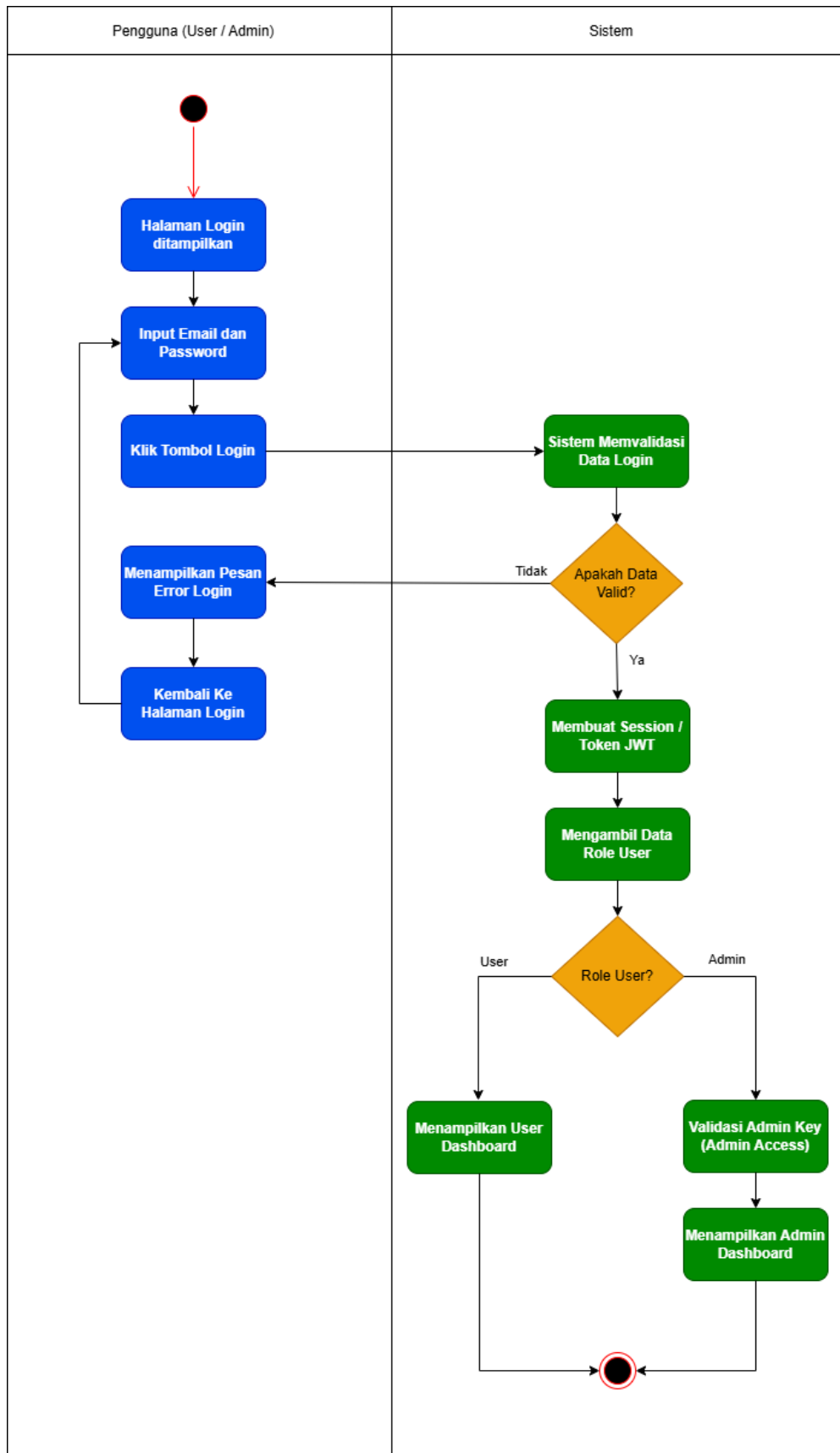
Berdasarkan Use Case Diagram Admin, administrator memiliki kewenangan untuk mengelola seluruh komponen sistem cloud hosting berbasis Docker Container, mulai dari manajemen pengguna, deployment aplikasi, monitoring server, hingga konfigurasi sistem secara terpusat melalui Admin Panel.

3.3.3 Activity Diagram

Activity Diagram digunakan untuk menggambarkan alur aktivitas yang terjadi pada sistem berdasarkan interaksi antara pengguna dan sistem. Diagram ini memberikan gambaran mengenai urutan proses yang dijalankan oleh sistem dalam menangani setiap fitur utama. Pada penelitian ini, Activity Diagram dibuat untuk menggambarkan proses login, pengelolaan container, monitoring container, dan akses web terminal.

1. Activity Diagram Login

Activity Diagram Login digunakan untuk menggambarkan alur proses autentikasi pengguna pada sistem cloud hosting berbasis Docker Container. Diagram ini menunjukkan interaksi antara pengguna dan sistem mulai dari proses memasukkan kredensial hingga validasi data login. Tujuan dari proses login adalah untuk memastikan bahwa hanya pengguna yang memiliki hak akses yang dapat menggunakan fitur-fitur yang tersedia pada sistem.

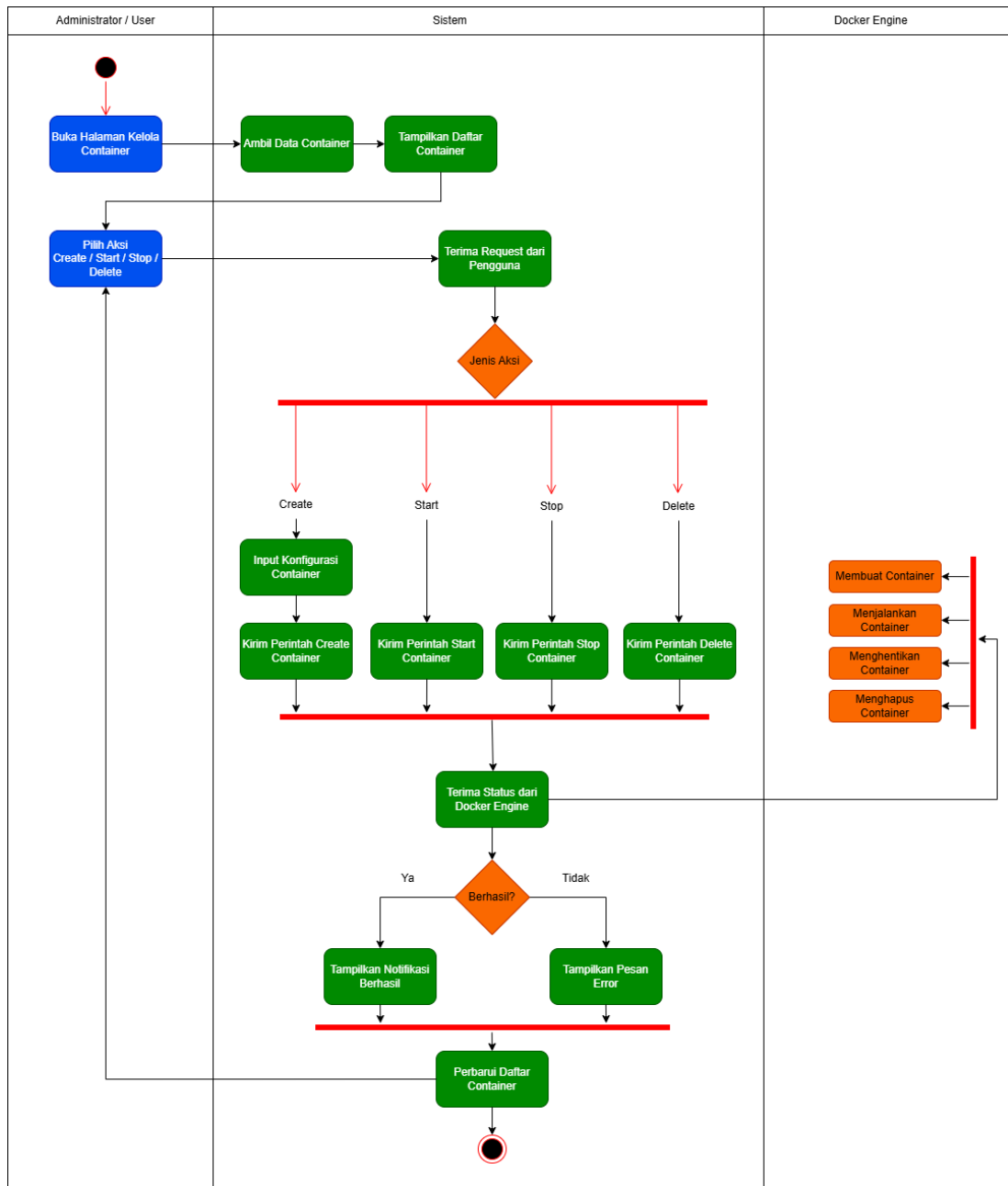


Gambar 3. 6 Activity Diagram Login

Berdasarkan Gambar 3.6, proses login dimulai ketika halaman login ditampilkan kepada pengguna. Selanjutnya pengguna memasukkan email dan password pada form login yang tersedia dan menekan tombol login untuk mengirimkan data autentikasi ke sistem. Setelah menerima data login, sistem melakukan proses validasi terhadap email dan password yang dimasukkan. Jika data yang diberikan valid, sistem akan membuat session atau token JWT sebagai identitas autentikasi pengguna. Setelah proses autentikasi berhasil, pengguna akan diarahkan ke halaman dashboard dan dapat mengakses seluruh fitur yang tersedia pada sistem.

2. Activity Diagram Kelola Container

Activity Diagram Kelola Container digunakan untuk menggambarkan alur proses pengelolaan container Docker yang dilakukan oleh pengguna melalui dashboard sistem. Diagram ini menunjukkan interaksi antara pengguna, sistem, dan Docker Engine dalam menjalankan berbagai operasi container seperti pembuatan, menjalankan, menghentikan, dan menghapus container secara otomatis melalui antarmuka web.



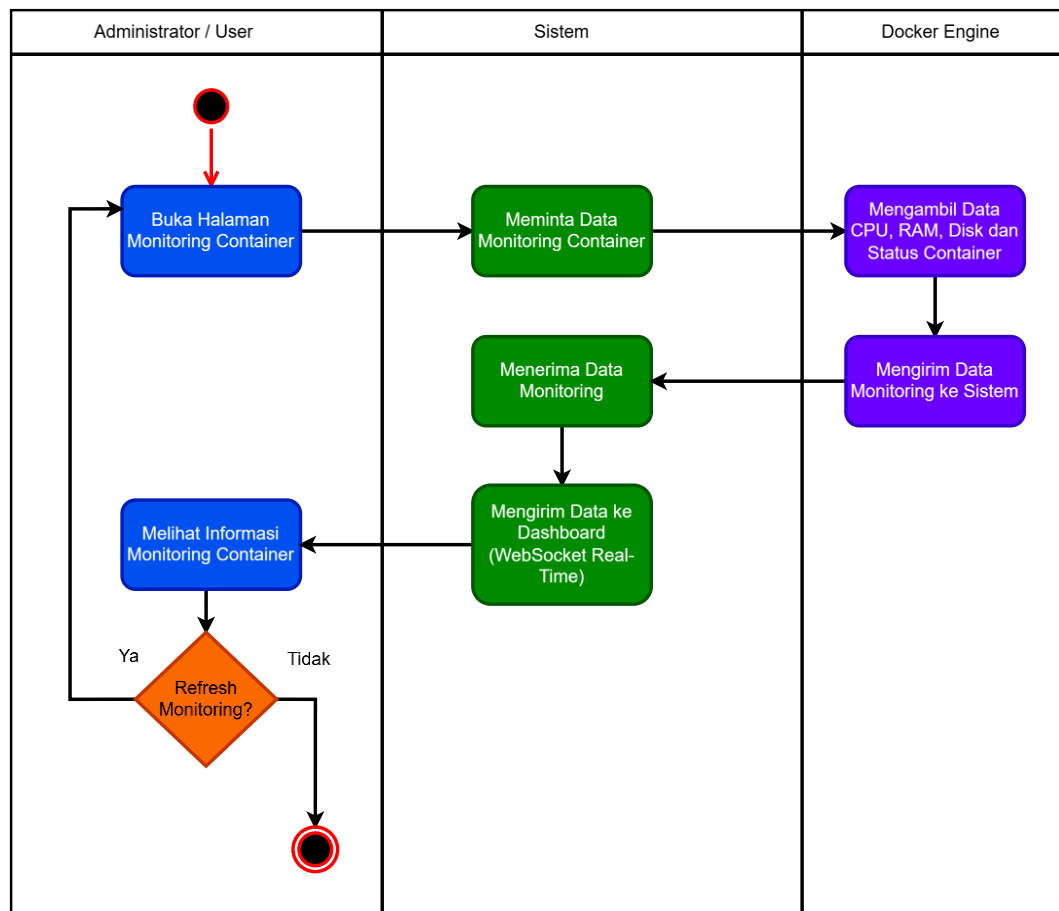
Gambar 3. 7 Activity Diagram Pengelolaan Container Docker

Berdasarkan Gambar 3.7, proses dimulai ketika Administrator membuka halaman Kelola Container pada dashboard sistem. Selanjutnya sistem mengambil data container yang tersedia dari Docker Engine dan menampilkan daftar container kepada pengguna.

Setelah daftar container ditampilkan, Administrator dapat memilih salah satu aksi yang tersedia, yaitu Create Container, Start Container, Stop Container, atau Delete Container. Sistem kemudian menerima request dari pengguna dan mengidentifikasi jenis aksi yang dipilih. Jika pengguna memilih Create Container, sistem akan meminta konfigurasi container terlebih dahulu sebelum mengirimkan perintah pembuatan container ke Docker Engine. Sedangkan untuk aksi Start, Stop, dan Delete, sistem akan langsung mengirimkan perintah yang sesuai kepada Docker Engine. Docker Engine kemudian mengeksekusi perintah yang diterima dan mengirimkan status hasil eksekusi kembali ke sistem. Sistem melakukan pengecekan terhadap status tersebut untuk menentukan apakah proses berhasil atau gagal.

3. Activity Diagram Monitoring Container

Activity Diagram Monitoring Container digunakan untuk menggambarkan alur proses pemantauan kondisi container Docker secara real-time melalui dashboard sistem. Diagram ini menunjukkan interaksi antara pengguna, sistem, dan Docker Engine dalam proses pengambilan, pengiriman, serta penyajian data monitoring container. Informasi yang ditampilkan meliputi penggunaan sumber daya seperti CPU, memori (RAM), disk, serta status container yang sedang berjalan. Proses monitoring dilakukan secara berkelanjutan menggunakan mekanisme WebSocket sehingga pengguna dapat memperoleh informasi kondisi container secara aktual tanpa perlu melakukan refresh halaman secara manual.

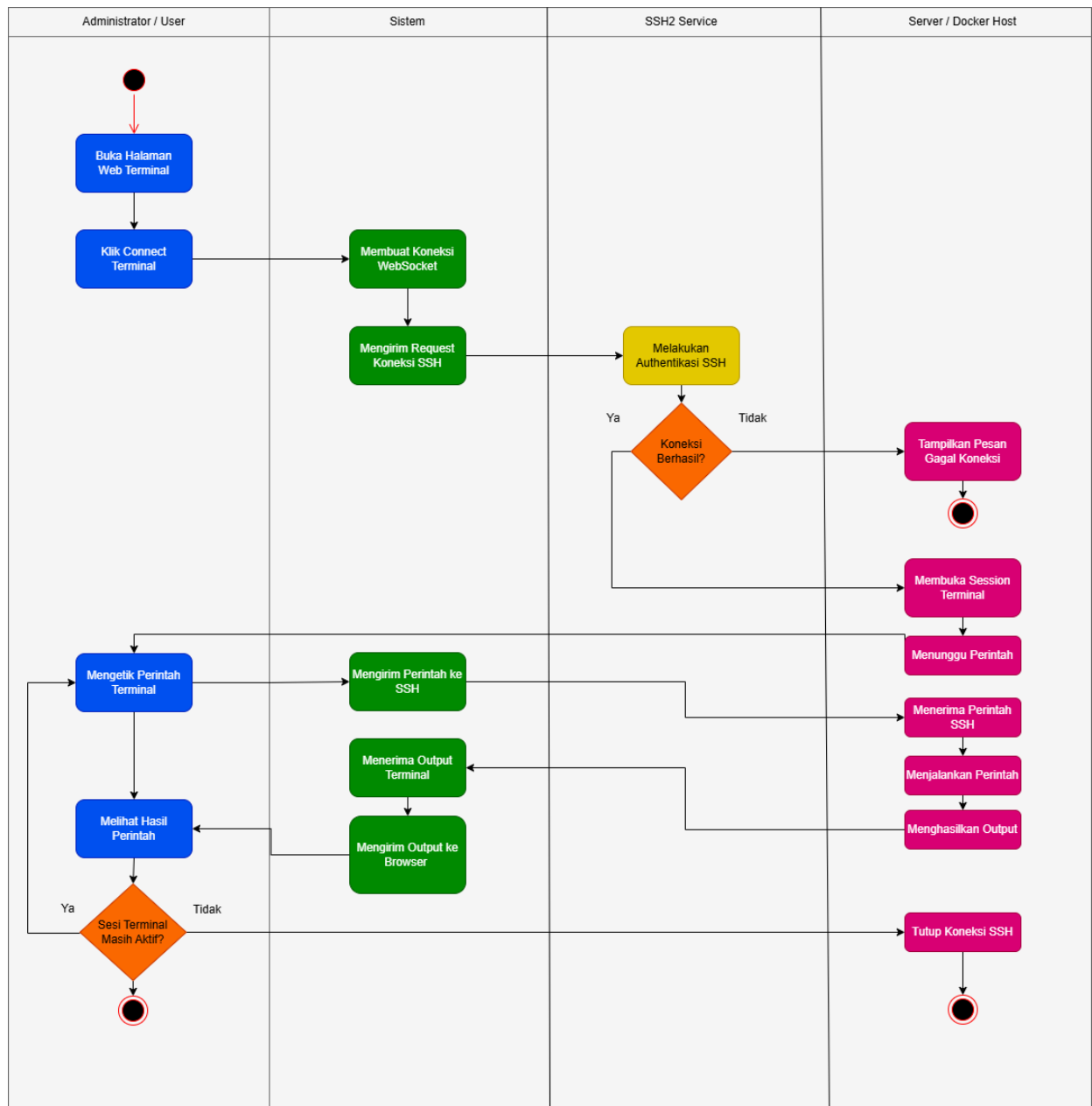


Gambar 3. 8 Activity Diagram Monitoring Container

Berdasarkan Gambar 3.8, proses monitoring container dimulai ketika Administrator membuka halaman Monitoring Container pada dashboard sistem. Setelah halaman monitoring diakses, sistem mengirimkan permintaan data monitoring kepada Docker Engine. Selanjutnya Docker Engine mengambil informasi terkait kondisi container yang sedang berjalan, meliputi penggunaan CPU, memori (RAM), disk, serta status container. Data hasil monitoring tersebut kemudian dikirimkan kembali ke sistem untuk diproses lebih lanjut.

4. Activity Diagram Web Terminal

Activity Diagram Web Terminal digunakan untuk menggambarkan alur proses akses dan penggunaan terminal berbasis web yang terintegrasi dengan server melalui protokol SSH. Diagram ini menunjukkan interaksi antara pengguna, sistem, layanan SSH2, serta server atau Docker Host dalam membangun koneksi terminal, mengirimkan perintah, menerima hasil eksekusi, dan menampilkan output secara real-time melalui browser. Fitur ini memungkinkan pengguna untuk melakukan pengelolaan server dan container secara langsung melalui antarmuka web tanpa harus menggunakan aplikasi SSH eksternal.



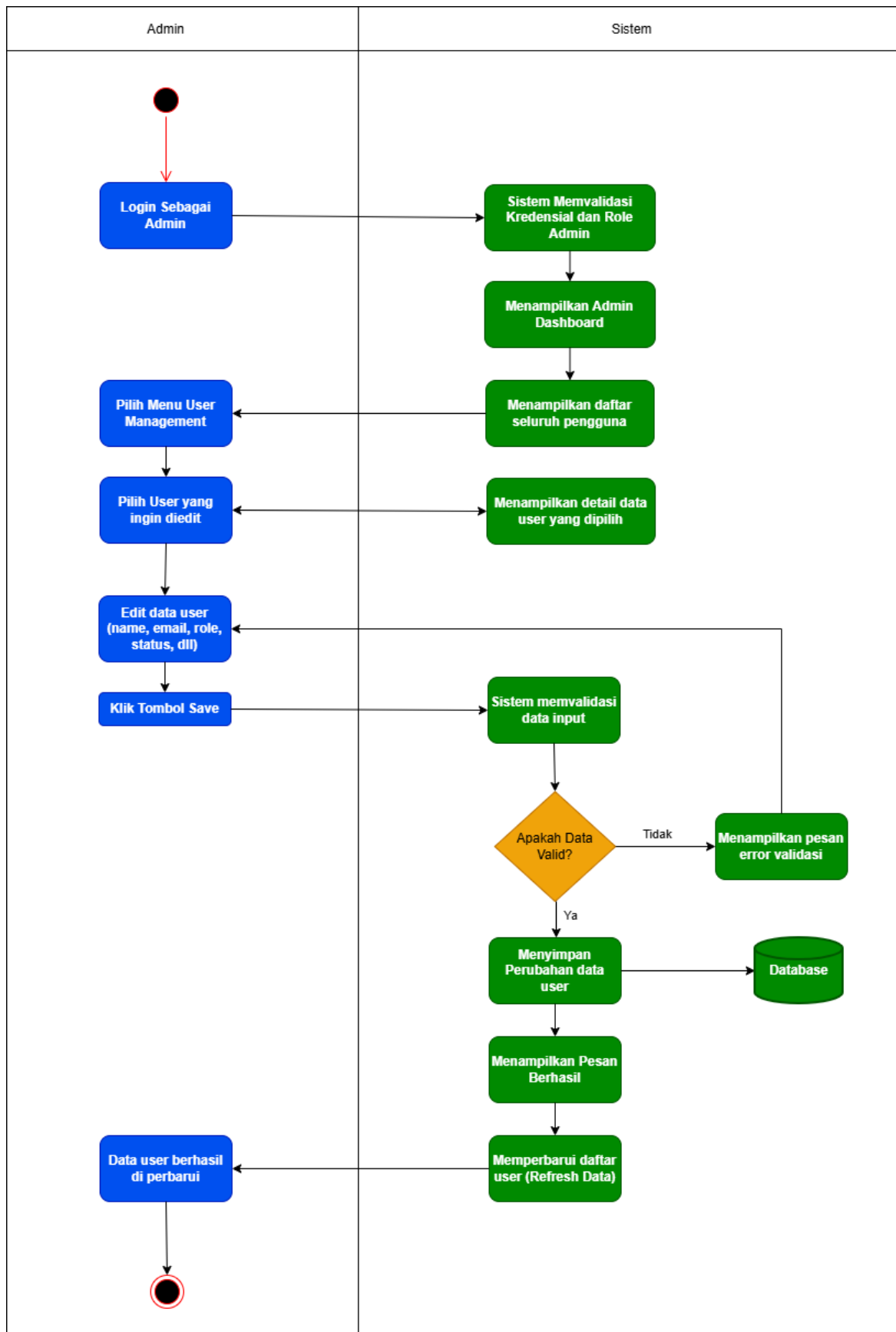
Gambar 3. 9 Activity Diagram Proses Web Terminal

Berdasarkan Gambar 3.9, proses dimulai ketika Administrator membuka halaman Web Terminal pada dashboard sistem dan memilih opsi Connect Terminal. Setelah itu sistem membangun koneksi WebSocket yang digunakan sebagai media komunikasi real-time antara browser dan server, kemudian mengirimkan permintaan koneksi SSH melalui layanan SSH2.

Layanan SSH2 selanjutnya melakukan proses autentikasi untuk memverifikasi kredensial yang digunakan dalam mengakses server. Jika proses autentikasi gagal, sistem akan menampilkan pesan gagal koneksi dan proses berakhir. Sebaliknya, jika autentikasi berhasil, sistem akan membuka sesi terminal pada server dan menunggu perintah dari pengguna.

5. Activity Diagram Admin Management

Activity Diagram Admin Management menggambarkan alur aktivitas yang dilakukan oleh administrator dalam mengelola data pengguna pada sistem. Diagram ini menunjukkan proses mulai dari administrator melakukan autentikasi, mengakses Admin Panel, memilih menu manajemen pengguna, melakukan perubahan data pengguna, hingga sistem memvalidasi dan menyimpan perubahan ke dalam basis data. Diagram ini bertujuan untuk memperlihatkan interaksi antara administrator dan sistem selama proses pengelolaan data pengguna berlangsung.



Gambar 3. 10 Activity Diagram Admin Management

Berdasarkan Gambar 3.10, proses diawali ketika administrator berhasil melakukan login ke dalam sistem menggunakan akun yang memiliki hak akses administrator. Setelah proses autentikasi berhasil, sistem melakukan validasi terhadap role administrator dan menampilkan dashboard utama. Administrator kemudian memilih menu *User Management* untuk mengakses daftar seluruh pengguna yang telah terdaftar pada sistem.

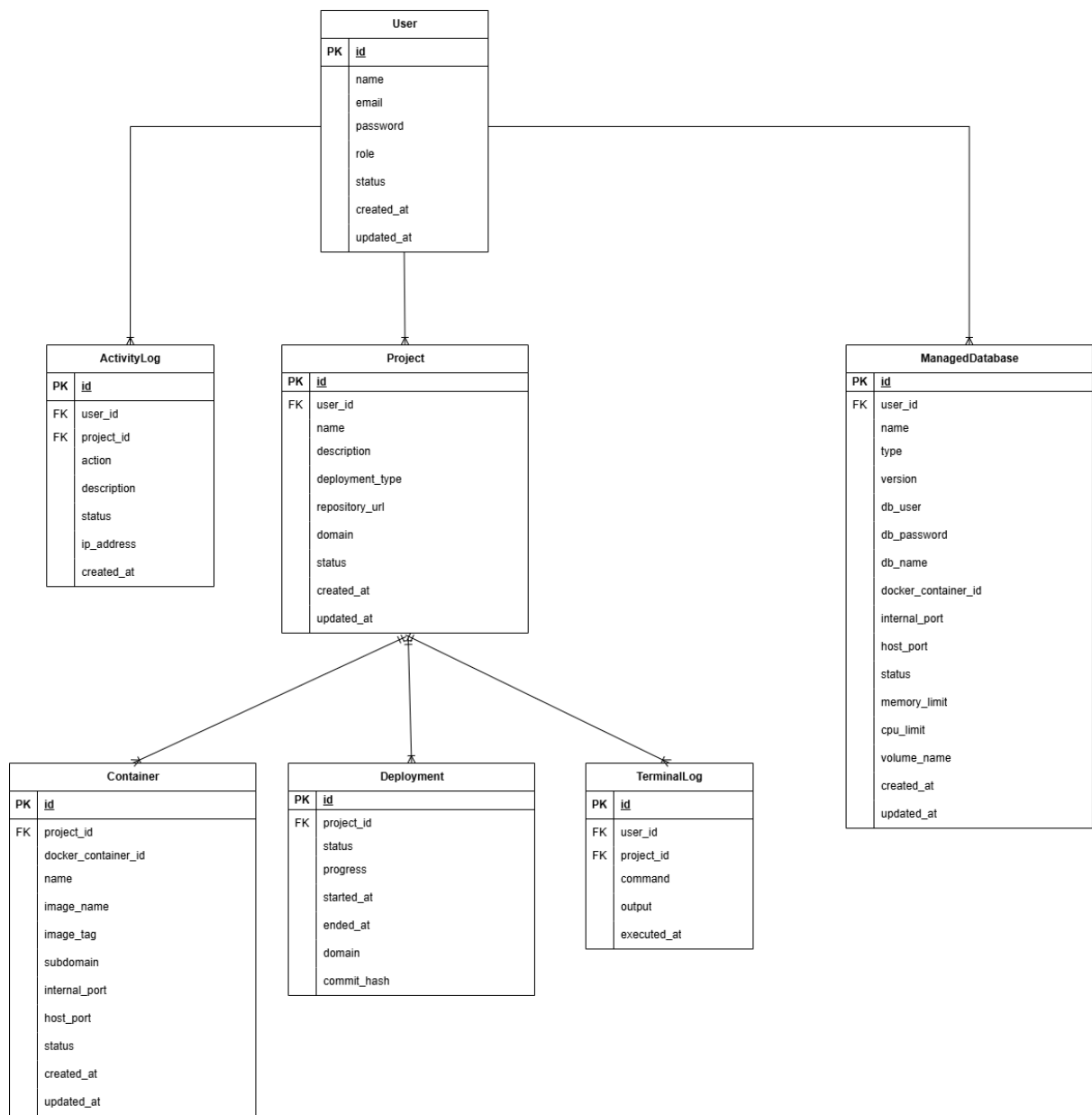
Sistem kemudian melakukan proses validasi terhadap seluruh data yang diinputkan. Apabila data yang dimasukkan tidak memenuhi ketentuan validasi, sistem akan menampilkan pesan kesalahan sehingga administrator dapat melakukan perbaikan terhadap data tersebut. Sebaliknya, apabila data dinyatakan valid, sistem akan menyimpan perubahan ke dalam basis data, memperbarui daftar pengguna, serta menampilkan notifikasi bahwa proses pembaruan data berhasil dilakukan. Setelah seluruh proses selesai, administrator dapat melihat data pengguna yang telah diperbarui pada halaman User Management.

3.3.4 Perancangan Database

Perancangan database merupakan tahap penyusunan struktur penyimpanan data yang digunakan oleh sistem. Database dirancang untuk mendukung proses autentikasi pengguna, pengelolaan container, monitoring sistem, serta penyimpanan informasi yang diperlukan oleh aplikasi. Pada penelitian ini, database menggunakan PostgreSQL yang dikelola melalui Prisma ORM untuk mempermudah proses pengelolaan data dan integrasi dengan backend Node.js.

1. Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) merupakan representasi visual yang digunakan untuk menggambarkan struktur database serta hubungan antar entitas yang digunakan dalam sistem. Perancangan database pada penelitian ini bertujuan untuk mendukung proses autentikasi pengguna, pengelolaan proyek, manajemen container Docker, pencatatan aktivitas sistem, dan penyimpanan riwayat penggunaan terminal berbasis web.



Gambar 3. 11 Entity Relationship Diagram Sistem

Berdasarkan Gambar 3.11, entitas User memiliki hubungan *one-to-many* dengan entitas Project, yang berarti satu pengguna dapat memiliki lebih dari satu proyek. Setiap proyek yang dibuat akan disimpan pada entitas Project dan terhubung dengan pengguna melalui atribut *user_id* sebagai *foreign key*.

2. Struktur Tabel Database

Struktur tabel database digunakan untuk menjelaskan atribut, tipe data, dan fungsi dari setiap field yang terdapat pada database sistem. Struktur tabel ini dirancang untuk mendukung proses autentikasi pengguna, pengelolaan proyek, manajemen container Docker, pencatatan aktivitas, dan penyimpanan riwayat penggunaan terminal.

1). Tabel User

Tabel User digunakan untuk menyimpan data pengguna yang memiliki akses ke sistem.

Field	Tipe Data	Keterangan
id	CUID	Primary Key
name	VARCHAR(100)	Nama pengguna
email	VARCHAR(255)	Email pengguna
password	VARCHAR(255)	Password terenkripsi
role	ENUM	Hak akses pengguna (ADMIN/USER)
status	ENUM	Status akun pengguna (ACTIVE/BLOCKED)
created_at	TIMESTAMP	Waktu pembuatan data
updated_at	TIMESTAMP	Waktu pembuatan data

Tabel 3. 1 Struktur Tabel User

2). Tabel Project

Tabel Project digunakan untuk menyimpan informasi proyek yang akan dideploy ke dalam container Docker.

Field	Tipe Data	Keterangan
id	CUID	Primary Key
user_id	CUID	Foreign Key ke User
name	VARCHAR(100)	Nama proyek
description	TEXT	Deskripsi proyek
deployment_type	VARCHAR(50)	Metode deployment (ZIP, GitHub, Dockerfile)
repository_url	TEXT	URL repository GitHub
domain	VARCHAR(255)	Domain atau subdomain aplikasi
status	VARCHAR(50)	Status proyek
created_at	TIMESTAMP	Waktu pembuatan data
updated_at	TIMESTAMP	Waktu pembaruan data

Tabel 3. 2 Struktur Tabel Project

3). Tabel Container

Tabel Container digunakan untuk menyimpan informasi container Docker yang dibuat dari suatu proyek.

Field	Tipe Data	Keterangan
id	CUID	Primary Key
project_id	CUID	Foreign Key ke User
docker_container_id	VARCHAR(225)	ID container dari Docker Engine
name	VARCHAR(100)	Nama container
image_name	VARCHAR(225)	Nama image Docker
image_tag	VARCHAR(50)	Versi image Docker

subdomain	VARCHAR(255)	Subdomain aplikasi
internal_port	INTEGER	Port internal container
host_port	INTEGER	Port pada host server
status	VARCHAR(50)	Status container
created_at	TIMESTAMP	Waktu pembuatan data
updated_at	TIMESTAMP	Waktu pembaruan data

Tabel 3. 3 Struktur Tabel Container

4). Tabel ActivityLog

Tabel ActivityLog digunakan untuk menyimpan riwayat aktivitas pengguna pada sistem.

Field	Type Data	Keterangan
id	CUID	Primary Key
user_id	CUID	Foreign Key ke User
project_id	CUID	Foreign Key ke Project
action	VARCHAR(100)	Jenis aktivitas
description	TEXT	Deskripsi aktivitas
status	VARCHAR(50)	Status aktivitas
ip_address	VARCHAR(50)	Alamat IP pengguna
created_at	TIMESTAMP	Waktu aktivitas dilakukan

Tabel 3. 4 Struktur Tabel ActivityLog

5). Tabel TerminalLog

Tabel TerminalLog digunakan untuk menyimpan riwayat penggunaan Web Terminal.

Field	Tipe Data	Keterangan
id	CUID	Primary Key
user_id	CUID	Foreign Key ke User
project_id	CUID	Foreign Key ke Project
command	TEXT	Perintah yang dijalankan
output	TEXT	Hasil eksekusi perintah
Executed_at	TIMESTAMP	Waktu eksekusi perintah

Tabel 3. 5 Struktur Tabel TerminalLog

6). Tabel ManagedDatabase

Tabel DatabaseInstance digunakan untuk menyimpan informasi layanan database yang dibuat oleh pengguna melalui fitur Portal Database. Tabel ini menyimpan konfigurasi database yang digunakan oleh aplikasi, seperti jenis database, versi, kredensial akses, serta status layanan database.

Field	Tipe Data	Keterangan
id	CUID	Primary Key
user_id	CUID	Foreign Key ke Project
name	VARCHAR	Nama layanan database
type	ENUM	Jenis database
version	VARCHAR(50)	Versi database
db_user	VARCHAR(100)	Username database
db_password	VARCHAR(255)	Password database terenkripsi
db_name	VARCHAR(255)	Nama database
docker_container_id	VARCHAR(255)	ID Docker Container database

internal_port	INTEGER	Port internal container
host_port	INTEGER	Port yang diekspos pada host
status	ENUM	Status container database
memory_limit	INTEGER	Batas penggunaan memori
cpu_limit	FLOAT	Batas penggunaan CPU
volume_name	VARCHAR	Nama Docker Volume untuk penyimpanan persisten
created_at	TIMESTAMP	Waktu pembuatan data
updated_at	TIMESTAMP	Waktu pembuatan data

Tabel 3. 6 Struktur Tabel ManagedDatabase

7). Tabel Deployment

Tabel Deployment sistem dilakukan pada lingkungan server berbasis Linux dengan memanfaatkan teknologi container menggunakan Docker. Proses deployment mencakup konfigurasi layanan frontend, backend, basis data, reverse proxy, serta penerapan sertifikat SSL untuk mendukung keamanan komunikasi melalui protokol HTTPS. Seluruh komponen sistem dikonfigurasi agar dapat saling terintegrasi sehingga aplikasi dapat berjalan secara stabil, aman, dan mudah dikelola pada lingkungan produksi. Rincian konfigurasi deployment yang digunakan pada penelitian ini ditunjukkan pada Tabel 3.7

Field	Tipe Data	Keterangan
Id	CUID	Primary Key
project_id	CUID	Foreign Key ke Project
status	VARCHAR(50)	Status proses deployment
progress	INTEGER	Persentase progres deployment
started_at	TIMESTAMP	Waktu proses deployment dimulai

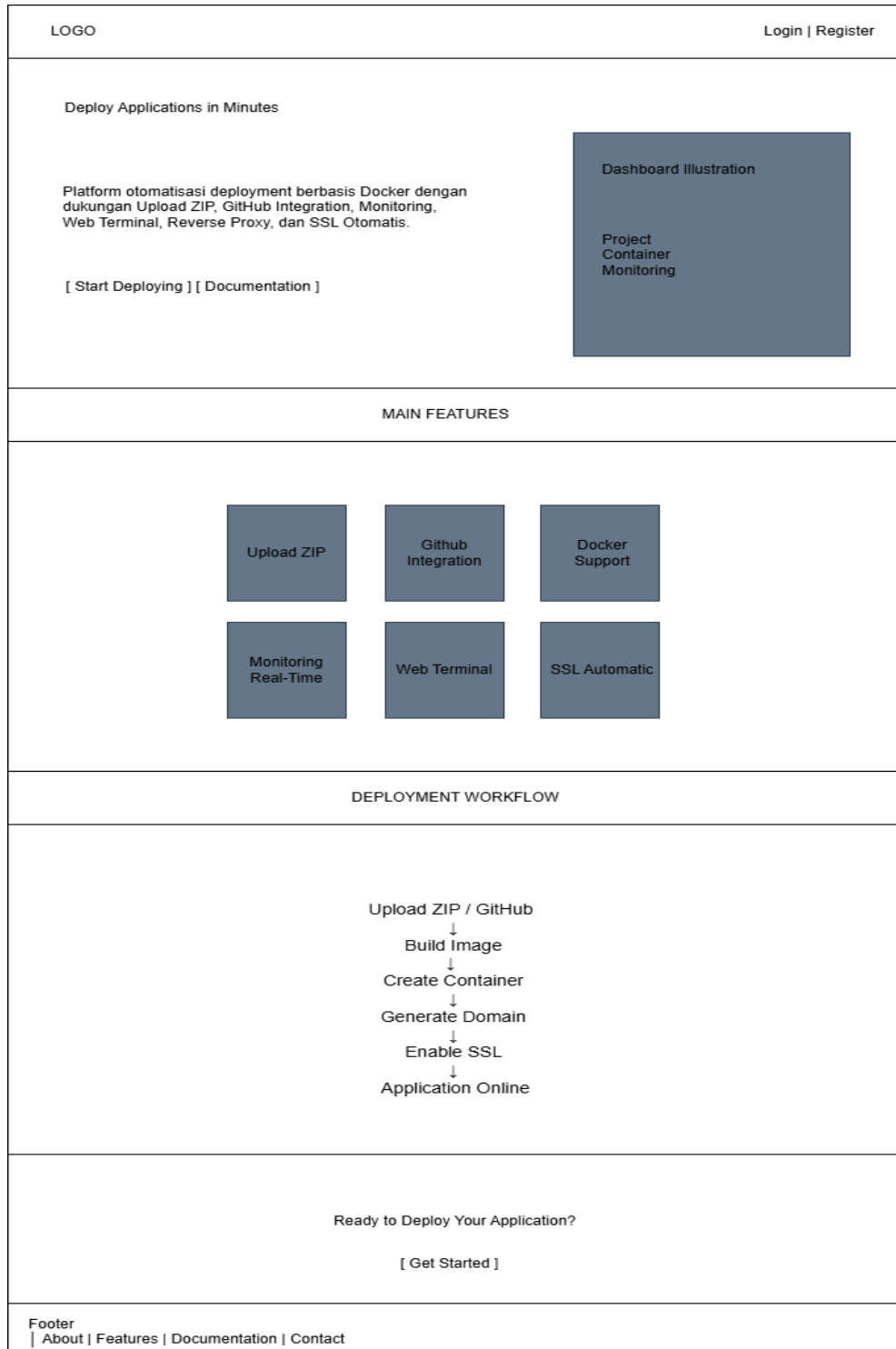
ended_at	TIMESTAMP	Waktu proses deployment selesai
domain	VARCHAR(255)	Domain aplikasi yang dideploy
commit_hash	VARCHAR(255)	Hash commit yang digunakan dalam deployment

Tabel 3. 7 Struktur Tabel Deployment

3.3.5 Perancangan User Interface

Perancangan User Interface (UI) dilakukan untuk menggambarkan tampilan antarmuka yang akan digunakan pada sistem. Antarmuka dirancang dengan mempertimbangkan kemudahan penggunaan (usability), kemudahan navigasi, serta kenyamanan pengguna dalam mengelola proses deployment aplikasi berbasis container Docker.

1. Halaman Landing Page



Gambar 3. 12 Wireframe Halaman Landing Page

Halaman Landing Page merupakan halaman utama yang ditampilkan kepada pengguna sebelum melakukan proses autentikasi. Halaman ini berfungsi sebagai media informasi mengenai sistem serta memperkenalkan fitur-fitur utama yang tersedia kepada pengguna.

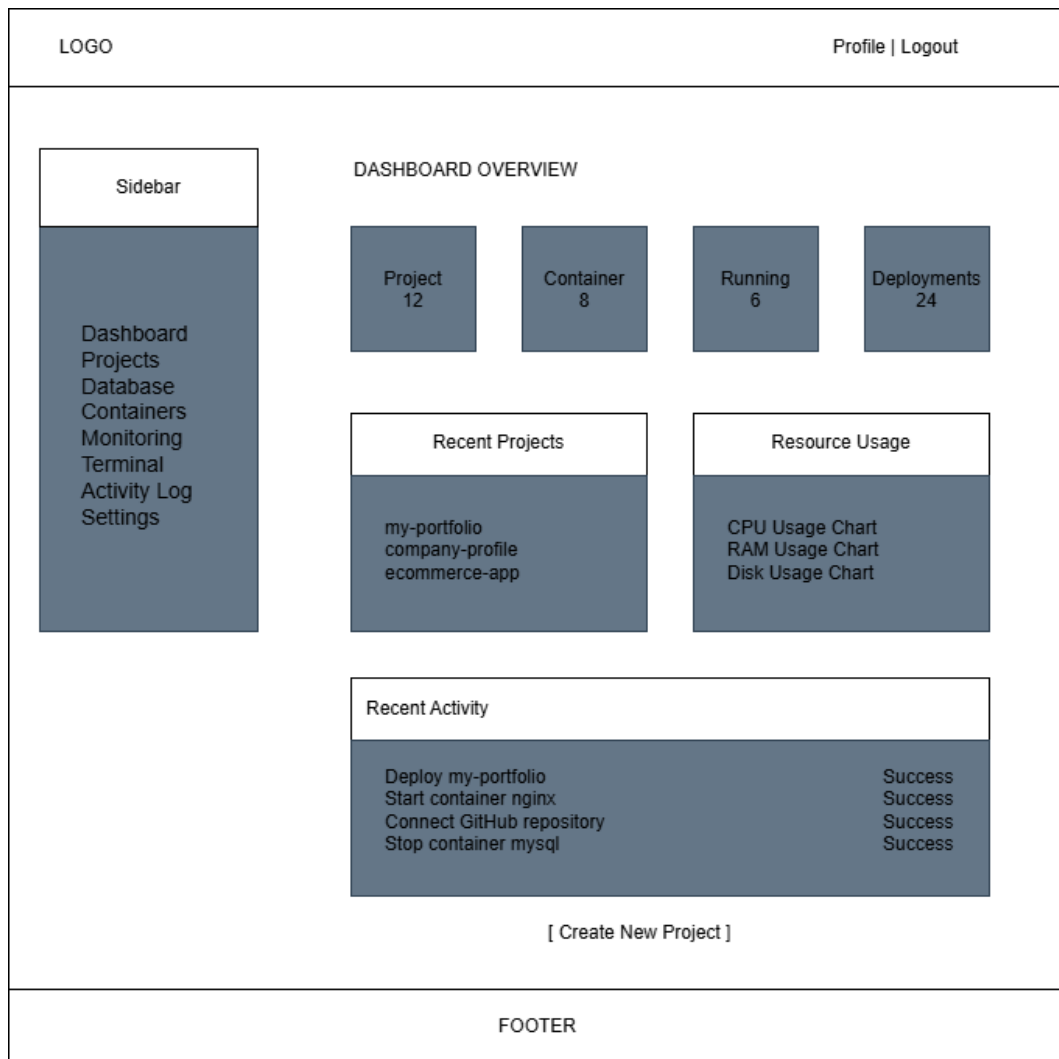
2. Halaman Login

NAVBAR
Welcome Back! Sign in to manage your deployments Email [] Password [] Forgot Password? [Login] Don't have an account? Register
FOOTER

Gambar 3. 13 Wireframe Halaman Login

Halaman Login digunakan sebagai sarana autentikasi pengguna sebelum mengakses fitur-fitur yang tersedia pada sistem. Pengguna diwajibkan memasukkan alamat email dan kata sandi yang telah terdaftar pada sistem untuk melakukan proses masuk (login).

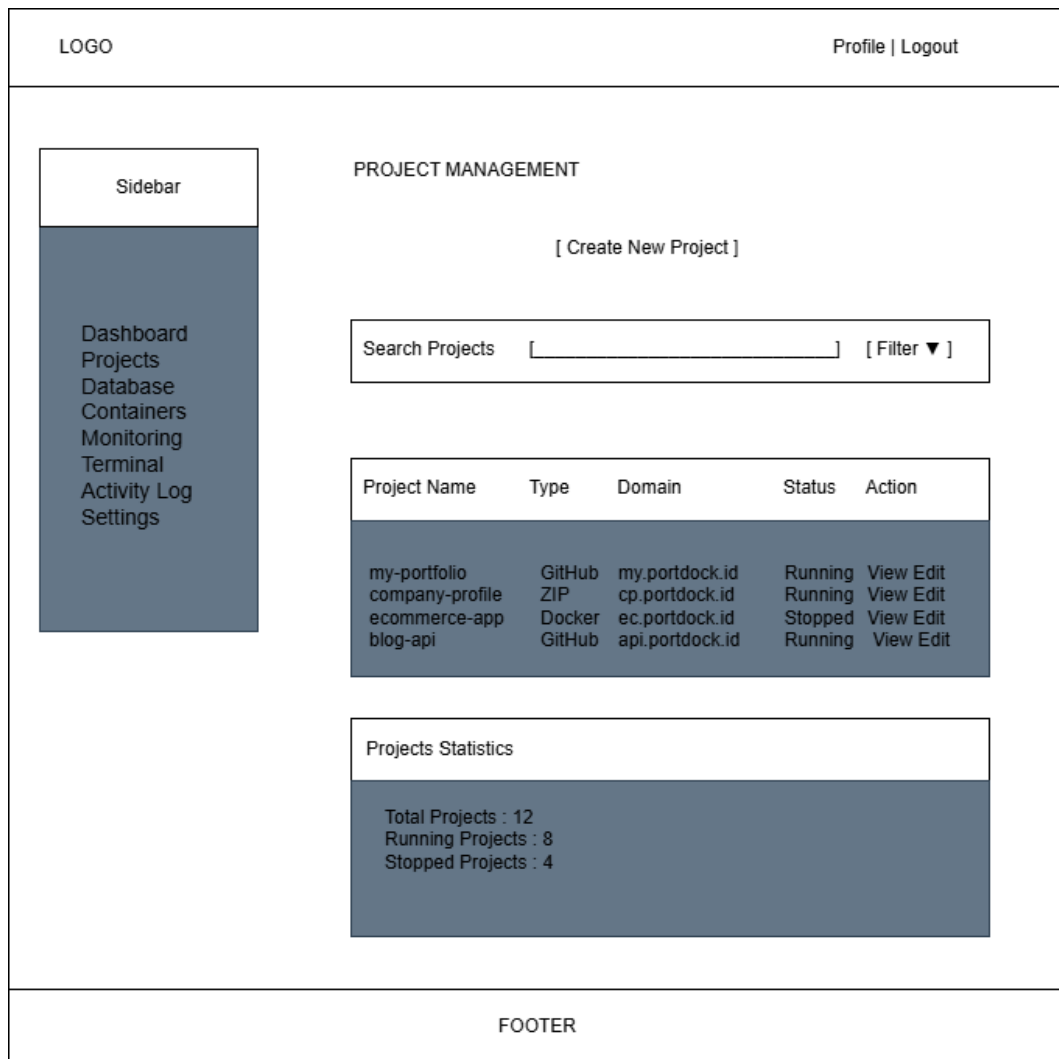
3. Halaman Dashboard



Gambar 3. 14 Wireframe Halaman Dashboard

Halaman Dashboard merupakan halaman utama yang ditampilkan setelah pengguna berhasil melakukan proses autentikasi pada sistem. Halaman ini berfungsi sebagai pusat pengelolaan dan monitoring seluruh aktivitas deployment yang dilakukan oleh pengguna.

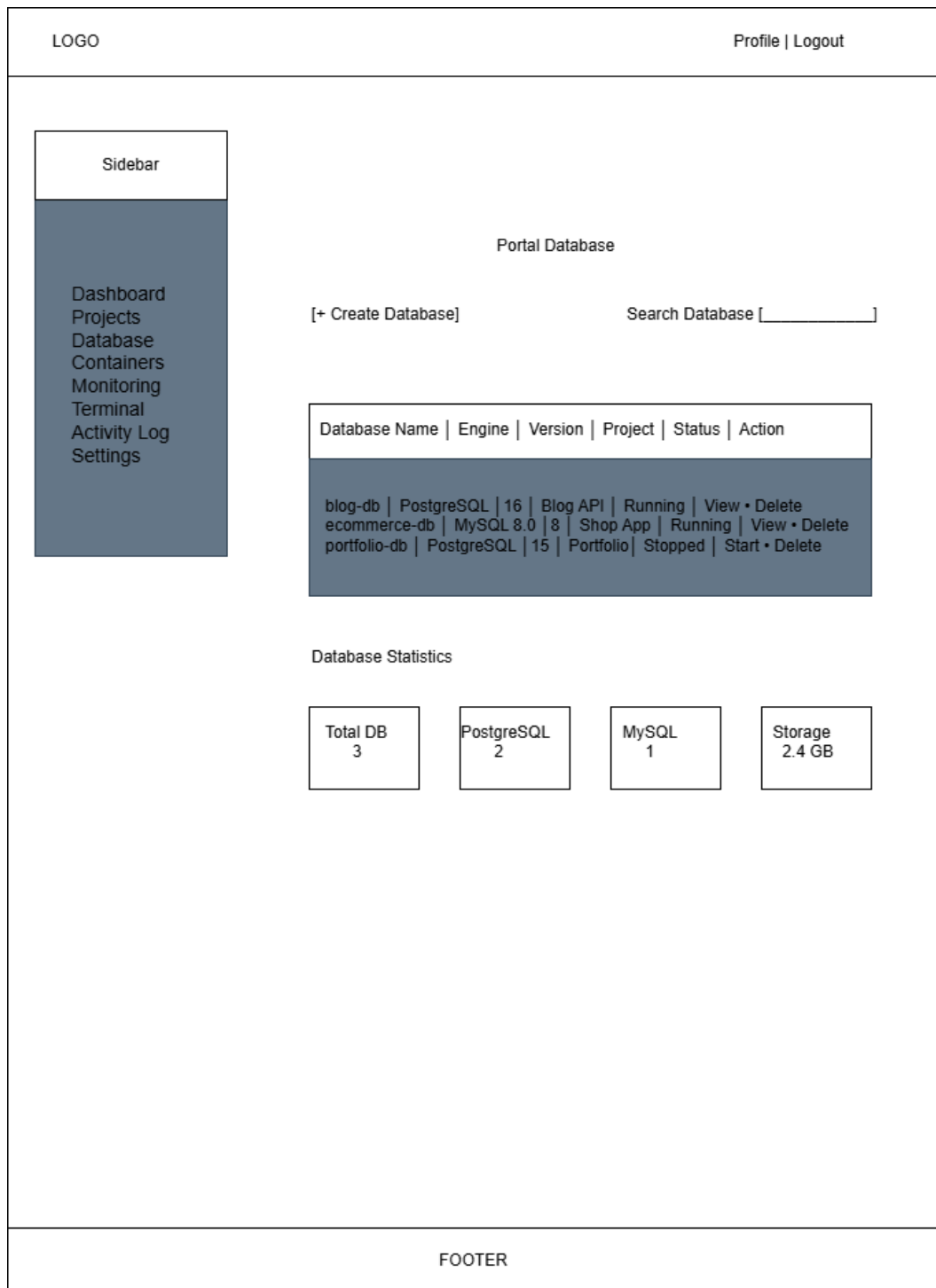
4. Halaman Project



Gambar 3. 15 Wireframe Halaman Project

Halaman Project digunakan untuk mengelola seluruh project yang telah dibuat oleh pengguna pada sistem. Halaman ini berfungsi sebagai pusat manajemen project yang memungkinkan pengguna untuk melihat daftar project, memantau status deployment, serta mengakses informasi terkait domain dan metode deployment yang digunakan.

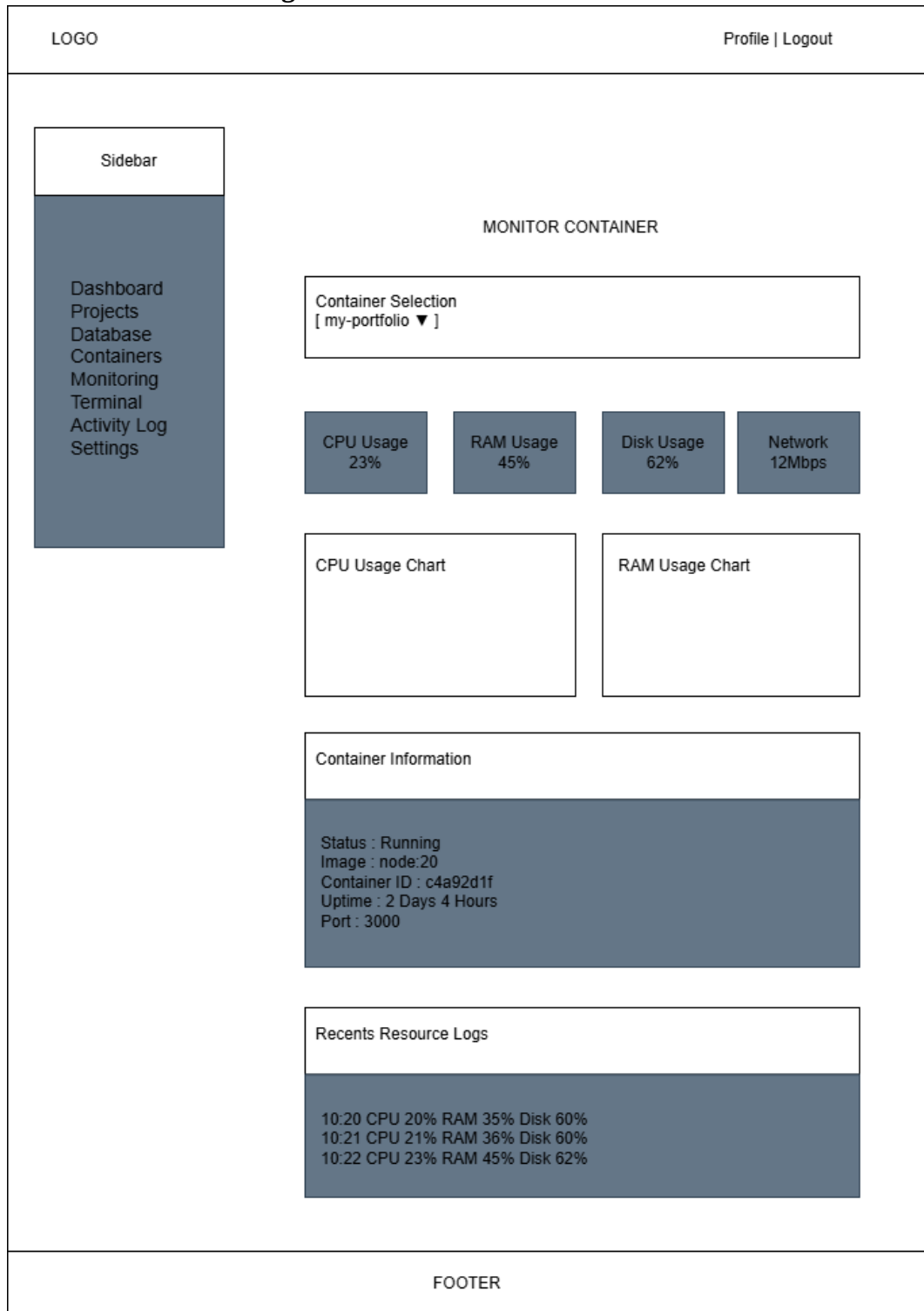
5. Halaman Database



Gambar 3. 16 Wireframe Halaman Portal Database

Halaman Database merupakan antarmuka yang digunakan oleh pengguna untuk mengelola layanan basis data yang terintegrasi dengan aplikasi yang telah dideploy pada sistem. Halaman ini dirancang untuk mempermudah pengguna dalam melakukan pengelolaan database tanpa harus melakukan konfigurasi secara langsung melalui command line atau server.

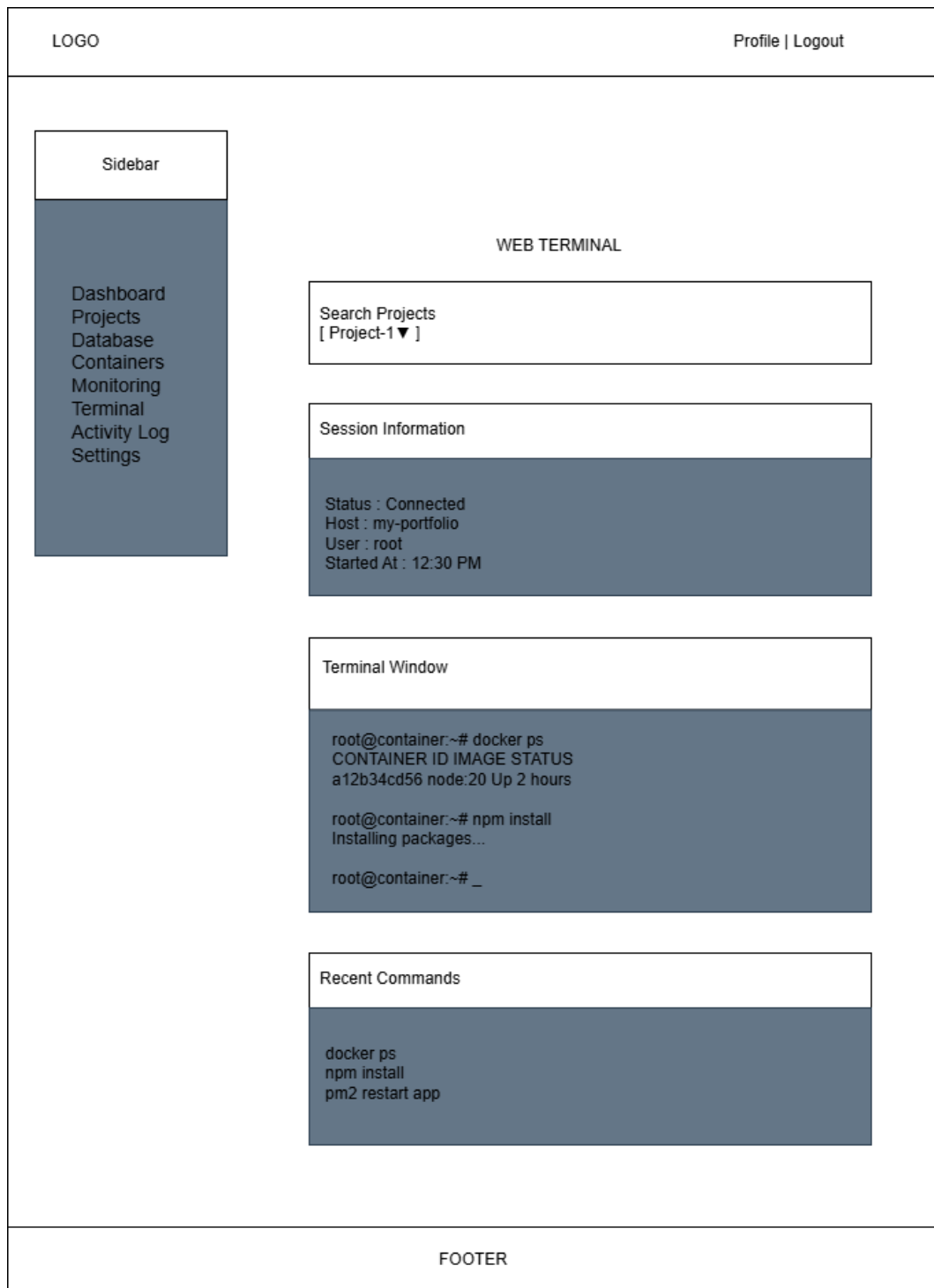
6. Halaman Monitoring Container



Gambar 3. 17 Wireframe Halaman Monitoring Container

Halaman Monitoring Container digunakan untuk memantau kondisi dan penggunaan sumber daya container Docker yang sedang berjalan pada sistem. Halaman ini dirancang untuk memberikan informasi monitoring secara real-time sehingga pengguna dapat mengetahui status dan performa aplikasi yang telah dideploy.

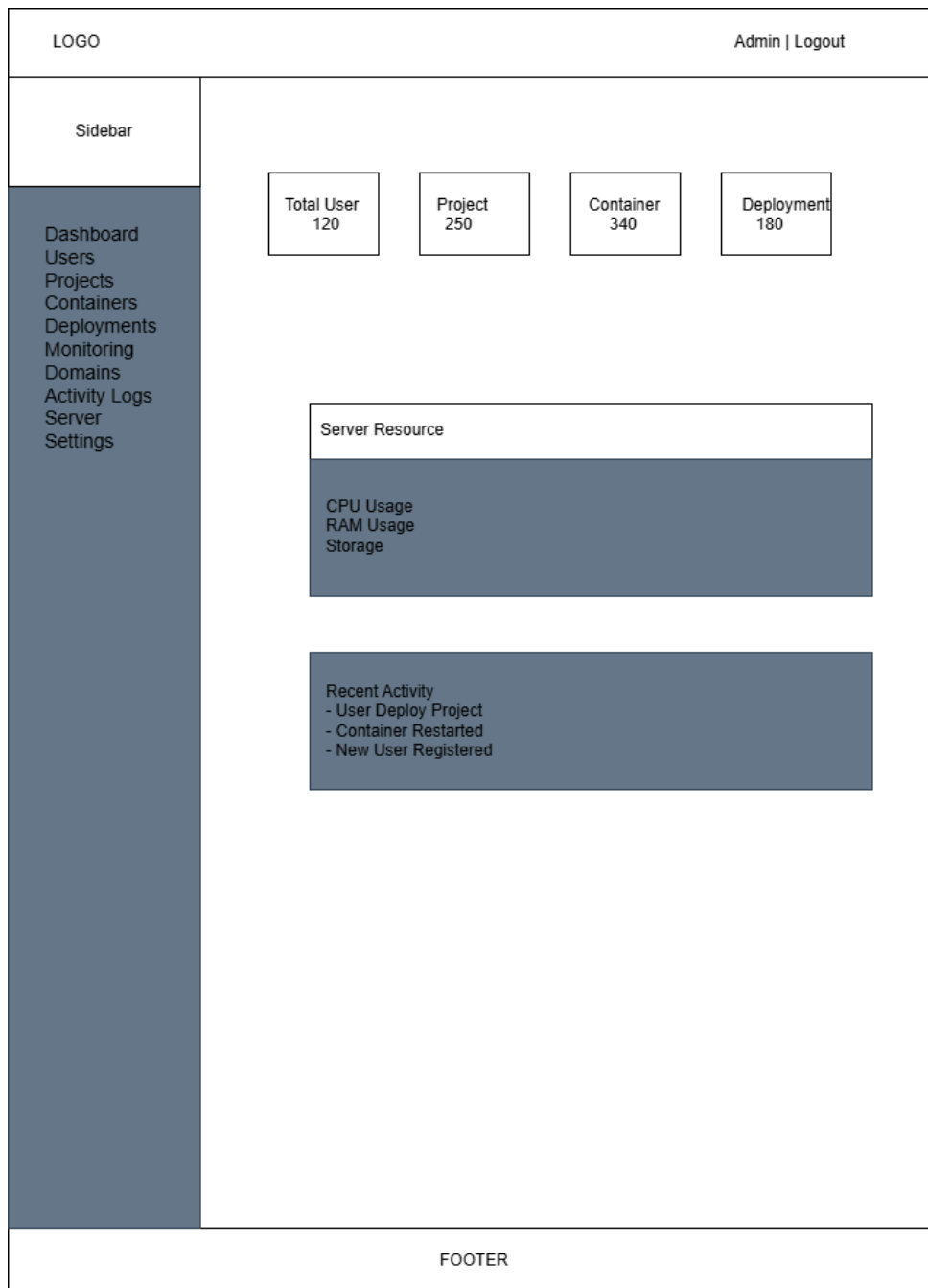
7. Halaman Web Terminal



Gambar 3. 18 Wireframe Halaman Web Terminal

Halaman Web Terminal digunakan untuk memberikan akses terminal berbasis web kepada pengguna sehingga dapat berinteraksi langsung dengan container atau server tanpa perlu menggunakan aplikasi terminal eksternal. Fitur ini memungkinkan pengguna menjalankan berbagai perintah administrasi, konfigurasi, maupun pemeliharaan aplikasi secara langsung melalui browser.

8. Dashboard Admin



Gambar 3. 19 Wireframes Halaman Dashboard Admin

Halaman Dashboard Admin merupakan halaman utama yang ditampilkan setelah administrator berhasil mengakses Admin Panel. Halaman ini berfungsi sebagai pusat informasi untuk memantau kondisi keseluruhan sistem secara real-time. Dashboard menampilkan berbagai informasi penting, seperti jumlah pengguna yang terdaftar, jumlah project yang aktif, jumlah container yang sedang berjalan, serta jumlah deployment yang telah dilakukan pada sistem.

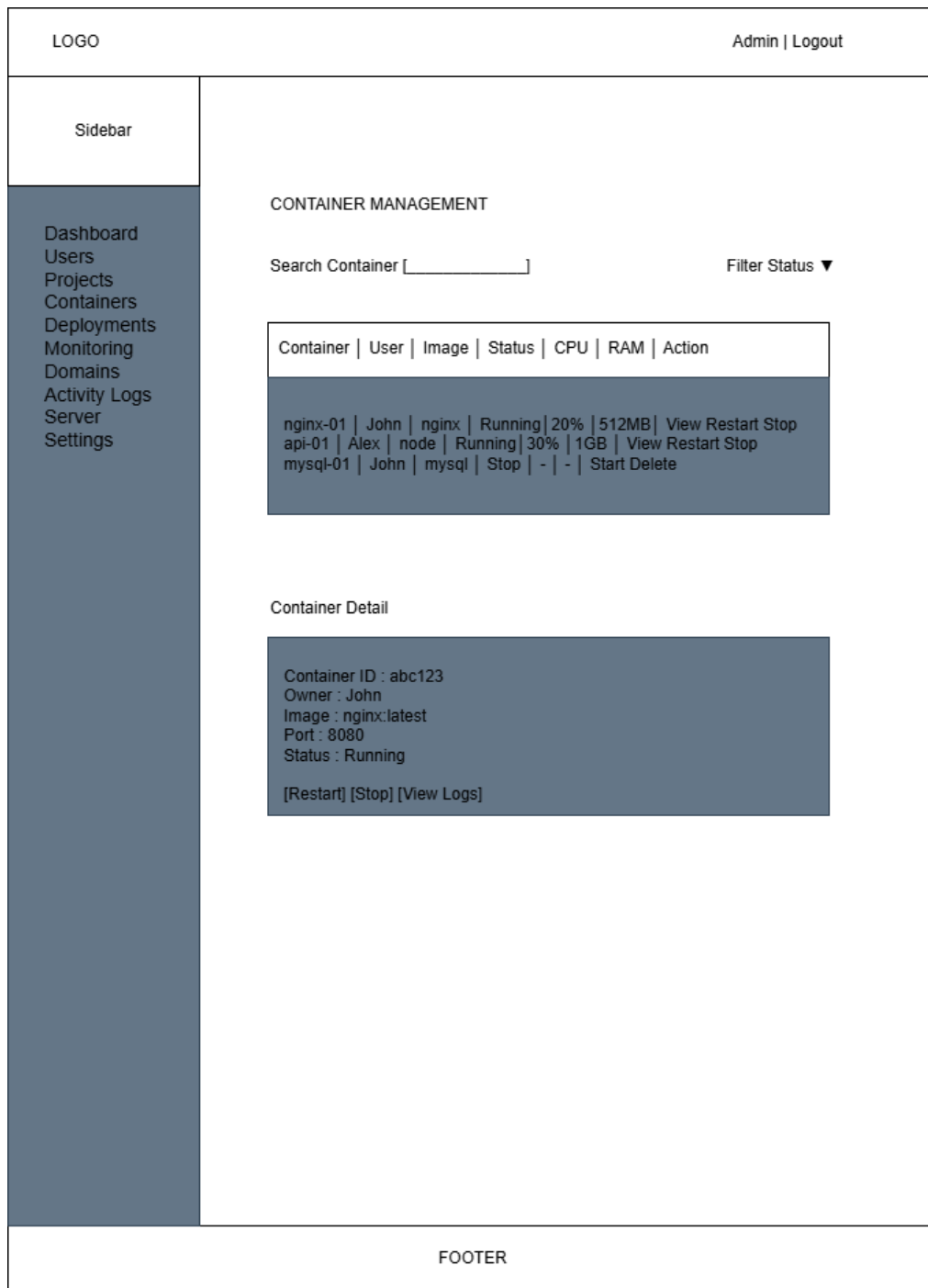
9. User Management

LOGO		Admin Logout																					
Sidebar																							
Dashboard Users Projects Containers Deployments Monitoring Domains Activity Logs Server Settings		USER MANAGEMENT																					
		Search User [] [+ Add User]																					
		<table border="1"><thead><tr><th>Name</th><th>Email</th><th>Role</th><th>Status</th><th>Action</th></tr></thead><tbody><tr><td>John Doe</td><td>john@mail.com</td><td>User</td><td>Active</td><td>View Edit Delete</td></tr><tr><td>Admin</td><td>admin@mail.com</td><td>Admin</td><td>Active</td><td>View Edit Delete</td></tr><tr><td>Alex</td><td>alex@mail.com</td><td>User</td><td>Block</td><td>View Edit Delete</td></tr></tbody></table>		Name	Email	Role	Status	Action	John Doe	john@mail.com	User	Active	View Edit Delete	Admin	admin@mail.com	Admin	Active	View Edit Delete	Alex	alex@mail.com	User	Block	View Edit Delete
		Name	Email	Role	Status	Action																	
John Doe	john@mail.com	User	Active	View Edit Delete																			
Admin	admin@mail.com	Admin	Active	View Edit Delete																			
Alex	alex@mail.com	User	Block	View Edit Delete																			
User Detail																							
		Username : John Doe Email : john@mail.com Role : User Project : 5 Container: 8 [Edit User] [Reset Password] [Delete User]																					
FOOTER																							

Gambar 3. 20 Wireframes Halaman User Management

Halaman User Management digunakan oleh administrator untuk mengelola seluruh akun pengguna yang terdaftar pada sistem. Melalui halaman ini administrator dapat melihat daftar pengguna beserta informasi seperti nama pengguna, alamat email, peran (role), dan status akun.

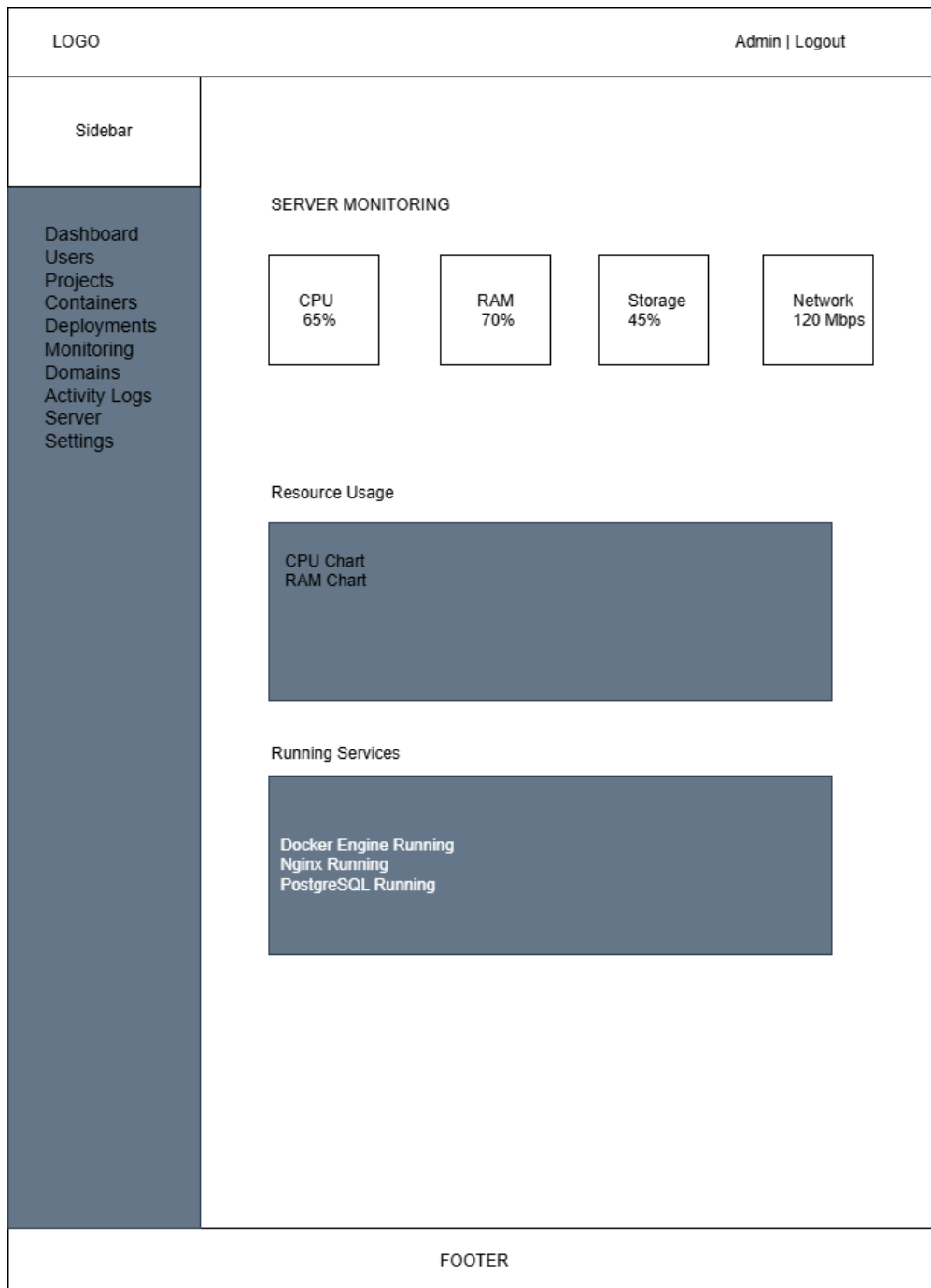
10. Container Management



Gambar 3. 21 Wireframes Halaman Container Management

Halaman Container Management digunakan oleh administrator untuk memantau dan mengelola seluruh Docker Container yang dijalankan pada server. Berbeda dengan pengguna biasa yang hanya dapat mengelola container miliknya sendiri, administrator memiliki hak akses untuk melihat seluruh container yang dimiliki oleh semua pengguna.

11. Monitoring Server



Gambar 3. 22 Wireframes Halaman Monitoring Server

Halaman Monitoring Server berfungsi sebagai media pemantauan kondisi server yang digunakan untuk menjalankan layanan. Halaman ini menampilkan informasi penggunaan sumber daya server secara real-time, seperti penggunaan prosesor (CPU), memori (RAM), penyimpanan (storage), dan aktivitas jaringan (network).

12. Server Settings

LOGO		Admin Logout
Sidebar		
Dashboard Users Projects Containers Deployments Monitoring Domains Activity Logs Server Settings	SERVER SETTINGS Docker Configuration Docker Version : 27.0 Docker Status : Running [Restart Docker] Server Configuration Server Host : ubuntu@portdock.id SSH Port : 22 Timezone : Asia/Jakarta [Save Configuration] Security Firewall Status : Active SSL Certificate: Valid [Update Security Setting]	
FOOTER		

Gambar 3. 23 Wireframes Halaman Server Settings

Halaman Server Settings merupakan halaman konfigurasi yang digunakan administrator untuk mengelola pengaturan server yang menjadi infrastruktur utama sistem. Halaman ini menyediakan berbagai konfigurasi penting yang berkaitan dengan layanan Docker, pengaturan server, serta aspek keamanan sistem.

3.3.6 Perancangan Infrastruktur Docker

Perancangan Infrastruktur Docker merupakan tahap perancangan yang bertujuan untuk mendefinisikan arsitektur sistem yang digunakan dalam proses deployment aplikasi secara otomatis pada platform. Infrastruktur ini dirancang untuk mengintegrasikan berbagai komponen utama, seperti Frontend, Backend, Database, Docker Engine, Reverse Proxy, dan layanan SSL, sehingga mampu mendukung proses deployment aplikasi secara cepat, aman, dan terotomatisasi. Melalui rancangan infrastruktur ini, sistem dapat melakukan proses build image, pembuatan container, konfigurasi domain, hingga aktivasi SSL secara otomatis tanpa memerlukan konfigurasi manual dari pengguna.

1. Deskripsi Infrastruktur

Infrastruktur pada sistem dirancang untuk mendukung proses deployment aplikasi berbasis Docker secara otomatis, mulai dari penerimaan source code hingga aplikasi dapat diakses melalui internet menggunakan domain dan protokol HTTPS. Infrastruktur ini terdiri dari beberapa komponen utama yang saling terintegrasi, yaitu Frontend, Backend, Database, Docker Engine, Reverse Proxy, dan layanan SSL.

Frontend berfungsi sebagai antarmuka yang digunakan pengguna untuk melakukan berbagai aktivitas seperti membuat proyek, melakukan deployment aplikasi, mengelola container, memantau penggunaan sumber daya, serta mengakses fitur Web Terminal. Frontend berkomunikasi dengan Backend melalui API untuk mengirim dan menerima data yang dibutuhkan sistem.

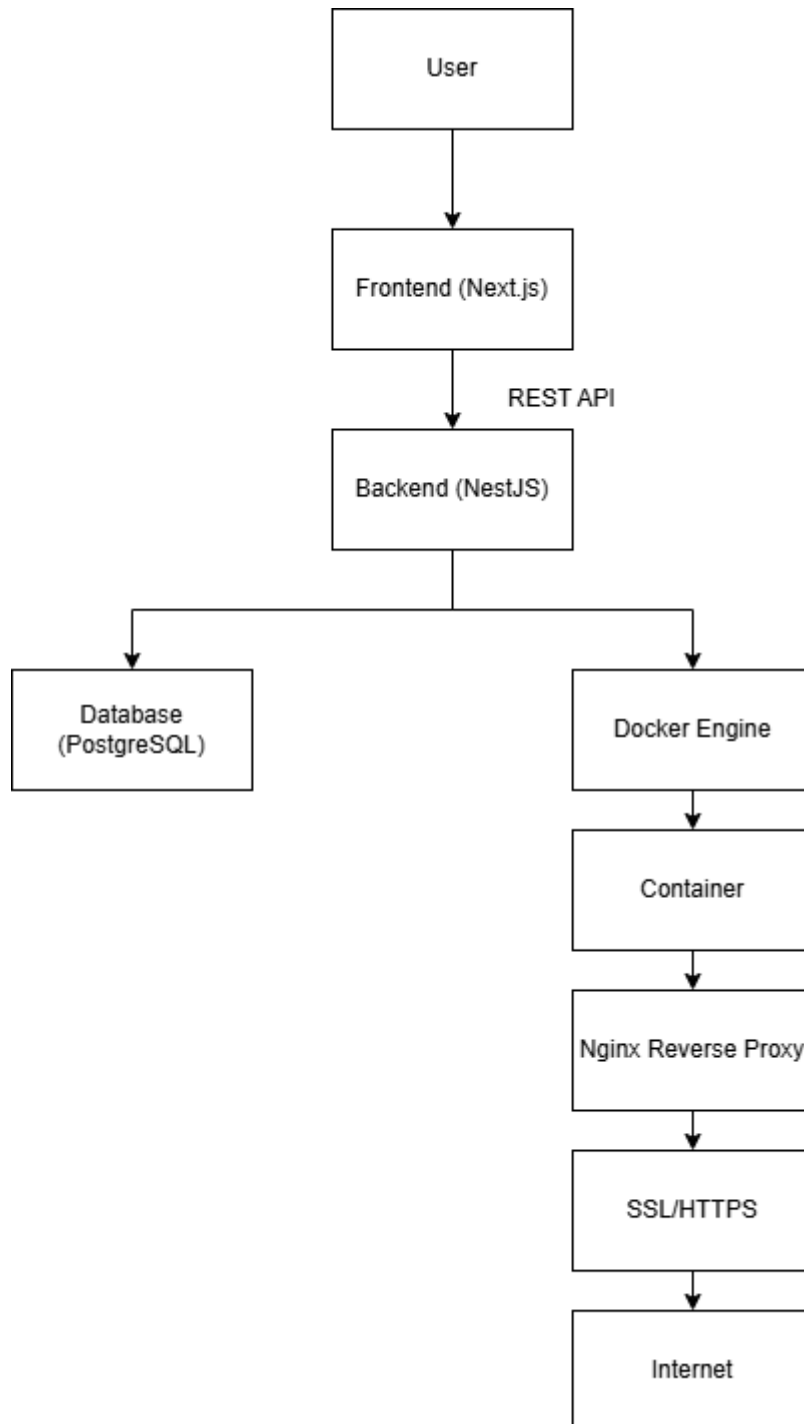
Backend bertugas sebagai pusat logika aplikasi yang menangani autentikasi pengguna, manajemen proyek, proses deployment, pengelolaan container Docker,

pencatatan aktivitas sistem, serta komunikasi dengan Docker Engine. Setiap permintaan yang dilakukan pengguna melalui antarmuka akan diproses oleh Backend sebelum diteruskan ke komponen terkait.

Database digunakan untuk menyimpan seluruh data yang diperlukan sistem, seperti data pengguna, proyek, container, riwayat aktivitas (Activity Log), dan riwayat penggunaan terminal (Terminal Log). Penyimpanan data dilakukan secara terstruktur sehingga memudahkan proses pengelolaan dan pelacakan aktivitas pengguna.

Docker Engine berperan sebagai layanan utama dalam proses deployment aplikasi. Docker Engine digunakan untuk melakukan build image, membuat container baru, menjalankan container, menghentikan container, serta menghapus container sesuai kebutuhan pengguna. Seluruh operasi Docker dilakukan secara otomatis melalui Backend menggunakan Docker API.

2. Diagram Infrastruktur Docker

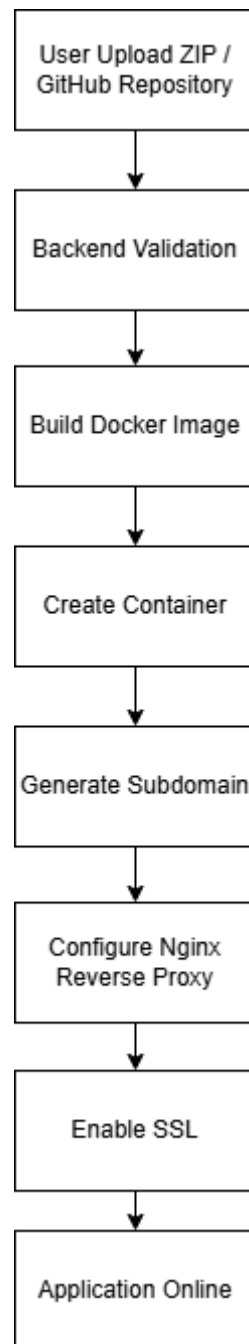


Gambar 3. 24 Diagram Infrastruktur Docker

Gambar 3.24 menunjukkan rancangan infrastruktur Docker yang digunakan pada sistem. Infrastruktur ini terdiri dari beberapa komponen utama, yaitu User, Frontend, Backend, Database, Docker Engine, Container, Nginx Reverse Proxy, dan SSL/HTTPS yang saling terhubung untuk mendukung proses deployment aplikasi secara otomatis.

3. Alur Deployment

Alur deployment pada sistem dirancang untuk mengotomatisasi proses publikasi aplikasi berbasis Docker. Pengguna dapat melakukan deployment melalui beberapa metode yang disediakan, yaitu upload file ZIP, integrasi repository GitHub, maupun penggunaan Dockerfile kustom. Setelah source code diterima oleh sistem, proses deployment akan dilakukan secara otomatis hingga aplikasi dapat diakses melalui internet menggunakan domain dan protokol HTTPS.



Gambar 3. 25 Alur Deployment

Gambar 3.25 menunjukkan alur deployment aplikasi pada sistem. Proses deployment dimulai ketika pengguna mengunggah source code aplikasi dalam bentuk file ZIP atau menghubungkan repository GitHub yang akan digunakan sebagai sumber aplikasi. Setelah source code diterima oleh sistem, Backend akan melakukan validasi terhadap data yang diberikan untuk memastikan bahwa aplikasi dapat diproses lebih lanjut.

4. Komponen Infrastruktur

Infrastruktur sistem terdiri dari beberapa komponen yang saling terintegrasi untuk mendukung proses deployment aplikasi secara otomatis. Setiap komponen memiliki peran yang berbeda namun saling mendukung dalam menjalankan fungsi sistem secara keseluruhan. Adapun komponen yang digunakan pada sistem meliputi User, Frontend, Backend, Database, Docker Engine, Container Docker, Nginx Reverse Proxy, dan SSL/HTTPS.

1) User

User merupakan aktor utama yang menggunakan sistem untuk melakukan berbagai aktivitas deployment dan pengelolaan aplikasi. Melalui antarmuka yang disediakan, pengguna dapat melakukan registrasi dan login ke dalam sistem, membuat proyek baru, melakukan deployment aplikasi, memantau kondisi container, mengakses Web Terminal, serta mengelola pengaturan akun. Seluruh aktivitas yang dilakukan pengguna akan diproses oleh sistem dan dicatat untuk mendukung kebutuhan monitoring dan audit aktivitas.

2) Frontend (Next.js)

Frontend merupakan komponen yang berfungsi sebagai antarmuka pengguna pada sistem. Komponen ini dikembangkan menggunakan framework Next.js yang mampu menyediakan tampilan web yang responsif, modern, dan mudah digunakan. Frontend bertugas menampilkan seluruh halaman yang digunakan oleh pengguna, seperti Landing Page, Login, Dashboard, Project Management, Deployment, Monitoring Container, dan

Web Terminal. Selain itu, Frontend juga bertanggung jawab mengirimkan permintaan data ke Backend melalui API dan menampilkan hasil pemrosesan data kepada pengguna secara interaktif.

3) Backend (NestJS)

Backend merupakan komponen yang menjalankan logika bisnis utama pada sistem. Backend dibangun menggunakan framework NestJS yang berfungsi sebagai penghubung antara Frontend, Database, dan Docker Engine. Komponen ini bertanggung jawab untuk menangani proses autentikasi pengguna, pengelolaan proyek, deployment aplikasi, monitoring container, pencatatan aktivitas pengguna, serta pengelolaan layanan Web Terminal. Selain itu, Backend juga berperan dalam mengatur konfigurasi domain, reverse proxy, dan aktivasi SSL yang diperlukan agar aplikasi dapat diakses secara aman melalui internet.

4) Database (PostgreSQL)

Database digunakan sebagai media penyimpanan utama untuk seluruh data yang dibutuhkan oleh sistem. Pada penelitian ini, PostgreSQL digunakan sebagai sistem manajemen basis data karena memiliki performa yang baik dan mendukung pengelolaan data relasional secara efisien. Database digunakan untuk menyimpan informasi pengguna, proyek, container, aktivitas sistem, serta riwayat penggunaan terminal. Data yang tersimpan dalam database akan digunakan oleh Backend untuk menjalankan berbagai fungsi sistem dan memastikan konsistensi data selama proses operasional berlangsung.

5) Docker Engine

Docker Engine merupakan layanan yang bertanggung jawab dalam menjalankan teknologi container pada sistem. Komponen ini digunakan untuk membangun Docker Image, membuat container baru, menjalankan container, menghentikan container, serta mengelola seluruh siklus hidup

container yang digunakan oleh aplikasi pengguna. Backend akan berkomunikasi secara langsung dengan Docker Engine melalui Docker API untuk mengotomatisasi seluruh proses deployment aplikasi sehingga pengguna tidak perlu melakukan konfigurasi Docker secara manual.

6) Container Docker

Container Docker merupakan lingkungan eksekusi aplikasi yang dihasilkan dari proses deployment. Setiap aplikasi yang dideploy oleh pengguna akan dijalankan di dalam container yang terisolasi sehingga tidak saling mempengaruhi satu sama lain. Container menyimpan seluruh kebutuhan aplikasi yang meliputi source code, runtime, library, dan dependensi yang diperlukan untuk menjalankan aplikasi. Dengan menggunakan container, aplikasi dapat berjalan secara konsisten pada berbagai lingkungan server dan mempermudah proses pengelolaan aplikasi.

7) Nginx Reverse Proxy

Nginx Reverse Proxy digunakan untuk menghubungkan domain atau subdomain aplikasi dengan container yang sesuai. Komponen ini berfungsi sebagai perantara yang menerima permintaan dari pengguna melalui internet dan meneruskannya ke container tujuan berdasarkan konfigurasi yang telah ditentukan. Dengan adanya Nginx Reverse Proxy, pengguna dapat mengakses aplikasi menggunakan domain atau subdomain yang mudah diingat tanpa perlu mengetahui port internal yang digunakan oleh container. Selain itu, Nginx juga berperan dalam pengelolaan virtual host dan mendukung implementasi sertifikat SSL pada sistem.

8) SSL/HTTPS

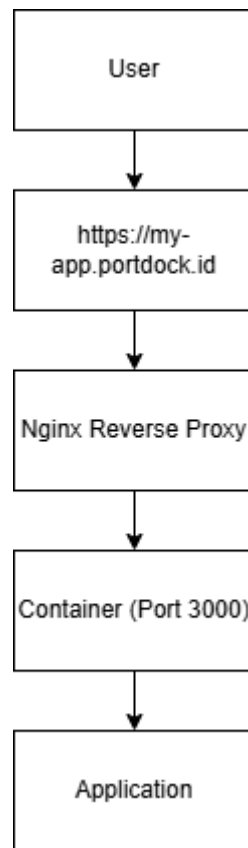
SSL (Secure Socket Layer) merupakan komponen yang digunakan untuk menyediakan komunikasi yang aman antara pengguna dan aplikasi yang telah dideploy. Sertifikat SSL memungkinkan seluruh data yang dikirimkan antara client dan server dienkripsi sehingga dapat mengurangi risiko

penyadapan dan manipulasi data oleh pihak yang tidak berwenang. Pada sistem, aktivasi SSL dilakukan secara otomatis setelah proses deployment dan konfigurasi domain berhasil diselesaikan. Dengan demikian, aplikasi yang dihasilkan dapat langsung diakses menggunakan protokol HTTPS yang lebih aman dan terpercaya.

5. Mekanisme Reverse Proxy dan SSL

Mekanisme Reverse Proxy dan SSL pada sistem dirancang untuk memungkinkan aplikasi yang berjalan di dalam container Docker dapat diakses melalui domain atau subdomain secara aman menggunakan protokol HTTPS. Mekanisme ini memanfaatkan layanan Nginx sebagai Reverse Proxy yang berfungsi menghubungkan permintaan pengguna dari internet ke container tujuan berdasarkan konfigurasi domain yang telah ditentukan.

Ketika proses deployment berhasil dilakukan, sistem akan membuat container baru dan menentukan port internal yang digunakan oleh aplikasi. Selanjutnya, Backend akan secara otomatis menghasilkan konfigurasi Nginx yang menghubungkan subdomain pengguna dengan container yang sedang berjalan. Dengan mekanisme tersebut, pengguna dapat mengakses aplikasi menggunakan domain yang lebih mudah diingat tanpa perlu mengetahui alamat IP server maupun nomor port container.



Gambar 3. 26 Mekanisme Reverse Proxy dan SSL

Berdasarkan Gambar 3.26, pengguna mengakses aplikasi melalui alamat domain atau subdomain yang telah diberikan oleh sistem. Permintaan tersebut pertama kali diterima oleh Nginx Reverse Proxy yang berjalan pada server. Nginx kemudian meneruskan permintaan ke container tujuan berdasarkan konfigurasi domain yang telah dibuat selama proses deployment.

3.4 Pengolahan dan Analisis Data

Data yang telah dikumpulkan melalui observasi teknis dan eksperimen laboratorium diolah dan dianalisis secara kualitatif untuk memastikan bahwa sistem memenuhi spesifikasi yang telah dirancang. Proses ini bertujuan untuk mengubah data teknis menjadi informasi yang valid mengenai kelayakan sistem. Tahapan pengolahan dan analisis data dalam penelitian ini adalah sebagai berikut:

3.4.1 Pengolahan Data Teknis

Data mentah yang diperoleh dari hasil pengkodean dan integrasi modul diproses dengan langkah-langkah:

1. Reduksi Data: Peneliti memilah hasil *log* dari *backend* NestJS dan respon dari *Docker Engine* untuk mengambil data yang paling relevan dengan fungsionalitas sistem.
2. Penyajian Data: Data teknis disajikan dalam bentuk diagram alur (*flowchart*), skema *database* Prisma, serta tangkapan layar (*screenshot*) antarmuka Next.js yang menunjukkan status kontainer secara *real-time*.
3. Verifikasi: Melakukan pengecekan konsistensi antara data yang dikirim melalui *WebSocket* dengan tampilan pada *terminal* Xterm.js untuk memastikan tidak ada latensi yang mengganggu operasional.

3.4.2 Analisis Fungsionalitas (*Black-Box Analysis*)

Analisis dilakukan terhadap setiap fungsi utama sistem untuk memastikan semua fitur berjalan sesuai rencana. Fokus analisis meliputi:

1. Manajemen Kontainer: Menganalisis keberhasilan proses pembuatan, penghentian, dan penghapusan kontainer melalui instruksi dari antarmuka *web*.
2. Otomatisasi *Reverse Proxy*: Menganalisis efektivitas pemetaan domain ke kontainer *Docker* secara otomatis.
3. Akses *Terminal*: Menganalisis stabilitas koneksi SSH2 yang dijumpai ke *browser* pengguna.