BAB III

PELAKSANAAN KERJA

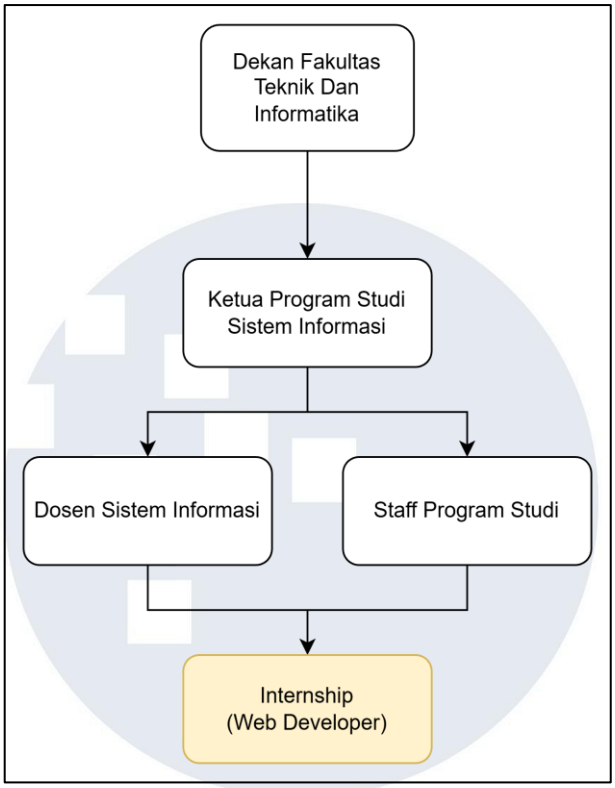
3.1 Kedudukan dan Koordinasi

Bab ini menguraikan pelaksanaan kerja magang di Fakultas Teknik dan Informatika, Universitas Multimedia Nusantara, dengan fokus pada kedudukan dalam struktur organisasi dan mekanisme koordinasi. Uraian mencakup posisi fungsional yang ditempati serta alur komunikasi yang diterapkan bersama pembimbing lapangan guna memastikan kelancaran pengembangan aplikasi Optimalisasi Manajemen Peminjaman Ruangan Penelitian.

3.1.1 Kedudukan

Dalam struktur operasional magang di Universitas Multimedia Nusantara, posisi yang ditempati adalah sebagai *Web Developer Intern*. Posisi ini berada secara administratif di bawah naungan Program Studi Sistem Informasi, Fakultas Teknik dan Informatika. Fokus utama dari peran ini meliputi pengembangan teknis, perancangan antarmuka, serta implementasi logika *backend* untuk sistem aplikasi manajemen peminjaman ruangan, yang pelaksanaannya diawasi oleh dosen pembimbing serta berkoordinasi dengan staf program studi.

Berdasarkan Gambar 3.1, divisualisasikan kedudukan posisi magang dalam hierarki organisasi di Fakultas Teknik dan Informatika. Pada tingkat tertinggi, tanggung jawab dipegang oleh Dekan yang membawahi Ketua Program Studi Sistem Informasi. Posisi *Internship (Web Developer)* ditempatkan pada level pelaksana teknis (operasional) yang menerima arahan langsung dari dua entitas utama, yaitu Dosen Sistem Informasi sebagai supervisor teknis dan Staff Program Studi sebagai verifikator kebutuhan sistem. Dalam skema tersebut, garis komando berjalan secara vertikal, di mana hasil kerja dilaporkan dan divalidasi oleh dosen serta staf sebelum dilaporkan lebih lanjut ke tingkat struktural yang lebih tinggi.



Gambar 3.1 Struktur Kedudukan Magang

Melalui penempatan strategis dalam struktur organisasi tersebut, sinergi antara aspek akademis dan kebutuhan operasional dapat terjalin secara efektif. Kedudukan ini memungkinkan proses pengembangan aplikasi tidak hanya berfokus pada kualitas kode dan arsitektur teknis semata, melainkan juga memastikan relevansi fungsionalitas sistem terhadap alur kerja administrasi yang sesungguhnya. Oleh karena itu, peran ini menjadi elemen vital dalam menjembatani solusi teknologi dengan permasalahan nyata di lingkungan Fakultas Teknik dan Informatika.

3.1.2 Koordinasi

Mekanisme koordinasi pelaksanaan kerja magang dilakukan menggunakan pendekatan terpusat melalui Supervisor (Dosen Pembimbing Lapangan). Dalam skema ini, arus informasi dan pendelegasian tugas bersifat satu pintu (*single point of contact*) guna menjaga konsistensi instruksi teknis

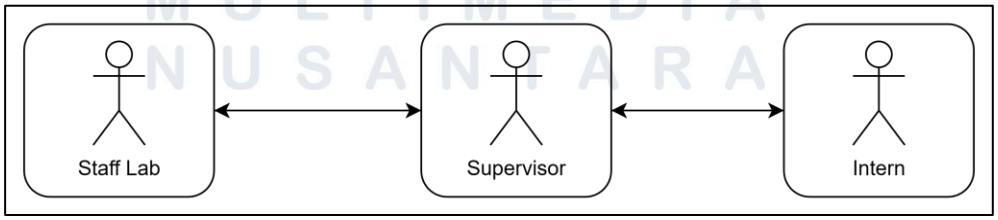
pengembangan aplikasi. Secara rinci, alur koordinasi yang berjalan dapat diuraikan sebagai berikut:

A. Perumusan Kebutuhan dan Diskusi Internal Tahap awal melibatkan kolaborasi dan diskusi antara Supervisor dengan Staf Laboratorium/Program Studi. Pada tahap ini, identifikasi masalah operasional dan kebutuhan fitur dilakukan secara mendalam oleh Supervisor bersama pihak staf untuk menentukan spesifikasi sistem yang tepat sasaran.

B. Pendelegasian Tugas (*Task Assignment*) Hasil diskusi dan spesifikasi fitur yang telah disepakati kemudian diterjemahkan oleh Supervisor menjadi instruksi kerja teknis. Instruksi tersebut selanjutnya diturunkan kepada pihak pengembang (*intern*) untuk dieksekusi. Hal ini memastikan bahwa setiap tugas yang dikerjakan telah melalui proses validasi awal dan memiliki prioritas yang jelas.

C. Pelaksanaan dan Pelaporan Hasil Berdasarkan instruksi yang diterima, proses pengembangan sistem (baik *frontend* maupun *backend*) dilaksanakan. Hasil pengerjaan kemudian dilaporkan kembali secara berkala kepada Supervisor untuk ditinjau (*review*). Apabila terdapat revisi atau penyesuaian, arahan akan diberikan kembali oleh Supervisor setelah mempertimbangkan masukan dari Staf Laboratorium.

Alur komunikasi berjenjang tersebut diterapkan untuk meminimalisir distorsi informasi dan memastikan pengembangan aplikasi berjalan sesuai dengan standar akademik maupun operasional laboratorium.



Gambar 3.2 Bagan Alur Koordinasi

Gambar 3.2 memvisualisasikan skema koordinasi linier terpusat di mana Supervisor bertindak sebagai jembatan penghubung utama (*intermediary*). Hubungan komunikasi dua arah (*bidirectional*) terjalin antara Staf Laboratorium dan Supervisor untuk keperluan analisis kebutuhan serta evaluasi, yang kemudian diterjemahkan menjadi instruksi kerja teknis kepada Intern. Sebaliknya, koordinasi antara Intern dan Supervisor difokuskan pada penerimaan tugas, diskusi teknis, serta pelaporan progres, sehingga memastikan seluruh alur informasi dan validasi hasil pengembangan tetap terstruktur melalui satu pintu kendali tanpa interaksi langsung antara pengembang dan pengguna akhir.

3.2 Tugas yang Dilakukan

Selama periode pelaksanaan program magang, serangkaian aktivitas teknis telah dilaksanakan guna merealisasikan pengembangan aplikasi Optimalisasi Manajemen Peminjaman Ruangan Penelitian. Fokus utama pekerjaan mencakup penerapan siklus hidup pengembangan perangkat lunak (*Software Development Life Cycle*) secara menyeluruh, dimulai dari tahap analisis kebutuhan pengguna, perancangan arsitektur sistem dan basis data, implementasi pengkodean pada sisi *frontend* maupun *backend*, hingga tahap pengujian fungsionalitas. Seluruh kegiatan yang telah diselesaikan tersebut didokumentasikan secara kronologis untuk memberikan gambaran komprehensif mengenai kontribusi teknis dalam proyek ini.

Guna memberikan pemaparan yang terstruktur, rincian tugas dalam Tabel 3.1 diklasifikasikan berdasarkan fase utama pengembangan sistem. Struktur tabel memuat informasi waktu pelaksanaan secara spesifik pada kolom 'Bulan & Minggu', uraian detil aktivitas teknis pada kolom 'Kegiatan', serta hasil konkret yang dicapai pada kolom 'Output', baik berupa dokumen perancangan maupun artefak kode program. Format penyajian ini ditujukan untuk menunjukkan transparansi progres dan akuntabilitas hasil kerja yang terukur di setiap tahapannya.

Tahap Analisis Kebutuhan & Perancangan Sistem		
Bulan & Minggu	**Kegiatan**	**Hasil**
September Minggu 1	Diskusi Awal analisis kebutuhan pengguna dan mempelajari *Tech Stack*	Dokumen Requirement
September Minggu 2	Melanjutkan Diskusi kebutuhan pengguna dan mulai menyusun *framework* yang digunakan	Dokumen Requirement dan struktur *framework*
September Minggu 3	Merancang kebutuhan system dan mulai membuat komponen awal	Dokumen Rancangan Sistem dan komponen awal pada *framework*
September Minggu 4	Merancang design system dan melanjutkan membuat komponen awal	Dokumen Rancangan Sistem dan komponen awal pada *framework*
Tahap *Development & Implementation*		
Bulan & Minggu	**Kegiatan**	**Hasil**
Oktober Minggu 1	Membuat navigasi, memperbaiki *responsive* komponen global, dan membuat komponen datatable template	Router aplikasi yang dapat dinavigasikan via *navbar/sidebar* dan menghasilkan komponen global yang *responsive* dan komponen template
Oktober Minggu 2	Membuat template form sekaligus filter untuk halaman monitoring dan mulai menyiapkan backend environtment	Menghasilkan Template form sekaligus filter dan juga enviroment backend yang siap di integrasikan
Oktober Minggu 3	Mulai mengerjakan halaman *authentication,* membuat scheduler, email notification, dan memperbaiki *responsive*	Halaman Monitoring *Role, User, User Access, Lab, Department, Login, Register, Forgot Password*, notifikasi email dan halaman yang *responsive*

Tahap *Development & Implementation*		
Minggu	**Kegiatan**	**Hasil**
Oktober Minggu 4	Mengerjakan fitur *facility, booking*, memperbaiki beberapa halaman dari fitur *authentication* dan memperbaiki *responsive*	Halaman untuk *manage facility* dan *booking* dan halaman yang lebih *responsive*
November Minggu 1	Membuat fitur *settings* untuk memilih tema aplikasi, membuat halaman *schedule* dan *booking* pada tampilan user	Halaman setting, fitur *darkmode/lightmode,* halaman schedule dan halaman booking pada tampilan user
November Minggu 2	Membuat *dashboard* admin dan membuat logika *scope* data dan membuat tampilan user landing page, labs, article	Halaman *dashboard* yang informatif dan data yang ditampilkan pada *website* ada ruang lingkup berdasarkan *role*
November Minggu 3	Membuat halaman activity dan security logs, penambahan logika pencatatan aktivitas aplikasi	Halaman yang menampilkan aktivitas pada aplikas
November Minggu 4	Finalisasi fitur integrasi chatbot, deploy tunneling dan mulai untuk testing	Fitur yang sudah final dan siap di test
Tahap *Testing & Evaluation*		
Minggu	**Kegiatan**	**Hasil**
November Minggu 4	Mempersiapkan UAT dan memperbaiki *minor bug*	Dokumen UAT
Desember Minggu 1	Mempersiapkan website ke tahap *production ready* dan melakukan *testing* sebelum di *test* langsung oleh *user*	Website yang sudah siap untuk deployment versi production
Desember Minggu 2	Testing langsung oleh user	Dokumen UAT yang sudah di isi oleh user

Tabel 3.1 Tabel Uraian Kegiatan Magang

Tahap Perencanaan dan Desain Sistem (September) Pada bulan pertama, fokus utama kegiatan ditujukan pada pembangunan fondasi teknis melalui tahapan

Requirement Engineering. Minggu-minggu awal September didominasi oleh diskusi analisis kebutuhan pengguna serta pendalaman terhadap *tech stack* agar selaras dengan spesifikasi kebutuhan perangkat lunak. Hal ini sejalan dengan prinsip rekayasa kebutuhan yang menyatakan bahwa pemahaman mendalam terhadap *domain* masalah adalah kunci keberhasilan sistem [6]. Setelah spesifikasi teknis dan bisnis teridentifikasi, proses dilanjutkan dengan perancangan sistem serta pembuatan komponen awal *framework*. Perancangan antarmuka pengguna (*User Interface*) pada tahap ini dilakukan dengan mengacu pada prinsip *User-Centered Design* (UCD) guna memastikan konsistensi visual dan kemudahan interaksi pengguna [7].

Tahap Pengembangan dan Implementasi (Oktober) Memasuki bulan Oktober, intensitas kegiatan meningkat pada tahap Pengembangan dan Implementasi (*Construction*). Pengembangan awal difokuskan pada penyusunan struktur navigasi dan pembuatan komponen global yang responsif dengan menerapkan paradigma *Component-Based Software Engineering* (CBSE). Pendekatan ini dipilih untuk meningkatkan *reusability* dan efisiensi kode dalam ekosistem Next.js [8]. Selanjutnya, dilakukan integrasi layanan *backend* berbasis arsitektur RESTful API, pembuatan templat formulir, hingga pengerjaan fitur krusial seperti Otentikasi dan Penjadwal (*Scheduler*). Periode ini diakhiri dengan penyelesaian fitur manajemen fasilitas dan peminjaman yang menjadi fungsi inti aplikasi.

Penyempurnaan Fitur dan Logika Keamanan (November) Bulan November didedikasikan untuk penyempurnaan fitur dan penguatan logika keamanan sistem. Pengembangan fitur Pengaturan dan personalisasi tampilan (*dark/light mode*) dilakukan untuk memenuhi heuristik kegunaan (*Usability Heuristics*) terkait fleksibilitas dan efisiensi penggunaan [9]. Pada pertengahan bulan, fokus dialihkan pada pembangunan Dasbor Admin dengan menerapkan model *Role-Based Access Control* (RBAC). Penerapan RBAC ini bertujuan untuk membatasi hak akses pengguna berdasarkan peran mereka dalam organisasi, sehingga meminimalisir risiko keamanan data [10]. Selain itu, fitur *Activity & Security Logs*

diimplementasikan sebagai bentuk *audit trail* untuk mencatat seluruh aktivitas sistem demi menjaga integritas dan akuntabilitas.

Tahap Pengujian dan Evaluasi (Desember) Terakhir, pada bulan Desember dilaksanakan tahap Pengujian dan Evaluasi (*Testing & Evaluation*) guna memverifikasi kualitas perangkat lunak. Dua minggu pertama dimanfaatkan untuk finalisasi fitur dan persiapan dokumen *User Acceptance Testing* (UAT). Metode UAT dipilih untuk memvalidasi bahwa sistem telah memenuhi kebutuhan bisnis dan dapat diterima oleh pengguna akhir sebelum dirilis [11]. Selain itu, dilakukan *deployment tunneling* dan pengecekan *bug* mayor untuk memastikan situs web mencapai status *production ready* sesuai dengan standar kualitas ISO/IEC 25010.

Secara keseluruhan, rangkaian kegiatan yang terangkum dalam Tabel 3.1 menunjukkan bahwa proses pengembangan aplikasi telah dilaksanakan secara sistematis dan terukur. Integrasi antara praktik pengembangan teknis dengan landasan teoritis rekayasa perangkat lunak memastikan bahwa luaran (*output*) yang dihasilkan tidak hanya fungsional, tetapi juga andal (*reliable*) dan aman (*secure*). Keberhasilan ini menegaskan kesiapan aplikasi untuk memasuki tahap implementasi penuh guna mendukung efisiensi operasional di Fakultas Teknik dan Informatika.

3.3 Uraian Pelaksanaan Kerja

Bagian ini menguraikan secara komprehensif seluruh aktivitas teknis dan non-teknis yang telah terlaksana selama periode program berlangsung. Fokus utama pelaksanaan kerja tertuju pada realisasi pengembangan sistem informasi peminjaman ruangan berbasis framework *Next.js* dan *FastAPI.* Uraian berikut akan memaparkan penerapan teori rekayasa perangkat lunak ke dalam praktik nyata, mulai dari proses penerjemahan kebutuhan pengguna menjadi spesifikasi teknis hingga implementasi kode program yang menghasilkan solusi digital fungsional bagi Fakultas Teknik dan Informatika.

3.3.1 Proses Pelaksanaan

Pelaksanaan kegiatan magang difokuskan pada penyelesaian satu proyek pengembangan sistem utama yang dikerjakan secara menyeluruh (*end-to-end*) guna memberikan solusi terhadap permasalahan manajemen fasilitas yang ada. Guna memastikan proses rekayasa perangkat lunak berjalan secara sistematis, terukur, dan menghasilkan produk yang berkualitas, metode pendekatan yang diterapkan mengacu pada kerangka kerja Siklus Hidup Pengembangan Perangkat Lunak atau *Software Development Life Cycle* (SDLC). SDLC didefinisikan sebagai serangkaian tahapan standar yang digunakan dalam industri rekayasa perangkat lunak untuk merancang, mengembangkan, dan menguji aplikasi berkualitas tinggi melalui alur kerja yang logis dan terstruktur. Penerapan metodologi ini dalam proyek bertujuan untuk meminimalisir risiko kegagalan sistem melalui tahapan perencanaan yang matang, sekaligus memastikan bahwa setiap kode program yang dibangun telah melalui validasi kebutuhan pengguna yang spesifik [12]. Pendekatan terstruktur ini memungkinkan seluruh aktivitas teknis, mulai dari analisis awal hingga tahap penyebaran aplikasi terdokumentasi dengan baik guna menjaga akuntabilitas dan integritas hasil pengembangan.

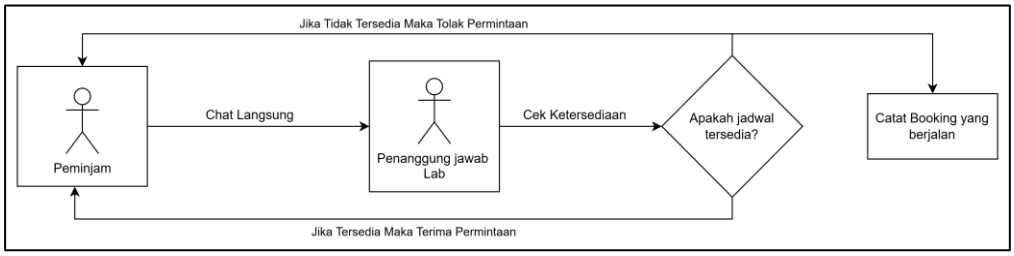
Berdasarkan kerangka kerja SDLC tersebut, serta merujuk pada uraian kegiatan yang telah dipaparkan sebelumnya, seluruh rangkaian pelaksanaan kerja diklasifikasikan ke dalam lima tahapan modular yang merepresentasikan alur logika pengembangan sistem secara komprehensif. Tahapan awal dimulai dengan analisis kebutuhan dan perancangan arsitektur sistem, di mana fokus utama adalah pemahaman domain masalah dan penentuan teknologi. Proses kemudian dilanjutkan dengan perancangan antarmuka pengguna serta skema basis data untuk memastikan konsistensi visual dan efisiensi penyimpanan data. Fase selanjutnya berfokus pada pengembangan layanan *backend* dan sistem keamanan yang mencakup implementasi logika bisnis serta proteksi akses data. Setelah fondasi sistem terbentuk, tahap implementasi *frontend* dan integrasi fitur utama dilaksanakan guna menyatukan antarmuka dengan logika sistem.

Rangkaian proses ini diakhiri dengan tahap pengujian, evaluasi, dan *deployment* untuk memverifikasi fungsionalitas perangkat lunak sebelum digunakan secara operasional di lingkungan fakultas.

3.3.1.1 Analisis Kebutuhan dan Perancangan Arsitektur Sistem

Tahap fundamental dalam pengembangan aplikasi ini dilaksanakan pada minggu pertama hingga kedua bulan September, dengan fokus utama pada proses elisitasi kebutuhan (*requirements elicitation*). Kegiatan ini melibatkan koordinasi intensif antara pengembang, dosen pembimbing, dan staf administrasi laboratorium untuk membedah prosedur operasional yang sedang berjalan. Pemahaman mendalam terhadap kebutuhan pemangku kepentingan (*stakeholders*) melalui komunikasi yang efektif merupakan langkah kritis untuk menghindari ambiguitas spesifikasi fitur yang dapat menghambat fase pengembangan selanjutnya [13]. Oleh karena itu, diskusi dilakukan secara berulang guna memastikan seluruh batasan masalah dan ekspektasi pengguna dapat terpetakan dengan akurat sebelum penulisan kode dimulai.

Berdasarkan hasil diskusi dan observasi lapangan, teridentifikasi akar permasalahan pada mekanisme reservasi ruangan penelitian yang berjalan saat ini. Proses peminjaman ruangan sepenuhnya bergantung pada koordinasi manual melalui komunikasi berbasis pesan singkat (*chat*) secara langsung kepada Dosen Penanggung Jawab (*Person in Charge*/PIC) laboratorium terkait. Metode yang terdesentralisasi ini dinilai tidak efisien karena mengharuskan pemohon menunggu konfirmasi ketersediaan dari dosen yang bersangkutan, yang sering kali menimbulkan jeda waktu (*bottleneck*) dalam layanan akademik. Ketergantungan pada respon manual perorangan ini menghambat fleksibilitas mahasiswa atau peneliti yang membutuhkan kepastian jadwal dengan cepat.



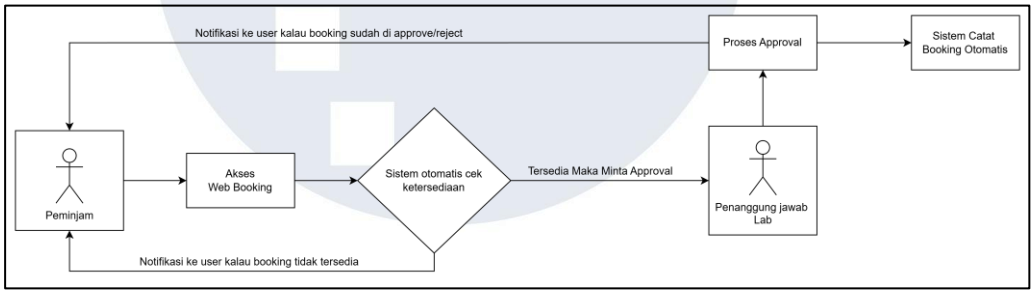
Gambar 3.3 Alur Proses Peminjaman Ruangan Sebelum Integrasi Sistem

Visualisasi alur prosedur peminjaman ruangan yang berjalan sebelum adanya integrasi sistem dapat dilihat pada Gambar 3.3. Sebagaimana tertera pada diagram, proses inisiasi dimulai oleh peminjam yang melakukan komunikasi langsung (*direct chat*) kepada Penanggung Jawab Laboratorium guna menanyakan ketersediaan slot waktu. Apabila jadwal dikonfirmasi tersedia, alur kerja menuntut Penanggung Jawab untuk tidak hanya memberikan konfirmasi persetujuan kepada peminjam, tetapi juga melakukan aktivitas administratif lanjutan berupa pencatatan manual atas peminjaman yang berjalan ("Catat Booking yang berjalan"). Ketergantungan pada mekanisme pencatatan konvensional ini dinilai menjadi titik lemah utama sistem seperti kelalaian dalam mendokumentasikan jadwal yang telah disetujui, terutama saat volume permintaan tinggi menjadi penyebab utama terjadinya ketidaksinkronan data yang berujung pada bentrokan jadwal di lapangan.

Selain masalah efisiensi administrasi, ditemukan pula kendala operasional yang signifikan terkait ketidaktertiban penggunaan fasilitas. Ketiadaan sistem pemantauan terpusat (*single source of truth*) yang dapat diakses publik menyebabkan informasi ketersediaan ruangan menjadi tidak transparan. Kondisi ini sering kali memicu insiden di mana mahasiswa atau dosen menggunakan ruangan tanpa izin atau persetujuan dari Penanggung Jawab, hanya karena berasumsi bahwa ruangan tersebut sedang tidak digunakan. Praktik penggunaan tanpa izin (*unauthorized usage*) ini kerap berujung pada bentrokan jadwal secara fisik di lapangan dengan pengguna

yang telah menempuh prosedur reservasi resmi, sehingga mengganggu kondusivitas kegiatan penelitian.

Risiko kesalahan juga diperparah oleh metode pencatatan jadwal yang tidak terintegrasi dalam satu *platform*, yang menyulitkan verifikasi data secara *real-time*. Hal ini berpotensi menyebabkan terjadinya jadwal tumpang tindih (*double booking*) secara administratif, di mana satu ruangan tercatat dipesan oleh dua pihak berbeda pada waktu yang sama. Guna mengatasi akumulasi permasalahan tersebut, dirumuskan kebutuhan mendesak akan sebuah sistem baru yang mampu mengotomatisasi validasi jadwal, mencegah penggunaan tanpa reservasi, serta menyediakan transparansi data ketersediaan ruangan yang dapat diakses oleh seluruh sivitas akademika.



Gambar 3.4 Alur Proses Peminjaman Setelah Integrasi Sistem

Transformasi alur kerja melalui integrasi sistem digital divisualisasikan pada Gambar 3.4. Dalam skema usulan ini, inefisiensi komunikasi manual dipangkas melalui mekanisme validasi otomatis. Sistem akan secara instan memverifikasi ketersediaan ruang saat peminjam mengakses situs web (*Web Booking*), sehingga permintaan pada slot waktu yang sudah terisi dapat langsung ditolak oleh sistem tanpa perlu melibatkan intervensi manusia. Peran Penanggung Jawab Laboratorium kini difokuskan secara eksklusif pada tahap persetujuan akhir (*approval*), di mana setelah disetujui, sistem secara otomatis mencatat reservasi ke dalam basis data terpusat (*Sistem Catat Booking Otomatis*) dan mengirimkan notifikasi status kepada peminjam. Otomatisasi pencatatan ini menjamin sinkronisasi data secara *real-time* dan menghilangkan risiko lupa catat, namun di sisi lain

menuntut adanya fondasi teknologi yang andal untuk menangani beban validasi data yang cepat dan presisi.

Sebagai respons terhadap tantangan teknis tersebut, arsitektur sistem dirancang menggunakan pendekatan aplikasi web modern yang memisahkan logika antarmuka (*frontend*) dan logika server (*backend*). Pada sisi klien, pengembangan dilakukan menggunakan kerangka kerja Next.js. Pemilihan teknologi ini didasarkan pada kemampuan *Server-Side Rendering* (SSR) yang dimanfaatkan untuk menghasilkan performa muat awal (*initial load*) yang cepat serta meningkatkan SEO pada mesin pencari. Sinergi antara Next.js dan arsitektur *client-server* ini dirancang untuk memberikan pengalaman pengguna (*User Experience*) yang responsif dan interaktif, terutama saat menavigasi jadwal ketersediaan ruangan yang padat data.

Secara spesifik dalam implementasi antarmuka, pengembangan dibangun di atas pustaka React dengan menerapkan paradigma *Component-Based Software Engineering* (CBSE). Dalam pendekatan ini, antarmuka pengguna dipecah menjadi komponen-komponen modular yang independen dan dapat digunakan kembali (*reusable*), seperti komponen tombol, formulir, hingga kartu jadwal. Penerapan struktur berbasis komponen ini tidak hanya mempercepat proses pengembangan melalui penggunaan ulang kode, tetapi juga mempermudah isolasi perbaikan tampilan (*bug fixing*) tanpa mengganggu keseluruhan fungsionalitas sistem, sehingga efisiensi perawatan kode (*code maintainability*) dapat terjaga dengan baik [14].

Sementara itu, untuk menangani logika bisnis dan pemrosesan data di sisi server, dipilih kerangka kerja *FastAPI* yang berjalan pada ekosistem *Python*. Keputusan ini didasarkan pada performa tinggi *FastAPI* dan dukungan natifnya terhadap pemrosesan *asynchronous* (AsyncIO). Kemampuan ini dinilai krusial untuk menangani potensi lonjakan permintaan reservasi secara bersamaan (*high concurrency*), misalnya pada periode

pengumpulan tugas akhir tanpa membebani kinerja server atau menyebabkan *downtime*, sehingga reliabilitas sistem tetap terjaga.

Guna menjamin kualitas dan keterbacaan kode (*clean code*), struktur direktori *backend* disusun dengan mengadopsi pola arsitektur *Model-View-Controller* (MVC). Penerapan pola ini memastikan adanya segregasi tanggung jawab yang tegas antarbagian sistem. *Model* bertanggung jawab mendefinisikan struktur data dan interaksi basis data, *Controller* menangani logika pemrosesan bisnis dan pengaturan rute API, sementara *View* (dalam konteks API) merepresentasikan format data yang dikirimkan kembali ke klien. Pemisahan tanggung jawab ini bertujuan untuk memudahkan pemeliharaan jangka panjang serta mendukung skalabilitas fitur tanpa risiko tumpang tindih logika program [15].

Selain keunggulan performa, pemilihan ekosistem Python memfasilitasi integrasi fitur cerdas berupa *AI Chatbot* yang bertujuan meningkatkan layanan bantuan mandiri. Modul kecerdasan buatan ini dikembangkan menggunakan teknologi *Large Language Model* (LLM) berbasis arsitektur GPT-2 open source. Penggunaan model ini memungkinkan sistem untuk memproses bahasa alami (*Natural Language Processing*) dari pengguna, sehingga interaksi antara mahasiswa dan sistem pertanyaan untuk peminjaman dapat dilakukan secara lebih luwes melalui percakapan teks, tidak hanya terpaku pada antarmuka formulir kaku.

Secara teoritis, *Large Language Model* (LLM) merupakan model pembelajaran mesin yang dilatih menggunakan korpus data teks berskala masif untuk memahami, memprediksi, dan menghasilkan teks baru dengan koherensi tinggi. Namun, keterbatasan utama LLM generik adalah ketidakmampuannya mengakses informasi spesifik yang bersifat privat atau terkini (*proprietary data*) di luar data latihnya. Untuk mengatasi hal tersebut, digunakan pendekatan *Retrieval-Augmented Generation* (RAG). RAG bekerja dengan mekanisme hibrida yang menggabungkan kemampuan

generatif model bahasa dengan sistem temu kembali informasi (*information retrieval*); sistem akan terlebih dahulu mencari dokumen relevan dari basis data lokal saat menerima pertanyaan pengguna, kemudian menyuntikkan konteks tersebut ke dalam *prompt* LLM untuk menghasilkan jawaban yang tidak hanya fasih secara linguistik, tetapi juga akurat secara faktual [16].

Agar respon yang dihasilkan oleh model kecerdasan buatan memiliki relevansi tinggi dengan konteks akademik universitas, implementasi metode RAG dalam sistem ini dirancang untuk mengakses basis pengetahuan (*knowledge base*) spesifik yang tersimpan dalam basis data sistem sebelum memberikan jawaban. Dengan demikian, risiko halusinasi AI di mana model memberikan jawaban yang meyakinkan namun salah dapat diminimalisir secara signifikan. Lebih lanjut, sistem ini dilengkapi dengan fitur manajemen pengetahuan dinamis yang memungkinkan peran Superadmin untuk memperbarui, menambah, atau mengoreksi informasi prosedur peminjaman secara langsung, memastikan *chatbot* selalu memberikan informasi terkini sesuai kebijakan fakultas.

Melengkapi fondasi arsitektur teknis, perancangan sistem juga mencakup aspek keamanan logika melalui penerapan mekanisme *Role-Based Access Control* (RBAC). Secara konseptual, RBAC didefinisikan sebagai mekanisme pembatasan akses sistem di mana izin diberikan berdasarkan peran pengguna dalam organisasi, memastikan bahwa setiap individu hanya dapat mengakses data dan fitur yang relevan dengan tanggung jawab kerjanya (*principle of least privilege*) [17]. Penerapan skema ini didasari oleh kebijakan sistem yang mewajibkan proses otentikasi (*login*) bagi setiap pengguna sebelum dapat melakukan transaksi peminjaman guna menjamin akuntabilitas data.

Hak akses dalam sistem kemudian diklasifikasikan ke dalam empat tingkatan hierarkis. Tingkat tertinggi, Superadmin, dipegang oleh Staf Laboratorium FTI yang memiliki otoritas penuh atas pengelolaan seluruh

fasilitas laboratorium di fakultas. Di bawahnya, peran Admin dialokasikan bagi Staf Program Studi (seperti Sistem Informasi, Teknik Komputer, dan Informatika) dengan tanggung jawab pengelolaan yang terbatas pada laboratorium di bawah naungan departemen masing-masing. Selanjutnya, peran PIC (*Person in Charge*) didelegasikan kepada mahasiswa atau staf yang bertanggung jawab atas satu laboratorium spesifik dengan wewenang yang dikhususkan pada manajemen persetujuan (*approval*). Terakhir, peran Visitor mencakup basis pengguna umum, termasuk mahasiswa dan dosen, yang memiliki hak akses terbatas untuk mengajukan permohonan peminjaman ruangan.

Menindaklanjuti stratifikasi hak akses tersebut, serta guna mengelola kompleksitas fitur yang luas, arsitektur perangkat lunak tidak dirancang sebagai satu kesatuan monolitik, melainkan didistribusikan ke dalam sejumlah modul fungsional yang terintegrasi. Segmentasi ini bertujuan untuk memisahkan logika bisnis antara sisi manajemen administratif (*SubApp Admin*) dan sisi layanan publik (*SubApp Client*), yang secara langsung berimplikasi pada partisi skema basis data (*Entity Relationship Diagram*) agar tetap terstruktur dan menjamin performa sistem. Penjabaran komprehensif mengenai spesifikasi desain antarmuka, struktur basis data, serta alur logika operasional untuk masing-masing modul tersebut diuraikan secara mendalam melalui poin-poin pembahasan berikut.

a. Modul Otentikasi dan Manajemen Pengguna (*Authentication & User Management Module*)

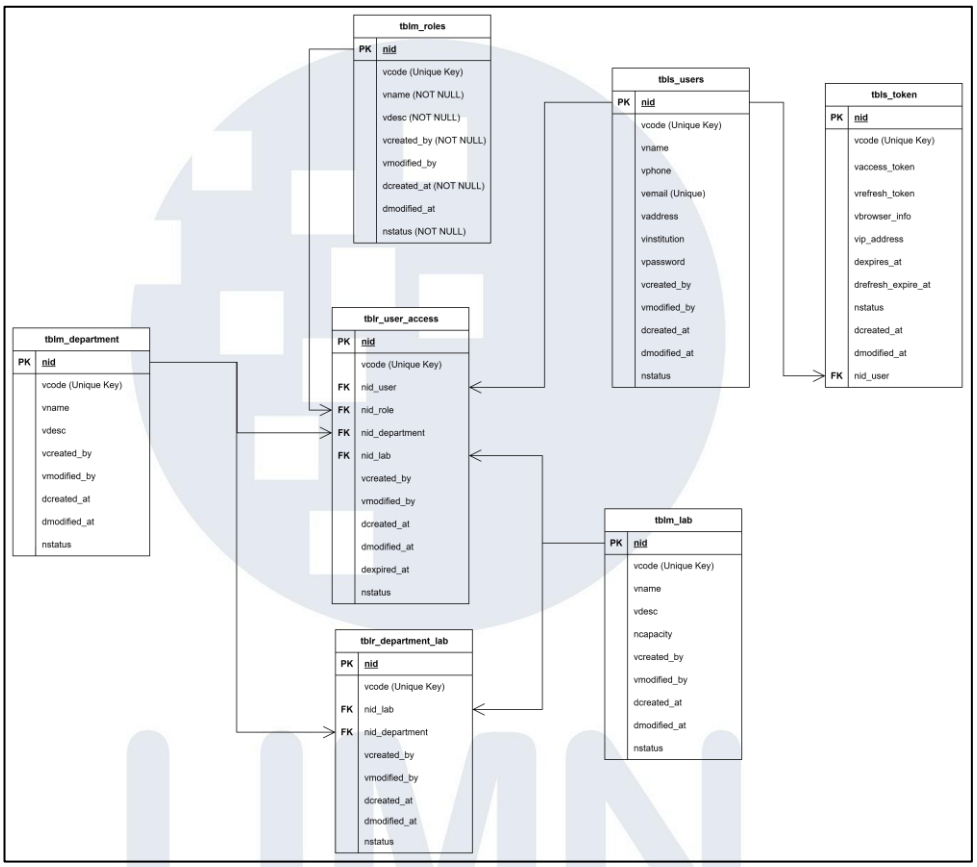
Modul ini dirancang sebagai komponen fundamental yang menjamin keamanan sistem melalui validasi identitas dan pengendalian hak akses secara terpusat. Dalam implementasi arsitektur terdistribusi antara *SubApp Admin* dan *SubApp Client*, mekanisme otentikasi dikembangkan dengan mengadopsi paradigma *Stateless Authentication* sesuai prinsip arsitektur REST (*Representational State Transfer*). Secara teknis, sistem

memanfaatkan standar *JSON Web Token* (JWT) sebagaimana didefinisikan dalam RFC 7519 [18]. Berbeda dengan pendekatan konvensional yang menyimpan sesi pengguna pada basis data server (*stateful*), pendekatan ini menyerahkan penyimpanan token terenkripsi pada sisi klien (*client-side storage*). Server hanya bertugas memvalidasi tanda tangan digital (*signature*) token pada setiap permintaan HTTP, sehingga beban komputasi server dapat diminimalisir dan skalabilitas sistem meningkat signifikan saat menangani konkurensi pengguna yang tinggi.

Pada lapisan otorisasi, sistem menerapkan model Role-Based Access Control (RBAC) yang telah dimodifikasi dengan pendekatan berbasis konteks (*context-aware access control*). Sesuai dengan taksonomi RBAC sistem ini tidak hanya memetakan pengguna ke dalam peran (*roles*), tetapi mengintegrasikan batasan lingkup organisasi secara hierarkis [19]. Melalui fitur pemetaan akses (*user access mapping*), peran Superadmin memiliki otoritas untuk mendefinisikan matriks akses yang mengikat pengguna, peran, departemen, dan laboratorium spesifik dalam satu entitas relasi. Strategi ini merupakan implementasi nyata dari Principle of Least Privilege (PoLP) [20], yang menegaskan bahwa setiap entitas dalam sistem hanya boleh diberikan privilese minimum yang diperlukan untuk menjalankan fungsinya, guna memitigasi risiko eskalasi hak akses (*privilege escalation*) yang tidak sah.

Implementasi fisik dari kebijakan otorisasi tersebut direpresentasikan dalam rancangan skema basis data pada Gambar 3.5. Struktur relasi dipusatkan pada tabel asosiatif tblr_user_access yang berfungsi sebagai penghubung vital (*junction table*) untuk meresolusi hubungan antar entitas. Tabel ini secara spesifik merekam kombinasi unik antara kunci primer pengguna (tbls_users), peran (tblm_roles), serta lingkup unit kerja (tblm_department dan tblm_lab), memungkinkan sistem memvalidasi hak akses yang bersifat granular tanpa melanggar aturan normalisasi data. Selain itu, terlihat adanya segregasi fungsional pada tabel

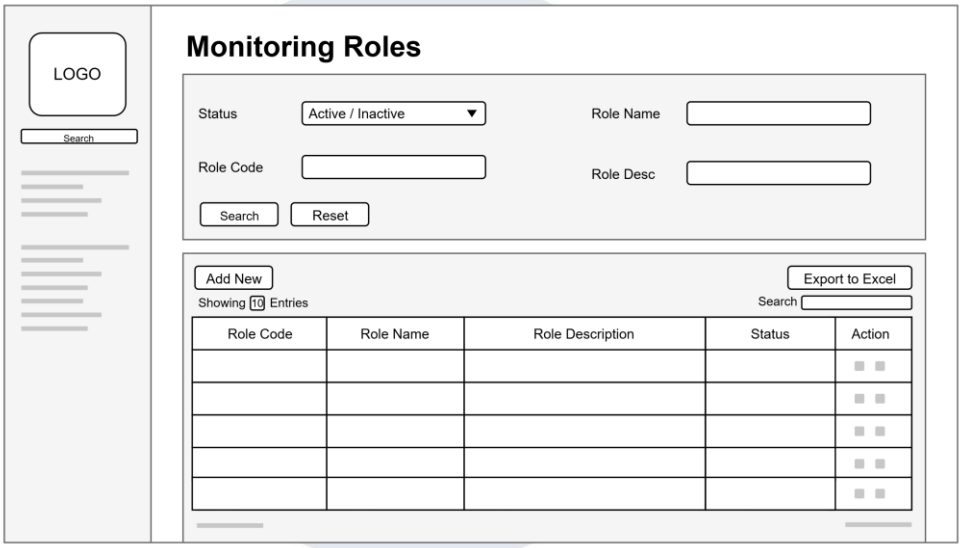
tbls_token, yang dirancang khusus untuk menyimpan token utilitas bersifat sementara (*ephemeral*) seperti token verifikasi email secara terpisah dari data master pengguna, guna menjaga integritas dan keamanan siklus hidup kredensial.



Gambar 3.5 ERD Modul Authentication

Realisasi dari struktur data relasional tersebut dimanifestasikan ke dalam antarmuka pengguna pada modul *SubApp Admin*. Perancangan antarmuka difokuskan pada kemudahan navigasi bagi Superadmin dalam mengelola entitas master, meliputi data Pengguna, Peran, Departemen, dan Laboratorium. Guna menjamin efisiensi operasional dan kurva pembelajaran (*learning curve*) yang cepat, sistem menerapkan pola desain antarmuka yang terstandarisasi (*standardized layout*) untuk seluruh halaman manajemen data master. Sebagaimana divisualisasikan pada Gambar 3.6, struktur halaman manajemen peran (*Monitoring Roles*)

merepresentasikan pola desain yang juga diadopsi secara identik pada halaman manajemen departemen dan laboratorium. Keseragaman ini mencakup tata letak formulir pencarian, penyajian data dalam bentuk tabel, serta penempatan tombol aksi, sehingga administrator dapat beralih antar-menu pengelolaan dengan pengalaman pengguna yang konsisten.



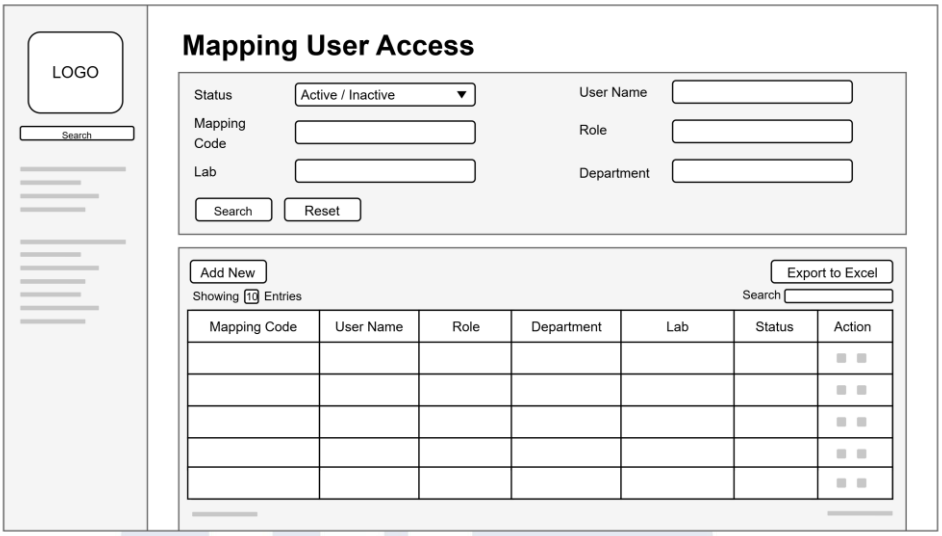
Gambar 3.6 Struktur Halaman Roles

Gambar 3.7 Struktur Formulir Add dan Update Halaman Role

Sementara itu, mekanisme interaksi untuk manipulasi data (*Create, Update*) dirancang menggunakan pendekatan antarmuka modal (*modal dialog*). Sebagaimana terlihat pada Gambar 3.7, formulir penambahan dan penyuntingan data disajikan dalam jendela *pop-up* di atas lapisan konten utama (*overlay*), sehingga pengguna tidak perlu berpindah halaman (*page refresh*) saat melakukan input data. Strategi desain ini bertujuan untuk mempertahankan konteks kerja pengguna agar tetap fokus pada tugas

administratif yang sedang berjalan. Konsistensi desain formulir juga dijaga di seluruh modul aplikasi, mulai dari keseragaman label input, validasi kolom wajib diisi (*mandatory fields*), hingga penempatan tombol aksi, guna menciptakan pengalaman pengguna yang intuitif dan kohesif.

Implementasi teknis dari konsep *context-aware role assignment* direalisasikan secara spesifik pada halaman Mapping User Access. Sebagaimana ditampilkan pada Gambar 3.8, antarmuka ini dirancang sebagai pusat kendali untuk menghubungkan entitas pengguna dengan peran jabatan serta lingkup kerja operasionalnya (Departemen dan Laboratorium). Berbeda dengan halaman master data standar, halaman ini dilengkapi dengan fitur filter tingkat lanjut (*advanced filtering*) yang memungkinkan administrator untuk melacak hak akses berdasarkan kombinasi spesifik misalnya, mencari seluruh "PIC" yang aktif di "Laboratorium Big Data". Tabel pemetaan menyajikan data komprehensif yang mencakup kode pemetaan unik, identitas pengguna, serta hierarki penugasan mereka, memberikan transparansi penuh terhadap siapa yang memiliki otoritas atas fasilitas tertentu. Fitur ini menjadi instrumen vital dalam menjaga akuntabilitas sistem, memastikan bahwa setiap hak akses yang diberikan tercatat dan dapat diaudit dengan mudah.



Gambar 3.8 Struktur Halaman User Access

Gambar 3.9 Struktur Formulir Untuk Halaman User Access

Operasional teknis untuk menetapkan hak akses tersebut difasilitasi melalui antarmuka formulir pemetaan sebagaimana ditampilkan pada Gambar 3.9. Dalam mekanisme input ini, sistem menerapkan validasi integritas data yang ketat dengan menggunakan elemen *dropdown* dinamis untuk pemilihan Nama Pengguna, Peran, Departemen, dan Laboratorium. Desain formulir ini memaksa administrator untuk mendefinisikan konteks penugasan secara lengkap sebelum data disimpan; artinya, sistem tidak akan mengizinkan pembuatan hak akses yang ambigu (misalnya, memberikan peran PIC tanpa menunjuk laboratorium spesifik). Selain itu, formulir ini juga menangani logika *Update* di mana status keaktifan pemetaan (*Active/Inactive*) dapat dikelola, memungkinkan pencabutan akses sementara tanpa perlu menghapus riwayat penugasan pengguna.

Penting untuk digarisbawahi bahwa rangkaian visualisasi antarmuka yang disajikan di atas merupakan sampel representatif (*representative samples*) dari keseluruhan artefak desain *wireframe* yang telah dikembangkan pada modul ini. Mengingat kompleksitas sistem yang mencakup berbagai variasi status interaksi (*states*) dan penanganan pengecualian (*exception handling*), laporan ini hanya memprioritaskan visualisasi antarmuka inti yang merepresentasikan logika bisnis utama guna menjaga efisiensi dan fokus pembahasan. Kendati demikian, seluruh rancangan antarmuka lain yang tidak ditampilkan secara eksplisit tetap dikembangkan dengan mengacu pada standar sistem desain (*design system*) dan pola interaksi yang konsisten dengan contoh yang telah didemonstrasikan.

Dengan tersedianya struktur hak akses yang terdefinisi secara granular serta antarmuka manajemen pengguna yang matang, sistem kini memiliki fondasi operasional yang kokoh untuk menangani pengelolaan aset fisik. Oleh karena itu, pembahasan selanjutnya akan beralih pada aspek objek manajemen utama, yaitu pengelolaan data master laboratorium dan fasilitas pendukungnya, yang akan diuraikan secara mendalam dalam Modul Manajemen Data Master Laboratorium.

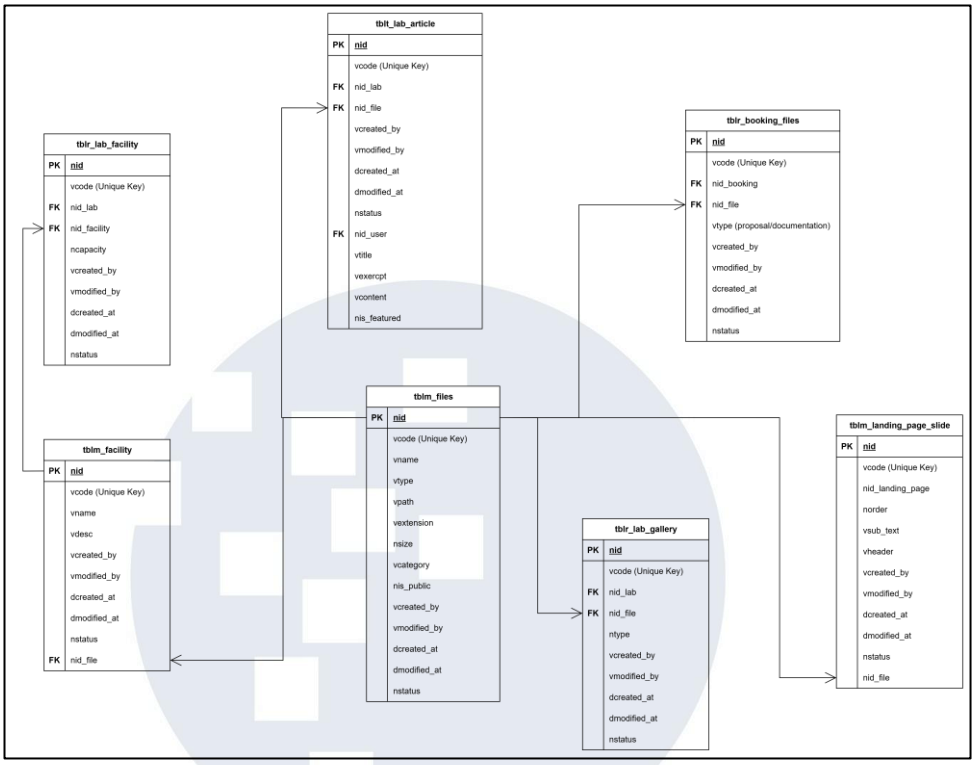
b. Modul Manajemen Inventaris Fasilitas dan Arsip Digital (*Facility Inventory & Digital Archive Management Module*)

Setelah terbentuknya infrastruktur keamanan dan manajemen pengguna yang solid, fokus pengembangan sistem beralih pada transformasi aset laboratorium menjadi entitas digital yang terstruktur. Modul ini dirancang dengan dua tujuan strategis: pertama, sebagai sistem inventarisasi perangkat keras yang menjamin transparansi ketersediaan alat bagi calon peminjam; dan kedua, sebagai platform manajemen pengetahuan (*Knowledge Management System*) yang mengarsipkan luaran intelektual laboratorium seperti proposal penelitian, laporan riset, dan dokumentasi

kegiatan. Berdasarkan pembagian peran pada *SubApp Admin*, modul ini memberikan kewenangan kepada Superadmin dan Admin Program Studi untuk mengkurasi data aset fisik serta mengelola konten digital yang merepresentasikan profil akademik setiap laboratorium.

Pada aspek manajemen fisik, sistem menerapkan mekanisme pemetaan fasilitas yang detail. Data inventaris seperti komputer spesifikasi tinggi, perangkat IoT, atau proyektor tidak hanya dicatat sebagai daftar statis, melainkan dipetakan relasinya terhadap laboratorium tertentu. Hal ini memungkinkan sistem untuk memvalidasi apakah sebuah ruangan memenuhi persyaratan teknis yang dibutuhkan pengguna sebelum reservasi dilakukan. Sementara itu, pada aspek manajemen arsip digital, sistem mengadopsi pendekatan repositori terpusat. Seluruh berkas digital baik itu lampiran persyaratan peminjaman (proposal) maupun konten publikasi laboratorium (artikel/foto) dikelola dalam satu entitas penyimpanan yang terintegrasi. Pendekatan ini selaras dengan konsep *Knowledge Management*, di mana data eksplisit (dokumen/file) harus dikelola secara sistematis agar dapat diakses kembali dan menjadi aset pengetahuan institusi yang berkelanjutan [21].

Implementasi teknis dari manajemen aset tersebut divisualisasikan melalui skema basis data pada Gambar 3.10. Struktur data berpusat pada dua entitas utama. Pertama, tabel tblm_facility berfungsi menyimpan atribut detail fasilitas fisik, yang kemudian dihubungkan ke laboratorium melalui tabel asosiatif tblr_lab_facility. Desain ini memungkinkan satu jenis fasilitas (misalnya, "PC High-Spec") dipetakan ke banyak laboratorium sekaligus dengan jumlah kapasitas yang berbeda-beda. Menariknya, pada skema terbaru ini, setiap data fasilitas dalam tblm_facility juga memiliki relasi langsung ke tabel penyimpanan file melalui kolom nid_file, yang memungkinkan sistem untuk menampilkan foto atau manual penggunaan alat secara spesifik tanpa memerlukan tabel perantara tambahan.



Gambar 3.10 ERD Pada Modul Manajemen Inventaris

Keunggulan strategis dari penerapan desain terpusat pada entitas tblm_files terletak pada fleksibilitasnya dalam membangun relasi dengan berbagai modul fungsional lain tanpa memicu redundansi struktur penyimpanan. Sebagaimana divisualisasikan pada diagram, tabel ini berperan sebagai induk referensi universal yang melayani berbagai kebutuhan arsip digital secara simultan. Secara spesifik, tabel ini berelasi dengan tblr_booking_files untuk memfasilitasi penyimpanan dokumen administratif seperti proposal atau legalitas peminjaman, serta terhubung dengan tblt_lab_article guna mengelola aset visual yang melekat pada artikel profil laboratorium. Selain itu, sistem kini mengakomodasi kebutuhan dokumentasi visual yang lebih luas melalui tabel tblr_lab_gallery, yang memungkinkan pengelola untuk mengunggah galeri foto kegiatan laboratorium secara terstruktur. Mekanisme ini juga dimanfaatkan oleh tblm_landing_page_slide sebagai basis data aset visual *banner* pada halaman utama. Melalui unifikasi repositori ini, desain skema

sistem tidak hanya menjamin efisiensi penyimpanan (*storage efficiency*), tetapi juga menyederhanakan proses pemeliharaan (*maintenance*) aset digital dalam jangka panjang.

Guna menjaga koherensi visual dan pengalaman pengguna (*User Experience*) yang optimal, implementasi antarmuka pada Modul Manajemen Inventaris dan Arsip Digital ini dikembangkan dengan mengacu pada sistem desain (*design system*) yang identik dengan modul sebelumnya. Komponen antarmuka untuk fitur pemantauan (*monitoring*) maupun manipulasi data (*CRUD*) seperti struktur tabel data, mekanisme penyaringan (*filtering*), hingga tampilan formulir modal, sepenuhnya mereplikasi pola interaksi yang telah dipaparkan pada Gambar 3.6 dan Gambar 3.7. Variasi visual pada halaman manajemen fasilitas dan konten hanya terbatas pada penyesuaian atribut formulir (*input fields*) dan kolom data yang relevan dengan konteks inventaris, tanpa mengubah kerangka dasar navigasi.

Dengan telah tersedianya struktur manajemen pengguna yang aman (Modul A) serta repositori data inventaris laboratorium yang lengkap (Modul B), sistem kini telah memiliki prasyarat infrastruktur data yang memadai untuk menjalankan fungsi operasional utamanya. Oleh karena itu, pembahasan selanjutnya akan difokuskan pada logika bisnis inti aplikasi, yaitu mekanisme transaksi peminjaman ruang dan validasi ketersediaan jadwal yang akan diuraikan secara mendalam dalam Modul Reservasi dan Penjadwalan.

c. Modul Manajemen Reservasi dan Penjadwalan (*Booking and Schdule Module*)

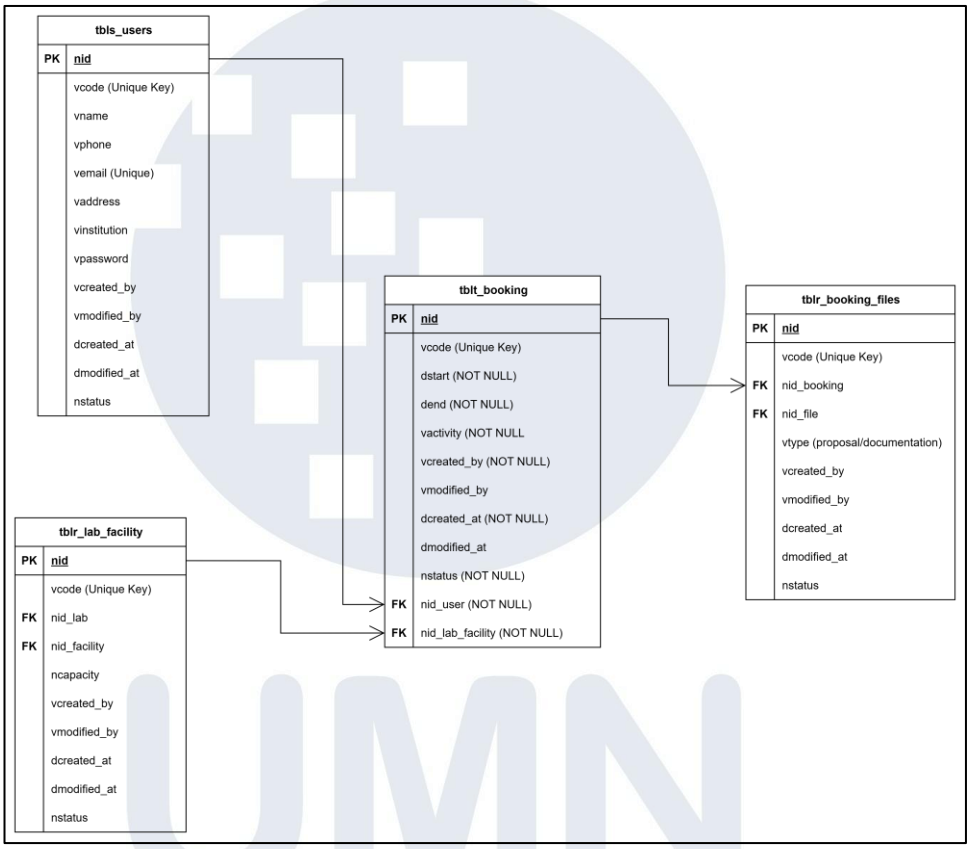
Modul Reservasi dan Penjadwalan dikembangkan sebagai komponen inti (*core business logic*) yang mengorkestrasi seluruh interaksi transaksional antara pengguna dan fasilitas laboratorium. Modul ini tidak hanya berfungsi sebagai pencatat jadwal, melainkan menerapkan Mesin

Keadaan Terhingga (*Finite State Machine*) untuk mengelola siklus hidup peminjaman (*booking lifecycle*) secara ketat. Berdasarkan alur kerja yang dirancang, sistem membagi proses reservasi menjadi tiga fase krusial: fase inisiasi dan validasi konflik, fase persetujuan berjenjang, dan fase pertanggungjawaban pasca-kegiatan. Pendekatan ini bertujuan untuk memitigasi masalah klasik manajemen laboratorium seperti tumpang tindih jadwal (*double booking*) serta rendahnya tingkat pelaporan hasil kegiatan mahasiswa.

Mekanisme operasional dimulai pada fase inisiasi, di mana pengguna mengajukan permohonan reservasi dengan parameter waktu tertentu. Pada tahap ini, algoritma deteksi konflik (*conflict detection algorithm*) bekerja secara *real-time* dengan memindai tabel tblt_booking untuk memastikan tidak ada irisan waktu (*time overlap*) dengan jadwal yang berstatus aktif pada fasilitas yang sama. Jika slot waktu valid, sistem akan menyimpan permohonan dengan status awal Pending. Transaksi ini kemudian masuk ke dalam antrean tinjauan (*approval queue*) yang memberikan otoritas kepada Admin Prodi atau PIC Laboratorium untuk melakukan verifikasi manual. Administrator dapat mengubah status menjadi Approved (disetujui) atau Rejected (ditolak) berdasarkan urgensi dan kelengkapan administrasi pemohon. Mekanisme hibrida yang menggabungkan validasi mesin dan manusia ini menjamin integritas jadwal tetap terjaga namun tetap fleksibel terhadap kebijakan fakultas.

Keunggulan distingtif dari modul ini terletak pada fase pasca-kegiatan yang menerapkan prinsip akuntabilitas tertutup (*closed-loop accountability*). Sistem dilengkapi dengan penjadwal otomatis (*scheduler*) yang memantau durasi setiap peminjaman yang aktif. Apabila waktu kegiatan telah berakhir (melewati dend), sistem secara otomatis memutakhirkan status transaksi dari *Approved* menjadi Waiting for Documentation. Pada status ini, sistem memblokir penyelesaian transaksi hingga pengguna memenuhi kewajiban administrasi, yaitu mengunggah

bukti pelaksanaan berupa dua foto dokumentasi kegiatan dan satu berkas PDF luaran hasil penelitian. Validasi kelengkapan ini menjadi syarat mutlak bagi sistem untuk mengubah status akhir menjadi Done, memastikan bahwa setiap penggunaan fasilitas terekam tidak hanya sebagai log waktu, tetapi juga menghasilkan aset arsip digital yang nyata bagi institusi.



Gambar 3.11 ERD Modul Booking

Struktur data yang mendukung logika transaksional tersebut divisualisasikan pada Gambar 3.11. Tabel tblt_booking bertindak sebagai entitas transaksi pusat yang merekam atribut krusial seperti rentang waktu (dstart, dend), jenis aktivitas (vactivity), serta status terkini (nstatus). Relasi antar-entitas dibangun secara terintegrasi, di mana setiap peminjaman terhubung langsung dengan pengguna (tbls_users) sebagai subjek peminjam dan fasilitas spesifik (tblr_lab_facility) sebagai objek yang dipinjam. Aspek dokumentasi dikelola melalui tabel relasi tblr_booking_files, yang

dirancang fleksibel untuk menampung berbagai jenis berkas. Kolom vtype pada tabel ini berfungsi membedakan antara berkas "Proposal" yang diunggah di awal pengajuan dan berkas "Documentation" yang diwajibkan di akhir kegiatan, memungkinkan sistem untuk melacak kelengkapan administrasi secara presisi dalam satu riwayat transaksi yang utuh.

Guna menjaga koherensi visual dan meminimalisir kurva pembelajaran pengguna (*learning curve*), perancangan antarmuka pada modul transaksi ini sepenuhnya mengadopsi standar sistem desain (*design system*) yang telah diimplementasikan pada modul manajemen pengguna dan inventaris. Struktur visual untuk fitur pemantauan daftar reservasi (*booking monitoring*) maupun formulir pengajuan tetap mempertahankan pola tata letak yang identik menggunakan penyajian data tabular yang dilengkapi fitur pencarian serta mekanisme formulir modal (*modal form*) untuk interaksi data. Diferensiasi antarmuka hanya bersifat kontekstual pada level atribut input, di mana formulir reservasi secara spesifik dilengkapi dengan elemen pemilih tanggal dan waktu (*datetime picker*) serta indikator status dinamis, tanpa mengubah kerangka navigasi utama. Dengan tuntasnya penjabaran mengenai logika reservasi yang mengintegrasikan validasi sistem otomatis dan verifikasi administratif ini, maka seluruh komponen fungsional utama aplikasi telah terdefinisi secara komprehensif, membentuk satu kesatuan ekosistem manajemen laboratorium yang siap dioperasionalkan.

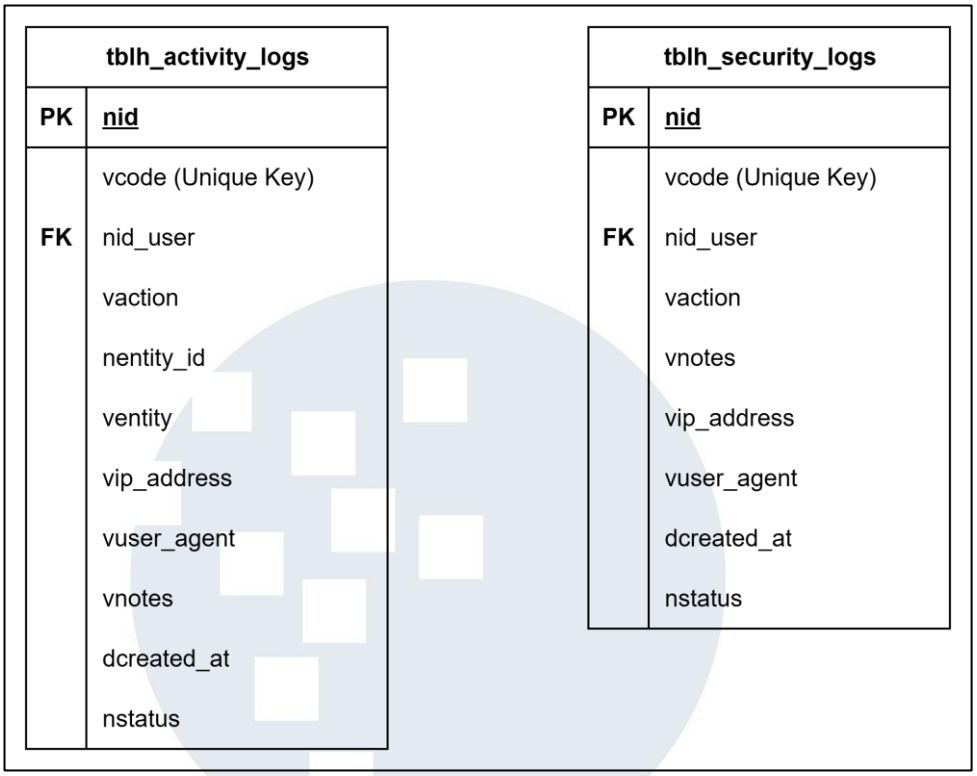
d. Modul Audit Aktivitas dan Visualisasi Dasbor (*Activity Audit & Dashboard Visualization Module*)

Sebagai lapisan pertahanan terakhir sekaligus pusat kendali informasi, Modul Audit dan Visualisasi Dasbor dikembangkan dengan pendekatan Antarmuka Adaptif Berbasis Peran (*Role-Based Adaptive Interface*). Modul ini dirancang untuk tidak menyajikan satu tampilan generik bagi seluruh pengguna, melainkan menyesuaikan komposisi *widget* dan metrik data

berdasarkan hierarki tanggung jawab pengguna yang sedang mengakses sistem. Strategi ini memastikan bahwa setiap aktor mulai dari Superadmin hingga PIC Laboratorium hanya disuguhi informasi yang relevan dengan konteks pengambilan keputusan mereka.

Bagi pengguna dengan privilese Superadmin, dasbor difungsikan sebagai panel Kesehatan & Pengawasan Sistem (*System Health & Oversight*). Antarmuka ini dirancang memuat rangkaian *widget* metrik teknis, antara lain: (1) System Pulse, yang memvisualisasikan status *uptime* server dan kesehatan koneksi basis data; (2) Threat Level, yang memantau anomali keamanan atau percobaan akses ilegal; serta (3) Infra Load, yang mengukur beban pemeliharaan infrastruktur. Selain itu, terdapat fitur kendali khusus berupa panel *Booking for Maintenance*, yang memberikan otoritas kepada Superadmin untuk memblokir jadwal laboratorium secara paksa guna keperluan perbaikan mendesak, sebuah fitur intervensi tingkat tinggi yang tidak diberikan kepada peran operasional lainnya.

Sebaliknya, bagi pengelola operasional seperti Admin Program Studi dan PIC Laboratorium, dasbor bertransformasi menjadi alat pemantauan siklus hidup transaksi. Visualisasi data difokuskan pada indikator kinerja layanan melalui *widget* statistik seperti: (1) Need Approval, yang menampilkan jumlah antrean reservasi yang membutuhkan persetujuan segera; (2) Pending Report, yang melacak daftar peminjam yang telah selesai menggunakan ruangan namun belum mengunggah dokumen pertanggungjawaban; serta (3) Lab Schedule, sebuah kalender interaktif yang memetakan okupansi ruangan secara visual. Meskipun menggunakan komponen yang serupa, sistem menerapkan isolasi data (*data scoping*) di mana Admin Prodi dapat memantau agregat data lintas-laboratorium, sedangkan PIC hanya mengakses data spesifik pada laboratorium yang dikelolanya.



Gambar 3.12 Struktur Table Activity & Security Logs

Keunggulan utama modul ini terletak pada penerapan strategi Audit Terpusat (*Centralized Logging Strategy*) yang memisahkan pencatatan aktivitas operasional dengan aktivitas keamanan. Sebagaimana divisualisasikan pada Gambar 3.12, sistem mengalokasikan dua entitas tabel riwayat khusus. Pertama, tabel tblh_activity_logs bertugas merekam setiap aksi manipulasi data (*CRUD Operations*). Tabel ini mencatat secara detail siapa pelakunya (nid_user), apa aksinya (vaction), entitas apa yang diubah (ventity), serta catatan perubahan spesifik (vnotes). Kedua, tabel tblh_security_logs didedikasikan khusus untuk merekam peristiwa keamanan krusial seperti upaya *login*, kegagalan otentikasi, atau perubahan kata sandi.

Implementasi kedua tabel log tersebut memenuhi standar Non-Repudiasi yang ketat dengan menyertakan atribut forensik digital, yaitu vip_address (alamat protokol internet pengguna) dan vuser_agent

(informasi perangkat dan peramban yang digunakan). Keberadaan data ini memungkinkan administrator untuk melacak asal muasal sebuah aksi hingga ke level perangkat fisik, yang sangat vital dalam proses investigasi insiden keamanan. Konsistensi desain antarmuka juga tetap dipertahankan pada modul ini, di mana data log disajikan menggunakan struktur tabel standar yang dilengkapi fitur penyaringan dan pencarian, memastikan pengalaman pengguna yang kohesif mulai dari manajemen aset, transaksi, hingga pemantauan jejak audit.

Guna menjaga koherensi visual dan meminimalisir kurva pembelajaran pengguna (*learning curve*), seluruh variasi komponen dasbor di atas dikembangkan menggunakan sistem desain (*design system*) yang konsisten, baik dari segi tipografi, palet warna indikator status, maupun pola interaksi. Dengan tuntasnya pemaparan pada Modul Audit dan Visualisasi Dasbor ini, maka seluruh rangkaian spesifikasi fungsional dan logika bisnis sistem mulai dari lapisan keamanan (Modul A), manajemen inventaris (Modul B), mesin reservasi (Modul C), hingga pengawasan bertingkat (Modul D) telah terdefinisi secara komprehensif. Integrasi keempat modul ini membentuk satu kesatuan arsitektur solusi yang solid, yang tidak hanya menjawab kebutuhan operasional peminjaman laboratorium, tetapi juga memenuhi standar keamanan dan akuntabilitas data yang dipersyaratkan.

Melengkapi spesifikasi fungsional di atas, perancangan skema basis data sistem menerapkan standar penamaan variabel yang ketat guna menjamin konsistensi struktur data. Konvensi yang diadopsi menggabungkan pendekatan Hungarian Notation dan Snake Case, di mana setiap atribut tabel diawali dengan prefiks huruf kecil yang merepresentasikan tipe datanya. Sebagai contoh, prefiks v digunakan untuk tipe data *varchar* (seperti vname, vcode), prefiks n untuk tipe data *numeric/integer* (seperti nstatus, ncapacity), dan prefiks d untuk tipe data *date/datetime* (seperti dcreated_at). Penerapan standar ini bertujuan untuk menciptakan dokumentasi mandiri (*self-documenting schema*) pada level basis data, sehingga memudahkan

pengembang dalam mengidentifikasi jenis data secara instan dan meminimalisir kesalahan tipe data (*type mismatch*) selama proses pengembangan.

Adapun penyusunan dokumen perancangan sistem ini dilakukan sepenuhnya di dalam platform Draw.io. Seluruh artefak visualisasi, mulai dari skema Diagram Hubungan Entitas (*Entity Relationship Diagram/ERD*), kerangka dasar antarmuka (*wireframe*), hingga rancangan desain visual akhir (*high-fidelity mockup*), dikerjakan dan diintegrasikan dalam satu lingkungan kerja (*workspace*) pada platform tersebut. Pendekatan terpusat ini dipilih untuk memastikan bahwa setiap komponen perancangan—baik struktur basis data maupun antarmuka pengguna—terdokumentasi dalam satu kanvas proyek yang kohesif. Selain menjamin presisi notasi standar industri, penggunaan Draw.io sebagai basis dokumentasi utama juga memudahkan proses pemeliharaan konsistensi visual dan manajemen versi desain secara terstruktur dalam satu berkas referensi.

Dengan terdefinisinya seluruh spesifikasi fungsional dan logika bisnis sistem yang terbagi dalam empat modul utama (A-D), perancangan ini kini diperkuat oleh standar teknis yang konsisten. Konsistensi dicapai melalui adopsi konvensi penamaan Hungarian Notation dan Snake Case pada skema basis data, yang bertujuan untuk menciptakan dokumentasi mandiri (*self-documenting schema*). Selain itu, seluruh artefak perancangan visual, termasuk ERD dan *wireframe*, dikerjakan secara terintegrasi menggunakan perangkat lunak Draw.io. Seluruh tahapan perancangan fungsional dan teknis ini secara kolektif menghasilkan cetak biru (*blueprint*) sistem yang solid dan akuntabel, siap untuk dilanjutkan ke tahap implementasi teknis.

3.3.1.2 Development & Implementation

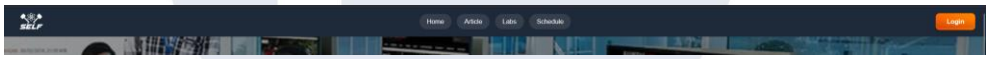
Tahap implementasi dan pengembangan sistem merupakan fase krusial yang dilaksanakan secara intensif mulai dari awal Oktober hingga akhir November. Fokus utama pada periode ini adalah merealisasikan cetak biru

sistem yang telah dirancang ke dalam kode program fungsional, memanfaatkan Next.js sebagai kerangka kerja *frontend* dan FastAPI untuk layanan *backend* (Tabel 3.1). Proses pengembangan diawali pada Oktober Minggu 1 dan 2 dengan pembangunan fondasi teknis, meliputi penyusunan *router* aplikasi yang *responsive*, pengembangan templat komponen *datatable* dan formulir untuk efisiensi *input*, serta penyiapan lingkungan *backend* agar siap diintegrasikan. Kegiatan dilanjutkan pada Oktober Minggu 3 dan 4 dengan pengimplementasian modul otentikasi (*Login*, *Register*) dan manajemen data master (Role, User, User Access, Lab, Department), dilanjutkan dengan pengerjaan fungsionalitas inti aplikasi, yaitu fitur facility dan booking. Tahap akhir pengembangan, yang berlangsung hingga akhir November, difokuskan pada penguatan sistem keamanan, seperti pembangunan Dasbor Admin dengan logika *scope data* berdasarkan peran (*role*), dan implementasi fitur pelacakan keamanan (Activity dan Security Logs), sebagai persiapan finalisasi fitur dan *deploy tunneling* menjelang *User Acceptance Testing* (UAT).

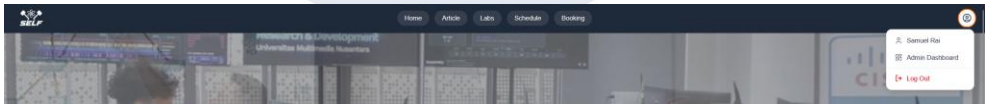
Pada minggu pertama bulan Oktober, fokus kegiatan diarahkan pada pembentukan fondasi *frontend* yang *responsive* dan komponen dasar sistem. Kegiatan utama yang dilaksanakan meliputi pembuatan navigasi, perbaikan responsivitas komponen global, dan pembuatan komponen *datatable template*. Aktivitas ini menghasilkan *Router* aplikasi yang fungsional dan dapat dinavigasikan melalui *navbar/sidebar*. Selain itu, dihasilkan komponen *global* yang *responsive* dan komponen *template* formulir yang mendukung berbagai tipe *input* data (termasuk *date picker*, *time picker*, dan *multi-file upload*), menegaskan penerapan paradigma *Component-Based Software Engineering* (CBSE) untuk mendukung efisiensi dan *reusability* kode pada arsitektur Next.js.

Pengembangan antarmuka menggunakan komponen modular yang *reusable* merupakan pilar dari pendekatan Component-Based Software Engineering (CBSE). CBSE merupakan cabang dari rekayasa perangkat

lunak yang menekankan pemanfaatan kembali artefak perangkat lunak yang sudah ada (*reuse*) sebagai strategi untuk meningkatkan kualitas sistem dengan biaya yang lebih rendah dan waktu pengembangan yang lebih singkat [14]. Secara fundamental, komponen-komponen yang telah dikembangkan seperti navigasi dan *form input* memecah kompleksitas antarmuka menjadi unit-unit independen. Strategi ini tidak hanya mempercepat proses pengembangan awal, tetapi juga mempermudah *maintainability* dan *bug fixing*. Selain itu, perancangan yang *responsive* sejak awal memastikan aplikasi dapat memberikan pengalaman pengguna (*User Experience*) yang konsisten di berbagai perangkat, sejalan dengan prinsip-prinsip User-Centered Design (UCD) yang mengedepankan pemahaman kebutuhan pengguna di setiap tahapan pengembangan [7].

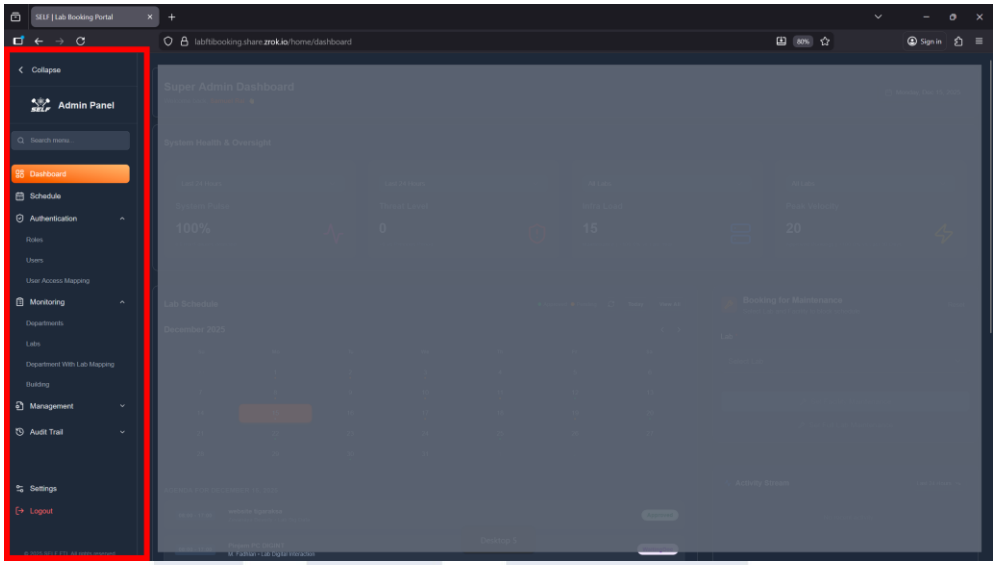


Gambar 3.13 Header Navigasi Sebelum Login

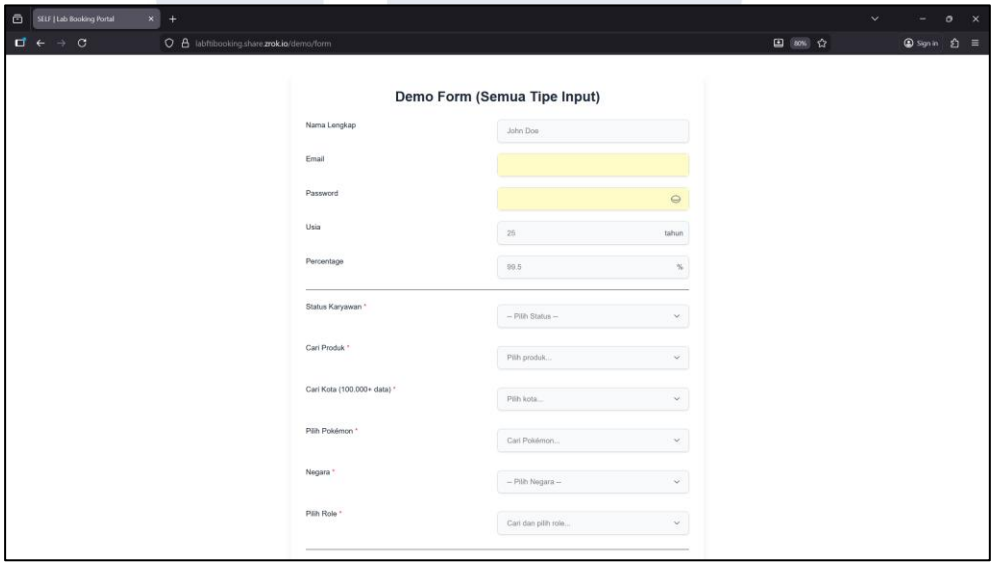


Gambar 3.14 Header Navigasi Setelah Login Sebagai Superadmin

Realisasi navigasi aplikasi dikerjakan dengan memisahkan antarmuka berdasarkan peran pengguna (*client* dan *admin*). Gambar 3.13 dan 3.14 merupakan struktur navigasi sisi publik (*Client*) direpresentasikan melalui *header* navigasi utama yang menyediakan akses langsung ke halaman informasi dan jadwal, serta tombol *Login* untuk autentikasi. Sementara itu pada gambar 3.15 diperlihatkan dihasilkan pula struktur *sidebar* pada *Admin Panel* yang memuat menu hierarkis seperti *Dashboard*, *Authentication* (*Roles*, *Users*, *User Access Mapping*), *Monitoring* (*Departments*, *Labs*), *Management*, dan *Audit Trail*. Implementasi ini memvisualisasikan arsitektur *fullstack* yang memisahkan logika antarmuka Next.js dengan logika server FastAPI, di mana jalur navigasi dirancang sesuai dengan hierarki akses Role-Based Access Control (RBAC).



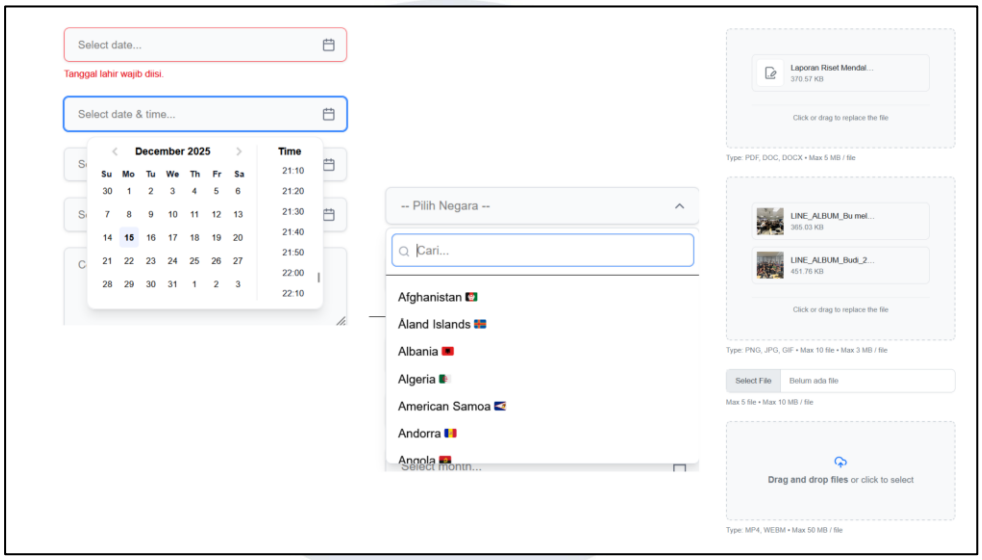
Gambar 3.15 Struktur Navigasi Sidebar Pada Admin Panel



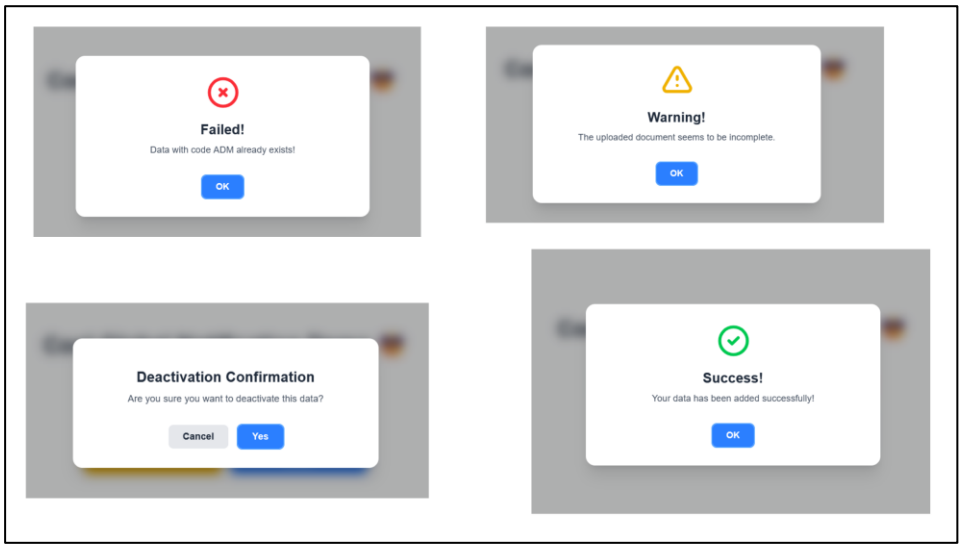
Gambar 3.16 Contoh Komponen Input Pada Form

Selain struktur navigasi, dihasilkan komponen *global* yang *responsive* dan komponen *template* formulir seperti pada gambar 3.16 yang mendukung berbagai tipe *input* data. Komponen-komponen ini mencakup *input fields* standar, *date picker* dan *time picker*, *dropdown* dinamis dengan fitur pencarian, hingga *input* untuk *multi-file upload* seperti pada gambar 3.17. Pengembangan komponen *form* ini vital karena menjamin konsistensi *User Experience* (UX) di seluruh sistem, mulai dari halaman pendaftaran hingga

pengajuan *booking*. Pemanfaatan komponen modular dengan standar validasi yang seragam di tahap awal ini mendukung proses *maintenance* dan efisiensi pengembangan fitur-fitur yang lebih kompleks di minggu-minggu berikutnya.



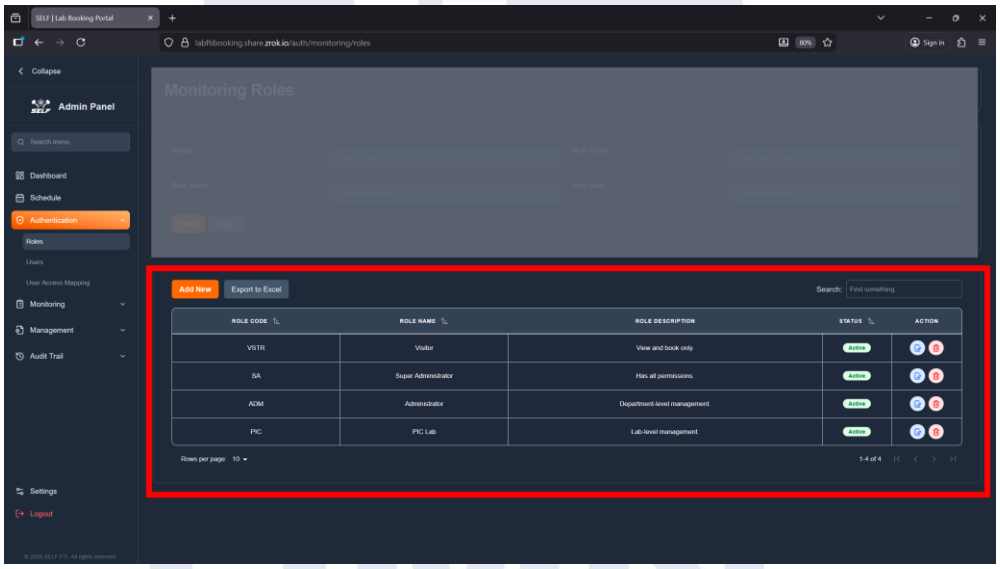
Gambar 3.17 Contoh Datepicker, Dropdown, dan Upload File



Gambar 3.18 Contoh Komponen Notifikasi Global

Sebagai bagian krusial dari komponen global, dirancang dan diimplementasikan serangkaian *modal dialog* terstandarisasi untuk

memberikan umpan balik (*feedback*) dan menangani kesalahan (*error handling*). Pada gambar 3.18 diperlihatkan pola desain ini mencakup berbagai status operasional, yaitu Success (notifikasi penambahan data berhasil), Failed (menangani konflik data seperti kode yang sudah ada), Warning (misalnya, dokumen tidak lengkap), dan Deactivation Confirmation (untuk memvalidasi aksi perubahan status data). Konsistensi visual dan fungsional dari *modal dialog* ini memastikan bahwa setiap interaksi pengguna dengan sistem selalu mendapatkan respons yang jelas dan intuitif, meminimalisir ambiguitas operasional.



Gambar 3.19 Contoh Design Layout Datatable

Melengkapi ekosistem komponen global, dikembangkan pula sebuah arsitektur presentasi data terstandarisasi berupa Datatable Template. Sebagaimana divisualisasikan pada gambar 3.19 , komponen ini menetapkan pola tata letak baku yang akan diterapkan pada seluruh halaman manajemen data dalam panel admin. Struktur tata letak dibagi menjadi tiga segmen strategis: area *filtering* di bagian atas untuk penyaringan data spesifik, bilah kontrol (*control bar*) yang memuat tombol aksi utama (*Add New*, *Export to Excel*) serta fitur pencarian global, dan area *grid* data itu sendiri. Komponen tabel ini telah dilengkapi dengan fungsionalitas interaktif bawaan, meliputi

pengurutan kolom (*sorting*), indikator status visual (*badges*), tombol aksi baris (*edit/delete*), serta navigasi halaman (*pagination*). Penerapan templat tunggal ini menjamin konsistensi visual di seluruh modul aplikasi sekaligus mereduksi redundansi kode secara signifikan saat pengembangan fitur manajemen data selanjutnya.

Memasuki minggu kedua bulan Oktober, kegiatan pengembangan dialihkan pada implementasi logika interaksi data di sisi *frontend* serta inisiasi infrastruktur di sisi *server*. Fokus utama pada periode ini adalah menciptakan efisiensi dalam pengelolaan data melalui standardisasi mekanisme *input* dan penyaringan (*filtering*), serta mempersiapkan landasan integrasi sistem.

Melanjutkan pengembangan komponen dasar, dilakukan perakitan komponen menjadi Templat Formulir dan Mekanisme Filter yang utuh untuk halaman *monitoring*. Templat formulir dirancang sebagai kesatuan antarmuka (*interface wrapper*) yang menggabungkan validasi dan tata letak responsif untuk fitur *Create* dan *Update*. Secara spesifik, logika penyaringan (*filtering*) pada *datatable* diimplementasikan menggunakan pendekatan *server-side*, selaras dengan mekanisme paginasi yang telah diterapkan. Dalam skema ini, proses penyaringan tidak membebani memori peramban (*client-side*), melainkan bekerja dengan memicu pemanggilan ulang (*re-fetching*) API ke *server* (metode GET) saat pengguna menerapkan filter. Permintaan data baru tersebut dikirimkan dengan menyertakan parameter *query* tambahan yang spesifik, sehingga *server* hanya mengembalikan himpunan data yang relevan. Pendekatan ini memastikan performa aplikasi tetap optimal dan responsif meskipun menangani volume data yang besar.

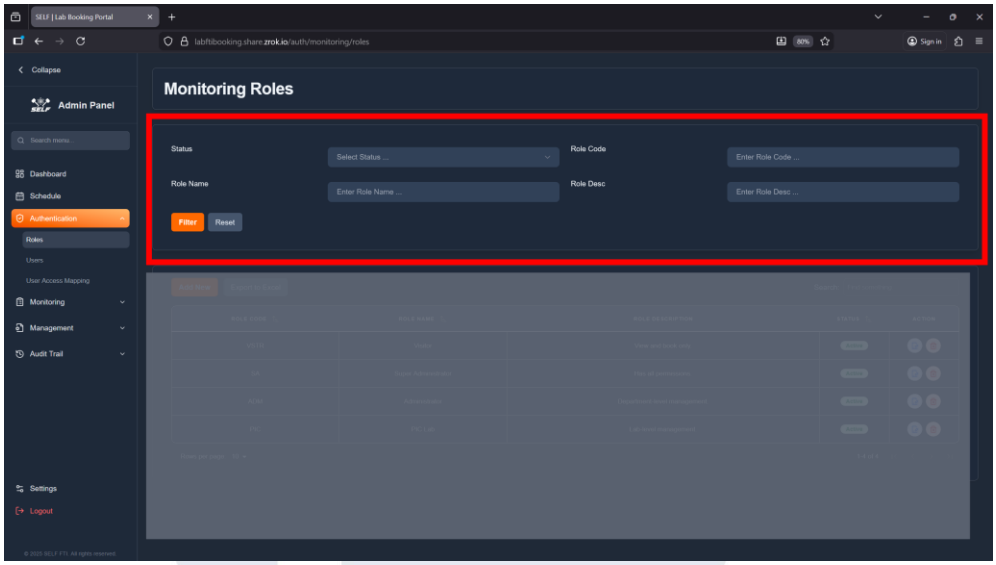
Secara paralel, dilakukan penyiapan lingkungan pengembangan di sisi *backend* sebagai persiapan menuju tahap integrasi. Kegiatan ini mencakup inisiasi proyek menggunakan kerangka kerja FastAPI berbasis Python, konfigurasi *virtual environment* untuk isolasi dependensi (*dependencies*), serta penyusunan struktur direktori awal yang mengadopsi pola arsitektur

Model-View-Controller (MVC). Tahap ini juga melibatkan konfigurasi koneksi dasar ke basis data MySQL serta penyiapan *CORS (Cross-Origin Resource Sharing) middleware* untuk mengizinkan komunikasi data yang aman antara aplikasi *client* (Next.js) dan *server*. Hasil akhir dari aktivitas ini adalah tersedianya *environment backend* yang stabil dan siap untuk menerima implementasi logika bisnis (*API Endpoints*) pada minggu berikutnya.

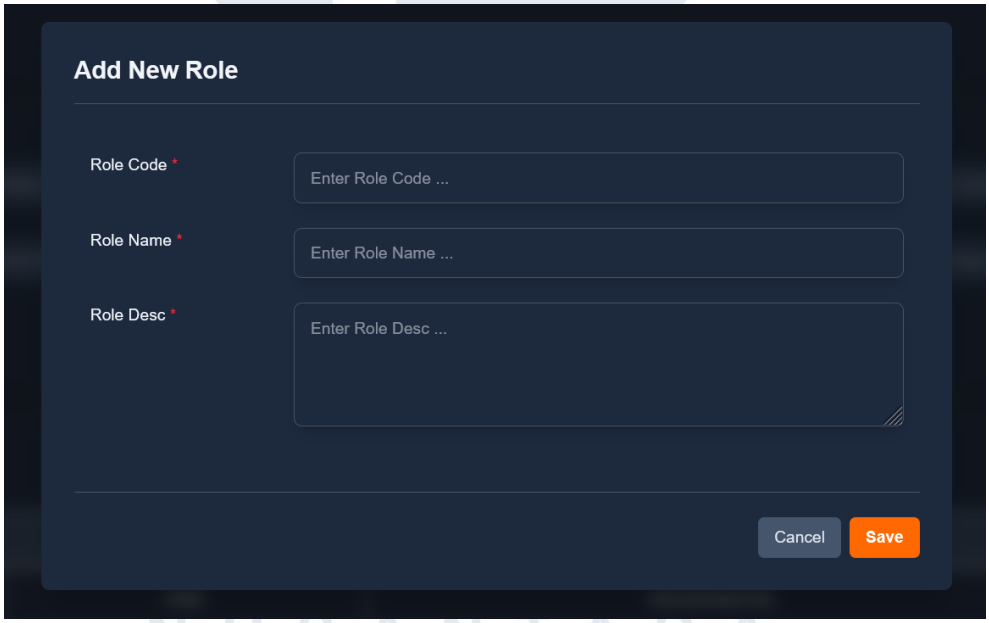
Server-side filtering dan *pagination* merupakan keputusan arsitektural strategis untuk menjamin performa aplikasi saat menangani volume data yang besar. Secara teoritis, memuat data dalam potongan kecil (*chunks*) alih-alih keseluruhan dataset secara signifikan mengurangi penggunaan memori pada sisi klien, yang krusial untuk mencegah penurunan kinerja (*slowdowns*) atau kegagalan aplikasi (*crashes*) pada perangkat pengguna dengan sumber daya terbatas [22]. Lebih lanjut, pendekatan ini mengoptimalkan efisiensi *bandwidth* jaringan dan mengurangi waktu respons API (*response time*), karena server hanya memproses dan mengirimkan subset data yang diminta oleh pengguna [22].Hal ini sejalan dengan prinsip desain sistem terdistribusi yang memprioritaskan konsistensi data dan skalabilitas, di mana pemrosesan logika penyaringan yang kompleks didelegasikan ke *server* untuk menjaga antarmuka pengguna tetap responsif dan ringan [22].

Implementasi antarmuka untuk logika penyaringan *server-side* tersebut divisualisasikan secara konkret pada gambar 3.20. Bagian yang ditandai (kotak merah) menunjukkan panel filter yang dirancang terintegrasi di atas tabel data, menyediakan parameter pencarian spesifik yang relevan dengan entitas yang dikelola, seperti *Role Code*, *Role Name*, deskripsi, serta *Status* keaktifan data. Panel ini dilengkapi dengan tombol aksi "Filter" yang berfungsi sebagai pemicu (*trigger*) pengiriman parameter *query* ke API *backend*, serta tombol "Reset" untuk membersihkan seluruh *input* pencarian dan memuat ulang data ke kondisi awal (*default state*). Desain antarmuka ini memastikan pengguna memiliki kendali penuh untuk menyaring informasi

secara presisi tanpa perlu memuat ulang keseluruhan halaman (*full page reload*), meningkatkan efisiensi operasional dalam pengelolaan data master.



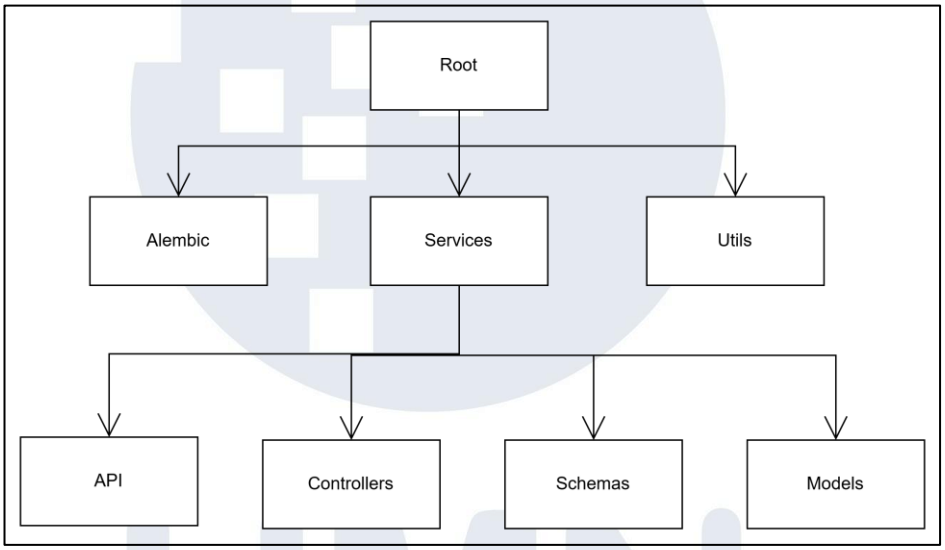
Gambar 3.20 Contoh Design Filter Untuk Datatable



Gambar 3.21 Contoh Template Form

Penerapan template formulir yang telah dikembangkan divisualisasikan lebih lanjut melalui antarmuka *input* data master pada gambar 3.21, sebagaimana ditampilkan pada formulir *Add New Role*. Komponen ini mengadopsi pola interaksi berbasis *modal dialog* yang memungkinkan

pengguna melakukan entri data tanpa kehilangan konteks navigasi halaman utama (*context switching*). Struktur formulir dirancang dengan standar validasi visuale yang jelas, di mana atribut wajib (*mandatory fields*) seperti *Role Code* dan *Role Name* ditandai dengan indikator asterisk merah, meminimalisir kesalahan *input* sebelum data dikirim ke *server*. Selain itu, tata letak elemen kontrol (tombol *Cancel* dan *Save*) ditempatkan secara konsisten untuk menjaga intuitifitas pengalaman pengguna di seluruh modul manajemen data.



Gambar 3.22 Struktur Folder Backend

Sebagai manifestasi teknis dari arsitektur yang diusung, struktur direktori *backend* diorganisasikan secara modular guna menjamin skalabilitas dan keterbacaan kode. Merujuk pada diagram struktur proyek, inisiasi *root* direktori membagi sistem ke dalam tiga komponen utama. Direktori Alembic difungsikan khusus untuk manajemen versi dan migrasi skema basis data, sedangkan direktori Utils disiapkan sebagai wadah fungsi pendukung (*helper*), seperti layanan pengiriman email dan *database seeder*. Inti fungsionalitas aplikasi dipusatkan di dalam direktori Services, yang secara internal mengelompokkan logika sistem menjadi empat lapisan terpisah.

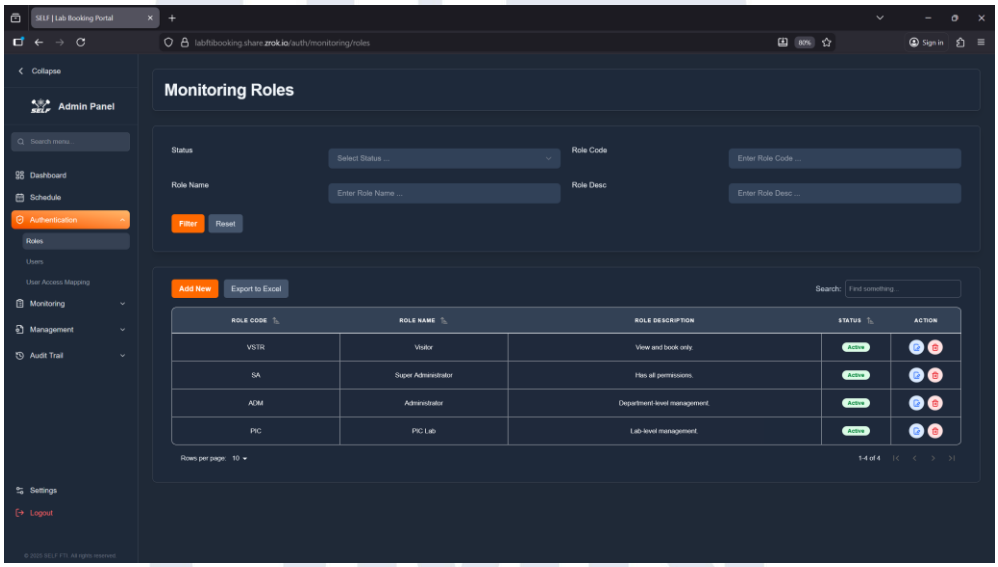
Lapisan pertama adalah direktori API yang bertugas mendefinisikan rute *endpoint* (*routes*), diikuti oleh Controllers untuk menangani eksekusi logika bisnis. Selanjutnya, terdapat direktori Models yang berfungsi sebagai lapisan abstraksi data menggunakan teknik *Object-Relational Mapping* (ORM). Penerapan ORM ini tidak hanya memetakan tabel basis data ke dalam bentuk objek, tetapi juga mengisolasi logika akses data dari logika bisnis, sebuah pola yang menurut [23] dapat meningkatkan produktivitas pengembang dan *maintainability* karena meminimalkan ketergantungan pada kueri SQL mentah.

Terakhir, direktori Schemas diimplementasikan untuk menangani validasi struktur data menggunakan pustaka Pydantic. Secara teoritis, lapisan ini mengadopsi pola *Data Transfer Object* (DTO) yang bertujuan untuk memisahkan representasi data internal (basis data) dari antarmuka eksternal (API). Dengan menegakkan validasi tipe data yang ketat (*strict type checking*) dan sanitasi input pada lapisan ini, sistem dapat mencegah anomali data serta memastikan bahwa hanya data yang memenuhi kontrak skema yang dapat diproses lebih lanjut, menjamin integritas dan keamanan aplikasi secara keseluruhan. Penataan hierarkis ini memastikan pemisahan tanggung jawab (*separation of concerns*) yang tegas, memfasilitasi pemeliharaan kode dan pengujian yang terisolasi [24].

Dalam konteks manajemen siklus hidup pengembangan sistem, keberadaan direktori Alembic mengimplementasikan konsep *Database Version Control* yang selaras dengan paradigma *Evolutionary Database Design*. Mekanisme ini mentransformasi cara pengelolaan skema basis data dari pendekatan manual menjadi berbasis kode (*code-based*), di mana setiap perubahan struktur seperti penambahan tabel atau modifikasi kolom dicatat sebagai skrip migrasi yang terurut secara kronologis. Penerapan migrasi basis data otomatis sangat krusial untuk menjaga konsistensi skema di berbagai lingkungan (*environment*), mulai dari pengembangan lokal (*development*) hingga produksi. Praktik ini tidak hanya menciptakan jejak audit (*audit trail*)

yang transparan atas evolusi basis data, tetapi juga memfasilitasi kemampuan *rollback* (pengembalian ke versi sebelumnya) yang aman. Hal ini secara efektif memitigasi risiko *configuration drift*, yaitu ketidaksesuaian skema antar lingkungan yang sering menjadi penyebab utama kegagalan *deployment* pada aplikasi berskala besar.

Memasuki minggu ketiga bulan Oktober, fase pengembangan bergerak ke tahap integrasi penuh antara antarmuka dan layanan *server*. Fokus utama pada periode ini adalah merealisasikan manajemen data master serta membangun sistem keamanan aplikasi melalui fitur otentikasi.

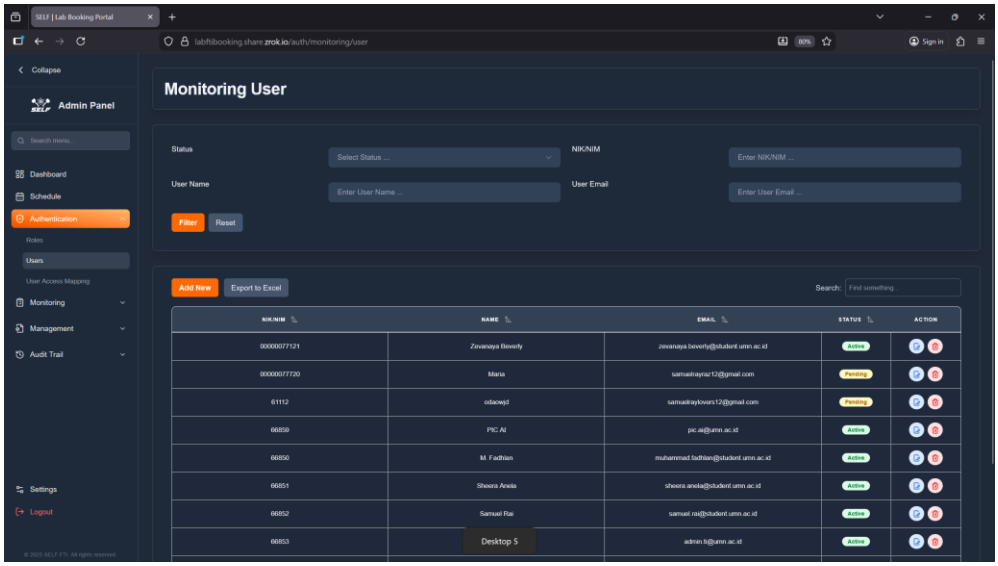


Gambar 3.23 Halaman Roles

Sebagai langkah awal integrasi, dihasilkan halaman manajemen data yang fungsional sepenuhnya, sebagaimana divisualisasikan pada halaman *Monitoring Roles* pada gambar 3.23. Antarmuka ini menggabungkan seluruh elemen yang telah dikembangkan sebelumnya navigasi, panel penyaringan data, dan tabel data interaktif ke dalam satu ekosistem visual yang kohesif. Pada halaman ini, administrator dapat memantau daftar peran (*roles*) yang terdaftar dalam sistem seperti *Super Administrator* (SA), *Administrator* (ADM), hingga *PIC Lab* lengkap dengan atribut deskriptif dan status keaktifannya. Keberhasilan perakitan halaman ini menandai dimulainya

implementasi modul manajemen data master secara menyeluruh, yang kemudian direplikasi untuk entitas lain meliputi Pengguna (*Users*), Pemetaan Akses (*User Access Mapping*), serta struktur organisasi (*Departments* dan *Labs*), sesuai dengan target luaran pada tabel kegiatan.

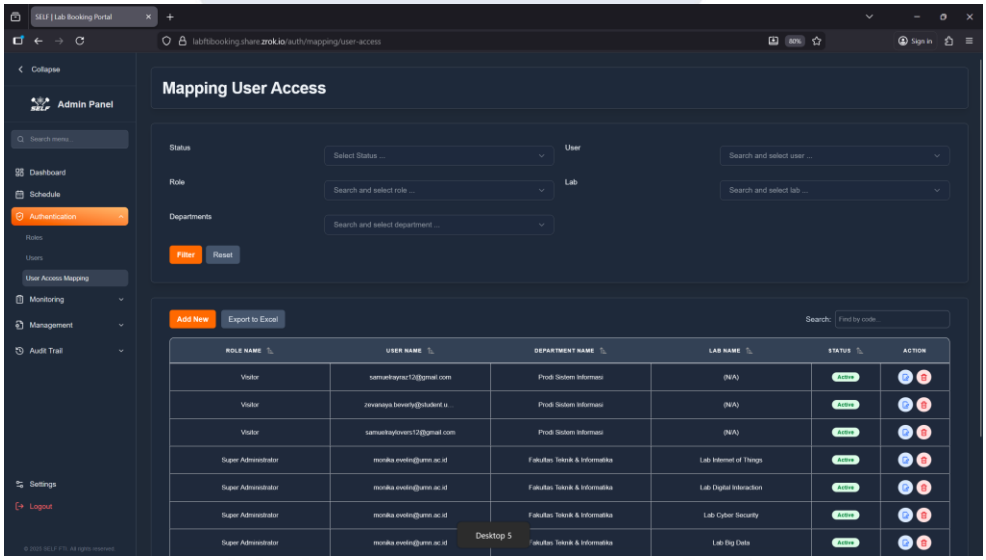
Dalam pengelolaan siklus hidup data, sistem menerapkan strategi *Soft Delete* (penghapusan logis) alih-alih penghapusan permanen (*Hard Delete*). Mekanisme ini bekerja dengan memutakhirkan status data dari *Active* menjadi *Inactive* saat tombol hapus dieksekusi, sebagaimana divalidasi melalui dialog konfirmasi deaktivasi. Implikasi dari pendekatan ini adalah data "terhapus" tetap tersimpan dalam basis data dan terlihat di panel admin dengan status *Inactive*, memungkinkan administrator untuk melakukan audit atau memulihkan data tersebut (*reactivate*) melalui fitur penyuntingan (*edit*) jika diperlukan. Namun, sistem secara otomatis memfilter data berstatus *Inactive* agar tidak muncul pada antarmuka pengguna (*Client App*) dan memblokir penggunaannya dalam transaksi operasional baru, menjamin konsistensi data tanpa risiko kehilangan informasi historis yang vital.



Gambar 3.24 Halaman Manajemen User

Perluasan implementasi data master dilakukan pada entitas Pengguna (*Users*), bisa dilihat pada gambar 3.24 di mana sistem memfasilitasi

administrasi akun secara terpusat. Pada modul ini, administrator memiliki kewenangan untuk mendaftarkan pengguna baru secara manual. Proses ini memicu layanan *backend* untuk mengirimkan notifikasi email otomatis ke alamat yang didaftarkan, berisi tautan aman bagi pengguna untuk menetapkan kata sandi mereka sendiri (*set password*). Sistem juga menerapkan logika siklus hidup akun yang ketat melalui empat status visual status Pending menandakan pengguna baru yang sedang dalam tahap verifikasi atau belum melakukan aktivasi status Active mengindikasikan akun telah terverifikasi dan memiliki hak akses penuh untuk *login*; status Expired diterapkan secara otomatis apabila batas waktu verifikasi telah terlampaui tanpa aktivitas serta status Inactive atau Deactivated yang menunjukkan bahwa akses akun telah dicabut oleh administrator, meskipun data historisnya tetap tersimpan dalam sistem.

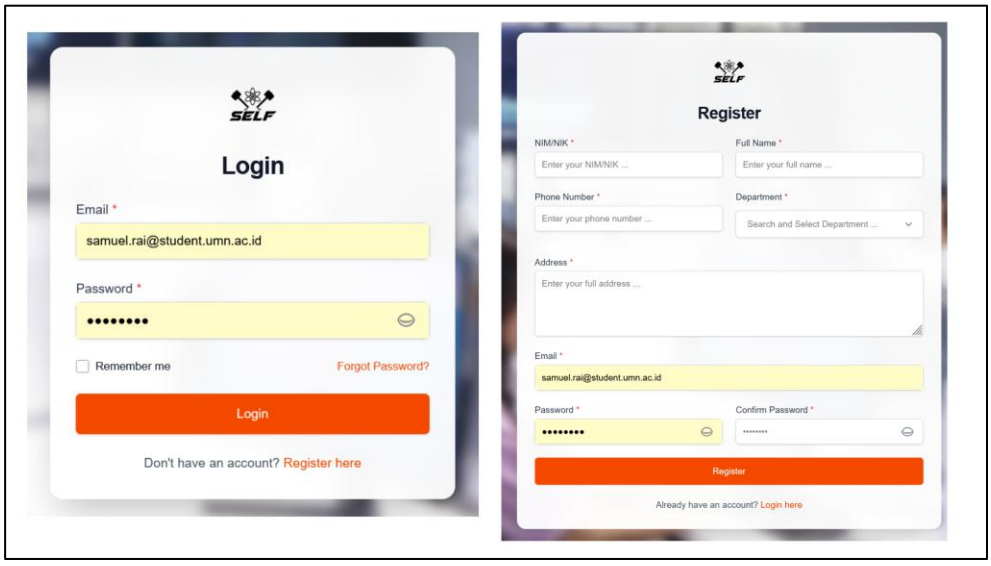


Gambar 3.25 Halaman User Access Mapping

Melengkapi infrastruktur manajemen pengguna, dikembangkan modul *User Access Mapping* yang berfungsi sebagai pusat kendali otorisasi sistem. Sebagaimana ditampilkan pada antarmuka pemetaan akses, modul ini memfasilitasi administrator untuk mendefinisikan hubungan relasional antara entitas Pengguna, Peran (*Role*), Departemen, dan Laboratorium. Mekanisme

ini merupakan implementasi teknis dari strategi *Role-Based Access Control* (RBAC) yang bersifat granular, di mana hak akses pengguna tidak hanya ditentukan oleh jabatannya, tetapi juga dibatasi oleh lingkup operasionalnya (misalnya, seorang pengguna dapat memiliki peran 'Super Administrator' spesifik untuk 'Lab Internet of Things'). Antarmuka ini dilengkapi dengan fitur penyaringan multidimensi yang memungkinkan pencarian cepat berdasarkan kombinasi atribut akses, memastikan pengelolaan hak akses yang presisi dan transparan.

Setelah manajemen data master terbentuk, pengembangan dilanjutkan pada lapisan keamanan sistem melalui pengerjaan halaman Authentication. Fitur ini mencakup antarmuka dan logika untuk *Login*, *Register*, dan *Forgot Password*. Implementasi ini mengintegrasikan formulir *frontend* dengan *endpoint* API *backend* untuk memvalidasi kredensial pengguna dan menerbitkan token akses (JWT). Hal ini memastikan bahwa hanya pengguna yang terdaftar dan terverifikasi yang dapat mengakses halaman *dashboard* dan menu internal lainnya, menegakkan prinsip keamanan akses tertutup.



Gambar 3.26 Form Login dan Register

Setelah manajemen data master terbentuk, pengembangan dilanjutkan pada lapisan keamanan sistem melalui pengerjaan halaman Authentication.

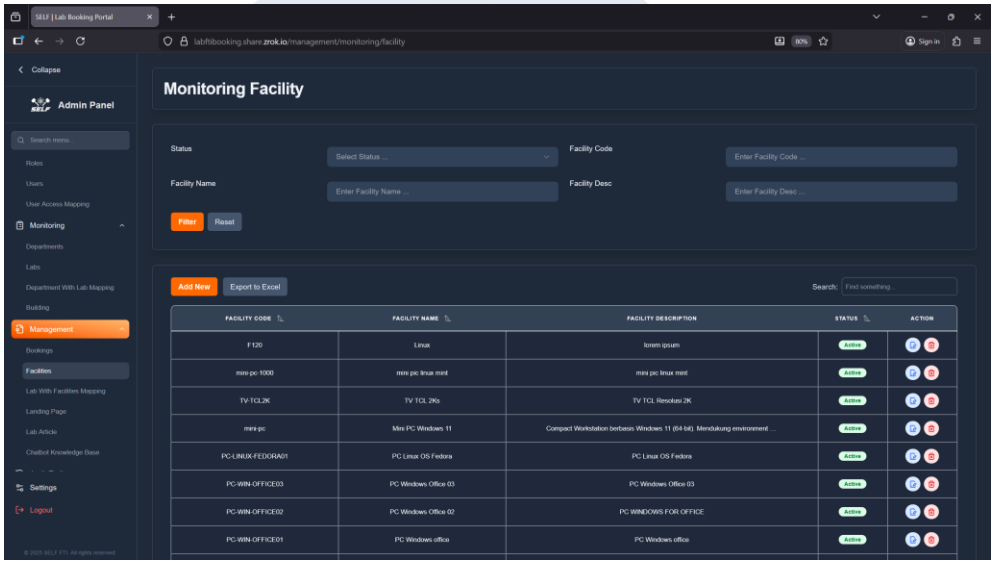
Implementasi modul ini divisualisasikan melalui antarmuka Login dan Register pada gambar 3.26 yang dirancang responsif dan aman. Pada halaman *Login*, pengguna diwajibkan memasukkan kredensial email dan kata sandi yang terenkripsi untuk mendapatkan token sesi (JWT). Sementara itu, halaman *Register* memfasilitasi pendaftaran pengguna mandiri dengan formulir isian lengkap meliputi Nomor Induk (NIM/NIK), Nama Lengkap, Nomor Telepon, Departemen, hingga Alamat Domisili. Kedua antarmuka ini dilengkapi dengan validasi *input* sisi klien (*client-side validation*) untuk memastikan kelengkapan data sebelum permintaan otorisasi dikirimkan ke server, serta fitur *toggle visibility* pada kolom kata sandi untuk kenyamanan pengguna.

Secara paralel di sisi *backend*, dilakukan pengembangan logika bisnis lanjutan berupa Scheduler dan sistem Email Notification. Fitur *scheduler* dirancang untuk menjalankan tugas-tugas otomatis terjadwal, seperti pengecekan status peminjaman yang kedaluwarsa atau mengubah status akun *Pending* menjadi *Expired*. Sementara layanan notifikasi *email* diintegrasikan untuk memberikan umpan balik *real-time* kepada pengguna terkait aktivitas akun (seperti verifikasi pendaftaran atau reset kata sandi).

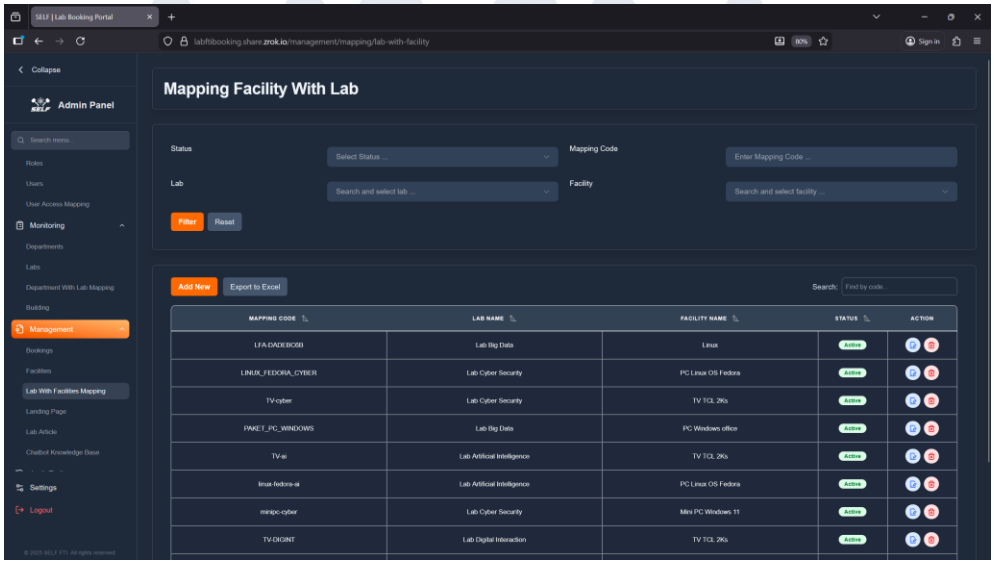
Minggu terakhir bulan Oktober didedikasikan untuk pengembangan logika bisnis inti (*core business logic*) yang menjadi tujuan utama pembuatan sistem. Kegiatan difokuskan pada realisasi modul manajemen inventaris fasilitas serta mekanisme transaksi peminjaman ruangan, diiringi dengan proses penyempurnaan kualitas antarmuka (*interface polishing*) secara menyeluruh.

Pengembangan dimulai dengan fitur pengelolaan data fasilitas laboratorium sebagai entitas pendukung utama kegiatan penelitian. Sebagaimana divisualisasikan pada gambar 3.27, modul ini menyediakan antarmuka terpusat bagi administrator untuk menginventarisasi aset fisik laboratorium. Tabel data dirancang untuk menampilkan informasi krusial

seperti Kode Fasilitas, Nama Aset (misalnya "PC-LINUX-FEDORA01" atau "TV TCL 2K"), serta deskripsi spesifikasi teknisnya. Halaman ini juga dilengkapi panel penyaringan (*filter*) di bagian atas yang memungkinkan pencarian aset berdasarkan status ketersediaan atau kata kunci tertentu, memastikan pengelolaan ratusan aset laboratorium dapat dilakukan dengan efisien dan terorganisir.



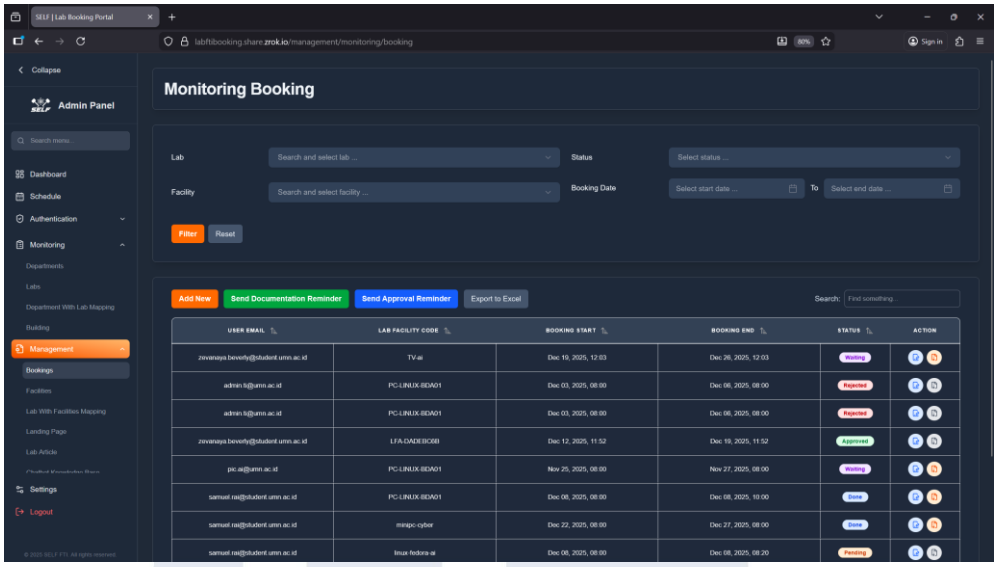
Gambar 3.27 Halaman Facility Management



Gambar 3.28 Halaman Mapping Facility dengan Lab

Melengkapi inventarisasi fisik, pada gambar 3.28 dikembangkan modul relasional Mapping Facility With Lab untuk menetapkan lokasi spesifik dari setiap aset. Antarmuka ini berfungsi sebagai jembatan logis yang menghubungkan entitas *Facility* dengan entitas *Lab*. Melalui fitur ini, administrator dapat memetakan bahwa fasilitas tertentu (misalnya "PC Linux OS Fedora") terinstalasi secara fisik di laboratorium spesifik (misalnya "Lab Cyber Security"). Tabel pemetaan menampilkan kolom *Mapping Code*, *Lab Name*, dan *Facility Name* secara berdampingan, memberikan visibilitas total mengenai distribusi aset di seluruh gedung. Mekanisme ini krusial untuk memastikan akurasi data saat pengguna melihat detail fasilitas yang tersedia pada saat hendak meminjam ruangan tertentu.

Di tengah pengembangan fitur inti, dilakukan serangkaian perbaikan teknis (*code refactoring*) untuk menjamin kenyamanan pengguna (*User Experience*) dan stabilitas sistem. Fokus perbaikan diarahkan pada aspek responsivitas antarmuka (*responsive design*), memastikan seluruh halaman panel admin yang telah dibuat dapat menyesuaikan tampilan secara adaptif pada berbagai ukuran layar perangkat. Selain itu, dilakukan *debugging* mendalam pada modul otentikasi untuk menangani kasus-kasus tepi (*edge cases*) seperti penanganan sesi kadaluarsa (*session timeout*) dan validasi input guna meningkatkan keamanan proses *login/register* sebelum sistem menangani transaksi peminjaman yang krusial.



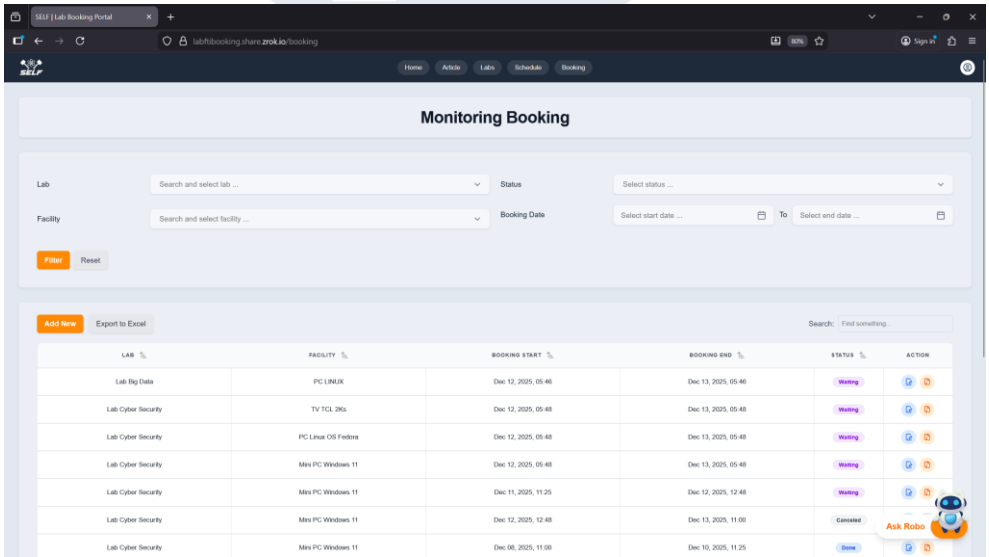
Gambar 3.29 Halaman Monitoring Booking Admin Panel

Dengan infrastruktur yang lebih stabil, pengembangan berlanjut ke pusat kendali operasional, yaitu halaman *Monitoring Booking* di panel admin. Seperti pada gambar 3.29 halaman ini memberikan visibilitas total serta kontrol penuh kepada administrator. Selain memantau, admin memiliki privilese untuk membuat peminjaman baru (*Add New*) secara manual guna mengakomodasi kebutuhan internal. Fitur komunikasi sistem dirancang fleksibel melalui dua mekanisme notifikasi: tombol aksi massal (*bulk action*) yang terletak pada *toolbar* di atas tabel data untuk mengirimkan *Documentation/Approval Reminder* ke banyak pengguna sekaligus, serta opsi pengiriman spesifik per transaksi yang tersedia di dalam halaman detail peminjaman. Fleksibilitas ini dirancang untuk meningkatkan efisiensi administrator dalam menindaklanjuti status peminjaman tanpa harus memproses data satu per satu.

Menutup kegiatan bulan Oktober, dilakukan implementasi logika bisnis (*backend logic*) guna menangani siklus hidup peminjaman yang kompleks melalui pendekatan *State Machine Workflow*, di mana antarmuka penggunanya akan direalisasikan sepenuhnya pada minggu berikutnya. Alur kerja otomatis ini dirancang secara berurutan, dimulai dari tahap *Submission* dengan status awal *Waiting/Pending* saat pengajuan, yang kemudian beralih

menjadi *Approved* setelah melalui validasi administrator. Selanjutnya, mekanisme otomasi melalui *Scheduler* akan memantau durasi peminjaman dan secara otomatis mengubah status menjadi *Waiting for Documentation* ketika masa pinjam berakhir. Siklus ini ditutup pada tahap *Completion*, di mana sistem mewajibkan pengguna untuk mengunggah bukti pertanggungjawaban berupa dua foto dokumentasi kegiatan dan satu dokumen hasil penelitian (PDF) guna mencapai status final *Done*. Pematangan logika ini menjadi fondasi krusial sebelum tim beralih ke pengembangan antarmuka sisi klien (*Client Side*) di awal bulan November.

Memasuki bulan November, orientasi pengembangan beralih dari sisi administrator ke sisi pengguna akhir (*Client-Side*). Fase ini bertujuan untuk menyediakan antarmuka publik yang informatif, responsif, dan mudah diakses bagi mahasiswa maupun dosen yang akan menggunakan layanan laboratorium.

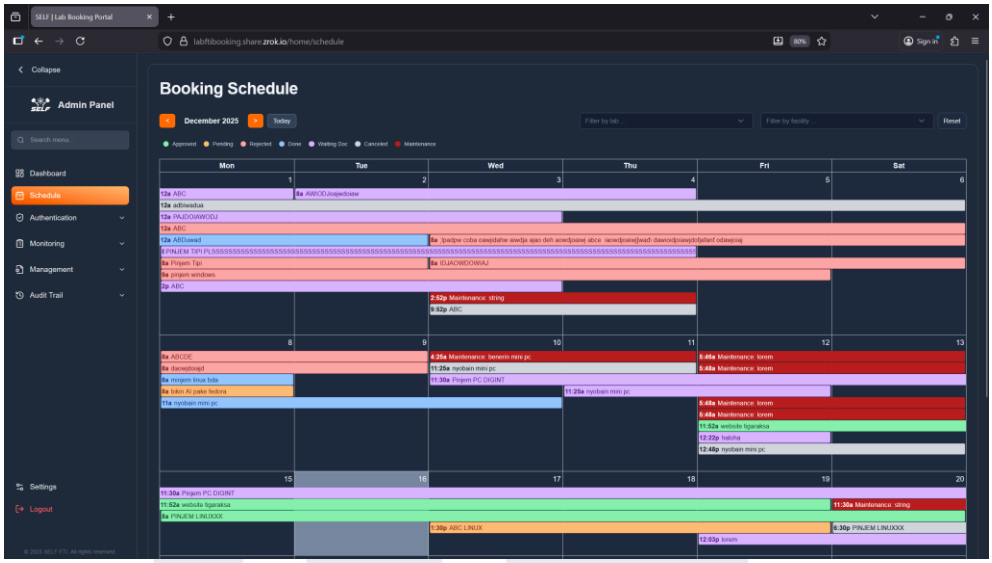


Gambar 3.30 Halaman Monitoring Booking Sisi User

Sebagai fitur utama layanan mandiri, dikembangkan halaman Monitoring Booking pada antarmuka pengguna. Pada gambar 3.30 diperlihatkan berbeda dengan panel administrator yang bersifat manajerial, halaman ini dirancang dengan logika isolasi data yang ketat (*data isolation*)

pengguna hanya diizinkan melihat riwayat dan status peminjaman yang mereka ajukan sendiri. Antarmuka ini berfungsi sebagai papan informasi personal di mana pengguna dapat melacak progres pengajuan apakah masih berstatus *Waiting*, telah *Approved*, atau *Cancelled* tanpa memiliki kewenangan untuk mengubah status tersebut. Hal ini kontras dengan logika pada Admin Panel, di mana administrator memiliki akses global (sesuai *Role-Based Access Control*) untuk melihat seluruh transaksi peminjaman pengguna lain dan memiliki hak eksekusi untuk melakukan persetujuan (*approval*) atau penolakan (*rejection*). Pembatasan fitur di sisi pengguna ini bertujuan untuk menjaga integritas alur kerja persetujuan yang terpusat hanya pada pengelola laboratorium.

Untuk kebutuhan pengawasan operasional, dikembangkan fitur Admin Schedule yang menyajikan transparansi data secara total. Kalender pada gambar 3.31 ini dikonfigurasi untuk menampilkan tujuh status peminjaman secara lengkap yang dibedakan melalui kode warna visual: *Approved* (Hijau), *Pending* (Oranye), *Rejected* (Merah Muda), *Done* (Biru), *Waiting Doc* (Ungu), *Canceled* (Abu-abu), hingga *Maintenance* (Merah Tua). Tingkat granularitas informasi ini memungkinkan administrator untuk memantau kepadatan laboratorium secara menyeluruh, mendeteksi potensi bentrokan jadwal (*conflicts*) yang melibatkan pengajuan berstatus *Pending*, serta melacak aktivitas pemeliharaan fasilitas dalam satu tampilan terintegrasi.



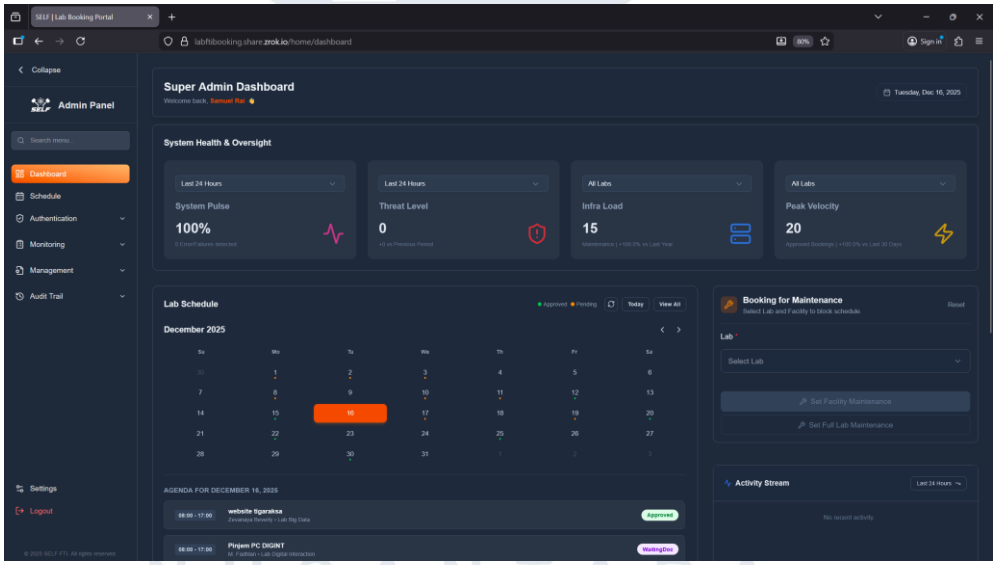
Gambar 3.31 Halaman Schedule Sisi Admin Panel

Sebaliknya, pada sisi antarmuka publik, fitur User Schedule menerapkan mekanisme penyaringan data yang ketat demi menjaga privasi dan kejelasan informasi. Sistem secara otomatis memfilter data sehingga pengguna hanya dapat melihat slot waktu dengan status Approved (Hijau) dan Maintenance (Merah). Pendekatan *Privacy by Design* ini bertujuan untuk menyembunyikan data internal yang sensitif seperti pengajuan yang ditolak atau masih dalam antrean persetujuan sekaligus memberikan kepastian visual kepada pengguna mengenai slot waktu mana yang benar-benar "Terisi" atau "Tidak Tersedia", sehingga proses pencarian jadwal kosong dapat dilakukan dengan lebih efisien.

Bersamaan dengan penyelesaian antarmuka pengguna, dilakukan peningkatan aspek ergonomi visual pada sisi administrator melalui fitur Pengaturan Tema. Modul ini memfasilitasi administrator untuk mengubah preferensi tampilan antarmuka antara Light Mode dan Dark Mode secara instan. Implementasi ini krusial untuk mengurangi ketegangan mata (*eye strain*) saat pengelola melakukan pemantauan data dalam durasi panjang, memberikan fleksibilitas lingkungan kerja yang adaptif terhadap kondisi pencahayaan.

Memasuki minggu kedua bulan November, kegiatan pengembangan diarahkan pada dua sektor vital penyediaan portal informasi terpadu bagi pengguna umum dan pembangunan papan instrumen (*dashboard*) analitik bagi pengelola sistem. Tahap ini bertujuan untuk meningkatkan transparansi informasi laboratorium serta menyediakan visualisasi data strategis yang kontekstual.

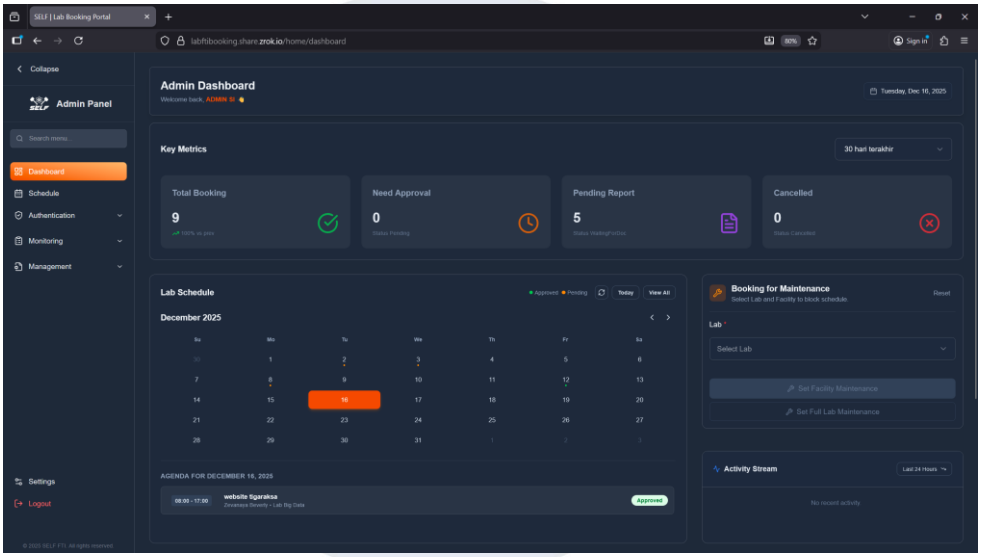
Sesuai dengan target luaran kegiatan, dikembangkan antarmuka sisi klien yang berfungsi sebagai pusat informasi statis dan dinamis. Implementasi mencakup tiga halaman utama Landing Page sebagai gerbang visual, Halaman Labs yang menyajikan katalog fasilitas, dan Halaman Artikel. Fitur Artikel dirancang sebagai kanal komunikasi resmi untuk mempublikasikan berita terbaru atau pengumuman pemeliharaan, memastikan pengguna memiliki akses terhadap informasi yang akurat sebelum melakukan peminjaman.



Gambar 3.32 Halaman Dashboard Superadmin

Pada tingkat manajerial tertinggi, seperti gambar 3.32 dikembangkan Super Admin Dashboard yang memberikan wawasan makro mengenai kesehatan sistem. Dashboard ini dilengkapi panel *System Health & Oversight* yang menampilkan metrik teknis krusial seperti *System Pulse* (stabilitas

server), *Threat Level* (keamanan), dan *Infra Load*. Selain itu, Super Admin memiliki akses penuh terhadap fitur Booking for Maintenance, memungkinkan pemblokiran jadwal laboratorium secara paksa jika terjadi insiden mendadak, serta melihat statistik global seluruh laboratorium tanpa batasan lingkup (*scope*).

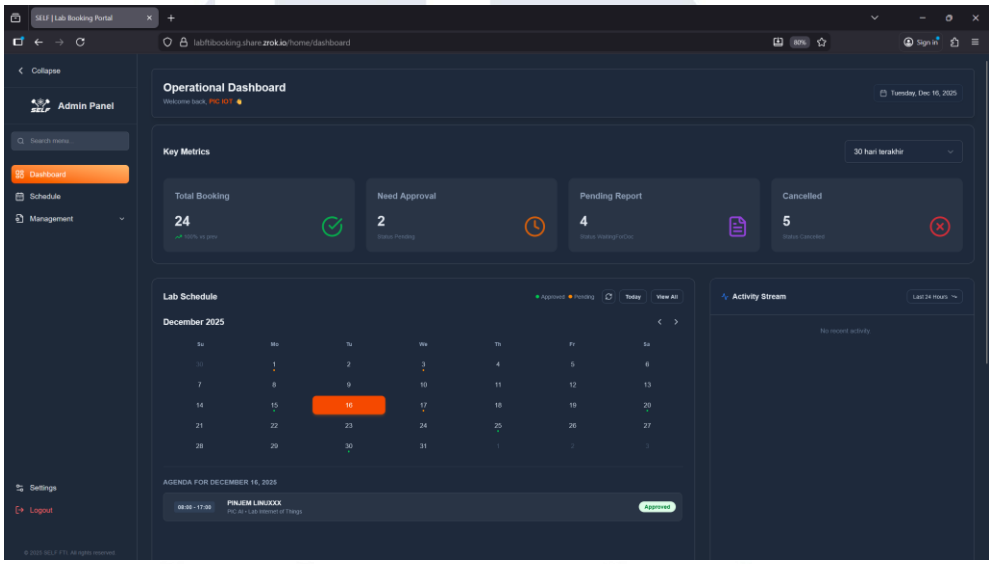


Gambar 3.33 Halaman Dashboard Admin

Berbeda dengan Super Admin, antarmuka Admin Dashboard difokuskan murni pada metrik operasional peminjaman tanpa menampilkan status teknis server seperti pada gambar 3.33. Halaman ini menyajikan widget statistik utama seperti *Total Booking*, *Need Approval* (menunggu persetujuan), dan *Pending Report* untuk lingkup departemen yang dikelola. Fitur pembeda utama pada level ini adalah administrator masih memiliki kewenangan untuk mengakses panel Booking for Maintenance di sisi kanan layar, memberikan fleksibilitas bagi manajer departemen untuk menutup sementara akses laboratorium di bawah naungannya demi keperluan perawatan atau perbaikan fasilitas.

Pada tingkatan operasional paling dasar, PIC Dashboard menerapkan logika lingkup data (*data scoping*) yang paling ketat. Antarmuka pada gambar 3.34 ini serupa dengan dashboard admin dalam hal penyajian metrik statistik

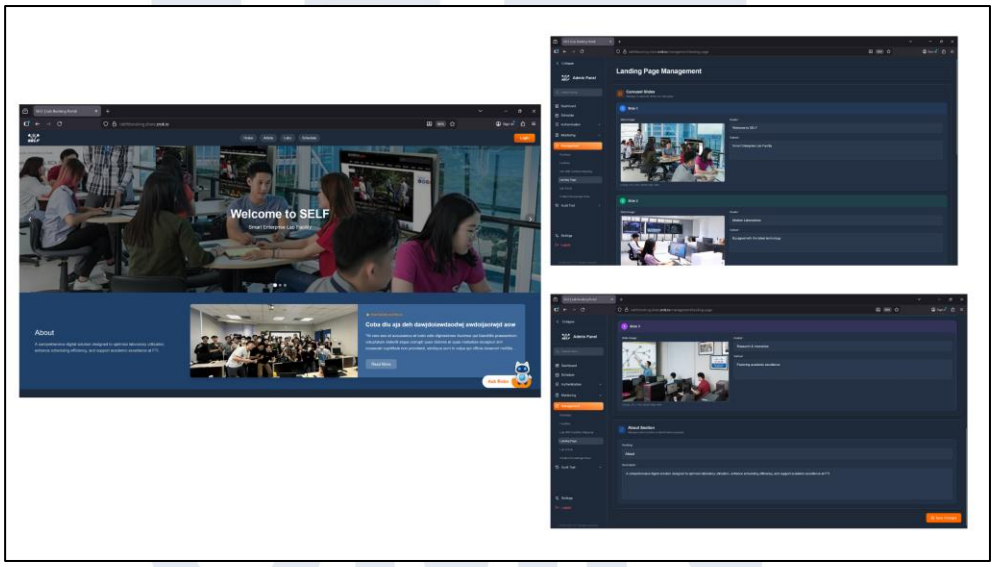
(*Key Metrics*), namun data yang ditampilkan secara otomatis difilter hanya untuk laboratorium yang menjadi tanggung jawab PIC tersebut (misalnya, hanya data untuk 'PIC AI'). Perbedaan signifikan terletak pada hilangnya panel kontrol *Booking for Maintenance*; hal ini menegaskan batasan wewenang di mana seorang PIC hanya bertugas memantau permohonan masuk dan memastikan kelancaran operasional harian tanpa hak untuk mengubah ketersediaan jadwal laboratorium secara sepihak.



Gambar 3.34 Halaman Dashboard PIC

Sebagai titik sentuh pertama (*first touchpoint*) pengguna dengan sistem, dikembangkan Landing Page yang dirancang dengan estetika modern dan profesional. Diperlihatkan pada gambar 3.35 ini mengusung struktur *Hero Section* dinamis yang menampilkan slogan "Smart Enterprise Lab Facility", diikuti oleh bagian *About* yang memberikan ringkasan visi misi laboratorium, serta *Featured Article* untuk menyoroti berita terkini. Implementasi antarmuka ini tidak hanya berfokus pada keindahan visual semata, tetapi juga pada navigasi yang intuitif, memandu pengguna untuk mengakses menu utama seperti *Labs*, *Schedule*, dan integrasi asisten virtual *Ask Robo* dengan mudah.

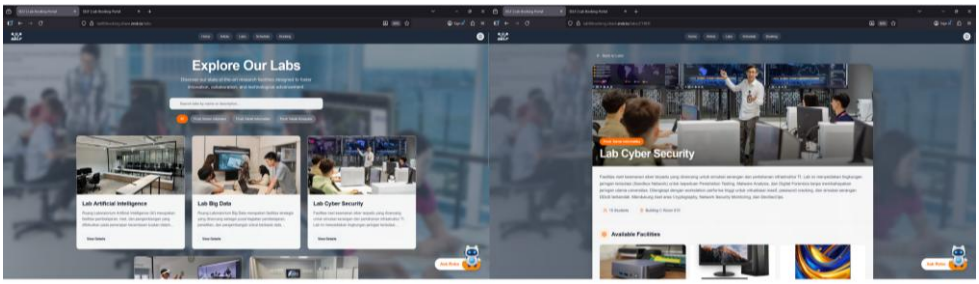
Untuk menjamin fleksibilitas informasi tanpa ketergantungan pada tim teknis, dikembangkan modul Landing Page Management di sisi administrator. Modul ini berfungsi layaknya *Content Management System* (CMS) sederhana yang memungkinkan administrator untuk memutakhirkan konten beranda secara *real-time*. Melalui antarmuka formulir yang ramah pengguna, pengelola dapat mengganti gambar *carousel slides*, menyunting teks tajuk (*header*) dan sub-teks, serta memperbarui deskripsi pada bagian *About Section*. Mekanisme ini memastikan bahwa informasi yang ditampilkan kepada publik selalu relevan dan dapat disesuaikan dengan agenda atau kampanye terbaru laboratorium secara mandiri.



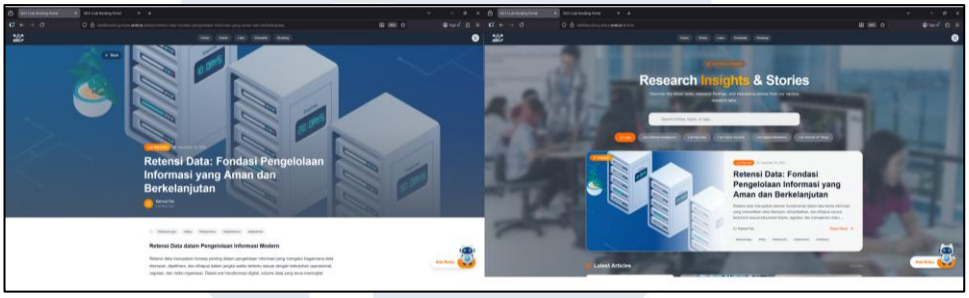
Gambar 3.35 Halaman Landing Page & Management Landing Page

Sebagai kelanjutan dari navigasi utama, pengembangan difokuskan pada halaman Labs yang berperan sebagai katalog fasilitas riset terpusat. Antarmuka pada gambar 3.36 ini dirancang responsif dengan mengintegrasikan fitur *smart filtering* berbasis program studi dan *keyword search* untuk memudahkan pencarian aset. Lebih lanjut, sistem ini menerapkan *dynamic routing* untuk menangani halaman detail laboratorium secara otomatis, di mana setiap entitas menampilkan informasi teknis mendalam mulai dari deskripsi fungsi spesifik (seperti *Sandbox Network* untuk Cyber Security), kapasitas ruangan, hingga daftar inventaris perangkat

keras guna menjamin transparansi data fasilitas bagi pengguna sebelum melakukan reservasi.



Gambar 3.36 Halaman Labs



Gambar 3.37 Halaman Article

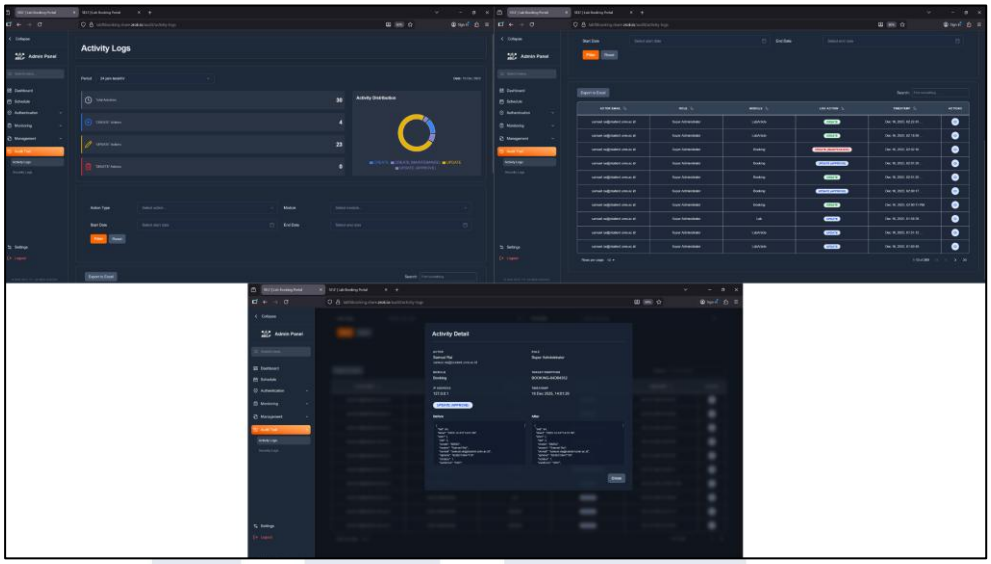
Melengkapi ekosistem informasi, dikembangkan modul 'Research Insights & Stories' sebagai medium diseminasi pengetahuan dan berita internal kampus. Antarmuka pada gambar 3.37 katalog artikel dirancang dengan hierarki visual yang menonjolkan *Featured Article* pada *hero section* untuk isu prioritas, didukung oleh fitur pencarian (*search bar*) dan *filtering chips* berbasis laboratorium (seperti 'Lab Big Data', 'Lab AI') guna mempermudah navigasi topik. Pada level konten, halaman baca (*Read Page*) menyajikan tata letak tipografi yang fokus pada keterbacaan (*readability*), menampilkan metadata penulis (Author), tanggal publikasi, serta sistem pengelompokan berbasis *tags* (misalnya #datastorage, #datacenter), memastikan pengguna mendapatkan konteks informasi yang utuh dan relevan.

Memasuki minggu ketiga, fokus pengembangan dialihkan sepenuhnya pada penguatan arsitektur keamanan data (*data security architecture*) dan akuntabilitas sistem. Tahap ini bersifat fundamental untuk memastikan bahwa

setiap interaksi di dalam aplikasi dapat dipantau, diaudit, dan dipertanggungjawabkan melalui mekanisme teknis yang terstandarisasi.

Guna menjamin integritas dan ketelusuran data (*data traceability*), dikembangkan sistem pencatatan aktivitas otomatis di sisi *backend*. Implementasi ini mencakup dua aliran data krusial: Activity Logs yang merekam seluruh mutasi data operasional (Create, Update, Delete) dan Security Logs yang memantau sesi otentikasi serta mendeteksi anomali akses. Di sisi antarmuka administrator, log ini divisualisasikan dalam bentuk tabel kronologis yang dilengkapi filter forensik (berdasarkan aktor, waktu, atau IP Address), memungkinkan pengelola untuk melakukan investigasi insiden secara presisi dan transparan.

Secara fundamental, pengembangan modul *Activity & Security Logs* pada minggu ini didasarkan pada prinsip Akuntabilitas Individu dalam keamanan sistem informasi. *Audit trail* didefinisikan sebagai mekanisme teknis yang merekam kronologi aktivitas sistem untuk memungkinkan rekonstruksi kejadian (*event reconstruction*) dan deteksi intrusi pasca-insiden. Keberadaan jejak audit ini vital untuk menegakkan prinsip *non-repudiation* (nirsangkal), yang menjamin bahwa pengguna tidak dapat menyangkal aksi yang telah mereka lakukan baik itu modifikasi data maupun upaya akses ilegal karena seluruh bukti digital telah tersimpan secara permanen dan transparan. Tanpa implementasi *audit trails* yang komprehensif, organisasi berisiko kehilangan visibilitas terhadap ancaman internal (*insider threats*) dan gagal memenuhi standar kepatuhan pengelolaan data yang aman.

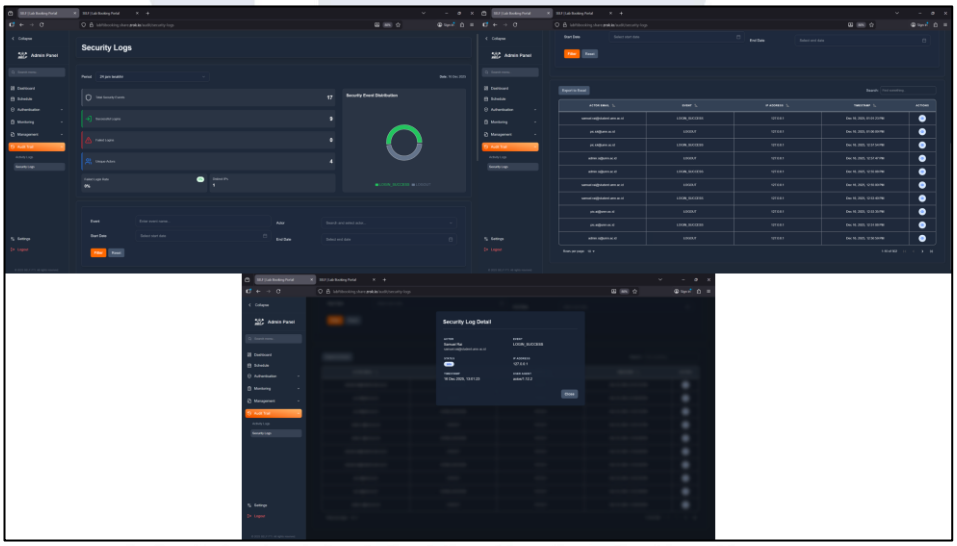


Gambar 3.38 Halaman Dashboard Activity Logs

Secara spesifik pada implementasi Activity Logs, dikembangkan antarmuka pemantauan komprehensif yang menggabungkan visualisasi statistik makro dengan detail forensik mikro. Pada gambar 3.38 diperlihatkan halaman utama menyajikan ringkasan metrik aktivitas dalam bentuk diagram distribusi (*donut chart*) dan penghitung aksi (*action counters*) untuk kategori *Create, Update,* dan *Delete*, memberikan wawasan instan mengenai intensitas operasional sistem. Di bawah panel statistik, tersedia tabel data kronologis yang mencatat identitas aktor, peran (*role*), serta modul yang diakses, lengkap dengan fitur ekspor data (*Export to Excel*) untuk keperluan pelaporan eksternal. Keunggulan teknis utama dari modul ini terletak pada fitur Activity Detail, yang tidak hanya merekam alamat IP pengguna untuk pelacakan lokasi, tetapi juga mengimplementasikan mekanisme perbandingan data (*state comparison*) berbasis JSON; fitur ini menampilkan *snapshot* data *"Before"* dan *"After"* secara berdampingan, memungkinkan administrator untuk memverifikasi perubahan spesifik yang terjadi pada setiap transaksi seperti persetujuan peminjaman atau publikasi artikel dengan tingkat akurasi tinggi.

Berjalan paralel dengan pencatatan aktivitas, modul Security Logs dikembangkan sebagai garda pertahanan untuk memantau integritas

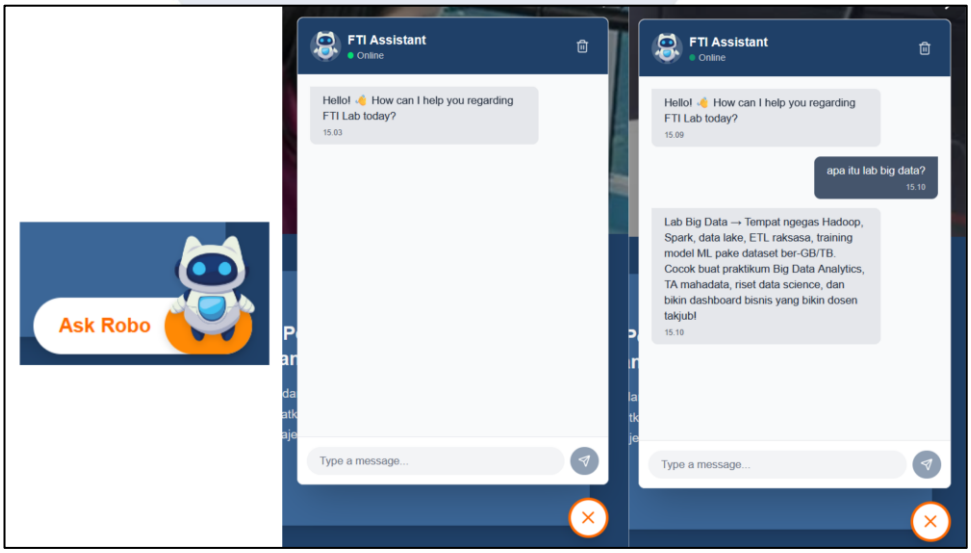
otentikasi sistem. Gambar 3.39 menyajikan halaman utama *Security Event Dashboard* yang memvisualisasikan kesehatan akses melalui metrik *Successful Logins* banding *Failed Logins* serta penghitung *Unique Actors*, yang berfungsi sebagai indikator awal untuk mendeteksi potensi serangan *brute-force* atau anomali akun. Di bawah panel metrik, sistem menyajikan tabel kronologis yang secara spesifik merekam siklus sesi pengguna (event *LOGIN_SUCCESS* dan *LOGOUT*) lengkap dengan identitas aktor dan alamat IP asal. Tingkat kedalaman audit diperkuat pada fitur Security Detail, di mana administrator dapat menelusuri metadata teknis setiap sesi, termasuk informasi *User Agent* (tipe peramban/perangkat) dan status respon server, memberikan visibilitas forensik penuh untuk memvalidasi legitimasi setiap akses yang terjadi dalam jaringan.



Gambar 3.39 Halaman Dashboard Security Logs

Menutup rangkaian kegiatan implementasi pada bulan November, fokus utama dialihkan pada penyempurnaan fitur interaktif dan persiapan lingkungan pengujian. Tahap ini menandai transisi dari fase konstruksi (*coding*) menuju fase stabilisasi, memastikan bahwa seluruh modul yang telah dibangun dapat beroperasi secara sinergis sebelum diserahkan kepada pengguna akhir untuk validasi.

Agenda prioritas minggu ini adalah merampungkan integrasi modul *Artificial Intelligence* pada fitur "Ask Robo" alias "FTI Assistant" pada gambar 3.40. Pengembangan difokuskan pada penyambungan *endpoint* API untuk memperluas kapabilitas respons asisten virtual agar tidak terbatas pada pertanyaan statis. Kini, sistem dirancang mampu melayani pengecekan ketersediaan jadwal laboratorium saat ini (*real-time availability*) serta memberikan panduan instan terkait Standar Operasional Prosedur (SOP) dan tata tertib peminjaman. Berdasarkan hasil pengujian antarmuka, chatbot terbukti responsif dalam memproses input bahasa alami (*Natural Language Processing*) baik untuk pertanyaan teknis spesifik seperti definisi "Lab Big Data" maupun pertanyaan administratif sehingga siap difungsikan sebagai lapisan dukungan pengguna pertama (*Tier-1 Support*) yang efektif mengurangi beban tim operasional dalam melayani pertanyaan rutin (FAQ).



Gambar 3.40 Ask Robo UI

Guna memfasilitasi akses pengujian jarak jauh tanpa memerlukan infrastruktur server produksi penuh, diterapkan mekanisme *network tunneling* (menggunakan layanan seperti zrok.io sesuai terlihat pada *address bar* antarmuka). Langkah ini memungkinkan aplikasi yang berjalan di lingkungan pengembangan lokal (*localhost*) untuk diekspos ke jaringan

publik secara aman, sehingga tim penguji eksternal dapat mengakses sistem secara langsung. Bersamaan dengan proses ini, dilakukan pemindaian kode menyeluruh (*code review*) untuk mengidentifikasi dan memperbaiki *minor bugs* atau inkonsistensi antarmuka (UI Glitches) yang mungkin terlewat selama fase pengembangan fitur.

Secara arsitektural, penerapan teknologi *tunneling* pada fase ini mengadopsi prinsip NAT Traversal guna mengatasi hambatan isolasi jaringan tanpa mengorbankan keamanan perimeter. Mekanisme ini bekerja dengan mengenkapsulasi paket data dari protokol privat ke dalam *secure tunnel* terenkripsi, sehingga layanan yang berjalan di lingkungan lokal (*localhost*) dapat diakses oleh jaringan publik (*ingress*) tanpa memerlukan konfigurasi *port forwarding* statis pada infrastruktur fisik. Pendekatan ini mendukung konsep *Ephemeral Staging*, di mana lingkungan pengujian yang bersifat sementara dapat disajikan secara instan untuk kebutuhan validasi pemangku kepentingan, mempercepat siklus umpan balik (*feedback loop*) sebelum *deployment* final dilakukan [25].

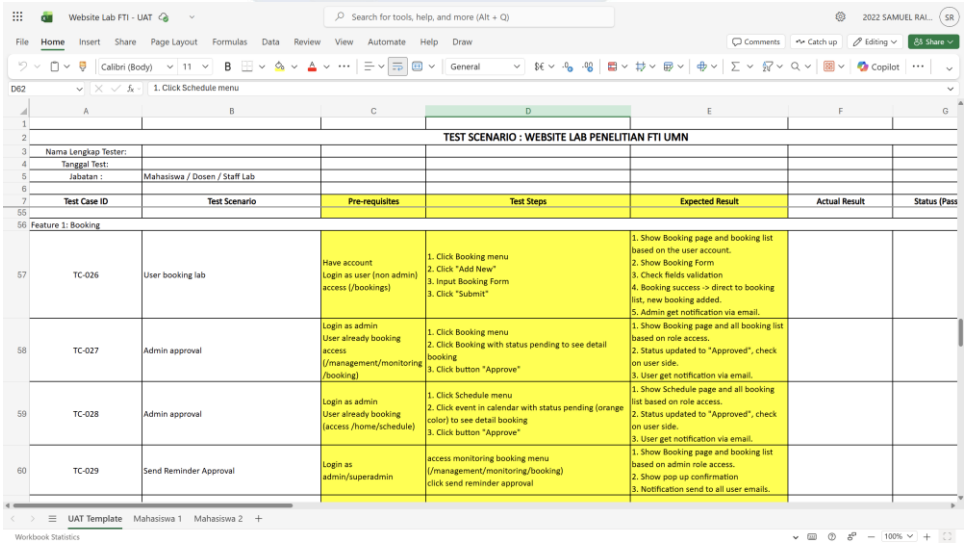
Dengan selesainya infrastruktur *staging* melalui tunneling, fokus pengembangan kini bergeser secara fundamental dari konstruksi fitur menuju User Acceptance Testing (UAT). Tahap ini krusial untuk memvalidasi bahwa sistem tidak hanya berfungsi secara teknis (*verification*), tetapi juga memenuhi kebutuhan bisnis dan ekspektasi pengguna akhir (*validation*). Melalui pengujian ini, potensi kesenjangan (*gap*) antara desain sistem dan alur kerja operasional dunia nyata dapat diidentifikasi dan dimitigasi secara dini, menjamin tingkat kepuasan dan adopsi pengguna yang optimal saat peluncuran resmi.

3.3.1.3 Testing & Evaluation

Pasca selesainya tahap pengembangan fitur dan konfigurasi infrastruktur *tunneling* pada akhir November, kegiatan di minggu pertama Desember difokuskan sepenuhnya pada pelaksanaan User Acceptance

Testing (UAT). Tahap ini menjadi gerbang validasi krusial untuk memastikan bahwa sistem yang dibangun tidak hanya berjalan secara teknis, tetapi juga memenuhi ekspektasi dan kebutuhan operasional pengguna akhir (*end-users*) sebelum peluncuran resmi.

Langkah awal dimulai dengan penyusunan dokumen skenario uji (*test scripts*) yang komprehensif. Instrumen ini dirancang mencakup 55 butir uji (*test cases*) yang memetakan seluruh fitur sistem mulai dari otentikasi, manajemen peminjaman, hingga fitur keamanan guna merepresentasikan alur kerja nyata di lapangan secara utuh. Lingkungan aplikasi yang telah diekspos ke jaringan publik melalui layanan *tunneling* (zrok.io) divalidasi stabilitasnya untuk memastikan dapat diakses secara lancar oleh para responden dari berbagai perangkat, sehingga menjamin hasil pengujian mencerminkan pengalaman pengguna yang otentik di luar lingkungan lokal (*localhost*).



TEST SCENARIO : WEBSITE LAB PENELITIAN FTI UMN

Nama Lengkap Tester:
Tanggal Test:
Jabatan : Mahasiswa / Dosen / Staff Lab

Test Case ID	Test Scenario	Pre-requisites	Test Steps	Expected Result	Actual Result	Status (Pass
Feature 1: Booking						
TC-026	User booking lab	Have account Login as user (non admin) access (/bookings)	1. Click Booking menu 2. Click "Add New" 3. Input Booking Form 3. Click "Submit"	1. Show Booking page and booking list based on the user account. 2. Show Booking Form 3. Check fields validation 4. Booking success -> direct to booking list, new booking added. 5. Admin get notification via email.		
TC-027	Admin approval	Login as admin User already booking access (/management/monitoring /booking)	1. Click Booking menu 2. Click Booking with status pending to see detail booking 3. Click button "Approve"	1. Show Booking page and all booking list based on role access. 2. Status updated to "Approved", check on user side. 3. User get notification via email.		
TC-028	Admin approval	Login as admin User already booking (access /home/schedule)	1. Click Schedule menu 2. Click event in calendar with status pending (orange color) to see detail booking 3. Click button "Approve"	1. Show Schedule page and all booking list based on role access. 2. Status updated to "Approved", check on user side. 3. User get notification via email.		
TC-029	Send Reminder Approval	Login as admin/superadmin	access monitoring booking menu (/management/monitoring/booking) click send reminder approval	1. Show Booking page and booking list based on admin role access. 2. Show pop up confirmation 3. Notification send to all user emails.		

Gambar 3.41 Dokumen Testing

Gambar 3.41 menyajikan sampel dokumen instrumen pengujian yang digunakan dalam fase UAT. Tabel tersebut merekam jejak validasi secara rinci, mencakup kolom Skenario Pengujian untuk mendefinisikan fitur yang diuji, Langkah Pengujian (*Test Steps*) yang menjelaskan urutan interaksi pengguna, serta perbandingan antara Hasil yang Diharapkan

(*Expected Result*) dengan Hasil Aktual (*Actual Result*). Sebagaimana terlihat pada kolom status, sistem berhasil merespons input sesuai dengan logika bisnis yang dirancang, di mana seluruh skenario mulai dari validasi formulir peminjaman hingga respons *chatbot* dinyatakan 'Valid'. Dokumen ini menjadi bukti empiris bahwa dari total 55 skenario uji yang dilakukan, tidak ditemukan penyimpangan fungsional (*functional deviation*) yang menghambat operasional sistem.

Setelah lingkungan siap, tautan akses sistem didistribusikan kepada kelompok responden terbatas yang merepresentasikan variasi peran pengguna, yaitu terdiri dari dua orang perwakilan mahasiswa, dosen, serta staf laboratorium. Dalam fase ini, para partisipan diminta untuk menyelesaikan ke-55 tugas spesifik (*tasks*) tersebut pada aplikasi dan memberikan penilaian subjektif terkait kemudahan penggunaan (*usability*), kejelasan informasi, dan kesesuaian fitur. Seluruh umpan balik (*feedback*) yang masuk dicatat secara sistematis sebagai bahan evaluasi untuk penyempurnaan akhir sistem.

Berdasarkan rekapitulasi umpan balik dari responden, secara umum sistem mendapatkan penilaian positif dari aspek usabilitas dan interaktivitas. Mayoritas penguji menilai antarmuka aplikasi berjalan intuitif dan dinamis, di mana alur navigasi dari *login* hingga penyelesaian reservasi (*booking flow*) dirasakan mulus tanpa hambatan kognitif yang berarti. Aspek *User Experience* (UX) dinilai "sangat baik" karena responsivitas sistem dalam memberikan umpan balik visual (*visual feedback*) pada setiap interaksi, sehingga pengguna merasa terbantu dalam memahami status permohonan mereka secara *real-time*.

Meskipun fungsi utama sistem berjalan lancar, pengujian tahap ini berhasil mengidentifikasi sejumlah anomali teknis (*minor defects*) yang menjadi catatan penting untuk perbaikan. Kendala utama yang ditemukan meliputi Inkonsistensi Visual (*Visual Inconsistency*), di mana terdapat

ketidakseragaman saturasi pada palet warna oranye terkadang tampil terlalu gelap atau terlalu terang yang sedikit mendegradasi estetika desain. Selain itu, teridentifikasi pula isu Tata Letak (*Layout Overlapping*) berupa penumpukan elemen antarmuka pada resolusi layar spesifik yang menghambat keterbacaan teks. Pada aspek Responsivitas Mobile (*Mobile View Glitches*), pengalaman pengguna di sisi klien (*client-side*) masih terkendala oleh ketidakstabilan struktur (*layout shift*), yang mengakibatkan navigasi pada perangkat seluler terasa kurang rapi dan intuitif dibandingkan versi *desktop*.

Evaluasi khusus pada fitur *AI Chatbot* ("Ask Robo") menunjukkan hasil yang variatif. Meskipun mampu menjawab definisi dasar, fitur ini masih memiliki keterbatasan signifikan dalam menangani pertanyaan yang bersifat instruksional spesifik, seperti permintaan tautan (*direct link*) atau navigasi kompleks. Selain itu, akurasi jawaban terkait jadwal *real-time* belum mencapai tingkat presisi 100%, di mana terkadang terjadi jeda sinkronisasi data yang menyebabkan respons bot kurang relevan dengan kondisi aktual database.

Terlepas dari catatan perbaikan visual dan limitasi pada modul kecerdasan buatan, fungsi inti (*core functionalities*) sistem dinyatakan valid dan berjalan 100%. Modul kritikal mulai dari registrasi akun, otentikasi login, validasi jadwal, hingga dokumentasi peminjaman beroperasi tanpa kegagalan sistem (*system crash*) atau *logic error*. Dengan demikian, sistem dinilai telah siap untuk digunakan secara operasional dengan catatan rekomendasi perbaikan pada aspek kosmetik dan penyempurnaan logika *bot* di pengembangan selanjutnya. Menindaklanjuti temuan kendala teknis tersebut, serangkaian langkah optimalisasi lanjutan sangat krusial guna menjamin stabilitas sistem sebelum dilakukan peluncuran skala penuh. Fokus utama perbaikan mencakup standardisasi *Design System* melalui penyelarasan variabel warna (*CSS variables*) dan komponen antarmuka untuk mengeliminasi inkonsistensi visual. Selain itu, penanganan isu

responsivitas dilaksanakan melalui teknik *refactoring mobile view* dengan pendekatan *Mobile-First Design* guna memastikan presisi tata letak pada berbagai dimensi layar. Sebagai upaya peningkatan fungsionalitas fitur 'Ask Robo', diperlukan pelatihan ulang (*retraining*) model LLM menggunakan dataset yang lebih spesifik, serta optimalisasi *webhook* jadwal demi menjamin sinkronisasi data yang akurat dan minim latensi antara basis data dan respons sistem secara *real-time*.

Terakhir, untuk meningkatkan kecerdasan "Ask Robo", disarankan untuk melakukan Ekspansi Knowledge Base pada Arsitektur RAG. Mengingat sistem menggunakan metode *Retrieval-Augmented Generation*, akurasi jawaban sangat bergantung pada kelengkapan data referensi. Oleh karena itu, perlu dilakukan pengayaan *dataset* dokumen vektor yang mencakup instruksi navigasi spesifik, tautan internal (*direct links*), serta variasi skenario tanya-jawab operasional yang lebih luas guna meminimalisir "halusinasi" jawaban dan meningkatkan relevansi konteks.

Sebelum adanya integrasi sistem, manajemen peminjaman ruangan penelitian di Fakultas Teknik dan Informatika (FTI) menghadapi kendala operasional yang signifikan akibat ketergantungan pada prosedur manual dan semi-manual, seperti penggunaan formulir kertas dan komunikasi via aplikasi pesan singkat. Absennya basis data terpusat (*single source of truth*) mengakibatkan rendahnya transparansi informasi mengenai ketersediaan ruangan secara *real-time*, yang sering memicu terjadinya konflik jadwal (*double booking*) akibat *human error*. Selain itu, proses verifikasi manual menciptakan alur birokrasi yang lambat dan beban administratif yang berlebihan bagi staf, sehingga menghambat efisiensi dukungan terhadap aktivitas riset mahasiswa dan dosen.

Merespons permasalahan tersebut, kegiatan magang ini bertujuan untuk merancang dan mengimplementasikan solusi digital berupa aplikasi web *fullstack* yang terintegrasi, menggunakan teknologi Next.js pada sisi

frontend dan FastAPI pada sisi *backend*. Fokus utama pengembangan adalah menciptakan sistem yang mampu memberikan visibilitas ketersediaan ruangan secara *real-time*, mengotomatisasi alur persetujuan untuk mengeliminasi risiko jadwal ganda, serta menyederhanakan proses pelaporan administratif. Melalui implementasi ini, diharapkan tercipta ekosistem manajemen fasilitas yang transparan, akuntabel, dan efisien, sekaligus menghasilkan dokumentasi teknis yang komprehensif sebagai landasan pengembangan sistem lebih lanjut oleh pihak fakultas.

Hasil pengujian fungsional dan validasi pengguna (*User Acceptance Testing*) yang telah dilaksanakan memberikan bukti empiris bahwa sistem berhasil menjawab tantangan operasional yang diuraikan pada latar belakang masalah. Secara spesifik, keberhasilan skenario pencegahan *double booking* secara efektif mengeliminasi risiko konflik jadwal yang sebelumnya kerap terjadi akibat *human error* dalam pencatatan manual. Selain itu, validasi fitur jadwal *real-time* memecahkan masalah asimetri informasi, memberikan kepastian data ketersediaan ruangan kepada pengguna tanpa perlu melalui birokrasi tanya-jawab yang berbelit.

Lebih jauh, digitalisasi alur persetujuan yang teruji valid dalam pengujian *User Acceptance Testing* (UAT) membuktikan bahwa sistem mampu mereduksi beban administratif staf secara signifikan. Berdasarkan perbandingan dengan proses manual sebelumnya yang mengandalkan pesan singkat (WhatsApp), durasi pengecekan jadwal hingga konfirmasi sering kali memakan waktu 1 hingga 2 jam, bahkan dalam beberapa kasus bisa mencapai hitungan hari apabila dosen penanggung jawab memiliki keterbatasan waktu untuk merespons pesan di tengah kepadatan jadwal. Dengan beralihnya prosedur ke platform terpusat, kendala ketidakteraturan data dan hambatan komunikasi tersebut dapat diminimalisir. Durasi pemrosesan teknis kini menjadi jauh lebih singkat, yakni hanya memerlukan waktu kurang lebih 3 menit, yang memberikan kepastian serta transparansi status permohonan secara *real-time* bagi pengguna.

Sebagai simpulan, rangkaian kegiatan pengembangan yang dimulai dari analisis kebutuhan, perancangan arsitektur berbasis Next.js dan FastAPI, hingga validasi akhir melalui *User Acceptance Testing* (UAT) telah berhasil menghasilkan solusi digital yang tidak hanya kokoh secara teknis, tetapi juga transformatif secara operasional. Implementasi sistem ini terbukti mampu mengeliminasi *bottleneck* birokrasi dan ketidakpastian informasi yang sebelumnya menjadi kendala utama dalam manajemen fasilitas laboratorium. Dengan tercapainya seluruh kriteria keberhasilan pada tahap pengujian, sistem ini dinyatakan layak untuk diimplementasikan pada lingkungan produksi (*production environment*) guna mewujudkan ekosistem layanan akademik di Fakultas Teknik dan Informatika yang lebih efisien, transparan, dan berkelanjutan.

3.3.2 Kendala yang Ditemukan

Selama pelaksanaan kegiatan magang dan pengembangan sistem, teridentifikasi sejumlah kendala yang mempengaruhi alur kerja, baik dari aspek manajemen maupun teknis. Kendala-kendala tersebut diuraikan sebagai berikut:

a. Tidak adilnya pembagian tugas teknis di antara anggota tim menyebabkan masalah besar dalam manajemen waktu. Tingkat penguasaan teknologi yang digunakan, seperti stack teknologi Next.js dan FastAPI, berbeda, sehingga sebagian besar fitur inti (core features) dan logika sistem yang kompleks dikerjakan secara bersamaan. Dengan demikian, waktu yang tersedia untuk pendalaman riset fitur dan optimalisasi kode sangat terbatas. Fokus utama tersita untuk memastikan bahwa fungsionalitas dasar sistem dapat beroperasi sesuai tenggat waktu yang ketat.
b. Dalam proses kerja tim, komunikasi internal menjadi tantangan tersendiri. Dalam kaitannya dengan progres pengerjaan antar-modul, terjadi celah informasi karena tingkat komunikasi yang berbeda dan kurangnya sinkronisasi berkala. Disebabkan perlunya menyelaraskan

ulang logika program yang dibuat secara terpisah agar dapat bekerja secara sinergis tanpa konflik, proses integrasi sistem akan memakan waktu lebih lama daripada yang direncanakan sebelumnya.

c. Dalam hal infrastruktur teknis, ada hambatan untuk menyediakan lingkungan produksi yang efisien dari segi biaya tetapi tetap teknis. Salah satu masalah utama adalah mencoba menemukan penyedia layanan cloud hosting atau PaaS (Platform as a Service) bertier gratis yang dapat mendukung arsitektur microservices (seperti Python FastAPI dan Node.js) dan manajemen basis data secara bersamaan. Untuk mencapai hal ini, perlu dilakukan penyelidikan teknis menyeluruh untuk menemukan metode publikasi aplikasi yang lebih stabil yang tidak membebani anggaran operasional proyek.

Identifikasi kendala ini krusial sebagai fondasi manajemen risiko proyek. Pemetaan tantangan tersebut memberikan gambaran objektif situasi lapangan, sekaligus melandasi perumusan strategi mitigasi adaptif demi menjamin keberlanjutan pengembangan sistem di tengah keterbatasan yang ada.

3.3.3 Solusi atas Kendala yang Ditemukan

Guna mengatasi berbagai kendala yang telah diuraikan sebelumnya, diterapkan serangkaian langkah strategis dan teknis sebagai berikut:

a. Strategi manajemen prioritas yang ketat digunakan untuk mengatasi keterbatasan waktu yang disebabkan oleh ketimpangan beban kerja. Fokus pengembangan terlebih dahulu tertuju pada penyelesaian fitur-fitur penting yang merupakan nilai utama sistem (Minimum Viable Product), seperti modul otentikasi dan algoritma reservasi untuk menjamin bahwa fungsionalitas inti tetap berjalan stabil sebelum tenggat waktu. Fitur-fitur pendukung lainnya dikerjakan secara bertahap atau disederhanakan implementasinya, sehingga target

penyelesaian proyek tetap dapat tercapai secara realistis dengan kualitas yang terjaga.

b. Jadwal pertemuan rutin yang lebih intensif, dapat membantu mengurangi hambatan komunikasi. Mekanisme ini berfungsi sebagai forum diskusi teknis dan wadah validasi progres harian untuk membedah logika pemrograman yang kompleks secara bersama-sama. Selain itu, standar pelaporan status pengerjaan yang lebih terorganisir diterapkan melalui kanal komunikasi digital. Ini memungkinkan setiap hambatan teknis diidentifikasi dan diselesaikan secara bersamaan untuk mengurangi kemungkinan kegagalan integrasi.

c. Pendekatan teknis yang menggunakan teknologi open-source tunneling melalui layanan zrok.io digunakan untuk menyelesaikan kendala infrastruktur deployment. Implementasi ini membutuhkan sejumlah konfigurasi teknis khusus, termasuk migrasi seluruh lingkungan proyek ke dalam Windows Subsystem for Linux (WSL) untuk memastikan kompatibilitas sistem operasi dan pengaturan Caddy Web Server sebagai proxy balik. Solusi ini terbukti berhasil menyediakan lingkungan staging yang stabil, aman, dan dapat diakses publik untuk keperluan pengujian (UAT) tanpa memerlukan biaya sewa server dedikasi. Ini meskipun memerlukan upaya konfigurasi awal yang cukup menantang.

Implementasi solusi strategis dan teknis terbukti efektif dalam memitigasi risiko dan menjaga stabilitas kerja. Sinergi manajerial serta inovasi infrastruktur menjadi kunci keberhasilan penyelesaian proyek sesuai standar kualitas dan tenggat waktu yang ditetapkan.