PROYEK TELEMATIKA GASAL 2024/2025

PRATIKA APP

Kelompok 2

I Wayan Agus Darmawan
NRP 5024211070

Dosen Pengampu

Ahmad Zaini, S.T., M.Sc

Prof. Dr. I Ketut Eddy Purnama, S.T.,M.T.

Dr. Arief Kurniawan, S.T., M.T.

Departemen Teknik Komputer
Fakultas Teknik Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2024

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan Laporan Proyek Telematika dengan judul **Praktika App** ini dengan baik. Dalam proses penyusunan laporan ini, kami ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah memberikan dukungan, baik secara langsung maupun tidak langsung, sehingga laporan ini dapat terselesaikan dengan lancar, yaitu:

- Tuhan Yang Maha Esa atas segala kelancaran, keberkahan, dan bimbingan yang diberikan selama pelaksanaan proyek ini.
- Bapak Dr. Supeno Mardi Susiki Nugroho, S.T., M.T., selaku Kepala Departemen Teknik Komputer FTEIC-ITS, atas dukungan dan arahannya.
- Bapak Ahmad Zaini, S.T., M.T., Dr. Arief Kurniawan, S.T., M.T., dan Prof. Dr. I Ketut Eddy Purnama, S.T., M.T., selaku para Dosen Pengampu Proyek Telematika Departemen Teknik Komputer FTEIC-ITS, atas bimbingan dan ilmunya selama proses penyusunan laporan ini.
- Rekan-rekan kelompok proyek telematika, yang telah bekerja sama dengan penuh dedikasi selama pelaksanaan proyek ini.
- Rekan satu tim dan semua pihak lainnya yang tidak dapat kami sebutkan satu per satu namun telah memberikan kontribusi yang sangat berarti.

Kami menyadari bahwa laporan ini masih jauh dari kesempurnaan. Oleh karena itu, kami memohon maaf jika terdapat kekurangan atau hal-hal yang kurang berkenan selama pelaksanaan proyek maupun dalam penyusunan laporan ini. Kami berharap, laporan ini dapat memberikan manfaat, baik bagi pembaca maupun bagi pengembangan keilmuan di bidang telematika di masa yang akan datang. Demikian, semoga laporan ini dapat menjadi referensi dan inspirasi bagi para pembaca.

Surabaya, Desember 2024

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pelaksanaan praktikum di Departemen Teknik Komputer ITS merupakan komponen penting dalam proses pembelajaran, di mana mahasiswa dapat mengaplikasikan teori yang telah dipelajari ke dalam praktik nyata. Namun, dalam pelaksanaannya, sejumlah kendala sering kali muncul yang menghambat kelancaran proses praktikum dan berdampak pada hasil belajar mahasiswa. Salah satu masalah utama yang dihadapi adalah kesulitan dalam mengakses informasi terkait praktikum. Saat ini, penyebaran informasi mengenai pendaftaran, pelaksanaan, teknis, modul, pengumpulan tugas, dan asistensi sering kali dilakukan melalui platform seperti WhatsApp. Meskipun WhatsApp merupakan alat komunikasi yang umum digunakan, penyampaian informasi melalui platform ini cenderung kurang terstruktur dan mudah terabaikan. Akibatnya, banyak mahasiswa yang terlambat atau bahkan tidak mendapatkan informasi penting yang berkaitan dengan pelaksanaan praktikum. Hal ini tentunya mengganggu persiapan dan partisipasi mahasiswa dalam praktikum, yang pada gilirannya mempengaruhi kualitas hasil belajar mereka.

Selain itu, kurangnya pemahaman mahasiswa terhadap materi yang disampaikan dalam praktikum juga menjadi tantangan yang signifikan. Meskipun modul panduan telah disediakan, mahasiswa masih sering mengalami kesulitan dalam memahami langkah-langkah praktikum yang harus dilakukan. Praktikum yang dilakukan secara berkelompok juga tidak jarang menyebabkan adanya ketidakmerataan dalam kontribusi antar anggota kelompok. Banyak mahasiswa yang hanya mengikuti instruksi dari rekan sekelompok tanpa terlibat secara aktif dalam pelaksanaan percobaan, sehingga pemahaman mereka terhadap materi menjadi kurang mendalam. Kebingungan yang terjadi selama praktikum juga disebabkan oleh minimnya simulasi atau prakondisi yang dapat memberikan gambaran awal sebelum praktik dimulai, yang pada akhirnya menghambat alur pelaksanaan praktikum. Minat dan antusiasme mahasiswa dalam mendalami materi praktikum juga terbilang rendah. Banyak mahasiswa yang lebih fokus pada penyelesaian laporan yang bersifat administratif, daripada benar-benar memahami dan menguasai materi yang dipraktikkan. Selain itu, kurangnya tantangan atau kompetisi yang dapat memotivasi mahasiswa juga menjadi faktor yang menurunkan minat mereka dalam mengikuti praktikum. Dengan latar belakang masalah-masalah tersebut, perlu adanya upaya untuk memperbaiki sistem pelaksanaan praktikum di Teknik Komputer ITS agar dapat memberikan pengalaman belajar yang lebih efektif dan mendalam bagi mahasiswa.

1.2 Permasalahan

Berdasarkan hal yang telah dipaparkan di latar belakang, terdapat beberapa masalah yang diangkat oleh peneliti dalam pembuatan tugas proyek telematika ini di antaranya:

1. **Kesulitan Akses Informasi Praktikum:** Informasi terkait pelaksanaan praktikum di Teknik Komputer ITS, seperti pendaftaran, teknis, modul, pengumpulan tugas, dan asistensi, saat ini disebarkan melalui WhatsApp. Meskipun WhatsApp merupakan alat komunikasi yang umum, penyampaian informasi melalui platform ini sering kali kurang terstruktur dan mudah terabaikan. Akibatnya, banyak mahasiswa yang terlambat mendapatkan informasi penting atau bahkan tidak menerima informasi sama sekali. Hal ini berdampak negatif pada persiapan dan partisipasi mahasiswa dalam praktikum, mengganggu alur kegiatan, dan menurunkan kualitas hasil pembelajaran.

2. **Kurangnya Pemahaman Materi Praktikum:** Meskipun modul panduan praktikum telah disediakan, banyak mahasiswa masih mengalami kesulitan dalam memahami materi yang disampaikan. Pelaksanaan praktikum yang dilakukan dalam kelompok sering kali tidak menjamin pemahaman yang merata di antara anggota kelompok. Banyak mahasiswa cenderung hanya mengikuti instruksi dari rekan sekelompok tanpa terlibat secara aktif dalam percobaan. Kondisi ini diperburuk oleh kurangnya simulasi atau prakondisi yang dapat memberikan gambaran awal sebelum praktikum dimulai, sehingga kebingungan sering terjadi selama proses pelaksanaan praktikum. Akibatnya, alur praktikum menjadi terhambat dan tujuan pembelajaran tidak tercapai secara optimal.

1.3 Solusi

Untuk mengatasi masalah-masalah tersebut, solusinya adalah dengan mengembangkan sebuah sistem berbasis web yang mencakup berbagai fitur untuk memudahkan pelaksanaan praktikum. Solusi ini akan memberikan integrasi informasi, simulasi praktikum, serta sistem kompetisi yang mampu meningkatkan partisipasi dan pemahaman mahasiswa. Beberapa fitur utama dari sistem ini adalah:

1. **Sistem Administrasi Praktikum Berbasis Web:** Sistem ini akan memfasilitasi mahasiswa untuk mendaftar praktikum, memilih jadwal, dan mendapatkan notifikasi terkait informasi penting seperti modul, teknis pelaksanaan, hingga deadline pengumpulan. Sistem ini mirip dengan platform LARAS, namun dikembangkan khusus untuk kebutuhan praktikum Teknik Komputer. Dengan adanya notifikasi otomatis, mahasiswa tidak akan melewatkan informasi yang diperlukan.

2. **Praktikum Rangkaian Digital Berbasis Simulasi dan IoT:** Dalam praktikum rangkaian digital, solusi ini akan mencakup pretest berbasis simulasi rangkaian, yang dapat dilakukan sebelum praktikum dimulai. Selain itu, sistem akan terintegrasi dengan modul fisik melalui IoT, yang memungkinkan sistem mendeteksi apakah mahasiswa dapat menyelesaikan tugas modul atau tidak, secara real-time.

BAB 2 KONTRIBUSI

2.1 Perancangan Arsitektur

Perancangan arsitektur sistem merupakan tahap awal yang sangat penting dalam pengembangan perangkat lunak. Sebagai seorang *software architect*, perancangan arsitektur yang baik memastikan bahwa sistem dapat memenuhi kebutuhan fungsional dan non-fungsional, serta mendukung kemudahan pengelolaan, keamanan, dan skalabilitas.

2.1.1 Identifikasi Kebutuhan Pengguna

Tahap pertama dalam perancangan ini adalah melakukan identifikasi kebutuhan pengguna. Proses ini bertujuan untuk menentukan spesifikasi sistem secara terperinci, sehingga setiap kebutuhan pengguna dapat terakomodasi dalam desain sistem.

2.1.2 Pemilihan Teknologi

Pemilihan teknologi merupakan aspek kritis dalam desain arsitektur. Teknologi yang dipilih bertujuan untuk mendukung implementasi sistem yang efisien, handal, dan mudah dikembangkan di masa depan. Keputusan ini didasarkan pada tren teknologi terkini dan prinsip-prinsip desain perangkat lunak modern.

2.1.3 Perancangan Arsitektur Aplikasi

Perancangan arsitektur aplikasi dilakukan dengan mempertimbangkan kebutuhan pengguna dan teknologi yang dipilih. Arsitektur ini mencakup struktur frontend, backend, dan DevOps, yang dirancang untuk mendukung pengembangan sistem yang efisien, aman, dan mudah dikelola.

2.2 Perancangan Teknologi yang Digunakan

Teknologi yang digunakan dalam pengembangan sistem ini dipilih berdasarkan kebutuhan untuk efisiensi, keamanan, dan skalabilitas. Perancangan ini mencakup aspek general, frontend, backend, dan DevOps, yang dijelaskan secara terperinci berikut ini.

2.2.1 General

Untuk menjaga konsistensi dan efisiensi dalam pengembangan, sejumlah standar diterapkan. Penggunaan *ESLint* dan *Prettier* memastikan format kode yang konsisten di seluruh proyek, sehingga meminimalkan potensi kesalahan dan mempermudah kolaborasi antar pengembang. Selain itu, integrasi *Husky* dengan *pre-commit hooks* memastikan bahwa linting dan pemformatan dilakukan sebelum perubahan kode dikirim ke repositori. Langkah ini merupakan bagian dari kontrol kualitas sejak tahap awal pengembangan.

Pipeline *CI/CD* yang diimplementasikan menggunakan *GitHub Actions* mendukung proses pengembangan berkelanjutan. Pipeline ini mencakup linting, pengujian, dan *build* otomatis, yang memberikan umpan balik cepat atas setiap perubahan kode. Selain itu, *starter kit* yang mencakup konfigurasi linting, pipeline *CI/CD*, dan struktur proyek yang terorganisir disediakan untuk memastikan pengembang dapat memulai proyek dengan standar yang seragam.

2.2.2 Arsitektur Frontend

Frontend sistem ini dibangun menggunakan *Next.js* sebagai kerangka kerja utama. *Next.js* dipilih karena mendukung *server-side rendering (SSR)* dan *static site generation (SSG)*, yang meningkatkan waktu muat halaman serta memberikan pengalaman pengguna yang lebih baik. Untuk antarmuka pengguna, kombinasi *Shadcn/UI* dan *Tailwind CSS* digunakan, memungkinkan desain antarmuka yang responsif, konsisten, dan efisien. Selain itu, *Framer Motion*

diterapkan untuk menambahkan animasi dinamis dan transisi halaman, yang membuat pengalaman pengguna lebih menarik dan interaktif.

Berdasarkan penjelasan diatas dibuat *starter kit* yang mencakup konfigurasi *Next.js*, integrasi *Shadcn/UI* dan *Tailwind CSS*, serta *Framer Motion*. *Starter kit* ini juga dilengkapi dengan *CI/CD* yang telah dijelaskan pada subab general untuk mendukung pengembangan berkelanjutan. *Starter kit* ini memastikan pengembangan *frontend* yang konsisten dan efisien, serta mempercepat proses pengembangan antarmuka pengguna.

2.2.3 Arsitektur Backend

Di sisi *backend*, *AdonisJS v6* digunakan sebagai kerangka kerja utama. *AdonisJS* dipilih karena mendukung pengembangan berbasis *TypeScript* yang modular dan terorganisir. Untuk memastikan keamanan, sistem otentikasi dirancang menggunakan *JWT (JSON Web Token)*, sehingga hanya pengguna yang terotorisasi dapat mengakses data sensitif. Dalam pengelolaan data, *Lucid ORM* digunakan untuk mempermudah manipulasi database relasional, dengan *PostgreSQL* sebagai database utama karena keandalannya dalam menangani data terstruktur. Selain itu, *Redis* diintegrasikan sebagai *caching layer* untuk meningkatkan kinerja aplikasi, terutama dalam pengelolaan sesi dan antrian pekerjaan.

Berdasarkan penjelasan di atas, dibuat *starter kit backend* yang mencakup struktur proyek *AdonisJS*, integrasi *Redis*, dan pipeline *CI/CD*. *Starter kit* ini juga mengimplementasikan pengujian otomatis (*automated testing*) untuk memverifikasi integritas aplikasi setelah setiap pembaruan. Dengan demikian, pengembangan *backend* menjadi lebih andal, konsisten, dan terukur.

2.2.4 DevOps

Pendekatan *DevOps* yang digunakan mendukung pengelolaan sistem yang efisien dan terukur. *Docker* digunakan untuk mengisolasi layanan *frontend* dan *backend*, memastikan konsistensi lingkungan di berbagai tahap pengembangan. Selain itu, *docker-compose.yml* dirancang untuk mengelola layanan seperti *backend*, *PostgreSQL*, *Redis*, dan *Adminer*, sehingga integrasi layanan menjadi lebih mudah.

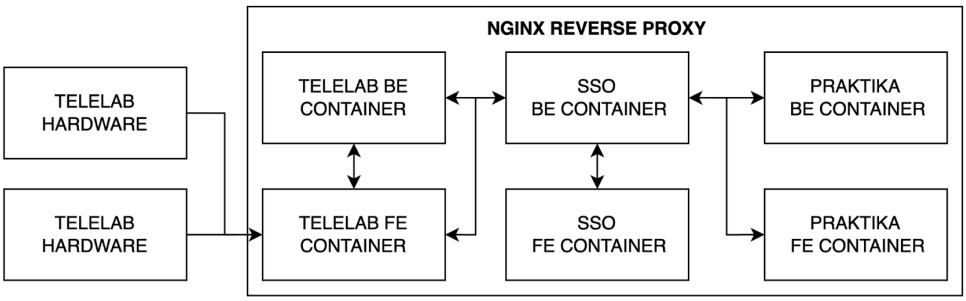
Untuk orkestrasi layanan, *Kubernetes* digunakan. Konfigurasi mencakup *namespace* untuk setiap layanan, *deployment* yang mendukung skalabilitas, serta *load balancer* untuk mengatur lalu lintas jaringan. *Nginx* diterapkan sebagai *reverse proxy* untuk mengelola permintaan ke sistem, memastikan pengelolaan lalu lintas yang efisien dan terstruktur.

Starter kit DevOps mencakup template *Dockerfile*, *docker-compose.yml*, konfigurasi *Nginx*, serta *YAML Kubernetes* untuk *deployment*. Kit ini juga mendukung pengelolaan log terpusat menggunakan *ELK Stack (Elasticsearch, Logstash, dan Kibana)* untuk memantau performa dan memastikan sistem berjalan optimal. Dengan pendekatan ini, sistem dapat dikelola dengan standar yang tinggi, baik di lingkungan pengembangan maupun produksi.

2.3 Arsitektur Aplikasi

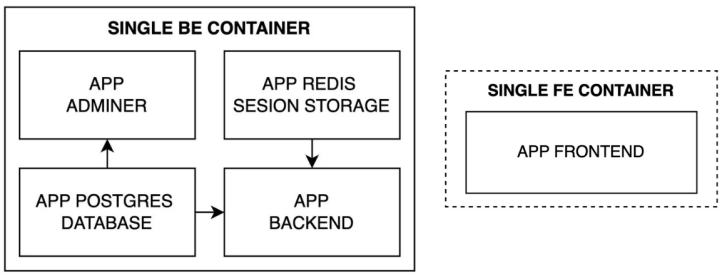
Arsitektur aplikasi PraktikaApp dirancang dengan pendekatan *microservices* dan *containerization* untuk mendukung modularitas, skalabilitas, dan kemudahan manajemen sistem. Pendekatan ini memungkinkan setiap layanan utama dapat dikembangkan, dikelola, dan diterapkan secara independen tanpa mengganggu layanan lainnya. Arsitektur ini terdiri dari tiga layanan utama, yaitu PraktikaApp, PraktikaTelelab, dan PraktikaSSO, yang masing-masing dipisahkan ke dalam kontainer terisolasi. PraktikaApp berfungsi sebagai sistem inti yang menangani administrasi praktikum, mencakup pengelolaan data mahasiswa, jadwal praktikum, pengaturan modul, dan pelaporan hasil. Layanan ini terdiri dari dua komponen utama, yaitu *frontend* dan *backend*, yang masing-masing diimplementasikan dalam kontainer terpisah untuk memastikan pengelolaan kode yang modular dan mempermudah skalabilitas. PraktikaTelelab

dirancang untuk mendukung praktikum berbasis simulasi dan IoT, terutama untuk eksperimen rangkaian digital. Sistem ini terhubung dengan perangkat keras berbasis *ESP* seperti *ESP8266* atau *ESP32*, yang memungkinkan pengguna untuk melakukan eksperimen jarak jauh. PraktikaTelelab tidak hanya menyediakan kemampuan simulasi berbasis perangkat lunak, tetapi juga memungkinkan integrasi langsung dengan perangkat keras fisik untuk memperkaya pengalaman praktikum. Hal ini memungkinkan mahasiswa untuk memahami implementasi praktis teknologi IoT secara langsung. Sementara itu, PraktikaSSO bertindak sebagai sistem *Single Sign-On (SSO)* yang mengintegrasikan PraktikaApp, PraktikaTelelab, serta aplikasi lainnya yang akan dikembangkan di masa depan. PraktikaSSO memastikan bahwa pengguna dapat mengakses seluruh layanan dengan satu kredensial tunggal, sehingga meningkatkan efisiensi dan pengalaman pengguna secara keseluruhan. Sistem ini dirancang dengan keamanan yang tinggi untuk melindungi data sensitif dan mencegah akses tidak sah. Diagram arsitektur aplikasi secara keseluruhan dapat dilihat pada Gambar 2.1.



Gambar 2.1: Diagram Arsitektur Aplikasi

Untuk mendukung implementasi layanan-layanan tersebut, setiap kontainer dirancang dengan komponen-komponen utama yang saling melengkapi, seperti yang digambarkan pada Gambar 2.2. Kontainer *Single BE Container* mencakup beberapa elemen penting, yaitu *App Adminer*, sebuah alat antarmuka yang mempermudah pengelolaan database PostgreSQL melalui antarmuka web yang intuitif; *App Postgres Database*, yang digunakan sebagai sistem manajemen basis data relasional untuk menyimpan data pengguna dan operasional aplikasi; *App Backend*, yang bertanggung jawab untuk menjalankan logika bisnis aplikasi dan memproses permintaan dari *frontend*; serta *App Redis Session Storage*, yang digunakan untuk penyimpanan sesi berbasis memori guna meningkatkan performa dan manajemen sesi pengguna. Setiap komponen ini dirancang untuk bekerja secara terintegrasi, tetapi tetap terisolasi untuk memastikan keamanan dan stabilitas sistem.



Gambar 2.2: Diagram Isi Kontainer

Selain itu, kontainer *Single FE Container* terdiri dari *App Frontend*, yaitu antarmuka pengguna berbasis web yang berfungsi sebagai titik interaksi utama bagi pengguna aplikasi. Antar-

muka ini dirancang dengan teknologi modern untuk memastikan pengalaman pengguna yang responsif, cepat, dan intuitif. Semua komunikasi antara *frontend* dan *backend* dikelola melalui protokol yang aman, memastikan integritas dan kerahasiaan data. Dengan pendekatan ini, arsitektur aplikasi PraktikaApp tidak hanya mendukung integrasi yang mudah antar layanan, tetapi juga memfasilitasi pengelolaan, pengujian, dan penerapan sistem di lingkungan pengembangan maupun produksi. Keunggulan ini memastikan bahwa aplikasi dapat terus berkembang seiring dengan kebutuhan pengguna dan perubahan teknologi tanpa mengorbankan stabilitas atau efisiensi.

2.4 Pengembangan Backend

Pengembangan *backend* PraktikaApp dibatasi dalam dua langkah utama, yaitu pengembangan struktur kode dan pengujian fungsional. Pengembangan struktur kode dilakukan dengan memperhatikan prinsip-prinsip desain perangkat lunak yang baik, seperti modularitas, kohesi, dan dekapsulasi. Struktur kode yang baik memastikan bahwa sistem dapat dikembangkan, dikelola, dan diperbarui dengan mudah di masa depan. Selain itu, pengujian fungsional dilakukan untuk memverifikasi bahwa setiap fitur berjalan sesuai dengan spesifikasi yang ditentukan. Pengujian ini mencakup pengujian *unit*, *integration*, dan *end-to-end* untuk memastikan bahwa setiap komponen berinteraksi dengan benar dan menghasilkan hasil yang diharapkan.

2.4.1 Pengembangan Struktur Kode

Struktur kode PraktikaApp dibatasi dalam dua bagian, yaitu *API* dan *Database*. Pengembangan *API* dilakukan dengan memperhatikan prinsip *RESTful API* yang baik, seperti penggunaan metode HTTP yang tepat, pengelolaan status kode yang konsisten, dan dokumentasi yang jelas. Struktur *API* dirancang untuk mendukung operasi CRUD (Create, Read, Update, Delete) yang efisien, serta menyediakan antarmuka yang intuitif bagi pengguna. Selain itu, pengembangan *database* dilakukan dengan memperhatikan desain basis data yang baik, seperti normalisasi, indeksasi, dan relasi yang tepat. Struktur *database* dirancang untuk mendukung operasi pengelolaan data yang efisien, aman, dan terukur.

2.4.2 Pengujian Fungsional

Pengujian fungsional PraktikaApp dilakukan dengan menggunakan framework *Jest* dan *Supertest* yang memudahkan dalam pengujian *unit* dan *integration*. Pengujian ini mencakup pengujian operasi CRUD, autentikasi pengguna, otorisasi akses, serta pengelolaan sesi dan antrian pekerjaan. Setiap pengujian dirancang untuk memverifikasi bahwa setiap fitur berjalan sesuai dengan spesifikasi yang ditentukan, serta memberikan umpan balik yang jelas atas setiap perubahan kode. Dengan pendekatan ini, pengembangan *backend* PraktikaApp menjadi lebih terstruktur, terukur, dan andal.

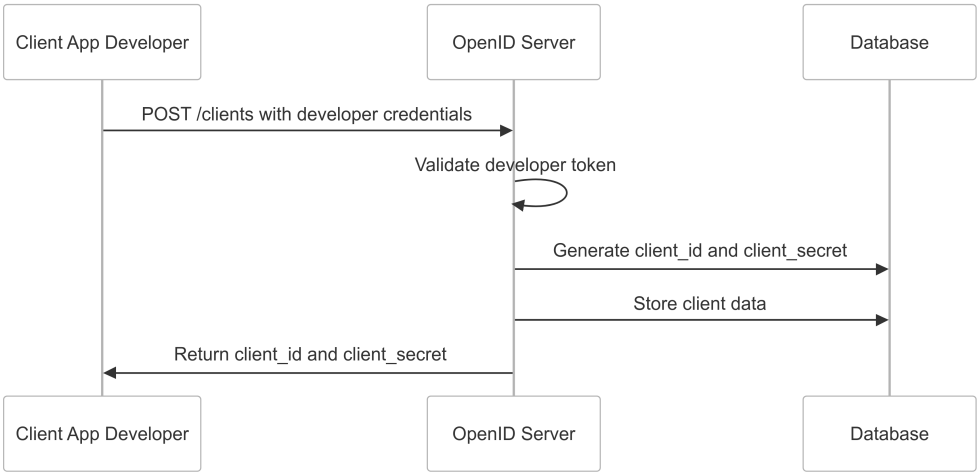
2.5 Implementasi PraktikaSSO

Implementasi PraktikaSSO memanfaatkan OpenID Connect dan OAuth 2.0 sebagai dasar untuk mengelola registrasi klien dan autentikasi pengguna secara aman. Sistem ini dirancang untuk memberikan pengalaman pengguna yang efisien dan nyaman, sekaligus memastikan keamanan data dan integritas sistem. Proses ini terdiri dari dua alur utama, yaitu registrasi klien dan autentikasi pengguna, yang masing-masing dijelaskan secara rinci berikut ini.

2.5.1 Alur Registrasi Klien

Registrasi klien merupakan langkah awal untuk mengintegrasikan aplikasi eksternal dengan PraktikaSSO. Proses ini diawali oleh pengembang aplikasi yang mengirimkan permintaan POST ke endpoint `/clients` di server OpenID dengan menyertakan kredensial pengembang. Server OpenID memvalidasi token pengembang untuk memastikan otentikasi dan otorisasi per-

mintaan. Validasi ini penting untuk menjaga keamanan sistem dan mencegah akses tidak sah. Setelah token divalidasi, server OpenID menghasilkan *client_id* dan *client_secret* yang unik untuk setiap klien. Data ini disimpan dengan aman di basis data, memastikan bahwa setiap klien dapat dikenali dan dilacak dengan jelas. Akhirnya, server mengirimkan *client_id* dan *client_secret* kepada pengembang sebagai respons. Diagram alur registrasi klien ditunjukkan pada Gambar 2.3.



Gambar 2.3: Diagram Alur Registrasi Klien

2.5.2 Alur Autentikasi Pengguna

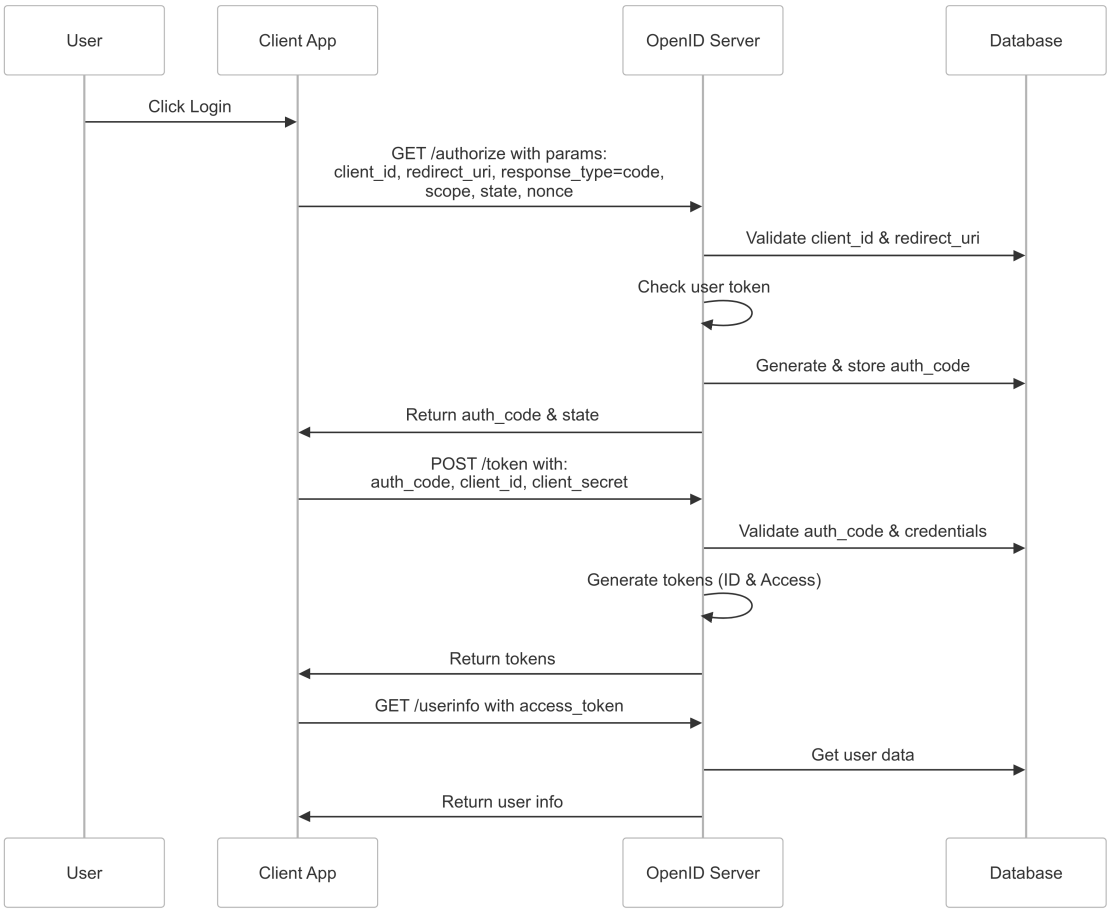
Setelah klien terdaftar, pengguna dapat mengakses aplikasi melalui alur autentikasi yang dirancang menggunakan protokol OpenID Connect dan OAuth 2.0. Alur ini bertujuan untuk memastikan bahwa hanya pengguna yang sah dapat mengakses sumber daya yang dilindungi. Proses autentikasi dimulai ketika pengguna menekan tombol login pada aplikasi klien. Pada tahap ini, aplikasi klien mengirimkan permintaan GET ke endpoint `/authorize` di server OpenID. Permintaan ini mencakup beberapa parameter penting, yaitu *client_id*, *redirect_uri*, *response_type=code*, *scope*, *state*, dan *nonce*. Parameter *client_id* digunakan untuk mengidentifikasi aplikasi klien, sementara *redirect_uri* menentukan URL tempat pengguna akan diarahkan setelah proses autentikasi berhasil. Parameter *response_type=code* menunjukkan bahwa alur yang digunakan adalah Authorization Code Flow, yang merupakan salah satu alur OAuth 2.0 yang paling aman. Parameter *scope* mendefinisikan tingkat akses yang diminta oleh aplikasi, seperti akses ke data pengguna. Sementara itu, *state* dan *nonce* digunakan untuk mencegah serangan replay dan memastikan integritas proses autentikasi.

Setelah permintaan diterima, server OpenID memvalidasi parameter *client_id* dan *redirect_uri* terhadap data yang tersimpan di basis data. Validasi ini penting untuk memastikan bahwa permintaan berasal dari aplikasi klien yang telah terdaftar dan disetujui. Jika validasi berhasil, server memeriksa apakah token pengguna yang sah sudah ada. Jika token tidak ditemukan atau telah kedaluwarsa, server meminta pengguna untuk melakukan autentikasi ulang dengan memasukkan kredensial mereka, seperti email dan kata sandi. Setelah pengguna berhasil diautentikasi, server OpenID menghasilkan *auth_code* (kode otorisasi) yang bersifat sementara. Kode ini disimpan di basis data untuk keperluan validasi lebih lanjut dan dikirimkan kembali ke aplikasi klien melalui *redirect_uri* yang telah ditentukan sebelumnya.

Pada tahap berikutnya, aplikasi klien mengirimkan permintaan POST ke endpoint `/token`. Permintaan ini mencakup *auth_code* yang diterima sebelumnya, bersama dengan *client_id* dan

client_secret. Server OpenID memvalidasi *auth_code* untuk memastikan bahwa kode tersebut belum digunakan atau kedaluwarsa, serta memeriksa kredensial klien untuk memastikan bahwa permintaan berasal dari aplikasi yang sah. Setelah validasi berhasil, server menghasilkan *access token* dan *ID token*. *Access token* digunakan oleh aplikasi klien untuk mengakses API yang dilindungi, sedangkan *ID token* berisi informasi pengguna dalam format terstruktur yang memungkinkan aplikasi klien untuk mengidentifikasi pengguna tanpa memerlukan panggilan tambahan ke server.

Aplikasi klien kemudian dapat menggunakan *access token* untuk mengakses sumber daya yang dilindungi, seperti data pengguna. Untuk mendapatkan informasi pengguna secara spesifik, aplikasi klien dapat mengirimkan permintaan GET ke endpoint `/userinfo`, dengan menyertakan *access token* dalam header permintaan. Server OpenID memvalidasi *access token* dan mengambil data pengguna dari basis data. Data ini kemudian dikembalikan ke aplikasi klien dalam format JSON yang dapat langsung digunakan untuk menampilkan informasi pengguna atau keperluan lainnya. Alur autentikasi ini dirancang untuk memastikan keamanan dan efisiensi, dengan memanfaatkan fitur enkripsi dalam proses pengiriman data sensitif. Penggunaan *state* dan *nonce* memberikan perlindungan tambahan terhadap serangan seperti *Cross-Site Request Forgery (CSRF)* dan *replay attacks*. Selain itu, validasi berlapis terhadap *client_id*, *redirect_uri*, dan *auth_code* memastikan bahwa hanya aplikasi dan pengguna yang sah yang dapat mengakses sistem. Diagram alur autentikasi pengguna ditampilkan pada Gambar 2.4.



Gambar 2.4: Diagram Alur Autentikasi Pengguna

2.5.3 Pemilihan OpenID Connect dan OAuth 2.0

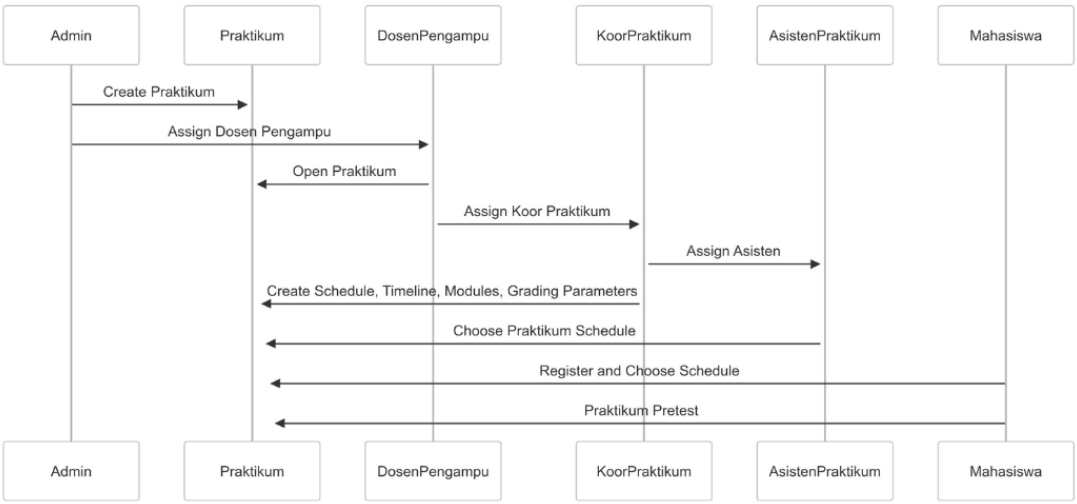
OpenID Connect dan OAuth 2.0 dipilih sebagai protokol dasar untuk PraktikaSSO karena beberapa alasan utama yang berkaitan dengan keamanan, fleksibilitas, dan skalabilitas. OpenID Connect adalah standar terbuka yang memungkinkan autentikasi berbasis web dengan cara yang sederhana namun aman. Dengan menggunakan OpenID Connect, PraktikaSSO dapat menyediakan mekanisme *Single Sign-On (SSO)* yang memungkinkan pengguna mengakses berbagai aplikasi dengan satu kredensial tunggal. Hal ini tidak hanya meningkatkan pengalaman pengguna tetapi juga mengurangi risiko keamanan terkait dengan pengelolaan banyak akun dan kata sandi.

OAuth 2.0, sebagai protokol otorisasi, memberikan fleksibilitas untuk mengatur izin akses ke sumber daya tertentu tanpa mengungkapkan kredensial pengguna. Dengan mendukung alur seperti *Authorization Code Flow*, OAuth 2.0 memungkinkan PraktikaSSO untuk mengelola otorisasi dengan aman di berbagai skenario, termasuk aplikasi berbasis web, seluler, dan layanan API.Kombinasi OpenID Connect dan OAuth 2.0 memberikan fondasi yang kuat untuk PraktikaSSO, memastikan bahwa autentikasi dan otorisasi dilakukan sesuai dengan standar industri. Selain itu, protokol ini memungkinkan PraktikaSSO untuk berkembang dan mendukung lebih banyak aplikasi serta integrasi di masa depan, tanpa mengorbankan keamanan atau efisiensi.

2.6 Implementasi PraktikaApp Backend

Implementasi PraktikaApp Backend merupakan realisasi dari desain arsitektur yang telah dijelaskan sebelumnya, dengan menggunakan *AdonisJS v6* sebagai kerangka kerja utama. Sistem ini dikembangkan untuk mengelola seluruh aspek administrasi praktikum, mulai dari perencanaan hingga evaluasi, dengan memanfaatkan fitur modular dan keamanan berbasis *JWT (JSON Web Token)*. Dengan pendekatan ini, PraktikaApp mampu mengintegrasikan berbagai peran pengguna dan alur kerja yang kompleks ke dalam satu sistem yang terorganisir. Pada tahap implementasi, struktur proyek yang dirancang dengan *starter kit* memungkinkan pengembangan modul berbasis *Domain-Driven Design (DDD)*. Modul ini mencakup fitur utama seperti pembuatan praktikum, penjadwalan, penilaian, dan pengelolaan pengguna. Integrasi dengan *PostgreSQL* melalui *Lucid ORM* memungkinkan pengelolaan data yang efisien, termasuk penyimpanan informasi praktikum, jadwal, modul, nilai, serta data pengguna. Selain itu, *Redis* digunakan untuk meningkatkan kinerja melalui penyimpanan sesi berbasis memori dan pengelolaan antrian pekerjaan, yang sangat berguna dalam memproses permintaan pengguna secara paralel.

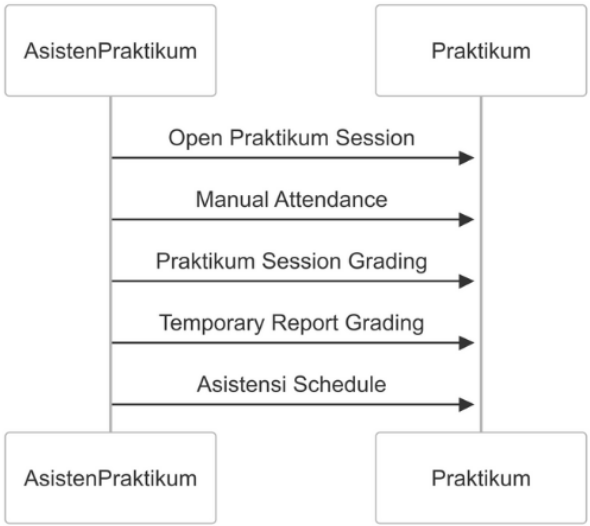
Alur implementasi PraktikaApp Backend dimulai dengan admin yang membuat praktikum dan menetapkan dosen pengampu sebagai pengelola utama. Proses ini mencerminkan fleksibilitas yang ditawarkan oleh kerangka kerja *AdonisJS*, di mana modul *CRUD* (Create, Read, Update, Delete) diterapkan untuk mengelola data praktikum secara dinamis. Setelah dosen pengampu membuka praktikum, sistem mengizinkan mereka untuk menetapkan koordinator praktikum yang bertanggung jawab untuk tugas-tugas operasional, seperti pembuatan jadwal, penambahan modul, dan pengaturan parameter penilaian.Selama pelaksanaan praktikum, asisten praktikum menggunakan modul *backend* untuk mengelola sesi praktikum, termasuk membuka sesi, mencatat kehadiran, dan memberikan penilaian individu. Sistem ini didukung oleh pipeline *CI/CD* yang mengotomatisasi pengujian dan penerapan kode, memastikan bahwa setiap pembaruan tidak mengganggu fungsionalitas yang ada. Diagram alur awal yang mencakup peran admin, dosen pengampu, koordinator praktikum, dan mahasiswa ditunjukkan pada Gambar 2.5.



Gambar 2.5: Diagram Alur Awal PraktikaApp

Sistem PraktikaApp Backend juga dirancang untuk menangani perubahan jadwal (reschedule). Fitur ini mengandalkan kekuatan *AdonisJS* dalam memproses permintaan dinamis, di mana mahasiswa yang berhalangan hadir dapat mengajukan reschedule melalui sistem, yang kemudian akan diproses oleh asisten praktikum. Jadwal baru disimpan ke dalam *PostgreSQL* dan diinformasikan kepada mahasiswa melalui pengumuman otomatis.

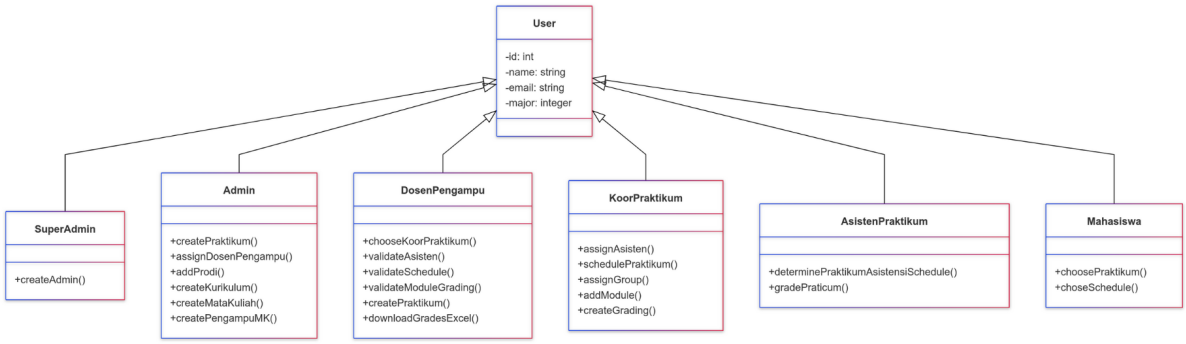
Diagram alur pelaksanaan praktikum, seperti yang ditampilkan pada Gambar 2.6, mencerminkan kemampuan sistem untuk mendukung interaksi yang berkelanjutan antara asisten praktikum dan mahasiswa. Setiap penilaian, baik untuk pelaksanaan praktikum maupun laporan sementara, disimpan dengan aman menggunakan mekanisme *JWT*, memastikan bahwa hanya pengguna yang berwenang dapat mengakses data tersebut.



Gambar 2.6: Diagram Alur Pelaksanaan Praktikum

Dalam proses penilaian akhir, koordinator praktikum memvalidasi semua nilai yang telah dimasukkan oleh asisten praktikum sebelum memberikan persetujuan. Setelah itu, dosen pengampu dapat melakukan tinjauan akhir dan mengesahkan nilai mahasiswa. Proses ini memanfaatkan pengelolaan data relasional di *PostgreSQL* untuk memastikan bahwa semua data

penilaian terdokumentasi dengan baik dan dapat diaudit. Sistem ini dirancang untuk mengelola berbagai peran pengguna dalam administrasi praktikum, dengan akses dan fungsi yang berbeda berdasarkan jenis pengguna, seperti yang ditunjukkan pada Gambar 2.7.



Gambar 2.7: Diagram Hak Akses Pengguna pada PraktikaApp

Diagram pada Gambar 2.7 menunjukkan bahwa PraktikaApp memiliki hierarki pengguna yang berbasis pada satu entitas utama, yaitu *User*. Kelas *User* memiliki atribut inti seperti *id*, *name*, *email*, dan *major*, yang dikelola di *PostgreSQL* sebagai database utama. Berdasarkan atribut ini, pengguna dapat diberi peran yang lebih spesifik, termasuk *SuperAdmin*, *Admin*, *DosenPengampu*, *KoorPraktikum*, *AsistenPraktikum*, dan *Mahasiswa*. Setiap peran memiliki akses ke fungsi tertentu yang diimplementasikan dalam modul *backend*. Sebagai contoh, *SuperAdmin* memiliki hak penuh untuk membuat pengguna baru, termasuk *Admin*, dan mengelola semua data dalam sistem. *Admin* bertanggung jawab untuk membuat praktikum, menetapkan dosen pengampu, menambahkan program studi, serta membuat kurikulum dan mata kuliah. *DosenPengampu* memiliki akses untuk memilih koordinator praktikum, memvalidasi jadwal, modul, dan penilaian, serta mengunduh nilai mahasiswa dalam format Excel. *KoorPraktikum* bertugas untuk menetapkan asisten praktikum, menjadwalkan kegiatan praktikum, menambahkan modul, serta membuat parameter penilaian. *AsistenPraktikum* bertanggung jawab untuk menentukan jadwal praktikum dan asistensi, serta memberikan penilaian terhadap pelaksanaan praktikum. Sementara itu, *Mahasiswa* hanya dapat memilih praktikum dan jadwal yang sesuai dengan kebutuhan mereka.

Implementasi fungsi ini menggunakan *AdonisJS*, yang menyediakan kemampuan modular untuk menangani logika bisnis secara terpisah berdasarkan peran pengguna. Sistem otentikasi menggunakan *JWT (JSON Web Token)* dari Praktika SSO memastikan bahwa setiap pengguna hanya dapat mengakses fitur yang sesuai dengan perannya. Integrasi dengan *Redis* digunakan untuk mempercepat proses otentikasi dan pengelolaan sesi pengguna. Melalui arsitektur ini, PraktikaApp Backend memberikan solusi yang terstruktur untuk mendukung berbagai alur kerja praktikum, mulai dari pembuatan hingga evaluasi. Setiap modul backend dirancang untuk mendukung komunikasi yang aman dan efisien antara klien dan server, memastikan bahwa data pengguna dan operasi sistem tetap terjaga integritasnya.

2.7 Implementasi dan Alur Kerja Sistem PraktikaTelelab

PraktikaTelelab merupakan platform yang dirancang untuk mendukung eksperimen praktikum berbasis simulasi dan IoT, dengan memanfaatkan teknologi modern seperti *AdonisJS*, *PostgreSQL*, dan *Redis*. Sistem ini dirancang agar mahasiswa dapat menjalankan eksperimen secara fleksibel, baik secara daring melalui simulasi maupun luring dengan perangkat keras

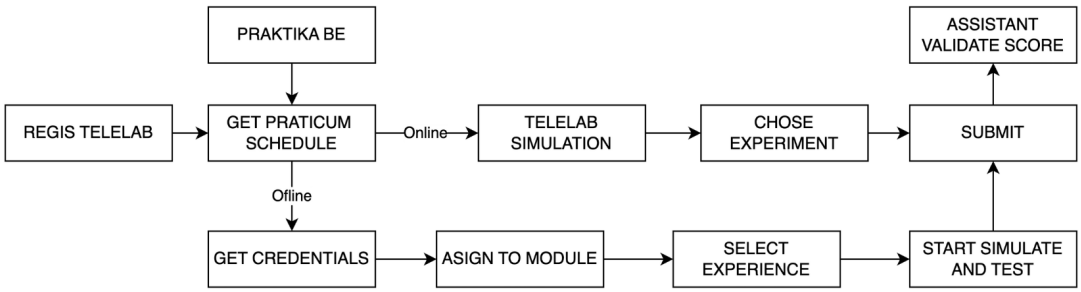
berbasis *ESP32* yang terpasang pada modul praktikum rangkaian digital. Berikut adalah penjelasan mendalam mengenai alur kerja sistem serta implementasi di sisi *backend*.

2.7.1 Alur Kerja Sistem PraktikaTelelab

Peserta yang telah terdaftar dalam praktikum rangkaian digital di PraktikaApp akan dapat mengakses PraktikaTelelab. PraktikaTelelab kemudian melakukan pengecekan terhadap jadwal dan jenis praktikum, baik yang dilakukan secara daring maupun luring, berdasarkan data yang terintegrasi dengan PraktikaApp. Setelah melakukan sinkronisasi data jadwal dengan PraktikaApp, peserta dapat memilih modul eksperimen yang ingin dijalankan, baik berupa simulasi daring maupun eksperimen luring.

Jika peserta mendapatkan giliran praktikum dalam mode luring, PraktikaTelelab memanfaatkan perangkat keras berbasis IoT, seperti *ESP32*, yang dipasang pada modul rangkaian digital. Mahasiswa menerima kredensial unik yang dihasilkan oleh backend, yang memungkinkan mereka mengakses modul eksperimen secara fisik. Kredensial ini disimpan sementara di *Redis* untuk memastikan akses yang cepat dan aman. Mahasiswa kemudian dapat menjalankan eksperimen menggunakan perangkat keras, dan hasil eksperimen dikirim kembali ke sistem backend untuk diverifikasi. Backend memastikan bahwa hasil eksperimen luring memiliki integritas yang setara dengan eksperimen simulasi.Sementara itu, jika praktikum dilaksanakan dalam mode daring, mahasiswa dapat langsung mengakses modul simulasi praktikum melalui antarmuka yang disediakan oleh sistem. Setelah memilih modul eksperimen yang diinginkan, mahasiswa dapat memulai simulasi dan menguji eksperimen secara langsung. Sistem ini memungkinkan mahasiswa untuk melakukan uji coba sebanyak yang diperlukan sebelum menyerahkan hasil akhir. Setelah eksperimen selesai, hasil simulasi dikirim melalui sistem untuk diverifikasi dan divalidasi oleh asisten praktikum.

Diagram alur kerja sistem PraktikaTelelab dapat dilihat pada Gambar 2.8.



Gambar 2.8: Diagram Alur Kerja Sistem PraktikaTelelab

2.7.2 Implementasi Backend PraktikaTelelab

Pada sisi *backend*, PraktikaTelelab dirancang menggunakan kerangka kerja *AdonisJS*, yang mendukung pengelolaan logika bisnis secara modular. Backend diimplementasikan dengan pendekatan *Domain-Driven Design (DDD)*, yang memungkinkan setiap modul—seperti pendaftaran, pengelolaan jadwal, dan penilaian eksperimen—untuk dikelola secara independen namun tetap terintegrasi dalam satu sistem.

Basis data relasional *PostgreSQL* digunakan untuk menyimpan data utama, seperti informasi mahasiswa, jadwal praktikum, hasil eksperimen, dan penilaian. Dengan dukungan fitur-fitur seperti indeks, relasi antar tabel, dan transaksi, *PostgreSQL* memastikan data disimpan dengan konsisten dan dapat diakses secara efisien.

Untuk meningkatkan performa, *Redis* digunakan sebagai *caching layer*. Teknologi ini mempermudah pengelolaan sesi pengguna, menyimpan kredensial sementara, dan memper-

cepat akses ke data yang sering digunakan selama proses simulasi. *Redis* juga berperan penting dalam menjaga responsivitas sistem, meskipun ada banyak permintaan yang datang secara bersamaan.

2.7.3 Validasi dan Penilaian Hasil Eksperimen

Proses validasi dan penilaian hasil eksperimen sepenuhnya dilakukan di sisi *backend*. Data hasil eksperimen, baik yang berasal dari simulasi maupun perangkat keras, akan diverifikasi untuk memastikan keaslian dan integritasnya sebelum diteruskan kepada asisten praktikum. Asisten menggunakan antarmuka yang terhubung dengan backend untuk memberikan penilaian berdasarkan parameter yang telah ditetapkan. Hasil penilaian ini kemudian disimpan di *PostgreSQL* dan dapat diakses oleh dosen pengampu atau koordinator praktikum untuk evaluasi lebih lanjut.

2.8 Implementasi keamanan dalam sistem *Backend*

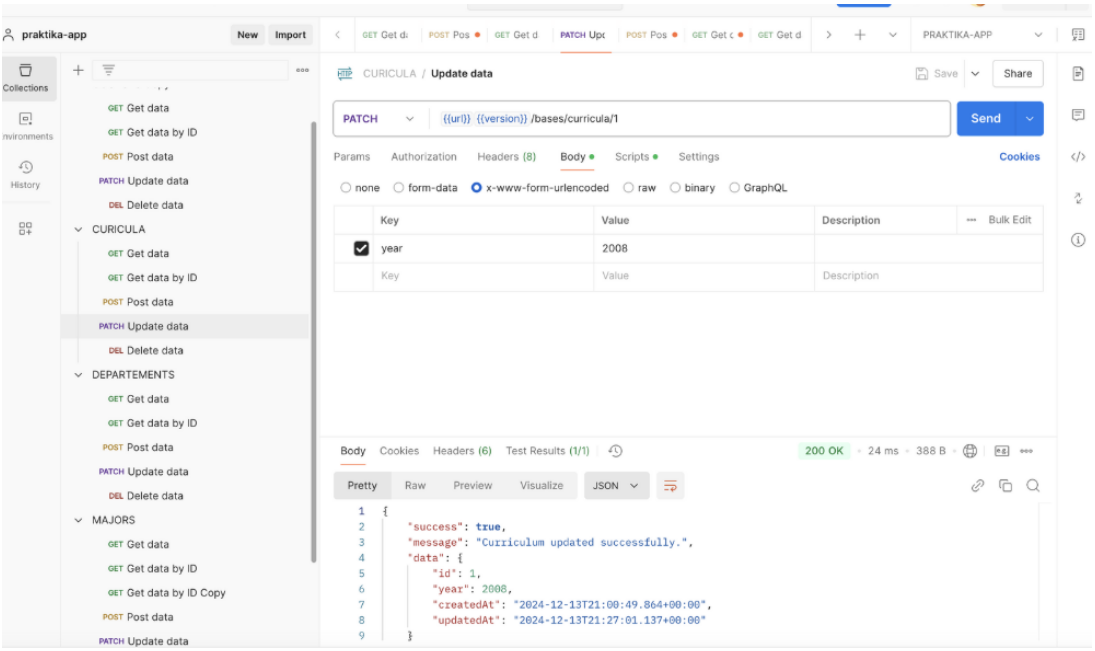
Keamanan sistem *backend* PraktikaApp dan PraktikaTelelab dijamin melalui beberapa lapisan perlindungan, mulai dari otentikasi pengguna hingga enkripsi data. Sistem ini dirancang dengan memperhatikan prinsip-prinsip keamanan informasi yang baik, seperti *least privilege*, *defense in depth*, dan *secure by design*. Berikut adalah beberapa langkah yang diambil untuk memastikan keamanan sistem *backend*. Sistem juga menggunakan OAuth 2.0 dan OpenID Connect untuk mengelola autentikasi pengguna dan otorisasi akses ke sumber daya yang dilindungi. Protokol ini memastikan bahwa setiap pengguna hanya dapat mengakses data yang sesuai dengan peran dan hak aksesnya, serta memberikan lapisan keamanan tambahan melalui enkripsi data dan validasi token. Selain itu juga diimplementasikan *CORS* untuk membatasi akses lintas domain dan mencegah serangan *Cross-Site Scripting (XSS)*.

2.8.1 Pengujian Backend

Sistem *backend* PraktikaApp dan PraktikaTelelab diuji secara menyeluruh melalui pengujian manual dan otomatis untuk memastikan fungsionalitas, keamanan, dan kinerja sistem yang optimal. Pengujian manual dilakukan oleh tim pengembang untuk memverifikasi apakah semua fungsi sistem berjalan sesuai dengan harapan dan untuk mendeteksi potensi masalah yang mungkin tidak terdeteksi oleh pengujian otomatis. Proses ini melibatkan pengujian antar muka pengguna, pengujian alur fungsionalitas, serta pengujian terhadap aspek keamanan sistem.Selain pengujian manual, pengujian otomatis dilakukan dengan menggunakan alat seperti *Jest* dan *Supertest*. *Jest* digunakan untuk pengujian unit untuk memastikan bahwa setiap fungsi atau metode dalam kode backend berfungsi sesuai dengan yang diinginkan. *Supertest* digunakan untuk pengujian API yang lebih mendalam, yang melibatkan pengujian end-to-end untuk memastikan bahwa seluruh alur sistem berjalan dengan baik dari awal hingga akhir.Pengujian ini mencakup beberapa jenis pengujian, seperti: - *Unit testing*: Menguji fungsi atau metode individual untuk memastikan bahwa komponen backend bekerja dengan benar. - *Integration testing*: Menguji bagaimana komponen backend berinteraksi satu sama lain, memastikan bahwa data yang diproses dapat mengalir dengan benar antar modul. - *End-to-end testing*: Menguji seluruh alur sistem, dari autentikasi pengguna hingga pengambilan data dan pemrosesan hasil eksperimen.

Hasil dari pengujian otomatis ini memberikan umpan balik cepat tentang status fungsionalitas dan kualitas kode, yang memungkinkan tim pengembang untuk segera mengidentifikasi dan memperbaiki bug. Setiap kali ada pembaruan kode atau penambahan fitur, pengujian otomatis dilakukan untuk memastikan bahwa pembaruan tidak menyebabkan kerusakan pada sistem yang sudah ada. Selain itu, pengujian ini juga digunakan untuk meningkatkan kualitas kode secara keseluruhan dan menjaga konsistensi antar modul. Pengujian API juga dilakukan

menggunakan *Postman*, di mana setiap endpoint API diuji untuk memastikan bahwa mereka mengembalikan respons yang benar dan sesuai dengan spesifikasi yang telah ditentukan. Dengan *Postman*, tim pengembang dapat mengirimkan permintaan HTTP ke server backend, menguji berbagai skenario respons (seperti status 200 untuk respons yang sukses, 404 untuk sumber daya yang tidak ditemukan, dll.), dan memverifikasi data yang diterima dalam respons. Proses pengujian ini membantu mengidentifikasi kesalahan dalam pengelolaan permintaan atau pengolahan data yang dikirimkan ke API.Gambar berikut menunjukkan contoh pengujian API menggunakan *Postman* untuk menguji berbagai endpoint dalam sistem backend PraktikaTelelab dan PraktikaApp:



Gambar 2.9: Pengujian API menggunakan Postman

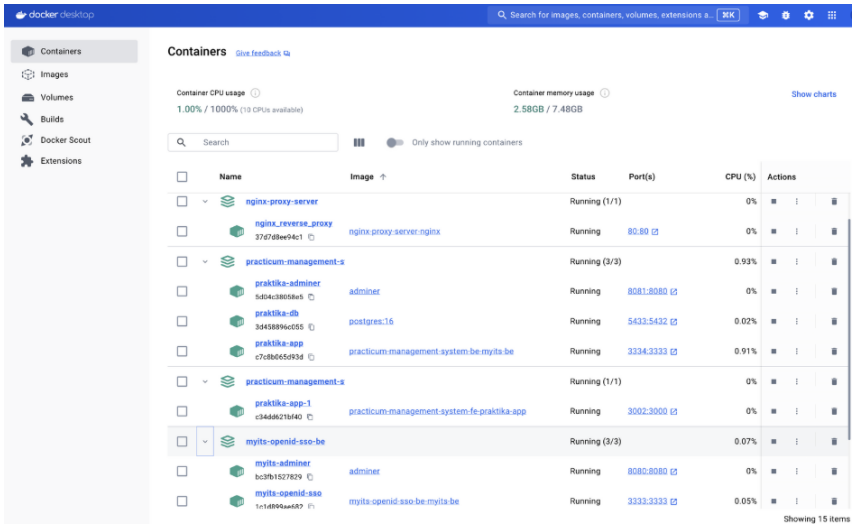
Gambar di atas menunjukkan bagaimana *Postman* digunakan untuk mengirim permintaan API ke server, memverifikasi respons yang diterima, dan memastikan bahwa setiap endpoint bekerja sesuai dengan yang diharapkan. Pengujian ini melibatkan berbagai skenario, seperti mengirimkan data valid, mengirimkan data yang tidak lengkap, atau menguji endpoint dengan parameter yang tidak valid.Melalui pengujian manual, otomatis, dan pengujian API menggunakan *Postman*, sistem backend PraktikaApp dan PraktikaTelelab dapat dijaga kualitasnya dan dijamin berfungsi sesuai dengan spesifikasi yang telah ditentukan, serta memberikan pengalaman pengguna yang optimal.

2.9 *Local Development*

Proses deployment dilakukan menggunakan Docker dan Nginx, seperti yang ditunjukkan pada gambar berikut. Pada tahap ini, *port mapping* dilakukan untuk mengarahkan trafik yang datang ke domain yang sudah dikonfigurasi ke kontainer yang sesuai melalui Nginx. Misalnya, jika aplikasi frontend dan backend menggunakan port yang berbeda, Nginx akan melakukan mapping antara port pada mesin host dengan port yang digunakan oleh kontainer Docker.

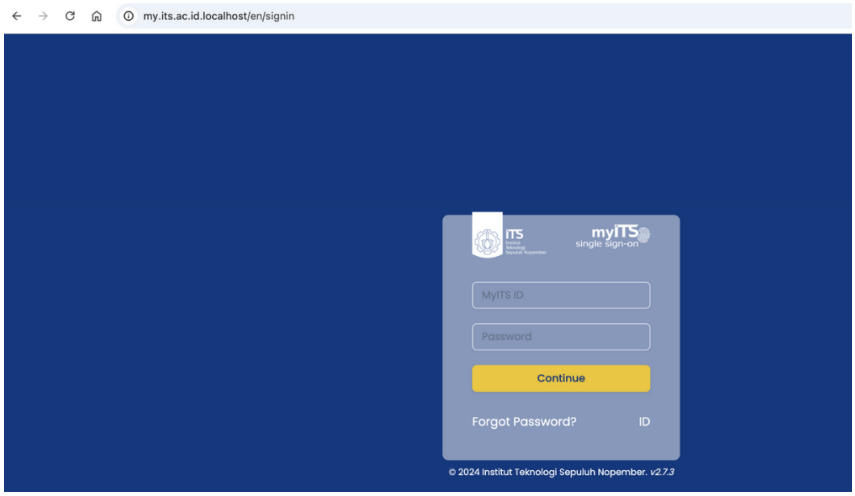
Sebagai contoh, jika `service_name.praktika.its.id.localhost` merujuk ke aplikasi frontend, maka Nginx akan mengarahkan trafik yang masuk pada port 3000 ke port yang digunakan oleh container frontend. Begitu juga dengan aplikasi backend yang dapat

diakses melalui subdomain `api.service_name.praktika.its.id.localhost`, yang akan diarahkan ke port backend di dalam kontainer Docker.Dalam hal ini, domain yang digunakan untuk mengakses aplikasi adalah `service_name.praktika.its.id.localhost` atau `api.service_name.praktika.its.id.localhost`, di mana `service_name` adalah nama dari aplikasi, misalnya `praktika_app`, `praktika_sso`, dan `praktika_telelab`. Setiap aplikasi ini akan memiliki kontainer-nya masing-masing, baik untuk bagian *backend* (BE) maupun *frontend* (FE), seperti yang ditunjukkan pada gambar di bawah 2.10.



Gambar 2.10: Diagram Deployment Sistem dengan Docker dan Nginx

Kontainer Docker akan dibungkus dengan Kubernetes, sehingga sistem dapat dikelola dengan lebih efisien dan dapat diskalakan sesuai kebutuhan. Dengan konfigurasi ini, aplikasi akan dapat diakses di browser menggunakan domain name yang telah dikonfigurasi pada Nginx, seperti pada gambar di bawah.



Gambar 2.11: Aplikasi PraktikaSSO pada browser

[Halaman ini sengaja dikosongkan]

BAB 3 KESIMPULAN DAN SARAN

3.1 Kesimpulan

Berdasarkan hasil pengembangan dan implementasi sistem, dapat disimpulkan bahwa sistem backend yang dibuat untuk aplikasi PraktikaApp, PraktikaSSO, dan PraktikaTelelab berjalan dengan baik. Sistem ini telah berhasil di-deploy secara lokal dan dilengkapi dengan berbagai fitur keamanan yang mendukung, seperti *OpenID Auth* untuk otentikasi pengguna. Selain itu, sistem backend ini dibangun dengan pendekatan *microservice*, yang memungkinkan pengelolaan dan pengembangan aplikasi secara modular dan terpisah sesuai dengan fungsinya.

Selain itu, untuk memastikan kemudahan deployment dan skalabilitas, aplikasi ini dibungkus menggunakan Docker dan Kubernetes. Dengan demikian, sistem dapat dengan mudah di-deploy pada berbagai lingkungan dan mendukung proses pengelolaan kontainer yang efisien. Fitur keamanan lain yang diterapkan adalah Cross-Origin Resource Sharing (*CORS*), yang memberikan perlindungan terhadap akses tidak sah dari sumber yang berbeda, memastikan aplikasi tetap aman dan terkontrol.

3.2 Saran

Sebagai pengembangan lebih lanjut, ada beberapa saran yang dapat dipertimbangkan untuk meningkatkan kualitas dan fungsionalitas sistem, antara lain:

- Pengembangan aplikasi **Praktika App** dapat diperluas untuk mencakup berbagai komponen selain rangkaian digital, sehingga dapat mencakup berbagai aspek yang lebih luas dalam dunia pendidikan dan teknologi.
- Melakukan *deployment* aplikasi di lingkungan *production* yang saat ini belum dilakukan, agar aplikasi dapat digunakan oleh pengguna secara lebih luas dan memberikan feedback yang lebih komprehensif terkait performa dan kestabilan aplikasi di dunia nyata.
- Menambahkan fitur baru yang dapat memperkaya pengalaman pengguna, seperti analisis data penggunan aplikasi atau integrasi dengan layanan pihak ketiga untuk meningkatkan fungsionalitas sistem.

[Halaman ini sengaja dikosongkan]