PENGEMBANGAN RESTFUL API PADA SISTEM REMUNERASI DOSEN UNIVERSITAS LAMPUNG BIDANG PENGABDIAN DAN PENUNJANG DENGAN AUTENTIKASI JSON WEB TOKEN (JWT)

(Skripsi)

Oleh

FATHIMAH ABIYYI KHAIRUNNISA
NPM 2117051088

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

PENGEMBANGAN RESTFUL API PADA SISTEM REMUNERASI DOSEN UNIVERSITAS LAMPUNG BIDANG PENGABDIAN DAN PENUNJANG DENGAN AUTENTIKASI JSON WEB TOKEN (JWT)

Oleh

FATHIMAH ABIYYI KHAIRUNNISA

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

ABSTRAK

PENGEMBANGAN RESTFUL API PADA SISTEM REMUNERASI DOSEN UNIVERSITAS LAMPUNG BIDANG PENGABDIAN DAN PENUNJANG DENGAN AUTENTIKASI JSON WEB TOKEN (JWT)

Oleh

FATHIMAH ABIYYI KHAIRUNNISA

Sistem remunerasi dosen Universitas Lampung berperan penting dalam pengelolaan data kinerja dosen, khususnya pada bidang pengabdian dan penunjang. Namun, pemanfaatan datanya masih terbatas, sehingga diperlukan pengembangan RESTful API dengan menggunakan autentikasi JSON Web Token (JWT). API dibangun mengikuti prinsip REST, yaitu *stateless*, *client-server*, *uniform interface*, *cacheable*, dan *layered system*, serta dilengkapi autentikasi berbasis JWT untuk menjamin keamanan proses pertukaran informasi sehingga integritas dan kerahasiaan data tetap terjaga. Hasil pengujian fungsional menunjukkan seluruh *endpoint* berjalan sesuai rancangan, termasuk mekanisme autentikasi dan otorisasi. Adapun pengujian performa dengan Apache JMeter membuktikan bahwa sistem tetap andal meskipun berada pada beban tinggi, yaitu pada simulasi 225 *requests per second* dengan hasil *error rate* 0,00% dan rata-rata waktu respons 122 ms. Sementara itu, pengujian keamanan dengan OWASP ZAP hanya menghasilkan dua peringatan bersifat *informational* tanpa kerentanan kritis. Dengan demikian, RESTful API ini tidak hanya memenuhi aspek fungsional, tetapi juga memiliki performa yang efisien, keamanan andal, serta mencapai tujuan penelitian yaitu memberikan kemudahan akses data kinerja dosen untuk berbagai proses evaluasi, seperti akreditasi maupun pelaporan institusional lainnya.

Kata kunci: RESTful API, Data, JWT, Integrasi, Sistem Informasi.

ABSTRACT

DEVELOPMENT OF RESTFUL API IN LECTURER REMUNERATION SYSTEM AT THE UNIVERSITY OF LAMPUNG FOR COMMUNITY SERVICE AND SUPPORTING ACTIVITIES WITH JSON WEB TOKEN (JWT) AUTHENTICATION

By

FATHIMAH ABIYYI KHAIRUNNISA

The lecturer remuneration system at Universitas Lampung plays an essential role in managing lecturer performance data, particularly in the areas of community service and supporting activities. However, the utilization of this data remains limited, making it necessary to develop a RESTful API with JSON Web Token (JWT) authentication. The API was built following REST principles—stateless, client-server, uniform interface, cacheable, and layered system—while incorporating JWT-based authentication to ensure secure information exchange, maintaining both data integrity and confidentiality. Functional testing showed that all endpoints operated as designed, including authentication and authorization mechanisms. Performance testing using Apache JMeter demonstrated that the system remained reliable under high load, handling 225 requests per second with an error rate of 0.00% and an average response time of 122 ms. Furthermore, security testing with OWASP ZAP resulted in only two informational alerts, with no critical vulnerabilities detected. Therefore, this RESTful API not only meets functional requirements but also delivers efficient performance, robust security, and fulfills the research objective of facilitating easier access to lecturer performance data for various evaluation processes, such as accreditation and institutional reporting.

Keywords: RESTful API, JWT, Data, Integration, Information System.

Judul Skripsi	:	**PENGEMBANGAN RESTFUL API PADA SISTEM REMUNERASI DOSEN UNIVERSITAS LAMPUNG BIDANG PENGABDIAN DAN PENUNJANG DENGAN AUTENTIKASI JSON WEB TOKEN (JWT)**
Nama Mahasiswa	:	Fathimah Abiyyi Khairunnisa
Nomor Pokok Mahasiswa	:	2117051088
Program Studi	:	Ilmu komputer (S-1)
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

M. Iqbal Parabi, S.Si., M.T.
NIP. 199011302015041002

Igit Sabda Ilman, M.Kom.
NIK. 232111960101101

MENGETAHUI

2. Ketua Jurusan Ilmu Komputer
FMIPA Universitas Lampung

Dwi Sakethi, S.Si., M.Kom.
NIP. 196806111998021001

3. Ketua Program Studi Ilmu Komputer
FMIPA Universitas Lampung

Tristiyanto, S.Kom., M.I.S., Ph.D.
NIP. 198104142005011001

MENGESAHKAN

1. Tim Penguji

Ketua Penguji : **M. Iqbal Parabi, S.SI., M.T.**

Sekretaris Penguji : **Igit Sabda Ilman, M.Kom.**

Penguji Utama : **Rico Andrian, S.Si., M.Kom.**

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Dr. Eng. Heri Satria, S.Si., M.Si.
NIP. 197110012005011002

Tanggal Lulus Ujian Skripsi: 11 Agustus 2025

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Fathimah Abiyyi Khairunnisa

NPM : 2117051088

Dengan ini menyatakan bahwa skripsi saya yang berjudul "**Pengembangan RESTful API pada Sistem Remunerasi Dosen Universitas Lampung Bidang Pengabdian dan Penunjang dengan Autentikasi JSON *Web Token* (JWT)**" merupakan hasil karya saya sendiri dan bukan karya orang lain. Seluruh tulisan yang tertuang di skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti skripsi saya merupakan hasil jiplakan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi sesuai hukum yang berlaku.

Bandar Lampung, 28 Agustus 2025

Yang menyatakan,

Fathimah Abiyyi Khairunnisa

NPM. 2117051088

RIWAYAT HIDUP

Penulis dilahirkan di Jakarta Selatan pada hari Kamis, 8 Agustus 2002, sebagai anak kedua dari dua bersaudara dari Bapak Santosa Nusa Putra dan Ibu Nur Betty. Penulis menyelesaikan pendidikan formal pertama di Raudhatul Athfal (RA) Mulia Az Zahra Kota Tangerang lalu melanjutkan pendidikan Sekolah Dasar di Sekolah Dasar Islam Plus (SDIP) Baitul Maal Kota Tangerang Selatan sampai pada Tahun 2012 yang kemudian pindah ke Sekolah Dasar (SD) Fadilah Kota Tangerang Selatan dan lulus pada Tahun 2014. Selanjutnya penulis menempuh pendidikan Sekolah Menengah Pertama di Sekolah Menengah Pertama Islam Terpadu (SMPIT) Insan Mubarak *Boarding School* Kota Jakarta Barat dan lulus pada Tahun 2017, lalu melanjutkan pendidikan ke Sekolah Menengah Atas di Madrasah Aliyah (MA) Sahid *Islamic Boarding School* Kota Bogor dan lulus pada Tahun 2020 yang kemudian bermukim sejenak di Rumah Tahfidz Khodijah dan menyelesaikannya pada Tahun 2021.

Pada Tahun 2021 penulis terdaftar sebagai mahasiswa Jurusan Ilmu komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung melalui jalur SBMPTN. Selama menempuh pendidikan di perguruan tinggi, penulis melakukan beberapa kegiatan antara lain:

1. Menjadi Anggota Muda Ilmu Komputer (ADAPTER) Himpunan Mahasiswa Jurusan Ilmu Komputer pada periode 2021/2022.
2. Menjadi peserta Latihan Manajemen Mahasiswa Ilmu Komputer Tingkat dasar (LKMMIK-TD) pada Tahun 2022.

3. Menjadi Asisten Dosen Jurusan Ilmu Komputer pada mata kuliah Pemrograman Terstruktur Tahun Ajaran 2022/2023.
4. Menjadi anggota Biro Kesekretariatan Himpunan Mahasiswa Jurusan Ilmu Komputer periode 2022 yang kemudian diamanahkan menjadi Sekretaris Biro Kesekretariatan pada periode 2023.
5. Menjadi anggota riset ilmiah Program Riset Independen Merdeka Belajar Kampus Merdeka (MBKM) yang mengangkat topik Sistem Temu Kembali Informasi Publikasi Ilmiah Dosen Universitas Lampung Menggunakan *Vector Space Model* (VSM) dan *Latent Semantic Analysis* (LSA) pada Tahun 2023.
6. Menjadi peserta kegiatan kerja praktik di PT PLN (Persero) UP2D Bandar Lampung pada Tahun Ajaran 2023/2024.
7. Menjadi peserta studi independen Program Studi Independen MBKM *Batch* 6 di Dicoding *Academy* bersama Badan Pariwisata dan Ekonomi Kreatif (Baparekraf) *Digital Talent* dan meraih sertifikat pada *learning path Front-End* dan *Back-End Web Developer* Tahun 2024.
8. Menjadi peserta Kerja Kuliah Nyata (KKN) di Desa Nibung, Gunung Pelindung, Lampung Timur sebagai bentuk pengabdian kepada masyarakat Tahun 2024.

MOTTO

"... Dan Dia (Allah) bersama kamu di mana saja kamu berada, ..."
(Q.S. Al-Hadid : 4)

"Dan janganlah kamu (merasa) lemah, dan janganlah (pula) bersedih hati,
sebab kamu paling tinggi (derajatnya), jika kamu orang beriman"
(Q.S. Ali Imran : 139)

"*The way to know life is to love many things*"
– Van Gogh

"Belajarlah mengucap syukur dari hal-hal baik di hidupmu,
dan belajarlah menjadi kuat dari hal-hal buruk di hidupmu"
– B. J. Habibie

"*Even when I fall and hurt myself, I endlessly run toward my dream*"
– 방탄소년단 (*Epilogue: Young Forever*)

PERSEMBAHAN

Alhamdulillahirobbilalamin

Puji syukur kehadirat Allah Subhanahu Wa Ta'ala atas segala rahmat, nikmat, serta karunia-Nya sehingga skripsi ini dapat diselesaikan dengan sebaik-baiknya.

Shalawat serta salam selalu tercurahkan kepada junjungan
Nabi Agung Muhammad Shalallahu 'Alaihi Wasalam
yang telah menjadi suri tauladan kepada seluruh umatnya.

Kupersembahkan karya ini kepada:

Kedua orang tua dan kakakku tercinta
Yang senantiasa memberikan doa, dukungan, cinta dan kasih sayang tanpa henti.

Seluruh sahabat, teman, sobat karib
Yang telah membersamai dan membantu di seluruh masa perkuliahan.

Almamater yang kubanggakan, Universitas Lampung
Tempat bertumbuh, belajar dan mengemban ilmu.

Serta tak lupa untuk **diriku sendiri**
Sebagai bentuk apresiasi atas perjuangan dan ketekunan yang telah dilakukan.

SANWACANA

Puji syukur kehadirat Allah SWT atas segala rahmat, nikmat, serta karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "**Pengembangan RESTful API pada Sistem Remunerasi Dosen Universitas Lampung Bidang Pengabdian dan Penunjang dengan Autentikasi JSON Web Token (JWT)**". Banyak pihak yang telah membantu penulis dalam menyelesaikan skripsi ini. Dengan segala kerendahan hati dan rasa hormat, penulis mengucapkan terima kasih kepada:

1. Allah Subhanahu Wa Ta'ala, Sang Pemilik kehidupan, yang dengan kasih sayang-Nya telah memberikan kesehatan, kekuatan, serta kesempatan sehingga skripsi ini dapat terselesaikan. Tiada daya dan upaya melainkan atas izin-Nya, tiada ilmu yang tercapai melainkan karena karunia-Nya. Di setiap lelah Dia yang memberikan keteguhan, di setiap bimbang Dia yang menghadirkan jalan, di setiap doa Dia pula yang menghadirkan jawaban. Skripsi ini hanyalah sekelumit usaha yang tak luput dari kekurangan, namun penulis percaya bahwa setiap langkah, setiap huruf dan setiap hasil adalah bagian dari rencana-Nya yang sempurna. Semoga karya sederhana ini bernilai ibadah di sisi-Nya.
2. Ayah, Ibu juga Kakak yang selalu menyayangi dan menyemangati dalam setiap proses yang penulis lalui. Terima kasih atas cinta yang tak pernah surut, doa yang tak pernah putus dan pengorbanan yang tak terhitung nilainya. Terima kasih karena telah menjadi rumah yang paling aman dan nyaman serta selalu memberikan dukungan atas setiap keputusan terbaik yang penulis ambil. Kalian adalah alasan penulis mampu berdiri hingga titik ini. Semoga keberhasilan kecil ini menjadi persembahan tulus untuk cinta yang tak ternilai.
3. Bapak Dr. Eng. Heri. Satria, S.Si., M.Si. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung.

4. Bapak Dwi Sakethi, S.Si., M.Kom. selaku Ketua Jurusan Ilmu Komputer FMIPA Universitas Lampung.
5. Bapak Tristiyanto, S.Kom., M.I.S., Ph.D. selaku Ketua Program Studi Ilmu Komputer (S-1) sekaligus Pembimbing Akademik yang telah senantiasa membimbing penulis selama berkuliah di Jurusan Ilmu Komputer.
6. Bapak M. Iqbal Parabi, S.SI., M.T. selaku Dosen Pembimbing Utama yang telah membimbing penulis dengan semua dukungan, motivasi, nasihat, serta saran yang diberikan dalam proses penyusunan dan penyelesaian skripsi ini. Terima kasih atas dedikasi dan ketulusan yang tak ternilai.
7. Bapak Igit Sabda Ilman, M.Kom. selaku Dosen Pembimbing Kedua yang telah memberikan bimbingan dan arahan yang berguna untuk memperbaiki diri dan karya penulis di setiap kesempatannya.
8. Bapak Rico Andrian, S.Si., M.Kom. selaku Dosen Penguji yang telah memberikan saran dan masukan untuk kemajuan penyelesaian skripsi ini.
9. Bapak/Ibu Dosen dan seluruh Staf Jurusan Ilmu Komputer FMIPA Universitas Lampung yang telah memberikan segenap ilmu pengetahuan yang bermanfaat bagi penulis serta membantu dalam berbagai urusan akademik maupun administratif.
10. Teman seperjuangan yaitu Shafa, Chandra dan Amalia. Terima kasih telah berjalan bersama di jalur penuh lika-liku ini. Penulis memahami bahwa keberhasilan bukan hanya tentang ilmu yang ditulis tetapi juga tentang kebersamaan yang terjalin. Meski lembar skripsi telah usai, semoga langkah kita ke depan selalu indah sesuai dengan apa yang kita impikan.
11. Teman-teman dan sobat karib yaitu Ayuni, Vidya, Nabillah, Jihan, Cindy, Roy, Ikhsan dan Kartika. Terima kasih telah menjadi pelengkap perjalanan ini. Di antara ribuan baris kode yang telah ditulis, ratusan hari kebersamaan kita, serta puluhan lelahnya tugas, kalian hadir membawa semangat dan doa. Dari begadang tugas, skripsi, sampai ketawa receh tiap hari semua itu akan selalu penulis rindukan. Semoga jejak langkah kita, meski akan menyebar ke arah dan tempat yang berbeda, selalu dipenuhi keberkahan dan tetap terikat oleh kenangan yang tak lekang oleh waktu.

12. Sahabat kecil yaitu Salmaa, Alyaa, Ayu, Icha, Talitha, Intan dan Syifa. Terima kasih telah menjadi saksi pertumbuhan dari tawa polos hingga perjuangan hari ini. Dalam setiap permainan sederhana dan percakapan ringan, penulis menemukan arti kebahagiaan yang tulus. Meski jalan hidup membawa kita pada cerita masing-masing, persahabatan masa kecil kita akan selalu menuntun kembali pada kehangatan dan kenangan indah yang abadi.
13. Keluarga Ilmu Komputer Angkatan 2021 yang telah menjadi bagian dari perjalanan ini, terima kasih atas kebersamaan dan dukungan yang senantiasa diberikan selama masa perkuliahan di Jurusan Ilmu Komputer Universitas Lampung.
14. Kepada seluruh pihak yang tidak dapat disebutkan satu per satu namun telah memberikan dukungan dan kontribusi yang berarti dalam proses penyusunan skripsi ini, penulis mengucapkan terima kasih.

Penulis menyadari bahwa skripsi ini masih memiliki banyak kekurangan dan belum sepenuhnya sempurna. Meski demikian, besar harapan penulis agar karya ini dapat memberikan manfaat dan keberkahan bagi perkembangan ilmu pengetahuan terutama bagi mahasiswa Ilmu Komputer dan seluruh civitas akademika Universitas Lampung.

Bandar Lampung, 28 Agustus 2025

Fathimah Abiyyi Khairunnisa
NPM. 2117051088

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR KODE

Kode Halaman

I. PENDAHULUAN

1.1 Latar Belakang

Dosen merupakan salah satu sumber daya yang memegang peranan penting dalam keberhasilan dan pencapaian kinerja di perguruan tinggi. Kualitas dosen juga secara langsung berkorelasi dengan pencapaian target institusi, baik dalam skala akademis maupun non-akademis. Oleh karena itu, perguruan tinggi perlu senantiasa memberikan dukungan dan motivasi agar dosen dapat tetap produktif dan mampu mengembangkan potensi yang dimilikinya secara maksimal, sehingga dapat memberikan dampak positif dan kontribusi yang optimal dalam perkembangan institusi.

Universitas Lampung menyadari pentingnya memberikan dorongan yang efektif kepada para dosennya agar tetap termotivasi untuk terus berprestasi. Salah satu pendekatan yang dilakukan adalah melalui penerapan sistem remunerasi, di mana dosen dapat menerima kompensasi yang sesuai dengan kinerjanya (Mardikaningsih & Darmawan, 2022). Dengan sistem remunerasi, universitas dapat meningkatkan produktivitas dosen sekaligus menciptakan lingkungan kerja yang kompetitif dan kondusif bagi pengembangan karier dosen (Leuhery & Nahumury, 2023).

Sistem remunerasi yang saat ini diterapkan di Universitas Lampung memiliki peran penting dalam pengumpulan data kinerja dosen yang digunakan sebagai dasar penilaian kinerja. Selain itu, sistem ini juga berfungsi sebagai instrumen untuk mengukur insentif kinerja (*pay for performance*) berlandaskan capaian kontrak kinerja dalam menentukan besaran tunjangan atas capaian yang diperoleh (Universitas Lampung, 2024).

Meskipun telah menjadi indikator utama dalam penilaian kinerja dosen, sistem remunerasi dosen Universitas Lampung masih memiliki potensi yang belum sepenuhnya dimanfaatkan secara maksimal, terutama dalam hal pemanfaatan data kinerja dosen terkhusus bidang pengabdian kepada masyarakat dan penunjang. Salah satu peluang untuk mengoptimalkan pendayagunaan data kinerja tersebut adalah pada saat proses akreditasi, di mana data kinerja dosen dapat dimanfaatkan untuk mendukung kelancaran dan kualitas proses tersebut.

Akreditasi, sama seperti remunerasi, memakai instrumen berlandaskan *tri dharma* perguruan tinggi yang mencakup tiga pilar utama yang wajib dipenuhi oleh institusi pendidikan tinggi, yaitu pendidikan dan pelaksanaan pendidikan, penelitian, pengabdian kepada masyarakat, serta ditambah penunjang. Pada praktiknya, pengumpulan data kinerja dosen untuk kebutuhan akreditasi masih menggunakan cara yang manual. Kurangnya keterbukaan akses pada proses pengumpulan data ini tidak hanya memperpanjang waktu penyusunan laporan, tetapi juga berisiko menyebabkan ketidakakuratan atau ketidaksesuaian data yang dibutuhkan. Kondisi ini dapat menurunkan efisiensi kerja tim penyusun dan menghambat upaya program studi dalam memperoleh penilaian yang optimal.

Berdasarkan kondisi tersebut, muncul kebutuhan untuk melakukan integrasi data remunerasi dengan sistem pelacakan data kinerja dosen melalui teknologi *application programming interface* (API) yang dapat membantu memaksimalkan pendayagunaan data tersebut. Dengan API yang terintegrasi, program studi akan lebih mudah mendapatkan akses data kinerja dosen, terutama di bidang pengabdian dan penunjang, sehingga dapat mempercepat proses pengumpulan data untuk kebutuhan akreditasi ataupun pengambilan keputusan strategis lainnya. Pengembangan API pada data remunerasi dosen Universitas Lampung ini tidak hanya akan mempermudah proses pengumpulan data kinerja dosen, tetapi juga dapat memberikan kemudahan dalam menghubungkan berbagai sistem informasi lainnya di lingkungan universitas.

Pengembangan API sendiri telah banyak diterapkan dalam berbagai sistem informasi di lingkungan perguruan tinggi, seperti sistem informasi akademik (Husein dkk., 2024) (Prayogi dkk., 2020) (Adam dkk., 2020), sistem manajemen dosen (Septama dkk., 2022) (Kaniya dkk., 2022), *e-learning* (Budiman dkk., 2021) (Mowla & Kolekar, 2020), dan lain-lain.

Penggunaan teknologi *server* Node.js akan digunakan dalam pembangunan API pada penelitian ini. Berdasarkan penelitian Prayogi dan tim mengenai desain dan implementasi REST API untuk sistem informasi akademik, disimpulkan bahwa Node.js memberikan kinerja yang sangat baik terutama dalam pengujian waktu *response* (*response time*) dan ukuran jumlah data yang berhasil diproses atau dikirim dalam periode waktu tertentu (*throughput*) di kedua titik akhir (*endpoint*) yang diujicobakan (Prayogi dkk., 2020). Hal ini menunjukkan bahwa Node.js memiliki kemampuan dalam menangani permintaan REST API dengan efisien, terutama ketika dihadapkan pada operasi yang berat dan kompleks.

Penelitian ini juga mengimplementasikan autentikasi dan otorisasi pada sistem API menggunakan JSON *Web Token* (JWT). Berdasarkan penelitian yang dilakukan oleh Adam dan Tim mengenai implementasi *web service* pada sistem informasi Unklab dengan menggunakan JWT, disimpulkan bahwa penggunaan autentikasi berbasis JWT bertujuan untuk memastikan proses otorisasi dan hak akses dapat berlangsung secara aman dan terstruktur (Adam dkk., 2020). Sehingga, hanya pengguna yang memiliki otoritas yang dapat mengakses data tertentu, dan memastikan perlindungan data serta mencegah akses yang tidak sah.

Dengan demikian, penerapan teknologi RESTful API dengan menggunakan autentikasi JWT ini diharapkan dapat memberikan solusi yang efektif dalam optimalisasi pendayagunaan data kinerja pada sistem remunerasi dosen Universitas Lampung. Penelitian ini juga diharapkan dapat menjadi inovasi teknologi serta membantu dalam pengumpulan informasi dengan akses yang lebih luas bagi pihak terkait guna memberikan kemudahan dalam berbagai proses penilaian kinerja, seperti akreditasi, evaluasi, maupun pelaporan institusional lainnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah pada penelitian ini adalah:

1. Bagaimana cara mengintegrasikan data kinerja dosen pada Sistem Remunerasi Universitas Lampung bidang pengabdian dan penunjang?
2. Bagaimana memastikan keamanan dalam proses pertukaran data pada Sistem Remunerasi Dosen Universitas Lampung bidang pengabdian dan penunjang?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini meliputi:

1. Fokus penelitian terletak pada pengembangan RESTful API untuk Sistem Remunerasi Dosen Universitas Lampung.
2. API yang dibangun ini hanya mencakup data kinerja beban lebih dosen bidang pengabdian dan penunjang.
3. Sistem yang dikembangkan ini dirancang khusus untuk memberikan akses melihat data kinerja beban lebih dosen tanpa fungsi manipulasi data.
4. Akses terhadap data kinerja dosen dibatasi sesuai dengan hak dan peran pengguna dalam sistem ini.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diidentifikasi, tujuan dari penelitian ini adalah:

1. Mengembangkan API pada Sistem Remunerasi Dosen Universitas Lampung untuk integrasi data kinerja dosen terkhusus bidang pengabdian dan penunjang.
2. Membangun autentikasi dan otorisasi hak akses untuk menjaga keamanan pertukaran data Sistem Remunerasi Universitas Lampung dengan sistem informasi lainnya.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan beberapa manfaat, seperti:

1. Optimalisasi pengintegrasian data kinerja dosen Universitas Lampung dengan berbagai sistem informasi lainnya.
2. Efisiensi pendayagunaan data kinerja dosen untuk mempermudah akses dalam proses evaluasi, akreditasi, ataupun pelaporan institusional lainnya.
3. Peningkatan pemahaman dan keterampilan dalam pengembangan modul yang mendukung integrasi sistem.

II. TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Dalam menyusun penelitian ini, beberapa penelitian terdahulu dijadikan sebagai acuan dalam mengadopsi konsep, teori, dan pembangunan kerangka berpikir untuk memperkuat rangka penyusunan penelitian yang dilakukan. Adapun yang menjadi referensi untuk penelitian ini dijabarkan dalam Tabel 1.

Tabel 1. Penelitian terdahulu

Judul	Teknologi yang digunakan	Hasil
Pengamanan RESTful API Web Service Menggunakan JSON Web Token (Studi Kasus: Aplikasi Siakadu Mobile Unesa) (Utama & Indriyanti, 2023)	*Framework* PHP Laravel, JWT, MySQL	Penelitian tersebut berfokus pada pengembangan lebih lanjut dari aplikasi siakadu mobile Unesa, yaitu implementasi keamanan RESTful API menggunakan JWT. Pengembangan tersebut bertujuan untuk menyediakan standar komunikasi antar aplikasi yang lebih kompatibel dari yang sebelumnya serta menyederhanakan proses pertukaran data antar sistem yang terintegrasi nantinya. Pengembangan tersebut juga menerapkan mekanisme pengamanan tambahan dengan JWT, yang mana dengan mekanisme tersebut, data yang terenkripsi dapat terjamin keamanannya, sehingga tidak dapat diubah maupun diakses oleh pihak yang tidak berwenang selama proses pertukaran data berlangsung.

Tabel 1. (lanjutan)

Judul	Teknologi yang digunakan	Hasil
RESTful Web Service Implementation on Unklab Information System using JSON Web Token (JWT) (Adam dkk., 2020)	Metode Agile, JavaScript, JWT, MySQL	Penelitian tersebut mengembangkan RESTful *web service* untuk sistem informasi universitas dengan memanfaatkan JWT. Penggunaan mekanisme otorisasi JWT bertujuan untuk menyediakan akses data bagi pengguna yang memiliki izin, serta meningkatkan aksesibilitas dan mengurangi risiko redundansi data.
Design and Implementation of REST API for Academic Information System (Prayogi dkk., 2020)	*Server-side Technologies* Node.js dan PHP	Penelitian tersebut membandingkan implementasi REST API pada sistem informasi akademik menggunakan dua teknologi *server*, Node.js dan PHP. Evaluasi dilakukan berdasarkan *response time* dan *troughput*, dengan hasil bahwa Node.js memiliki performa yang lebih baik pada kedua metrik evaluasi. Penelitian tersebut juga menekankan perlunya uji lebih lanjut pada konfigurasi data dan *endpoint* yang lebih kompleks.
A Web Based System for Easy Secured Managing Process of University Accreditation Information (Kommey dkk., 2022)	React.js, Node.js, *Framework* Express.js, MongoDB	Penelitian tersebut mengembangkan sistem akreditasi berbasis web yang dirancang untuk menghadirkan solusi yang efisien dalam pengelolaan proses akreditasi akademik di perguruan tinggi. Sistem tersebut mengadopsi model *client-server* dari REST dengan tujuan peningkatan keamanan dan fleksibilitas proses pertukaran data.
Implementation of One-Data based Lecturer Profile Information System for Key Performance Indicators (Septama dkk., 2022)	Metode *Extreme Programming, Framework* PHP Laravel	Penelitian tersebut mengembangkan sistem profil dosen menggunakan arsitektur terintegrasi berbasis *one-data* universitas guna mendukung regulasi *one-data* dan meningkatkan efisiensi pengelolaan informasi. Hasil uji *User Experience Questionnaire* menunjukkan kesan yang positif, menandakan bahwa sistem efektif mendukung pengelolaan data dosen.

2.2 BKD (Beban Kerja Dosen)

BKD merupakan kegiatan yang dibebankan kepada dosen dalam menjalankan tugas dan kewajibannya sebagai pendidik profesional dan ilmuwan pada kurun waktu tertentu. BKD diatur dalam Undang-Undang Nomor 14 Pasal 72 tentang Guru dan Dosen Tahun 2005, yang mencakup kegiatan pokok yakni merencanakan pembelajaran, melakukan evaluasi pembelajaran, membimbing dan melatih, melakukan penelitian, tugas tambahan, serta pengabdian kepada masyarakat. Berdasar pada Pedoman Operasional BKD Tahun 2021, BKD memiliki batas kepatutan dan kelayakan, yakni minimal 12 sks dan maksimal 16 sks. Namun dalam kondisi nyata di perguruan tinggi, dosen melaksanakan tugas dan kewajiban melebihi 16 sks setiap semester. Maka perguruan tinggi dapat mempertimbangkan pemberian penghargaan lewat pembayaran insentif dan/atau pemberian remunerasi sesuai kemampuan lembaga bagi dosen yang melaksanakan beban lebih.

2.3 Remunerasi

Menurut Kamus Besar Bahasa Indonesia (KBBI), remunerasi diartikan sebagai pemberian hadiah (penghargaan atas jasa dan sebagainya). Remunerasi dapat mencakup pembayaran upah, bonus, tunjangan, dan hak-hak lain dari pegawai yang bekerja untuk institusi (Dekawati, 2022). Pemberian remunerasi biasanya dilakukan apabila pegawai memberikan kinerja melampaui target yang sudah ditetapkan untuk memotivasi agar pegawai dapat memberikan kinerja yang lebih baik lagi (Wiyati & Pradana, 2022).

Di Universitas Lampung, remunerasi diwujudkan dalam bentuk insentif kinerja (*pay for performance*) yang dinilai berdasarkan capaian poin kinerja beban lebih pada setiap periode tertentu, bersumber dari poin yang merujuk pada BKD dan rubrik remunerasi (Universitas Lampung, 2024). Penilaian kinerja di Unila sendiri berlandaskan *tri dharma* perguruan tinggi. Di antara pilar *tri dharma*, penelitian ini akan menitikberatkan pada bidang pengabdian dan bidang penunjang.

2.4 RESTful API

REST (*representational state transfer*) merupakan sebuah arsitektur API yang menyediakan komunikasi antara *client* dengan *server* dalam aplikasi web melalui protokol HTTP (*hypertext transfer protocol*). RESTful sendiri adalah istilah yang digunakan untuk menyebutkan bahwa API tersebut dirancang sepenuhnya mengikuti prinsip-prinsip REST. Menurut R. Fielding dalam disertasi terkenalnya, ada tujuh prinsip yang dapat dipenuhi ketika ingin membangun sistem REST (Fielding, 2000), yaitu:

1. NULL *style*

   REST dibangun dari keadaan tanpa batasan (*null style*) yang kemudian dibentuk dengan menambahkan batasan REST secara bertahap.

2. *Client-Server*

   REST memisahkan peran *client* dan *server*, yang dikenal dengan prinsip *client-server*. Prinsip ini memberikan keuntungan berupa penyederhanaan komponen *server*, memungkinkan perbaikan antarmuka *client* secara terpisah, dan meningkatkan skalabilitas sistem secara keseluruhan.

3. *Stateless*

   Dalam REST, hubungan *client-server* bersifat *stateless*. Artinya, setiap permintaan harus membawa semua informasi yang diperlukan tanpa bergantung pada konteks yang disimpan di *server* atau riwayat sebelumnya. Prinsip ini membantu meningkatkan visibilitas dan keandalan sistem.

4. *Cache*

   REST mengoptimalkan efisiensi jaringan melalui mekanisme *caching*. *Server* dapat menandai *response* sebagai *cacheable* atau non-*cacheable* yang memungkinkan *client* menyimpan data sementara. Hal ini dapat meningkatkan performa di sisi *client* dan mengurangi beban pada sisi *server*, sehingga sistem dapat menjadi lebih efisien serta skalabel.

5. *Uniform Interface*

   REST menggunakan antarmuka yang seragam (*uniform interface*) untuk memastikan informasi disajikan dalam format standar. Hal ini bertujuan memisahkan data dari penyedia layanan, sehingga *client* dapat berinteraksi dengan *server* tanpa bergantung pada implementasi spesifik di sisi *server*.

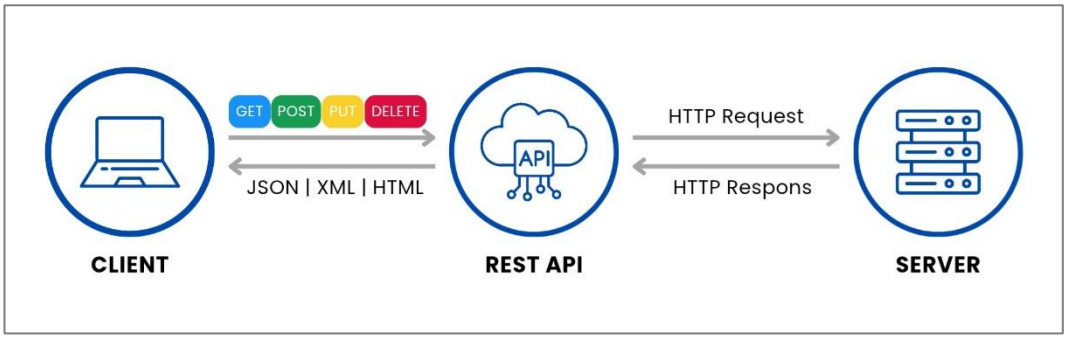
6. *Layered System*

   REST menerapkan sistem berlapis (*layered system*) di mana komponen-komponen di setiap lapisan tidak dapat mengakses atau mengetahui komponen lain di luar lapisannya. Prinsip ini memberikan keamanan tambahan, menyederhanakan desain, dan memungkinkan integrasi komponen warisan (*legacy components*).

7. *Code-on-Demand*

   REST dapat diperluas melalui fungsi *code-on-demand*, di mana *client* dapat mengunduh dan menjalankan kode selama *runtime*. Hal ini berguna untuk mengurangi jumlah fitur yang perlu diimplementasikan sebelumnya, sekaligus memungkinkan fleksibilitas lebih dalam pengembangan aplikasi.

Secara garis besar, hal yang perlu dipahami tentang REST API adalah terkait dengan HTTP beserta komponen-komponennya. Komponen-komponen tersebut dapat dikelompokkan menjadi dua, yaitu API *request* dan API *response*. Adapun gambaran dari proses API *request* dan API *response* pada skema model REST API dapat dilihat pada Gambar 1.



Gambar 1. Model REST API.

2.4.1 API *Request*

API *request* adalah permintaan yang dikirimkan oleh *client* ke *server*. Komponen utama dari API *request* meliputi:

a. API *Endpoint*

Endpoint adalah sebuah lokasi spesifik atau titik akses menuju API. *Endpoint* ini biasanya direpresentasikan sebagai *uniform resource indicators* (URI) yang terdiri dari *base*URL dan *path*.

b. HTTP *Methods*

HTTP memiliki sejumlah metode yang digunakan untuk menentukan jenis tindakan pada *resource*, meliputi:

- GET: Digunakan untuk mengambil data dari *server*.
- POST: Digunakan untuk mengirim data baru ke *server*.
- PUT: Digunakan untuk memperbarui data yang ada di *server*.
- DELETE: Digunakan untuk menghapus data dari *server*.

c. *Request Header*

Request header berisi pasangan *key-value* yang memberikan informasi tambahan dari *request*. Contohnya:

- *Content-Type*: Menunjukkan format data yang dikirim, seperti "*application/json*".
- *Authorization*: Berisi token untuk mengirimkan kredensial otorisasi.

d. *Request Body*

Request body digunakan ketika ingin mengirimkan data ke *server* dalam metode POST atau PUT. Kode 1 adalah contoh dari *request body*.

```
{
  "name": "Fathimah Abiyyi",
  "age": 21
}
```

Kode 1. Contoh penerapan *request body*.

2.4.2 API *Response*

API *response* adalah tanggapan yang dikirimkan *server* ke *client*. Komponen utama dari API *response* meliputi:

a. *Status Code*

Status code adalah sebuah kode numerik yang memberikan hasil dari *request*. *Status code* seperti pesan pendek berisi tiga digit yang menyampaikan apakah *request* berhasil mendapatkan *response*, mengalami masalah, atau menghadapi kesalahan.

- 1xx: *Informational Code*
- 2xx: *Success Code*
- 3xx: *Redirection Code*
- 4xx: *Client Error Code*
- 5xx: *Server Error Code*

b. *Response Header*

Response header sama seperti *request header* yang memberikan informasi tambahan tentang *response*. Contohnya:

- *Content-Type*: Menunjukkan format data yang dikirim, seperti *application/json*.
- *Set-Cookie*: Menyimpan *cookie* di sisi *client*.

c. *Response Body*

Response body mengandung data yang diminta oleh *client* dengan hasil dapat berupa HTML, JSON, ataupun XML. Bagian ini bersifat opsional karena digunakan ketika dibutuhkan saja. Kode 2 adalah contoh format *response body* dalam JSON.

```
{
  "message": "User created successfully"
}
```

Kode 2. Contoh *response body* dalam JSON.

RESTful API memungkinkan komunikasi yang efisien antara berbagai sistem atau aplikasi, menjadikannya salah satu fondasi utama dalam pengembangan layanan web modern (Ehsan dkk., 2022).

Dalam mendeskripsikan RESTful API, diperlukan format spesifikasi untuk memudahkan pengembang dalam mendokumentasikan, memverifikasi, dan memahami fungsi dari API. OpenAPI merupakan salah satu standar yang dapat memudahkan pengguna untuk memahami fungsionalitas sistem dan cara berinteraksi dengan logika implementasi minimum. OpenAPI menyediakan informasi tentang layanan *endpoint*, format pesan, dan ketentuan untuk memanggil layanan tersebut (Tzavaras dkk., 2023). Dalam laporan ini, OpenAPI akan disajikan dalam bentuk tabel agar lebih mudah dipahami, khususnya dalam konteks dokumentasi non-teknis. Adapun penjelasan elemen-elemen dalam spesifikasi API dijabarkan pada Tabel 2.

Tabel 2. Elemen spesifikasi API

Nama	Keterangan
Method	Merujuk pada metode HTTP yang digunakan untuk *endpoint* tersebut.
Path	Alamat URL yang akan diakses oleh *client*.
Description	Penjelasan singkat mengenai fungsi dari *endpoint*, yaitu apa yang akan dilakukan ketika *endpoint* tersebut diakses.
Parameter	Variabel yang menerima *input* ke dalam sebuah fungsi, dikirim melalui URL atau bagian *query*.
Request Header	Data tambahan yang dikirim lewat HTTP *header* menuju *server*.
Request Body	Data utama yang dikirimkan dalam *request*, biasanya dalam format JSON.
Response Success	Jawaban yang diberikan *server* jika berhasil dijalankan.
Response Failed	Jawaban yang diberikan *server* jika *request* terjadi kesalahan.
Asteris symbol (*)	Menunjukkan variabel yang bersifat *required* atau wajib.

2.5 JWT (JSON *Web Token*)

JSON *web token* atau JWT adalah sebuah cara untuk bertukar data atau informasi antara dua pihak, biasanya antara *server* dan *client*. Pada praktiknya, JWT digunakan dalam proses autentikasi dan otorisasi, di mana token akan dibuat berdasarkan struktur tertentu yang kemudian dapat memverifikasi identitas pengguna dan mengamankan akses ke API (Mahindrakar & Pujeri, 2020). Dalam *The* JWT *Handbook version* 0.14.2 (Peyrott, 2024), ada tiga elemen utama dalam struktur JWT, yaitu *header*, *payload*, dan *signature*/*encryption*. Kode 3 adalah contoh dari sebuah token JWT.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Kode 3. Contoh token JWT.

Pada tiga elemen JWT di atas, masing-masing elemennya akan dipisahkan oleh tanda titik (.) dan dikodekan secara independen. Dua elemen pertama adalah objek JSON dari struktur *header* dan *payload*, sedangkan elemen ketiga bergantung pada kombinasi algoritma yang digunakan untuk mengenkripsikan *signature/encryption*. Berikut penjelasan dari masing-masing elemen JWT:

a. *Header*

Elemen utama JWT adalah *header* berisi algoritma yang digunakan untuk menghasilkan *signature* yang terdiri dari informasi seperti jenis algoritma, tipe token, dan lain-lain. Kode 4 adalah contoh *decoded header* JWT.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Kode 4. Contoh *decoded header* JWT.

b. *Payload*

Payload berisikan klaim atau data yang akan dikirimkan. Dalam penerapannya, data yang disertakan biasanya bersifat unik, seperti *email*, ID/UUID, serta informasi terkait otoritas seperti peran yang berfungsi sebagai identitas pengenal pengirim token. Selain itu, disertakan juga hak akses dan waktu kedaluwarsa token untuk menambah keamanan ekstra dengan membatasi waktu di mana token yang disertakan dapat digunakan. Kode 5 adalah contoh dari bentuk *decoded payload* JWT.

```
{
  "id": "1234567890",
  "name": "John Doe",
  "admin": true,
  "iat": 1747745595,
  "exp": 1748350395
}
```

Kode 5. Contoh *decoded payload* JWT.

c. *Signature*

Signature digunakan untuk menjamin integritas token dengan memastikan bahwa isi token (*header* dan *payload*) tidak mengalami perubahan sejak token dibuat. *Signature* dihasilkan melalui proses *hashing* yang menggabungkan *header*, *payload*, dan *secret key*. *Secret key* ini bersifat rahasia dan sulit untuk diretas. Dengan *signature*, *server* dapat memverifikasi keaslian token, sehingga mencegah manipulasi terhadap *header* dan *payload*.

Pembuatan token yang unik dan terenkripsi ini dapat meningkatkan keamanan sistem dengan menghindari kebutuhan untuk mengirimkan informasi sensitif pada setiap permintaan. Hal ini tidak hanya melindungi data pengguna, tetapi juga meningkatkan pengalaman pengguna secara keseluruhan (Bucko dkk., 2023). Dengan formatnya yang sederhana dan ringkas, JWT dapat menjadi solusi yang fleksibel untuk berbagai kebutuhan implementasi autentikasi dan otorisasi dalam berbagai sistem informasi (Singh & Gupta, 2020).

2.6 Node.js

Node.js adalah salah satu teknologi sisi *server* (*server-side technology)* yang berperan sebagai *runtime environment* menggunakan mesin JavaScript V8 dari Chrome sebagai basisnya. Dikembangkan oleh Ryan Dahl pada tahun 2009, Node.js dirancang untuk membangun aplikasi *server* yang ringan dan mudah diskalakan. Teknologi ini memungkinkan sistem dengan performa tinggi, skalabilitas, pengembangan yang cepat, dan ekosistem yang luas (Doglio, 2015). Selain itu, Node.js menggunakan arsitektur *asynchronous* dan *non-blocking* yang sangat cocok untuk aplikasi yang membutuhkan penanganan banyak permintaan secara bersamaan. Penggunaan bahasa pemrograman JavaScript di sisi *server* dan *client* juga menyederhanakan proses pengembangan dan integrasi (Shukla, 2023).

2.7 *Database*

Database atau basis data adalah kumpulan data yang terorganisir dan memungkinkan untuk diakses, dikelola, dan diperbarui secara efisien. *Database* menyimpan data dalam bentuk terstruktur yang dapat diakses oleh berbagai aplikasi dan pengguna. *Database* juga mendukung penyimpanan data yang konsisten dan dapat diandalkan dengan memanfaatkan sistem manajemen basis data atau *database management system* (DBMS) untuk memastikan integritas, keamanan, dan pemulihan data (Ramakrishnan & Gehrke, 2002).

PostgreSQL adalah salah satu jenis sistem manajemen *database* relasional (RDBMS) yang bersifat *open-source*. PostgreSQL dirancang untuk mendukung standar SQL dan menawarkan fitur lanjutan seperti integrasi data dan dukungan untuk berbagai tipe data. Sistem ini memungkinkan pengelolaan basis data yang kompleks dengan kemampuan ekstensibilitas, seperti menambahkan tipe data baru, fungsi, dan metode indeks khusus. PostgreSQL juga dikenal karena keandalannya dan kompatibilitasnya untuk digunakan dalam berbagai platform dan aplikasi (Matthew & Stones, 2005).

2.8 Pengujian

Pengujian adalah suatu rangkaian proses yang bertujuan untuk memastikan kode komputer beroperasi sesuai dengan tujuannya dan tidak melakukan hal-hal yang tidak diinginkan (Kaur dkk., 2022). Pengujian juga diartikan sebagai kegiatan yang melibatkan penerapan kriteria pengujian umum yang sudah terstandarisasi pada model atau struktur sistem (Da Costa, 2021).

2.8.1 *White box Testing*

White box testing adalah teknik pengujian perangkat lunak yang berfokus pada pengujian struktur internal, desain, dan kode program (Tuya dkk., 2006). Pengujian dengan *white box testing* bertujuan untuk memverifikasi alur *input-output* sekaligus meningkatkan aspek desain, kegunaan, dan keamanan perangkat lunak. Metode ini mengacu pada pengujian yang didasarkan pada pemahaman struktur internal aplikasi dengan pendekatan yang melibatkan eksekusi atau meliputi elemen-elemen spesifik dalam kode (Kaur dkk., 2022).

2.8.2 Postman

Postman adalah platform kolaborasi API yang dirancang untuk memenuhi berbagai kebutuhan pengembangan API dan mendukung semua tahapan dalam *lifecycle* API sekaligus meningkatkan efisiensi kerja tim (Postman, 2024c). Melalui Postman, pengembang dapat menjalankan kueri HTTP/s, baik yang sederhana maupun kompleks, serta memeriksa *response* yang diterima untuk mempercepat alur kerja dan mengurangi beban kerja yang berlebihan (Kore dkk., 2022). Postman memungkinkan pengembang API dapat merancang, membangun, menguji, dan mendokumentasikan API secara kolaboratif, sementara pengguna API dapat menjelajahi, mencoba, dan mengintegrasikan API ke dalam sistem mereka sendiri (Krosnick, 2024).

Postman akan dimanfaatkan dalam proses pengujian fungsional dengan tujuan untuk memverifikasi setiap fungsi sistem dapat bekerja sesuai kebutuhan yang telah ditentukan. Pengujian ini untuk memastikan apakah masukan tertentu akan memberikan keluaran yang sesuai dengan harapan (Sommerville, 2011). Tabel 3 adalah elemen-elemen pengujian fungsional.

Tabel 3. Elemen dalam pengujian fungsional

Nama	Keterangan
Endpoint	Alamat URL dari API atau fitur yang diuji.
Scenario ID	ID unik untuk masing-masing skenario uji setiap *endpoint*. Format: TS.XX (contoh: TS.01).
Test Scenario	Deskripsi ringkas skenario yang diuji, dapat berupa alur atau tujuan utama.
Case ID	ID unik untuk masing-masing kasus uji, turunan dari *scenario* ID. Format: TC.XX.YY (contoh: TC.01.01).
Test Case	Deskripsi langkah uji yang dilakukan.
Type	Jenis uji fungsional untuk klasifikasi tipe pengujian.
Input Data	Data masukan yang digunakan untuk kasus uji, berupa *form input*, JSON *body*, parameter, dsb.
Expected Results	Hasil yang diharapkan dari sistem berdasarkan *input* dan skenario.
Actual Results	Hasil nyata dari sistem saat tes dijalankan.
Status	Menunjukkan apakah hasil sudah sesuai harapan.

2.8.3 Apache JMeter

JMeter adalah sebuah aplikasi untuk menguji kinerja dan perilaku fungsional dari aplikasi *client* atau *server*. Aplikasi ini berfungsi untuk mengukur semua sumber daya *server* yang digunakan dan dihasilkan, seperti penggunaan memori, sampai waktu dan ukuran *response* (Halili, 2008).

JMeter akan dimanfaatkan dalam pengujian performa, di mana aplikasi ini dapat menyimulasikan perilaku pengguna yang mengakses sistem secara bersamaan guna mengevaluasi kinerja sistem saat berada di bawah tekanan. Pengujian ini dilakukan dengan tujuan untuk memastikan bahwa API yang dikembangkan dapat menangani beban kerja yang tinggi serta mendeteksi potensi terjadinya hambatan (*bottleneck*) dalam prosesnya (Hendayun dkk., 2023). Adapun elemen-elemen yang digunakan dalam pengujian performa dijabarkan pada Tabel 4.

Tabel 4. Elemen dalam pengujian performa

Nama	Keterangan
Number of Threads (users)	Jumlah *virtual user* atau *thread* yang akan digunakan untuk menyimulasikan pengguna yang mengakses sistem secara bersamaan.
Ramp-up Period (s)	Waktu yang dibutuhkan untuk menaikkan seluruh *thread*.
Thread Spawn Rate (s)	Laju kemunculan setiap *thread* secara bertahap dan linier selama periode *ramp-up*.
Loop Count	Jumlah iterasi yang akan dilakukan setiap *thread*.
#Samples	Total permintaan (*sample*) yang dikirim, representasi dari aktivitas *virtual user* yang melakukan permintaan ke *server*.
Min (ms)	Waktu tercepat yang dibutuhkan untuk menyelesaikan satu permintaan.
Max (ms)	Waktu terlama yang dibutuhkan untuk menyelesaikan satu permintaan.
Average (ms)	Waktu rata-rata yang dibutuhkan untuk menyelesaikan semua permintaan.
Error %	Persentase permintaan yang gagal.
Throughput	Jumlah permintaan yang berhasil diproses oleh *server* per unit waktu.

2.8.4 OWASP ZAP (*Zed Attack Proxy*)

ZAP adalah sebuah alat pengujian yang dapat mendeteksi celah keamanan pada aplikasi berbasis web, termasuk API. Dibuat dan dikelola oleh sebuah organisasi internasional OWASP (*Open Web Application Security Project*), ZAP dirancang sebagai *proxy* intersepsi yang dapat memantau dan memindai lalu lintas web antara peramban dan aplikasi web (Maniraj dkk., 2024).

Dalam penelitian ini, ZAP akan dimanfaatkan dalam proses pengujian keamanan pada API yang dikembangkan, yaitu dengan bertindak sebagai *man-in-the-middle* (MITM) atau perantara yang memungkinkan untuk dapat melihat dan memanipulasi *request* dan *response* HTTP guna mengidentifikasi potensi kerentanan keamanan dari sistem yang diujikan (Soper dkk., 2023). Adapun penjelasan elemen-elemen yang digunakan dalam pengujian keamanan ada pada Tabel 5 berikut ini.

Tabel 5. Elemen-elemen dalam pengujian keamanan

Nama	Keterangan
Plugins	Jenis atau modul pengujian tertentu yang digunakan untuk mendeteksi kerentanan spesifik (misalnya: SQL *Injection*, XSS)
Elapsed	Lama waktu (dalam format hh:mm:ss) yang dibutuhkan *plugin* tersebut untuk menyelesaikan proses pemindaian.
Reqs	Jumlah total *request* HTTP yang dikirim oleh *plugin* selama pemindaian.
Alerts	Jumlah peringatan atau temuan (*alert*) yang berhasil dideteksi oleh *plugin* tersebut, baik berisiko tinggi maupun hanya informasional.
Status	Menunjukkan status dari proses pemindaian *plugin*, misalnya: *Completed*, *Skipped*, atau *Failed*.
Risk Level	Tingkat risiko dari *alert* yang ditemukan, diklasifikasikan sebagai *Informational*, *Low*, *Medium*, *High*, atau *False Positive*.
Number of Instances	Jumlah kejadian (*instance*) di mana kerentanan atau perilaku spesifik ditemukan oleh *plugin* dalam satu atau lebih *endpoint*.

2.9 API-*first Development*

API-*first development* adalah sebuah pendekatan dalam pengembangan perangkat lunak di mana perancangan dan pengembangan API menjadi prioritas utama (Postman, 2024b). Pendekatan ini berfokus pada merancang API lebih awal yang dapat digunakan oleh pengembang lain (baik internal maupun eksternal) untuk membangun solusi yang terintegrasi dengan baik, efisien, dan fleksibel (Chandrachood, 2021).

Dalam mengadopsi metode API-*first development*, beberapa hal yang perlu diperhatikan sejak awal yaitu memprioritaskan pengembangan API, mengenali peran berbagai jenis API (seperti API publik, pribadi, dan mitra), serta yang paling utama adalah memahami siklus hidup API atau API *lifecycle* dan berbagai *tools* yang diperlukan untuk mengimplementasikan metode API-*first* dengan efektif (Larsson dkk., 2021).

API *lifecycle* adalah serangkaian langkah yang perlu dilalui untuk memastikan keberhasilan dalam proses perancangan, pengembangan, penyebaran, dan pemanfaatan API. Dengan mengikuti siklus yang dirancang secara sistematis, tim dapat bekerja dengan lebih efisien dan mampu menghasilkan API yang andal dan berkualitas (Postman, 2024a). Adapun tahapan-tahapan dalam API *lifecycle* adalah sebagai berikut.

1. *Define* (Pendefinisian API)
   Pada tahap awal *define*, pengembang harus menentukan kebutuhan API dengan melakukan identifikasi kebutuhan dari pengguna atau pemangku kepentingan. Kebutuhan tersebut mencakup fungsi utama yang diharapkan dari API, termasuk skema apa saja yang akan dihasilkan oleh sistem yang akan dikembangkan. Proses ini dapat membantu mengarahkan pengembangan API agar sesuai dengan kebutuhan pengguna dan spesifikasi teknis yang diinginkan.

2. *Design* (Perancangan API)

   Tahap perancangan API dilakukan dengan melibatkan pengambilan keputusan secara terencana mengenai cara API menyajikan data kepada pengguna. Tahap ini bertujuan untuk mendeskripsikan struktur *database* serta API, seperti *endpoint*, skema data, metode permintaan, dan *response* yang akan dihasilkan. Hal ini memungkinkan semua pihak dapat memverifikasi apakah rancangan API sudah sesuai dengan kebutuhan sebelum masuk ke tahap pengembangan.

3. *Develop* (Pengembangan API)

   Tahap ini berfokus pada implementasi API sesuai dengan rancangan yang telah disepakati. Setelah API dirancang dan disetujui, pengembang akan menuliskan kode untuk mengimplementasikan fungsionalitas sesuai spesifikasi yang telah direncanakan. Selain implementasi kode, pengembang juga dapat mulai menyusun dokumentasi API untuk memudahkan konsumen API dalam menggunakannya.

4. *Test* (Pengujian API)

   Tahap ini dilakukan untuk memastikan bahwa API berfungsi sesuai dengan spesifikasi yang telah ditetapkan. Pengujian API dapat dilakukan baik secara manual maupun otomatis untuk mengidentifikasi potensi masalah sejak dini. API akan melewati beberapa pengujian, yaitu pengujian fungsional untuk memastikan apakah API berfungsi dengan benar, pengujian performa untuk memastikan API dapat menangani beban kerja yang kompleks, dan pengujian keamanan untuk deteksi dini celah keamanan dari API yang dibangun.

5. *Deploy* (Penerapan API)

   Tahap akhir yaitu *deploy* atau penerapan API. Tahap ini yang memungkinkan pengembang untuk mengirimkan atau menerbitkan API ke lingkungan *development*, *staging*, dan *production*. *Deployment* API biasanya dapat dilakukan melalui API *gateway* untuk menstandarisasi proses *deploy*. Proses ini bertujuan untuk memastikan bahwa API yang telah diuji dapat diimplementasikan dengan benar sebelum mencapai konsumen.

Pendekatan dengan menggunakan metode API-*first development* ini dapat memudahkan pengembangan dengan membaginya menjadi *microservice* yang mandiri dan saling terhubung melalui API unik untuk berkomunikasi dengan layanan lainnya. Pendekatan ini bermanfaat dalam membangun aplikasi yang memerlukan fleksibilitas dan kemudahan pengembangan fungsi baru tanpa mengganggu layanan lainnya (Dudjak & Martinović, 2020).

2.10 UML (*Unified Modeling Language*)

Unified modeling language atau UML adalah alat pemodelan visual yang fleksibel dan serbaguna untuk mendefinisikan, menggambarkan, merancang, mengembangkan, serta mendokumentasikan elemen-elemen dalam sistem. UML berfungsi untuk mengkomunikasikan ide dan pemahaman terkait sistem yang akan dibangun. UML digunakan dalam berbagai aktivitas seperti analisis, desain, penelusuran, konfigurasi, pemeliharaan, dan pengelolaan informasi sistem (Rumbaugh dkk., 1999).

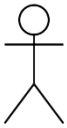
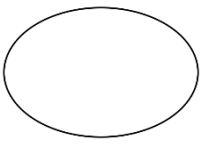
Terdapat berbagai jenis diagram dalam UML. Lima di antaranya yaitu *use case diagram*, *use case description*, *activity diagram*, *sequence diagram*, dan *class diagram*. Berikut adalah penjelasan mengenai diagram-diagram tersebut.

2.10.1 *Use Case Diagram*

Use case diagram adalah serangkaian tindakan yang dibutuhkan untuk merepresentasikan persyaratan sistem, memahami kebutuhan pengguna, dan mendokumentasikan fungsionalitas yang harus dimiliki oleh sistem. Diagram ini memadukan teks deskriptif dan beberapa simbol sederhana, sehingga dapat dengan mudah dipahami oleh berbagai pemangku kepentingan (Armour & Miller, 2000).

Tabel 6 adalah penjelasan dari simbol-simbol dalam *use case diagram*.

Tabel 6. Komponen *use case diagram*

No.	Gambar	Nama	Keterangan
1.		*Actor*	Orang, proses, atau sistem lain yang langsung berinteraksi pada sistem di luar pada sistem itu sendiri.
2.		*Association*	Komunikasi antara *actor* dan *use case* yang berpartisipasi.
3.		*Include*	Relasi antara *use case* di mana proses yang bersangkutan akan langsung dilanjutkan ke proses yang dituju.
4.		*Exclude*	Relasi antara *use case* yang ditambahkan dapat berdiri sendiri, walaupun tanpa *use case* tambahan.
5.		*Generalization*	Komunikasi antara satu fungsi umum dengan fungsi lain.
6.		*Use Case*	Fungsionalitas yang disediakan sistem sebagai sebuah unit yang saling bertukar antara unit satu dengan lainnya maupun *actor*.

2.10.2 *Use Case Description*

Use case description adalah bentuk dokumentasi yang menjelaskan secara rinci tentang suatu *use case* dalam sistem, mencakup interaksi antara aktor dan sistem serta urutan langkah-langkah yang diperlukan untuk mencapai tujuan spesifik. Tujuannya adalah untuk memberikan pemahaman mendalam dan terstruktur mengenai bagaimana sistem seharusnya berperilaku pada berbagai situasi dalam skenario tertentu (Bertolino dkk., 2002).

Tabel 7 merupakan penjelasan dari elemen pada *use case description*.

Tabel 7. Elemen *use case description*

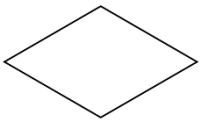
No.	Nama	Keterangan
1.	*Use Case Name*	Skenario atau aktivitas yang sedang dijelaskan dalam *use case*, biasanya berupa kalimat singkat yang mencerminkan tujuan utamanya.
2.	*ID*	Kode unik untuk mengidentifikasi setiap *use case* secara individual, memudahkan dokumentasi.
3.	*Priority*	Tingkat kepentingan atau urgensi *use case*.
4.	*Actor*	Pengguna yang berinteraksi langsung dengan sistem dalam skenario *use case* tersebut.
5.	*Description*	Ringkasan dari apa yang dilakukan dalam *use case* dan apa yang ingin dicapai oleh aktor melalui interaksi dengan sistem.
6.	*Trigger*	Aksi yang memicu dimulainya *use case*, biasanya oleh aktor atau hasil dari sistem eksternal.
7.	*Precondition*	Kondisi atau status sistem yang harus dipenuhi sebelum *use case* dapat dijalankan.
8.	*Normal Course*	Urutan langkah-langkah utama (alur normal) yang dilakukan oleh aktor dan sistem untuk menyelesaikan tujuan *use case*.
9.	*Postcondition*	Keadaan sistem setelah *use case* berhasil dijalankan atau hasil akhir yang diharapkan.
10.	*Sub Flows*	Langkah-langkah tambahan yang mungkin terjadi di dalam alur normal, sering kali merupakan proses opsional atau pengulangan.
11.	*Alternate/Exceptional Flows*	Alur alternatif atau pengecualian yang terjadi jika ada kesalahan, kondisi khusus, atau penyimpangan dari alur normal.

2.10.3 *Activity Diagram*

Activity diagram atau diagram aktivitas adalah diagram alur yang menunjukkan bagaimana aliran kontrol terjadi antar aktivitas yang dilakukan secara berurutan atau bersamaan dalam suatu proses komputasi. Diagram ini biasanya terdiri dari berbagai jenis *state*, *node*, serta panah yang menghubungkannya (Börger dkk., 2000). Diagram ini berguna untuk memodelkan logika proses bisnis, alur kerja sistem, atau skenario penggunaan dalam perangkat lunak secara visual dan mudah dipahami.

Tabel 8 adalah penjelasan dari simbol-simbol dalam *activity diagram*.

Tabel 8. Komponen *activity diagram*

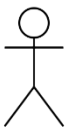
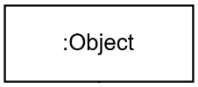
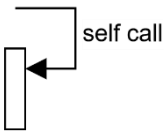
No.	Gambar	Nama	Keterangan
1.		*Initial State*	Status awal aktivitas pada sistem.
2.		*Transition*	Menunjukkan suatu objek akan melaksanakan tindakan tertentu ketika suatu aktivitas terjadi.
3.		*Activity*	Aktivitas yang dilakukan oleh suatu sistem, biasanya diawali dengan kata kerja.
4.		*Decision*	Asosiasi percabangan di mana jika terdapat lebih dari satu pilihan.
5.		*Fork/Join*	Pembagian aktivitas ke beberapa arah yang berbeda (*fork*) atau penggabungan beberapa aktivitas ke dalam satu arah yang sama (*join*).
6.		*Final State*	Status akhir aktivitas pada sistem.

2.10.4 *Sequence Diagram*

Sequence diagram atau yang disebut juga diagram urutan adalah sebuah diagram yang digunakan untuk menggambarkan interaksi antar objek dalam sebuah sistem secara berurutan. *Sequence diagram* berguna untuk menjelaskan serangkaian langkah yang dilakukan sebagai *response* dari sebuah peristiwa untuk menghasilkan suatu *output* tertentu (Bell, 2004).

Tabel 9 adalah penjelasan dari elemen-elemen *sequence diagram*.

Tabel 9. Komponen *sequence diagram*

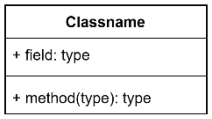
No.	Gambar	Nama	Keterangan
1.		*Actor*	Sebuah entitas eksternal yang berinteraksi dengan sistem.
2.		*Object*	Sebuah entitas di dalam sistem yang melakukan tugas atau aktivitas tertentu.
3.		*Lifeline*	Sebuah titik waktu di mana elemen berinteraksi dengan *actor* atau *object*.
4.		*Action Bar*	Proses durasi aktivitas sebuah *actor* atau *object*.
5.		*Message*	Sebuah aksi mengirimkan *request*.
6.		*Reply Message*	Sebuah aksi mengembalikan *response*.
7.		*Self Message*	Kondisi ketika *actor* atau *object* sedang melakukan tugas atau aktivitas pada diri sendiri.

2.10.5 *Class Diagram*

Class diagram atau diagam kelas adalah representasi visual dari tampilan statis suatu sistem yang menggambarkan kumpulan elemen model deklaratif seperti kelas, tipe data, serta isi dan hubungan antar elemen tersebut (Rumbaugh dkk., 1999). Unsur utama dalam *class diagram* adalah kelas dan hubungan antar kelas (Mcgill dkk., 2001).

Tabel 10 adalah penjelasan elemen-elemen dalam *class diagram*.

Tabel 10. Komponen *class diagram*

No.	Gambar	Nama	Keterangan
1.		*Class*	Kelas pada struktur sistem.
2.		*Association*	Relasi antar kelas yang dengan makna umum, dapat berupa relasi *one-to-one*, *one-to-many*, dan *many-to-many*.
3.		*Directed Association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
4.		*Generalization*	Relasi antar kelas dengan makna umum ke khusus.
5.		*Agregation*	Relasi antar kelas dengan makna semua-bagian.
6.		*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas.

III. METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian ini dilakukan di Laboratorium Rekayasa Perangkat Lunak yang bertempat di Jurusan Ilmu Komputer, Gedung Matematika dan Ilmu Pengetahuan Alam Terpadu, Universitas Lampung. Penelitian ini berlangsung pada tahun ajaran 2024/2025. Rincian waktu penelitian dapat dilihat pada Tabel 11.

Tabel 11. Rincian waktu penelitian

Aktivitas	2024	2025				
	Desember	Januari	Februari	Maret	April	Mei
Plan						
Define						
Design						
Develop						
Test						
Deploy						

3.2 Perangkat Penelitian

Beberapa perangkat yang digunakan untuk mendukung pelaksanaan penelitian ini terdiri dari dua, perangkat keras dan perangkat lunak. Penjelasan dari perangkat-perangkat penelitian tersebut ada pada halaman berikut ini.

3.2.1 Perangkat Keras

Perangkat keras yang digunakan dalam penelitian ini adalah sebuah laptop dengan spesifikasi sebagai berikut:

a. *System Manufacturer* : ACER
b. *Processor* : Intel Core i3-1005G1
c. RAM / *Storage* : 4,00 GB / 256 GB
d. *System Type* : 64-bit *operating system*

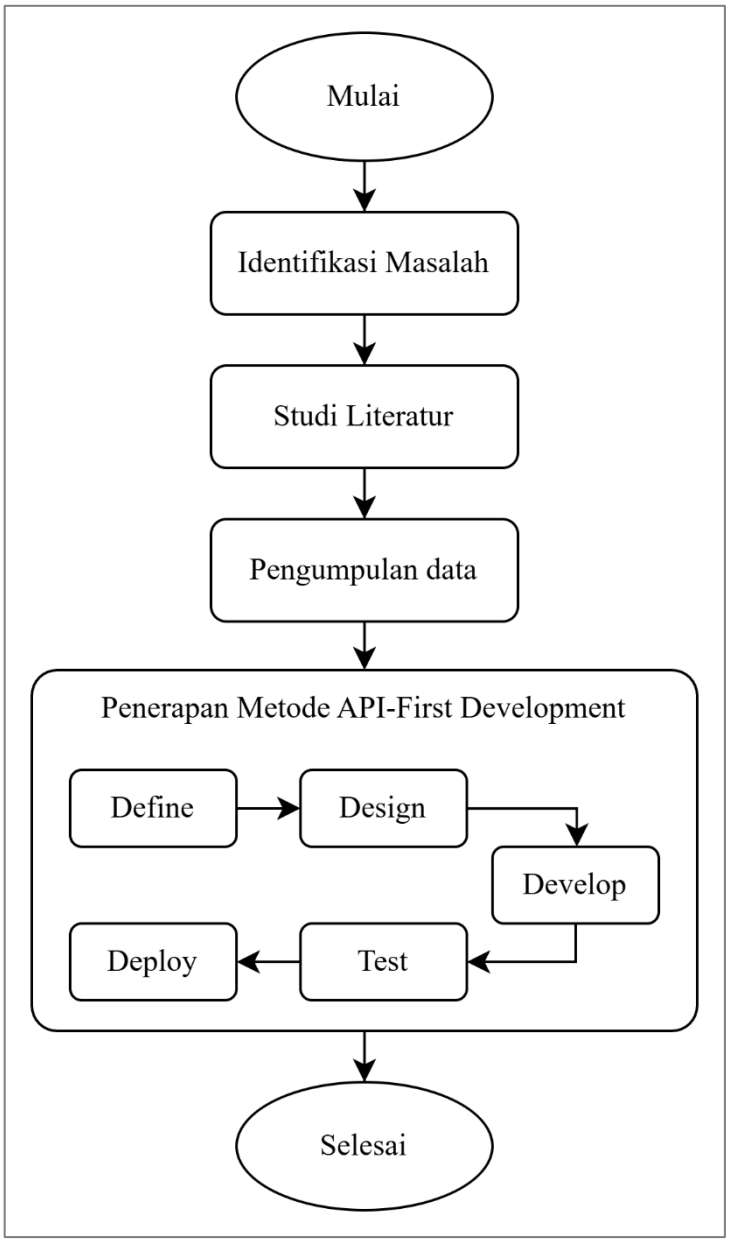
3.2.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam penelitian ini adalah sebagai berikut:

a. Windows 10 Home *Single Language*, sistem operasi untuk menjalankan perangkat lunak dan perangkat keras.
b. Visual Studio Code, editor kode ringan dengan dukungan berbagai bahasa pemrograman dan ekstensi.
c. PostgreSQL v17, sistem manajemen basis data (DBMS) *open-source* untuk penyimpanan dan pengelolaan data.
d. PgAdmin v4, antarmuka grafis untuk mengelola *database* PostgreSQL
e. Navicat *Premium Lite* v17, aplikasi manajemen basis data grafis.
f. Postman *Agent* v11, alat pengujian API untuk mengelola, menguji, dan menganalisis *request* dan *response*.
g. Apache JMeter v5, alat uji untuk mengukur kinerja aplikasi web dan API.
h. OWASP ZAP v2, alat uji keamanan aplikasi berbasis web, termasuk API.
i. Github, platform pengelolaan kode dan kolaborasi Git.
j. Google Chrome, peramban web dengan ekstensi dan fitur keamanan.
k. Publish or Perish, *software* analisis dari berbagai sumber.
l. Mendeley *Reference Manager*, alat pengelola dan penyusun referensi akademik dalam penulisan ilmiah.
m. Microsoft Word 365, *software* pengolah kata untuk membuat dokumen.
n. Draw.io, alat untuk membuat diagram dan skema visual lainnya.

3.3 Tahapan Penelitian

Tahapan-tahapan dalam penelitian ini digambarkan pada Gambar 2. Gambar tersebut merupakan diagram alir penelitian, di mana proses penelitian akan dibagi ke dalam 4 tahap, antara lain identifikasi masalah, studi literatur, pengumpulan data, dan penerapan metode API-*First Development*.

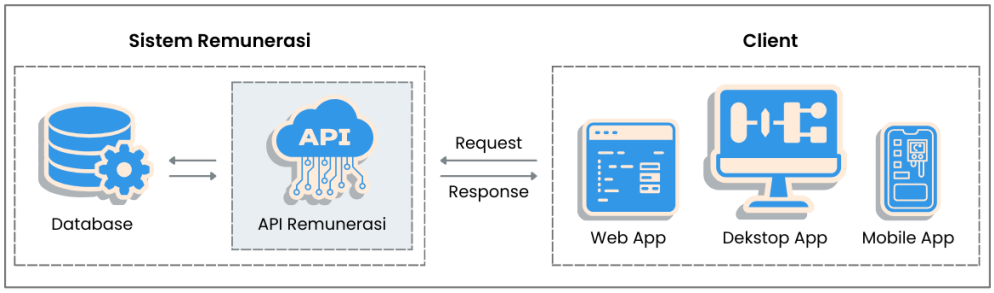


Gambar 2. Diagram alir tahapan penelitian.

3.3.1 Identifikasi Masalah

Penelitian ini dilakukan atas permasalahan yang terjadi pada sistem remunerasi dosen Universitas Lampung, khususnya terkait pemanfaatan data kinerja dosen bidang pengabdian dan penunjang yang sepenuhnya belum didayagunakan lebih luas. Dalam sistem remunerasi yang ada saat ini, kaprodi dan dekan tidak memiliki akses untuk memantau langsung data kinerja dosen fakultas serta program studinya. Padahal, data tersebut dapat berguna untuk berbagai penunjang proses penilaian, salah satunya proses akreditasi.

Dengan adanya akses yang lebih luas terhadap data kinerja dosen, yang dalam hal ini adalah pengembangan API remunerasi, kaprodi dan dekan akan dapat lebih mudah menghimpun data yang diperlukan dalam penyusunan dokumen pendukung. API ini dibangun dengan tujuan agar data kinerja dosen yang tersimpan dalam *database* remunerasi dapat diintegrasikan dan dimanfaatkan dalam berbagai proses yang membutuhkan informasi tersebut. Adapun proses integrasi API renumerasi ini digambarkan pada Gambar 3.



Gambar 3. Arsitektur integrasi API remunerasi.

Proses bisnis dari sistem remunerasi ini sendiri dimulai dari aktivitas dosen melaksanakan berbagai kegiatan *tri dharma*, mencakup bidang pendidikan, penelitian, pengabdian, serta penunjang. Setiap aktivitas tersebut kemudian dimasukkan ke dalam SISTER, sebuah sistem yang mengelola rekam jejak dan aktivitas akademik dosen. Setelah semua kegiatan tercatat, dosen dapat mengunduh portofolio BKD dari SISTER sebagai laporan kinerja mereka.

Selanjutnya, pada sistem remunerasi, dosen dapat melakukan sinkronisasi data dan mengunggah laporan BKD agar dapat melakukan klaim kinerja sebagai beban lebih. Proses ini bertujuan agar sistem dapat mengolah data kinerja tersebut menjadi poin capaian kinerja yang digunakan sebagai dasar perhitungan remunerasi. Sistem remunerasi kemudian akan menyimpan data kinerja dosen dalam *database* remunerasi.

Melalui pengembangan API remunerasi, sistem yang terhubung nantinya dapat mengambil dan menggunakan informasi dari data kinerja dosen tersebut. Dengan adanya integrasi ini, data kinerja dosen dapat digunakan oleh pihak yang berkepentingan dalam membantu pengelolaan sumber daya akademik, seperti akreditasi, pemantauan kinerja akademik, evaluasi kinerja dosen, serta pelaporan institusional lainnya.

3.3.2 Studi Literatur

Untuk mendukung pemahaman konsep, teori, dan pendekatan yang relevan atas pengidentifikasian masalah, dilakukan tahapan studi literatur sebagai salah satu langkah penting dalam penelitian ini. Melalui studi literatur, informasi dapat digali dari berbagai sumber terpercaya guna mendukung kerangka konseptual dan memperkuat argumentasi dalam pengembangan solusi yang ditawarkan.

Dalam penelitian ini, tahapan studi literatur dilakukan untuk memahami lebih dalam konsep dari arsitektur RESTful API, mengeksplorasi cara kerja Node.js, mempelajari pemanfaatan JWT sebagai sebuah metode otorisasi dan autentikasi, serta mendalami pengintegrasian sistem berbasis API dengan *database*. Hasil dari tahapan ini akan menjadi dasar dalam merancang dan mengimplementasikan sistem yang sesuai dengan kebutuhan penelitian.

3.3.3 Pengumpulan Data

Dalam penelitian ini, proses pengumpulan data dilakukan untuk menghasilkan dua jenis data, yaitu data primer dan data sekunder guna mendukung analisis kebutuhan sistem.

a. Data Primer

   Data primer yang digunakan dalam penelitian ini diperoleh langsung dari hasil wawancara dengan pihak yang akan terlibat, dalam hal ini ialah ketua program studi. Wawancara ini bertujuan untuk mendapatkan informasi mendalam terkait kebutuhan sistem dan kendala yang dihadapi dalam pengelolaan data kinerja dosen.

b. Data Sekunder

   Data sekunder yang digunakan dalam penelitian ini didapatkan dari informasi dalam sistem remunerasi dosen Universitas Lampung, khususnya data terkait kinerja dosen bidang pengabdian dan penunjang. Data ini digunakan dengan tujuan untuk menganalisis pola penggunaan sistem remunerasi saat ini serta mengidentifikasi peluang integrasi data yang dapat mendukung proses akreditasi.

3.3.4 Penerapan Metode API-*first Development*

Penerapan metode API-*first development* dalam penelitian ini dilakukan melalui serangkaian proses yang sistematis untuk memastikan pengembangan sistem berjalan efektif dan sesuai dengan kebutuhan. Metode ini berfokus pada perancangan dan pengembangan API sebagai elemen inti sebelum pengembangan komponen lain. Adapun tahapan-tahap yang diterapkan dalam metode ini ada pada halaman berikut.

1. *Define* (Pendefinisian API)

Proses pertama yaitu *define* yang dilakukan dengan tujuan untuk mengarahkan pengembangan API agar sesuai dengan spesifikasi teknis yang diinginkan. Dalam penelitian ini, dilakukan tahap identifikasi kebutuhan pengguna dan sistem untuk menghasilkan gambaran tentang fungsionalitas yang diharapkan. Selanjutnya, fungsionalitas tersebut divisualisasikan menggunakan *use case diagram* untuk menggambarkan interaksi antara pengguna dengan sistem, serta *activity diagram* dan *sequence diagram* untuk memvisualisasikan alur proses. Tahapan-tahapan tersebut dilakukan guna membantu mengurangi kesalahpahaman dalam perancangan API dan dapat meningkatkan pemahaman non-teknis serta efisiensi pengembangan API.

a. Identifikasi Kebutuhan Pengguna dan Sistem

Berdasarkan proses yang telah dilewati sebelumnya, diperoleh kebutuhan fungsional dan non-fungsional yang akan digunakan sebagai acuan dalam membangun API remunerasi. Penjelasan kebutuhan fungsional sistem dijabarkan sebagai berikut.

a) Sistem mampu melakukan proses *log in* dan *log out*;
   Seluruh pengguna (administrator, dekan, kaprodi, dosen) dapat mengakses sistem dengan melakukan *log in* menggunakan akun yang sudah didaftarkan sebelumnya. Pengguna juga memiliki opsi untuk keluar sistem dengan melakukan *log out* apabila ingin mengakhiri sesinya.

b) Sistem mampu memfasilitasi pengelolaan data pengguna;
   Pengguna (administrator) dapat melakukan pengelolaan data pengguna, seperti membuat, mengubah, serta menghapus data pengguna.

c) Sistem mampu menampilkan data kinerja dosen bidang pengabdian dan penunjang;

   Pengguna (admin, dekan, kaprodi, dosen) dapat melihat data kinerja sesuai dengan hak akses yang diberikan berdasarkan peran yang dimilikinya. Pengguna yang memiliki peran sebagai admin, dapat melihat seluruh data kinerja. Pengguna yang memiliki peran sebagai dekan, dapat melihat data kinerja seluruh dosen di bawah naungan fakultasnya. Pengguna yang memiliki peran sebagai kaprodi, dapat melihat data kinerja seluruh dosen di bawah naungan program studinya. Pengguna yang memiliki peran sebagai dosen, dapat melihat data kinerjanya pribadi.

Adapun kebutuhan non-fungsional dijabarkan sebagai berikut:

a) Performa: Sistem diharapkan memiliki performa yang lancar dan cepat dengan harapan *response time* kurang dari 1 detik.
b) Skalabilitas: Sistem diharapkan mampu menangani permintaan dalam jumlah yang besar tanpa mengurangi performa sistem.
c) Keamanan: Sistem diharapkan dapat menjaga seluruh data yang ada dalam sistem dengan aman dan terlindungi dari akses yang tidak sah atau penyalahgunaan.

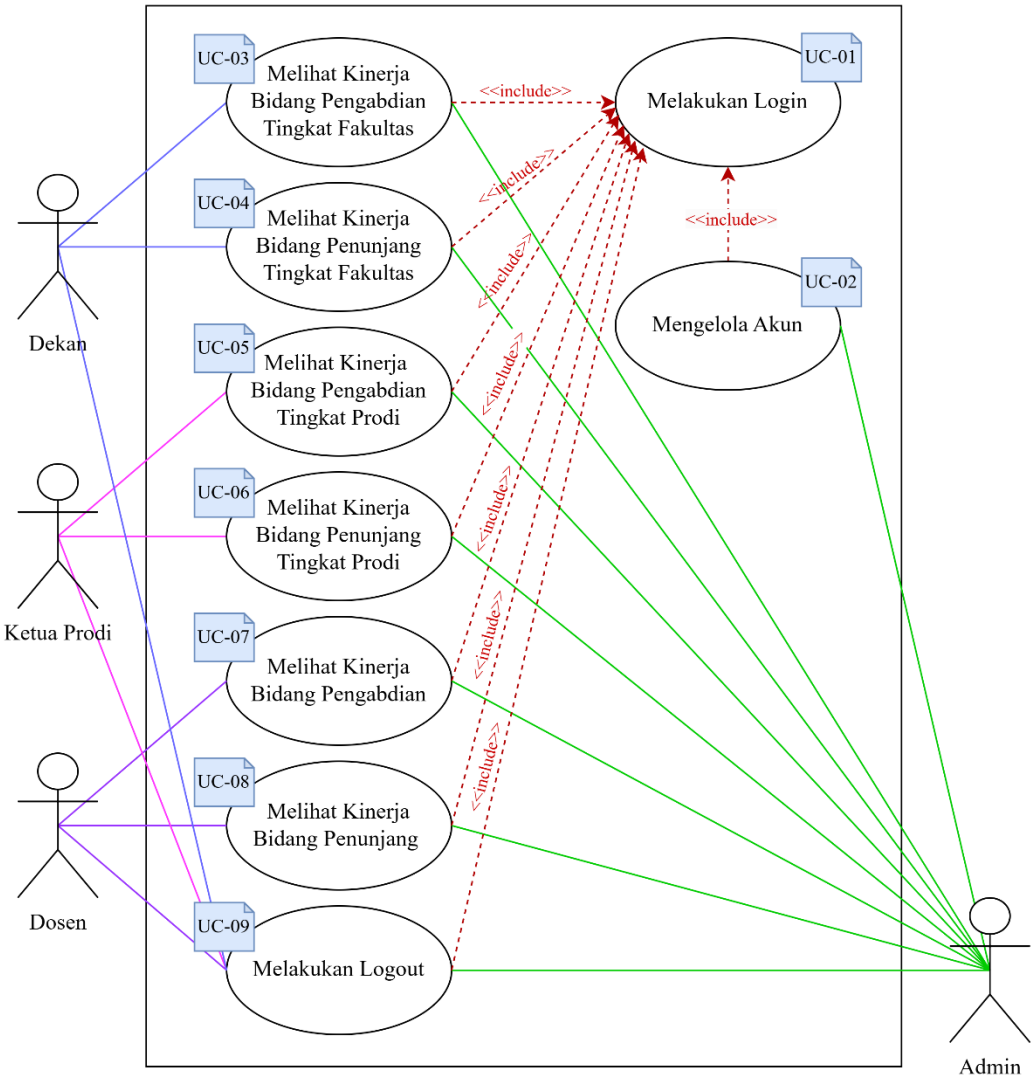
Pengembangan sistem ini akan melibatkan sejumlah pihak terkait. Pihak-pihak tersebut dapat dilihat pada Tabel 12.

Tabel 12. Pihak yang terlibat dalam penelitian

Nama	Peran
UPT TIK Universitas Lampung	*Business Stakeholder*
M. Iqbal Parabi, S.Si., M.T.	*Product Owner*
Amalia Nurul Rahmawati	*Front-End Developer*
Chandra Prasetya Putra	*Front-End Developer*
Fathimah Abiyyi Khairunnisa	*Back-End Developer*
Shafa Auliya	*Back-End Developer*

b. *Use Case Diagram*

Use case diagram pada Gambar 4 disusun berdasarkan hasil identifikasi kebutuhan pengguna dan sistem. Diagram ini berfungsi sebagai cetak biru (*blueprint*) untuk menggambarkan fungsi dan interaksi antara sistem dengan aktornya.



Gambar 4. *Use case diagram*.

Setelah *use case diagram* berhasil didefinisikan, langkah berikutnya ialah merinci setiap *use case* dengan lebih detail dan menjabarkannya dengan menggunakan tabel yang disebut *use case description*. Bidang pengabdian dan penunjang akan dijelaskan dalam tabel yang sama.

Tabel 13. *Use case description* melakukan *log in*

Use Case Name	Melakukan *Log in*
ID	UC-01
Priority	*High*
Actor	Admin, Dekan, Kaprodi, Dosen
Description	*Actor* dapat melakukan *log in* ke sistem menggunakan NIDN dan *password* yang sudah didaftarkan pada *database*
Trigger	*Actor* melakukan *log in*
Precondition	*Actor* sudah terdaftar di sistem
Normal Course	1. *Actor* mengakses halaman *login*. 2. *Actor* memasukkan NIDN dan *password*. 3. Sistem meminta data pengguna. 4. *Database* mengirim data akun. 5. Sistem mencocokkan kredensial *actor*. 6. Jika valid, sistem mengirim token *authorization*. 7. *Virtual* d*atabase* menyimpan token. 8. Sistem memvalidasi token. 9. Jika valid, sistem akan menampilkan halaman *user.*
Postconditions	*Actor* berhasil masuk dan dapat mengakses fitur-fitur sistem sesuai dengan perannya
Sub Flows	a. *Forgot password*
Alternate/ Exceptional Flows	A1: Jika NIDN atau *password* salah, maka sistem akan menampilkan pesan *error* dan meminta *actor* untuk mencoba lagi. A2: Jika *actor* telah melewati batas waktu sesi tertentu, secara otomatis token akan berubah menjadi tidak aktif dan sesi akan berakhir. A3: Jika *actor* lupa *password*, sistem menampilkan pesan untuk meminta *actor* menghubungi admin.

Tabel 13 adalah *use case description* untuk proses *login* dalam sistem. Tabel ini mencakup informasi seperti aktor yang terlibat, deskripsi proses *log in*, pemicu, prasyarat, alur normal, kondisi setelah *log in*, serta alur alternatifnya. Dalam skenario utama, aktor memasukkan NIDN dan *password*, kemudian sistem memverifikasi kredensial dan memberikan *authorization* jika valid. Tabel ini juga menjelaskan beberapa alur alternatif, seperti penanganan kesalahan kredensial, lupa *password*, serta batas waktu sesi *login*.

Tabel 14. *Use case description* mengelola akun pengguna

Use Case Name	Mengelola Akun Pengguna
ID	UC-02
Priority	*High*
Actor	Admin
Description	*Actor* dapat melakukan kelola data akun pengguna seperti menambah akun baru, mengubah data akun, atau menghapus akun pengguna
Trigger	*Actor* memilih salah satu aksi kelola akun
Precondition	*Actor* sudah *login* sebagai admin dan memiliki status sesi yang aktif
Normal Course	1. *Actor* sudah berada dalam sistem dan memiliki status sesi yang aktif. 2. Sistem menampilkan daftar data akun pengguna. 3. *Actor* dapat memilih salah satu aksi kelola akun. 3.1. *Actor* memilih aksi menambah akun. 3.1.1. Sistem menampilkan *form* tambah akun. 3.1.2. *Actor* mengisi *form* data-data yang dibutuhkan untuk tambah akun baru. 3.1.3. *Actor submit* form. 3.1.4. Sistem mengirimkan *request* tambah data berisi data akun baru yang dimasukkan *actor*. 3.1.5. *Database* menyimpan data baru dan mengembalikan *response* daftar akun. 3.2. *Actor* memilih aksi mengubah data akun. 3.2.1. Sistem menampilkan *form* ubah data akun. 3.2.2. *Actor* mengubah data pada formulir yang berisi data sebelumnya menjadi data baru yang ingin diubah. 3.2.3. *Actor submit form*. 3.2.4. Sistem mengirimkan *request* ubah data berisi data baru yang dimasukkan. 3.2.5. *Database* menyimpan data baru dan mengembalikan *response* daftar akun. 3.3. *Actor* memilih aksi menghapus akun. 3.3.1. *Actor* memvalidasi aksi hapus akun. 3.3.2. Sistem mengirimkan *request* hapus akun terpilih. 3.3.3. *Database* menghapus akun dan mengembalikan *response* daftar akun. 4. Sistem menampilkan seluruh daftar akun pengguna.

Tabel 14. (lanjutan)

Postconditions	*Actor* berhasil menambah, mengubah, atau menghapus akun pengguna
Sub Flows	a. Tambah akun b. Ubah data akun c. Hapus akun
Alternate/ Exceptional Flows	A1: Jika *actor* telah melewati batas waktu sesi tertentu, secara otomatis token akan berubah menjadi tidak aktif dan sesi akan berakhir. A2: Jika data pengguna yang dimasukkan pada kolom formulir tidak sesuai, maka sistem akan menampilkan pesan untuk meminta *actor* memperbaiki data. A3: Jika data pengguna yang ingin dihapus tidak divalidasi/dibatalkan oleh *actor*, maka sistem akan mengembalikannya tanpa perubahan ke halaman kelola akun.

Tabel 14 merupakan *use case description* yang menjelaskan proses pengelolaan akun pengguna oleh admin. Proses ini memiliki prasyarat yang harus dipenuhi, yaitu admin harus terlebih dahulu melakukan *log in* atau memiliki status sesi aktif sebelum melakukan tindakan.

Alur utama dalam tabel ini menjelaskan langkah-langkah admin mengelola akun pengguna. Admin dapat memilih aksi kelola, seperti menambah akun baru, mengubah data akun yang sudah ada, ataupun menghapus akun. Setiap aksi ini melibatkan interaksi dengan sistem dan *database*, di mana data yang dimasukkan akan divalidasi sebelum disimpan atau dihapus. Sistem kemudian memberikan *response* terhadap setiap perubahan yang dilakukan oleh admin, seperti menampilkan daftar akun yang telah diperbarui.

Dijabarkan juga akan kemungkinan adanya kegagalan dalam proses ini. Jika data akun yang dimasukkan tidak sesuai format, sistem akan menampilkan pesan kesalahan dan meminta admin untuk memperbaiki data tersebut. Jika admin membatalkan proses hapus akun, sistem akan mengembalikannya tanpa melakukan perubahan.

Tabel 15. *Use case description* melihat kinerja bidang pengabdian dan penunjang tingkat fakultas

Use Case Name	Melihat Kinerja Bidang Pengabdian dan Penunjang Tingkat Fakultas
ID	UC-03 & UC-04
Priority	*High*
Actor	Admin, Dekan
Description	*Actor* dapat mengakses fitur melihat kinerja bidang pengabdian dan penunjang tingkat fakultas berdasarkan program studi dan dosen yang dipilih
Trigger	*Actor* memilih program studi, dosen, dan bidang kinerja yang ingin dilihat
Precondition	*Actor* sudah *login* sebagai admin atau dekan dan memiliki status sesi yang aktif
Normal Course	1. *Actor* sudah berada dalam sistem dan memiliki status sesi yang aktif. 2. Sistem menampilkan data fakultas beserta daftar program studi. 3. *Actor* memilih program studi. 4. Sistem meminta data program studi. 5. *Database* mengembalikan data program studi. 6. Sistem menampilkan data program studi dan daftar dosen. 7. *Actor* memilih dosen yang ingin dilihat. 8. Sistem meminta data dosen. 9. *Database* mengembalikan data dosen. 10. Sistem menampilkan data dosen dan kinerja. 11. *Actor* memilih data kinerja bidang pengabdian/ penunjang. 12. Sistem meminta data pengabdian/penunjang. 13. *Database* mengembalikan data kinerja bidang pengabdian/penunjang. 14. Sistem menampilkan data pengabdian/penunjang.
Postconditions	*Actor* berhasil memilih program studi dan dosen serta mendapatkan data kinerja bidang pengabdian dan penunjang berdasarkan pilihan tersebut
Sub Flows	a. *Filter* data kinerja per tahun dan semester tertentu
Alternate/ Exceptional Flows	A1: Jika waktu sesi habis, maka secara otomatis token akan berubah menjadi tidak aktif dan sesi akan berakhir. A2: Jika data dosen tidak ditemukan, sistem akan menampilkan pesan dosen tidak ditemukan. A3: Jika hasil *filter* data kinerja tidak ada, maka sistem akan menampilkan pesan data kosong.

Tabel 16. *Use case description* melihat kinerja bidang pengabdian dan penunjang tingkat program studi

Use Case Name	Melihat Kinerja Bidang Pengabdian dan Penunjang Tingkat Program Studi
ID	UC-05 & UC-06
Priority	*High*
Actor	Admin, Kaprodi
Description	*Actor* dapat mengakses fitur melihat kinerja bidang pengabdian dan penunjang tingkat program studi berdasarkan dosen yang dipilih
Trigger	*Actor* memilih dosen dan bidang kinerja yang ingin dilihat
Precondition	*Actor* sudah *login* sebagai admin atau kaprodi dan memiliki status sesi yang aktif
Normal Course	1. *Actor* sudah berada dalam sistem dan memiliki status sesi yang aktif. 2. Sistem menampilkan data program studi beserta daftar dosen. 3. *Actor* memilih dosen yang ingin dilihat. 4. Sistem meminta data dosen. 5. *Database* mengembalikan data dosen. 6. Sistem menampilkan data dosen dan kinerja. 7. *Actor* memilih data kinerja bidang pengabdian/ penunjang. 8. Sistem meminta data pengabdian/penunjang. 9. *Database* mengembalikan data kinerja bidang pengabdian/penunjang. 10. Sistem menampilkan data pengabdian/penunjang.
Postconditions	*Actor* berhasil memilih dosen serta mendapatkan data kinerja bidang pengabdian dan penunjang berdasarkan pilihan tersebut
Sub Flows	a. *Filter* data kinerja per tahun dan semester tertentu
Alternate/ Exceptional Flows	A1: Jika waktu sesi habis, maka secara otomatis token akan berubah menjadi tidak aktif dan sesi akan berakhir. A2: Jika data dosen tidak ditemukan, sistem akan menampilkan pesan dosen tidak ditemukan. A3: Jika *actor* melakukan *filter* data kinerja berdasarkan tahun dan semester namun data kinerja tersebut tidak ada, maka sistem akan menampilkan data kosong.

Tabel 17. *Use case description* melihat kinerja bidang pengabdian dan penunjang

Use Case Name	Melihat Kinerja Bidang Pengabdian dan Penunjang
ID	UC-07 & UC-08
Priority	*High*
Actor	Admin, Dosen
Description	*Actor* dapat mengakses fitur melihat kinerja bidang pengabdian dan penunjang miliknya sendiri
Trigger	*Actor* memilih fitur melihat kinerja bidang pengabdian atau penunjang
Precondition	*Actor* sudah *login* sebagai admin atau dosen dan memiliki status sesi yang aktif
Normal Course	1. *Actor* sudah berada dalam sistem dan memiliki status sesi yang aktif. 2. Sistem menampilkan data dosen dan kinerja. 3. *Actor* memilih data kinerja bidang pengabdian/ penunjang. 4. Sistem meminta data pengabdian/penunjang. 5. *Database* mengembalikan data kinerja bidang pengabdian/penunjang. 6. Sistem menampilkan data pengabdian/penunjang.
Postconditions	*Actor* berhasil melihat data kinerja bidang pengabdian dan penunjang berdasarkan pilihan tersebut
Sub Flows	a. *Filter* data kinerja per tahun dan semester tertentu
Alternate/ Exceptional Flows	A1: Jika waktu sesi habis, maka secara otomatis token akan berubah dan sesi akan berakhir. A2: Jika hasil *filter* data kinerja berdasarkan tahun dan semester tertentu tidak ada, maka sistem akan menampilkan data kosong.

Tabel 15, 16, dan 17 adalah *use case description* yang menjabarkan proses melihat kinerja bidang pengabdian atau penunjang berdasarkan hak akses pengguna. Pada alur utamanya, Dekan dapat melihat data kinerja tingkat fakultas berdasarkan program studi dan dosen yang dipilih, Kaprodi dapat melihat data kinerja tingkat program studi berdasarkan dosen yang dipilih, dan Dosen dapat melihat data kinerjanya sendiri. Selain itu, ada skenario tambahan berupa *filter* tahun dan semester serta skenario yang menangani potensi kendala, seperti ketika data dosen atau hasil filter data kinerja per tahun dan semester tertentu tidak ada, sistem akan menampilkan data kosong.

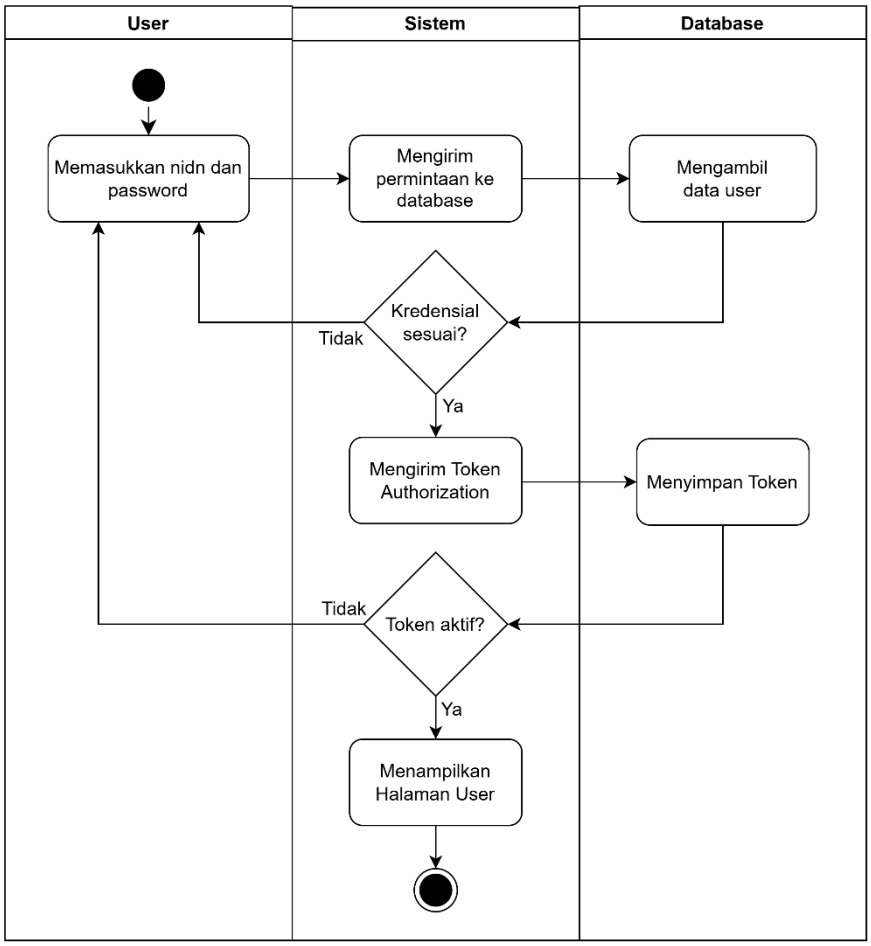
Tabel 18. *Use case description* melakukan *log out*

Use Case Name	Melakukan *Logout*
ID	UC-09
Priority	*High*
Actor	Admin, Dekan, Kaprodi, Dosen
Description	*Actor* dapat keluar dari sistem dengan melakukan *log out*
Trigger	*Actor* memilih aksi *log out*
Precondition	*Actor* sudah *log in* dan memiliki status sesi yang aktif
Normal Course	1. *Actor* sudah berada dalam sistem dan memiliki status sesi yang aktif. 2. *Actor* memilih aksi *log out.* 3. Sistem memproses permintaan *log out.* 4. *Database* memperbarui status sesi menjadi nonaktif. 5. Sistem akan mengarahkan *actor* ke halaman *log in.*
Postconditions	*Actor* berhasil keluar dari sistem
Sub Flows	a. Status sesi *actor* berubah menjadi nonaktif
Alternate/ Exceptional Flows	A1: Jika waktu sesi habis, maka sistem akan mengakhiri sesi secara otomatis.

Tabel 18 berisi *use case description* untuk proses *log out* dari sistem. Pada alur utamanya, proses ini diawali dengan kondisi apabila pengguna masih memiliki sesi yang aktif, maka pengguna dapat melakukan proses *log out*. Sistem nanti akan memproses permintaan *log out* dan mengubah status sesi. Kemudian sistem akan mengarahkan pengguna ke halaman *login* sebagai tanda bahwa sesi telah berakhir. Selain itu, terdapat skenario alternatif yaitu ketika waktu sesi pengguna telah habis, maka sesi akan berubah menjadi tidak aktif secara otomatis.

c. *Activity Diagram*

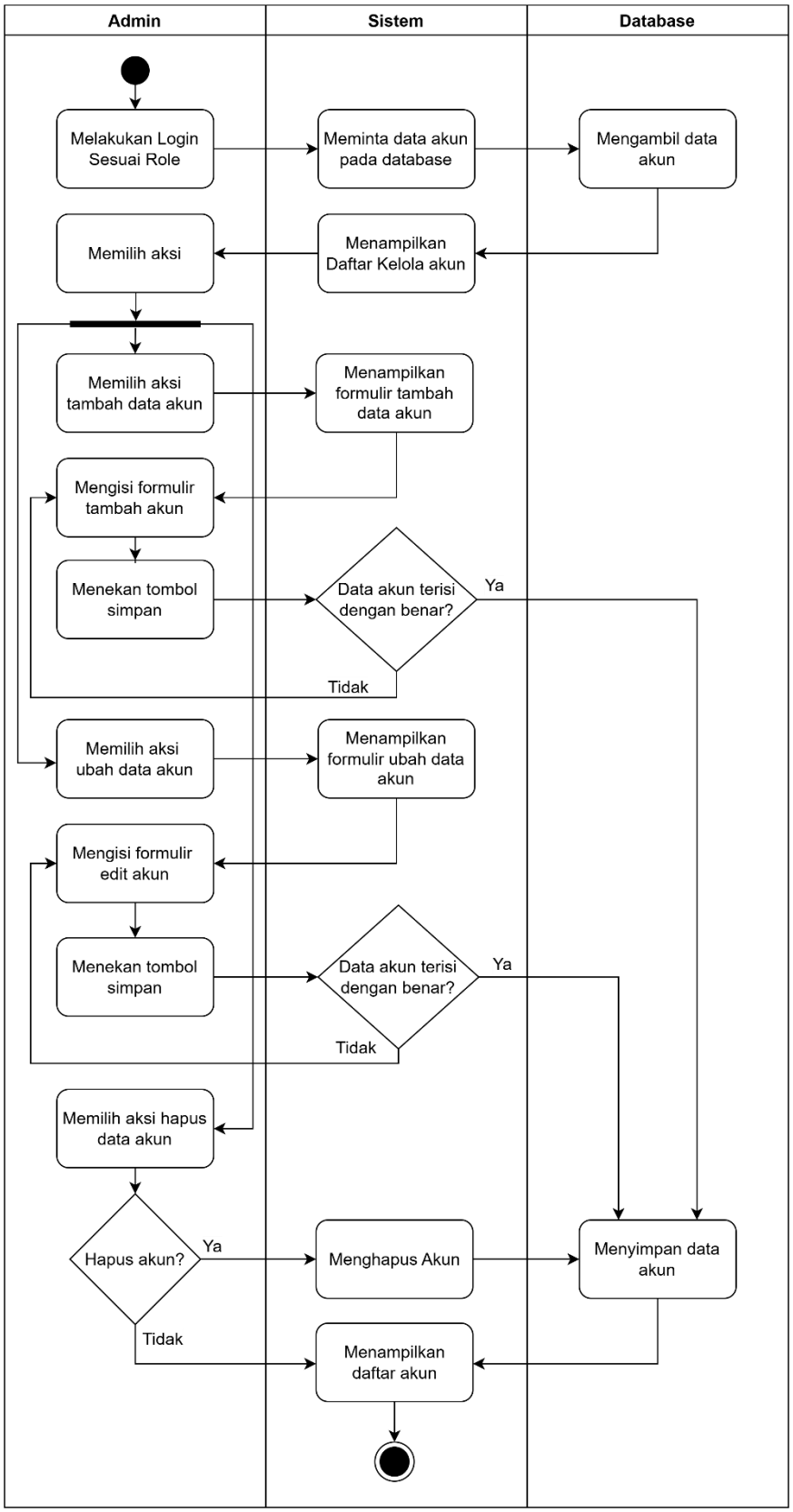
Activity diagram merupakan pengembangan dari *use case diagram*. Diagram ini digunakan untuk merancang alur kerja dari sistem yang akan dikembangkan. Dalam penelitian ini, terdapat 9 diagram yang dibuat sesuai dengan *use case* yang telah dirancang.



Gambar 5. *Activity diagram* melakukan *log in*.

Gambar 5 menjelaskan proses dari melakukan *log in*. Proses dimulai dengan pengguna mengakses halaman *login* yang berisi *form*. Pengguna dapat memasukkan NIDN dan *password* yang sudah didaftarkan. Jika data kredensial valid, sistem akan melanjutkan proses dengan menghasilkan dan mengirimkan token otorisasi.

Selanjutnya, token yang dihasilkan akan disimpan dalam 1 hari dan diverifikasi setiap pengguna mengirimkan permintaan. Apabila token masih aktif dan valid, sistem akan menampilkan halaman utama sesuai dengan hak akses pengguna. Proses ini menggambarkan mekanisme autentikasi dan otorisasi bekerja dalam sistem *log in* yang mengandalkan token, serta menegaskan pentingnya keamanan dalam pengelolaan akses pengguna berbasis kredensial dan validitas token.



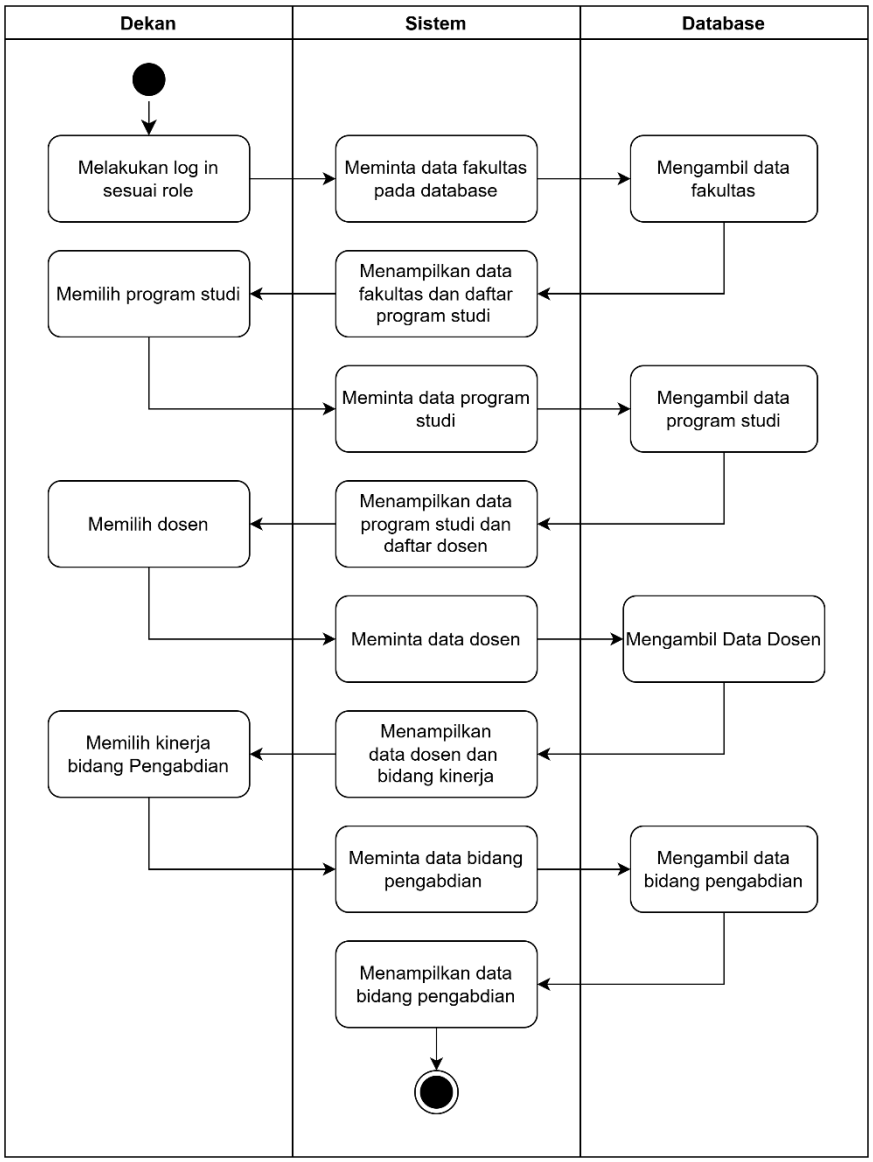
Gambar 6. *Activity diagram* mengelola akun.

Gambar 6 merupakan *activity diagram* yang menjelaskan proses pada *use case* mengelola akun. Proses pengelolaan data akun memiliki tiga fungsi, yaitu menambah, mengubah, dan menghapus data pengguna. Pengguna yang memiliki akses kelola akun adalah administrator, atau bisa juga disebut admin.

Untuk menambah akun baru, admin dapat memilih aksi tambah akun. Sistem akan menampilkan formulir tambah data dan admin dapat mengisinya dengan informasi yang diperlukan, seperti NIDN, *password*, peran, dan lainnya, lalu menekan tombol simpan. Sistem akan memvalidasi data yang dimasukkan. Jika data tersebut valid, akun baru tersebut akan disimpan ke dalam *database*. Jika terdapat kesalahan, admin akan diminta untuk memperbaiki data hingga validasi berhasil.

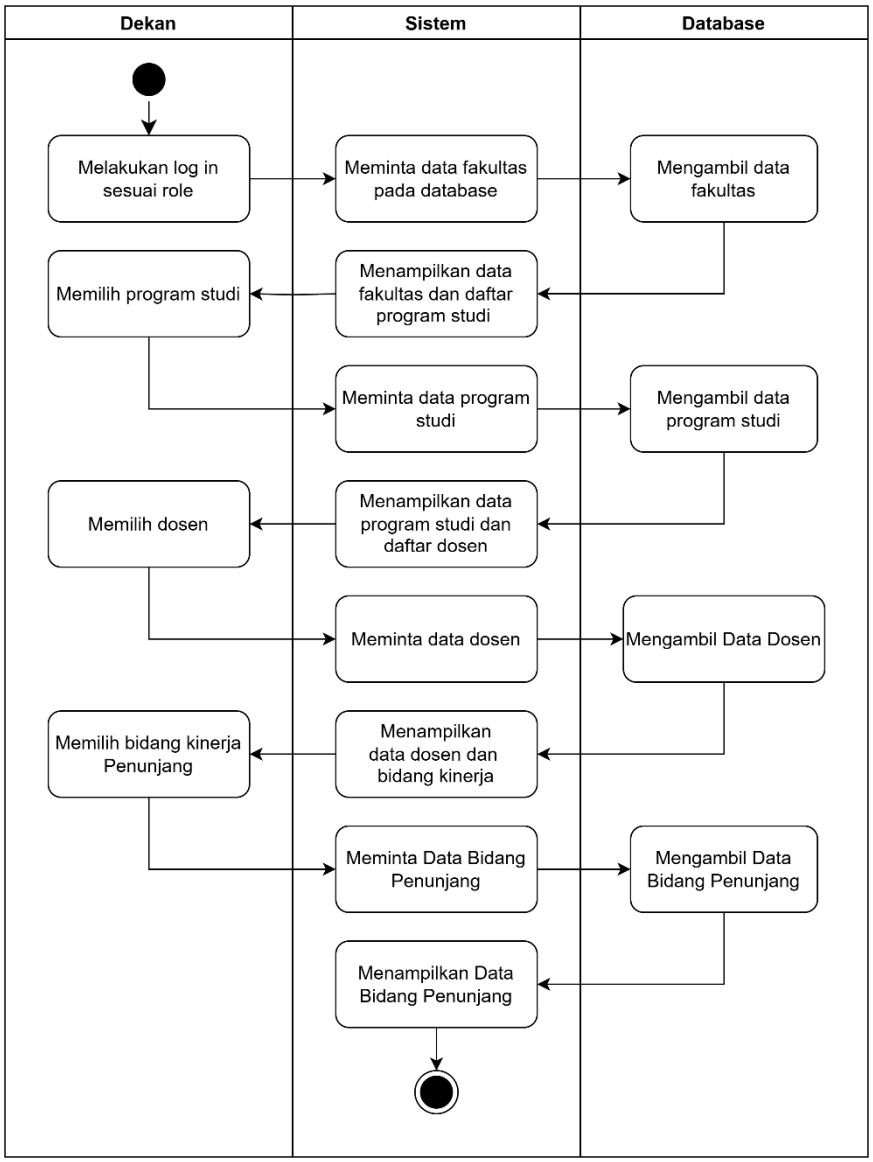
Untuk mengubah data akun, admin dapat memilih aksi ubah data pada akun yang ingin diubah. Sistem akan menampilkan formulir ubah data yang dapat diisi sesuai kebutuhannya untuk memperbarui informasi. Setelah melakukan perubahan, admin menekan tombol simpan, dan sistem memvalidasi data yang baru. Jika validasi berhasil, sistem akan memperbarui data akun ke dalam *database*. Jika terdapat kesalahan, admin perlu memperbaiki data hingga validasi berhasil.

Untuk menghapus akun, admin dapat memilih aksi hapus pada akun yang ingin dihapus. Sistem akan meminta konfirmasi untuk memastikan tindakan tersebut. Jika admin mengonfirmasi penghapusan akun, sistem akan menghapus akun dari *database*. Sebaliknya, jika admin membatalkan konfirmasi, proses penghapusan tidak dilakukan, dan kembali menampilkan halaman daftar akun tanpa perubahan.



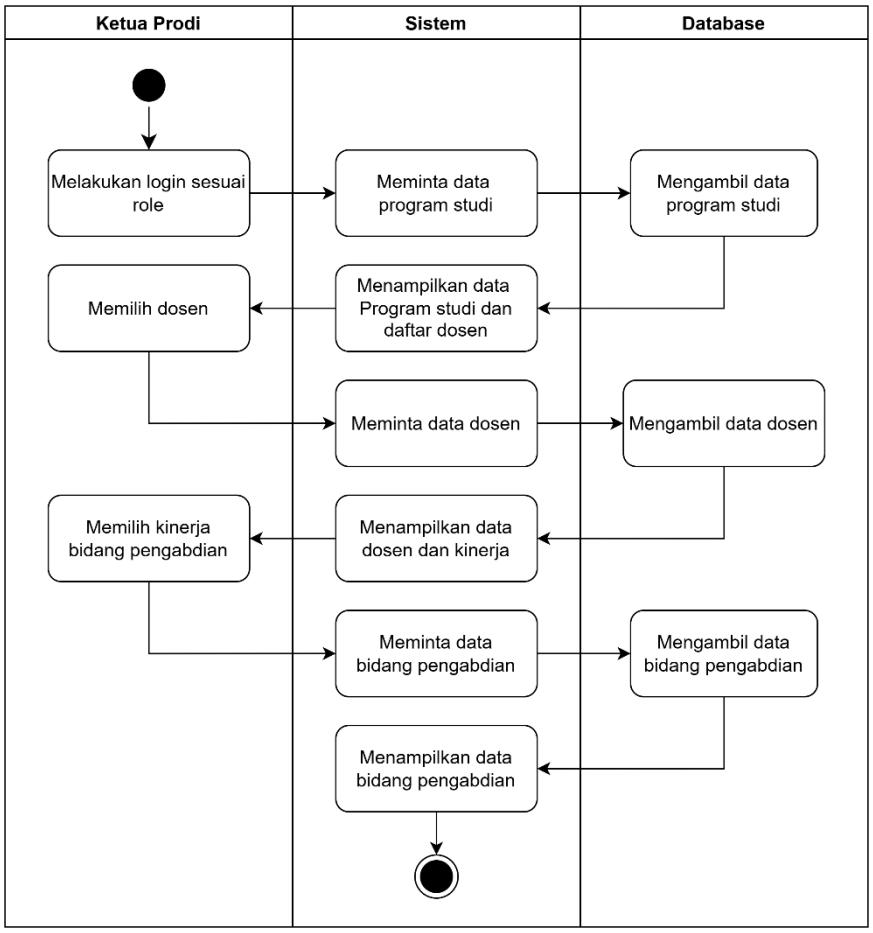
Gambar 7. *Activity diagram* melihat kinerja bidang pengabdian tingkat fakultas.

Gambar 7 adalah *activity diagram* yang menjelaskan proses *use case* melihat kinerja bidang pengabdian tingkat fakultas. Pengguna, dalam hal ini dekan, akan melakukan *log in* dan sistem akan menampilkan daftar program studi di fakultasnya. Setelah dekan memilih program studi tertentu, sistem akan menampilkan daftar dosen terkait. Dekan dapat memilih data dosen yang ingin dilihatnya, dan sistem akan menampilkan detail data dosen tersebut. Terakhir, dekan dapat melihat bidang kinerja yang ingin dilihat dari dosen tersebut, salah satunya bidang pengabdian.



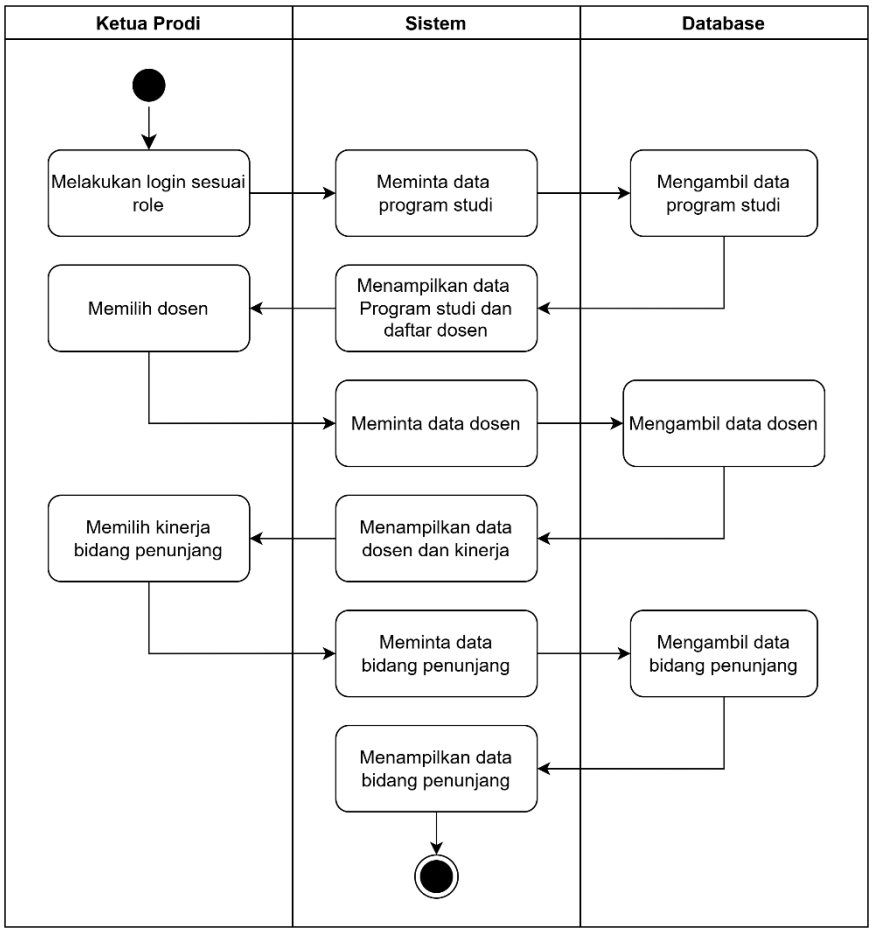
Gambar 8. *Activity diagram* melihat kinerja bidang penunjang tingkat fakultas.

Gambar 8 adalah *activity diagram* yang menjelaskan proses dari *use case* melihat kinerja bidang penunjang tingkat fakultas. Sama seperti bidang pengabdian, proses dimulai ketika pengguna, yaitu dekan melakukan *log in*. Kemudian dekan memilih program studi dari daftar yang ditampilkan, lalu memilih dosen untuk melihat detail kinerja. Selanjutnya dekan dapat memilih bidang kinerja yang ingin dilihat dari dosen tersebut, dalam hal ini bidang penunjang.



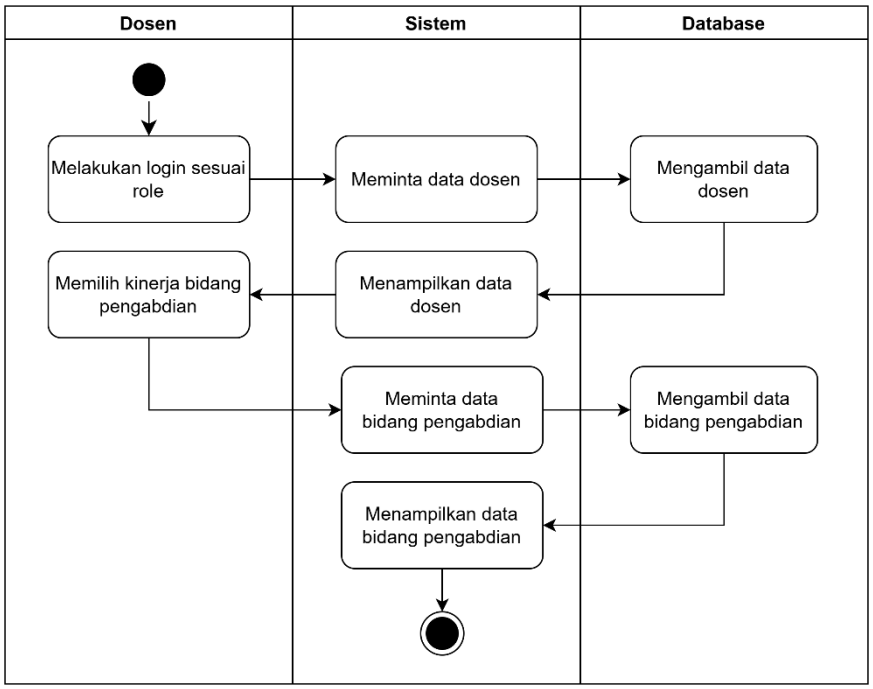
Gambar 9. *Activity diagram* melihat kinerja bidang pengabdian tingkat program studi.

Gambar 9 merupakan *activity diagram* yang menjelaskan proses dari *use case* melihat kinerja bidang pengabdian tingkat program studi. Dimulai dengan pengguna, yaitu kaprodi, melakukan *log in* sesuai perannya. Sistem kemudian meminta data dosen dan menampilkannya dalam bentuk daftar. Kaprodi dapat memilih dosen tertentu yang berada dalam daftar tersebut untuk melihat detailnya. Sistem akan menampilkan detail dosen tersebut beserta data kinerjanya. kemudian kaprodi dapat memilih bidang kinerja yang ingin dilihat, yaitu pengabdian, di mana sistem akan meminta data terkait bidang pengabdian dari *database*, kemudian menampilkan informasi tersebut.

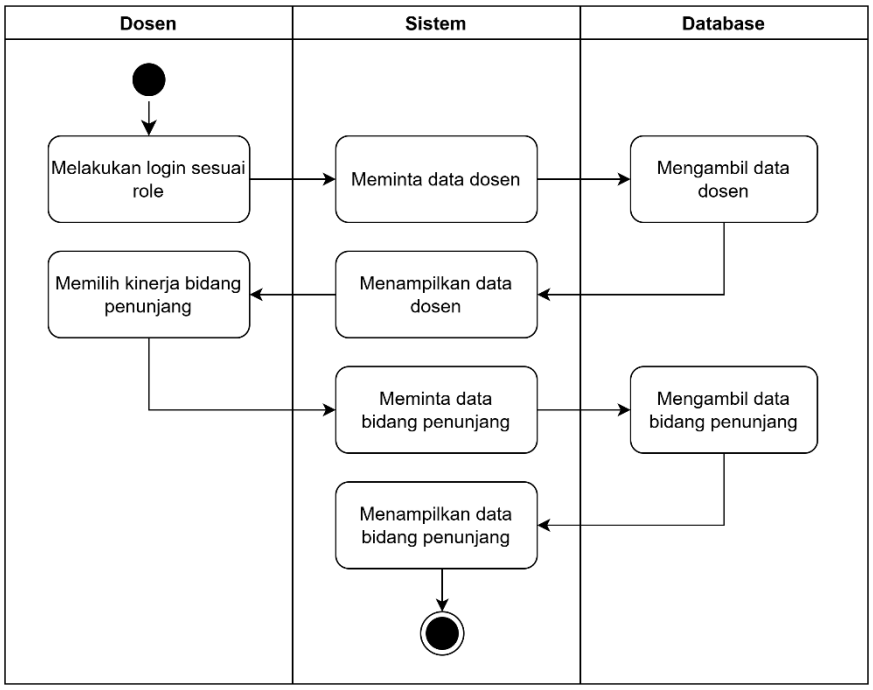


Gambar 10. *Activity diagram* melihat kinerja bidang penunjang tingkat program studi.

Gambar 10 adalah *activity diagram* yang menjelaskan proses dari *use case* melihat kinerja bidang penunjang tingkat program studi. *Activity diagram* ini memiliki alur kerja yang sama dengan bidang pengabdian. Setelah kaprodi memilih bidang kinerja, sistem akan meminta data terkait bidang penunjang dari *database*. Kemudian data tersebut akan ditampilkan oleh sistem untuk membantu kaprodi dalam mengakses informasi terkait bidang penunjang. Proses ini memastikan bahwa semua data bidang kinerja dapat diakses dengan mudah dan efisien oleh kaprodi dalam mendukung pengelolaan data yang diperlukan.



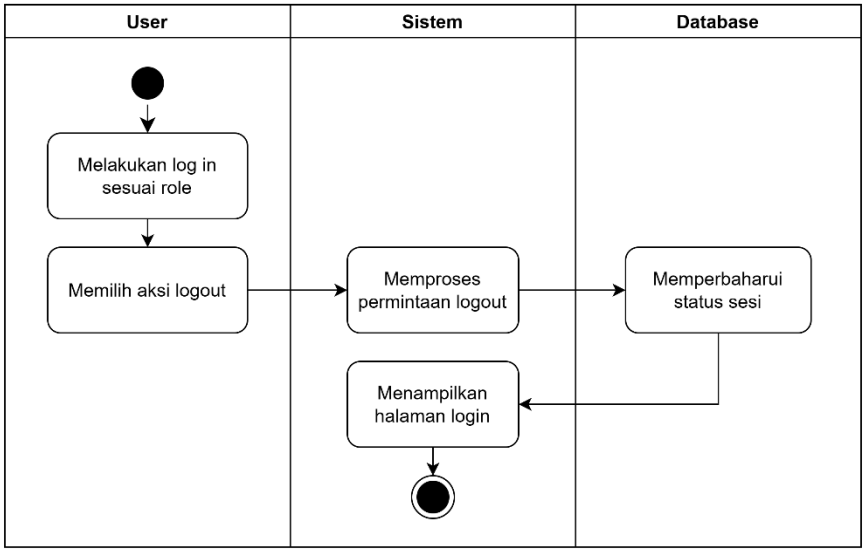
Gambar 11. *Activity diagram* melihat kinerja bidang pengabdian.



Gambar 12. *Activity diagram* melihat kinerja bidang penunjang.

Gambar 11 dan 12 adalah proses dari melihat kinerja bidang pengabdian dan penunjang.

Setelah *log in*, pengguna, yang dalam hal ini adalah dosen, akan meminta data dosen dan data kinerja yang dimilikinya kepada *database*. *Database* kemudian mengirimkan data yang diminta, dan sistem akan menampilkan data kinerja. Dosen kemudian dapat memilih bidang yang ingin dilihat, yaitu bidang pengabdian atau penunjang, dan sistem akan menampilkannya.



Gambar 13. *Activity diagram* melakukan *log out*.

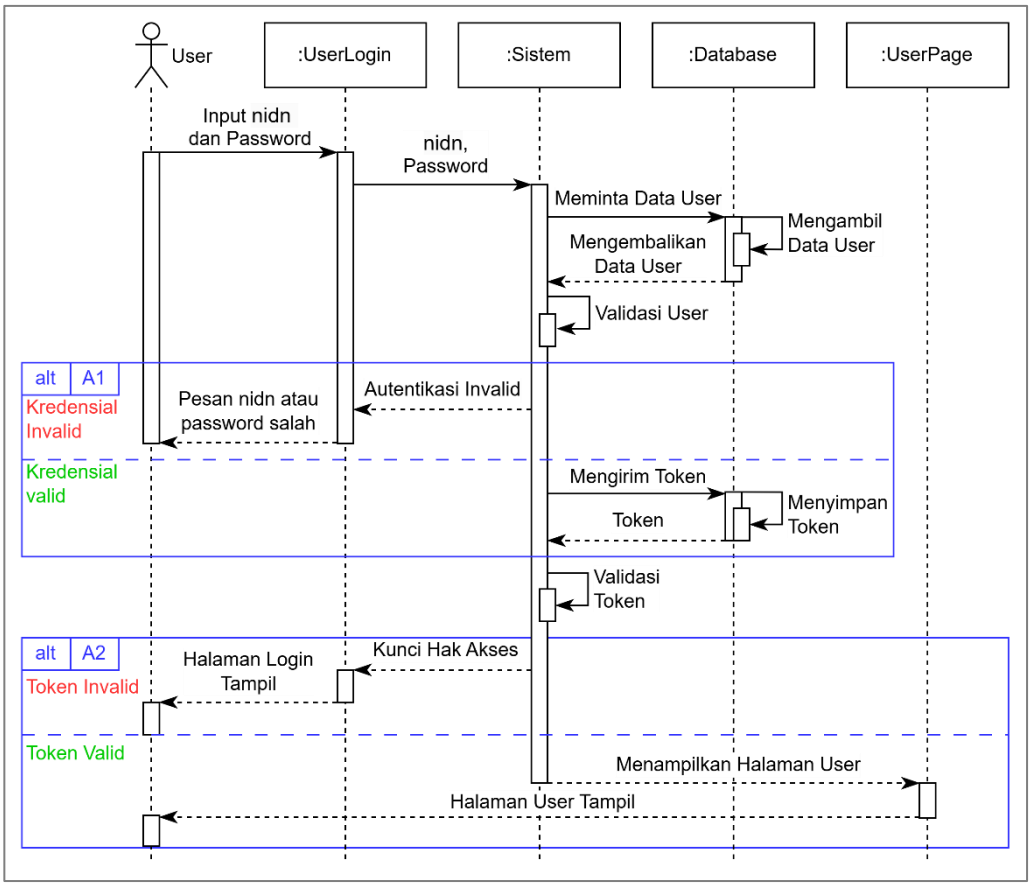
Gambar 13 adalah *activity diagram* yang menggambarkan proses dari *use case* melakukan *logout*. Proses ini dilakukan ketika pengguna sudah masuk ke dalam sistem, dan hendak keluar dari sistem tersebut. Pengguna dapat memilih aksi *logout* untuk mengakhiri sesi aktif mereka. Sistem akan memproses permintaan *logout* dengan mengirimkan perintah ke *database* untuk memperbarui status sesi pengguna menjadi tidak aktif. Setelah perubahan sesi berhasil disimpan, sistem akan mengarahkan pengguna kembali ke halaman *login* sebagai konfirmasi berhasil keluar dari akun. Proses ini akan memastikan keamanan dan mencegah dari akses yang tidak sah.

d. *Sequence Diagram*

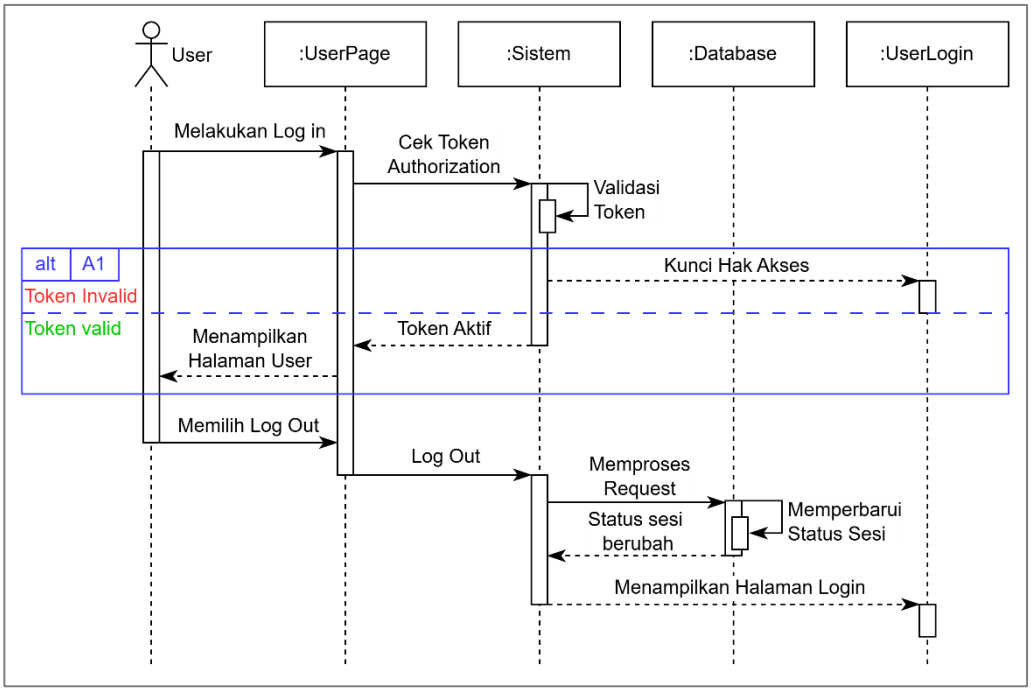
Sequence diagram merupakan penjelasan lebih lanjut dari *use case description* dan *activity diagram* yang berfungsi untuk menggambarkan alur komunikasi antara berbagai objek dalam suatu sistem berdasarkan urutan waktu. Diagram ini digunakan untuk memodelkan skenario sistem yang melibatkan berbagai entitas, seperti pengguna, antarmuka, sistem, maupun *database*, sehingga alur proses sistem dapat dipahami dengan lebih jelas.

Dalam konteks pengembangan API pada penelitian ini, *sequence diagram* memiliki tujuan untuk memvisualisasikan alur data dan interaksi antara sistem yang akan terintegrasi dengan *database* remunerasi melalui API yang akan dibangun. Dengan *seqeuence diagram*, pihak-pihak yang terlibat dalam pengembangan ini, seperti tim pengembang dan pemangku kepentingan, dapat memahami dengan mudah bagaimana setiap komponen akan berinteraksi. Selain itu, diagram ini juga berguna pada proses pengujian, karena dapat digunakan sebagai patokan untuk memastikan pengembangan API yang akan dibangun dapat bekerja sesuai dengan rancangan.

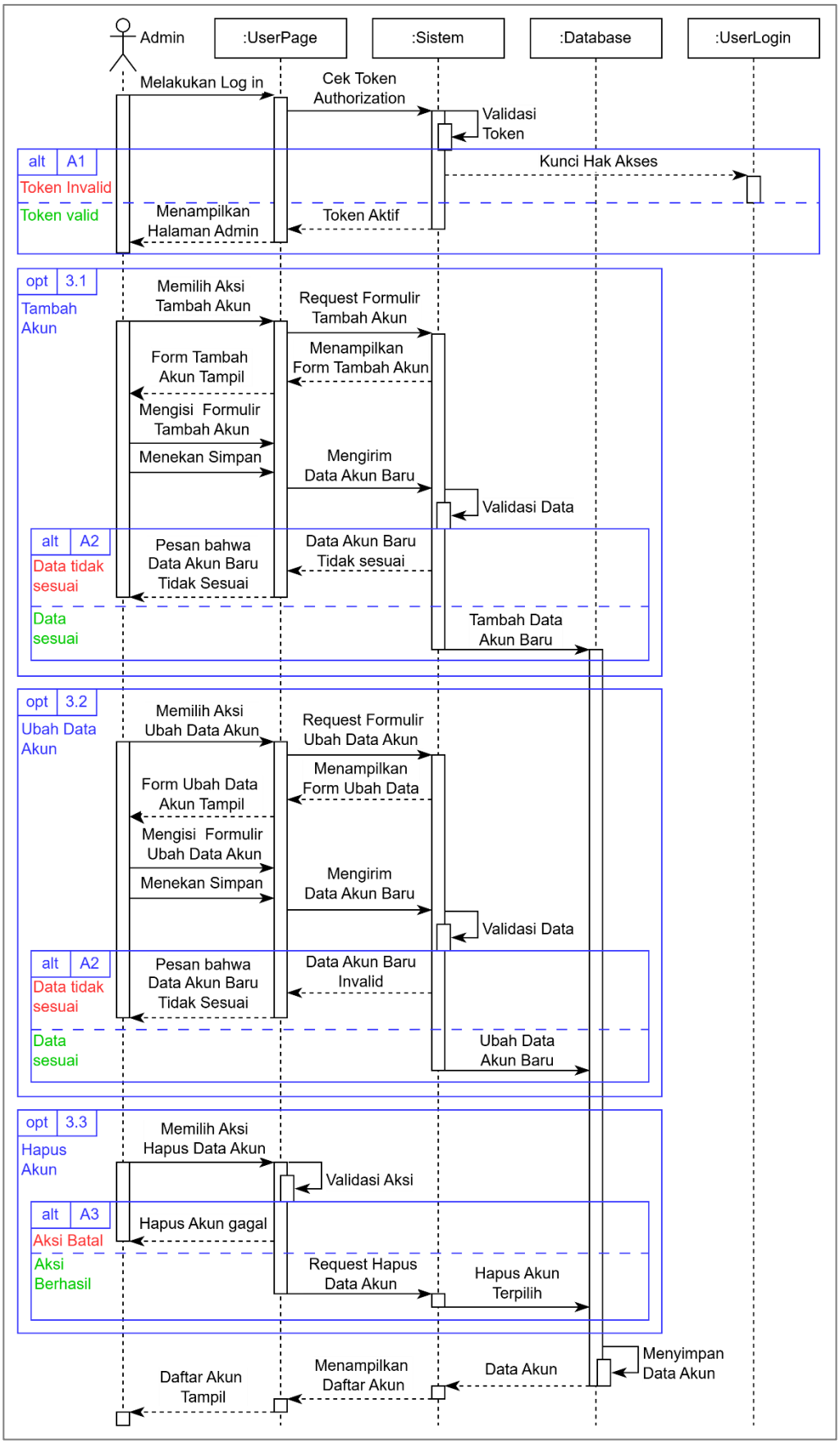
Gambar 14 hingga 22 adalah gambaran *sequence diagram* dari berbagai proses utama dalam sistem, mulai dari proses *log in*, pengelolaan akun pengguna, hingga fungsi melihat data kinerja dosen dan proses *log out*. Dalam tiap proses menjalankan fungsinya, tidak lupa digambarkan pula tentang alur autentikasi dan otorisasi guna memastikan keamanan data dalam sistem. Dengan memvisualisasikan seluruh alur proses melalui *sequence diagram*, pengembangan API akan menjadi lebih terstruktur serta dapat memudahkan identifikasi potensi masalah yang mungkin terjadi selama proses implementasi nantinya.



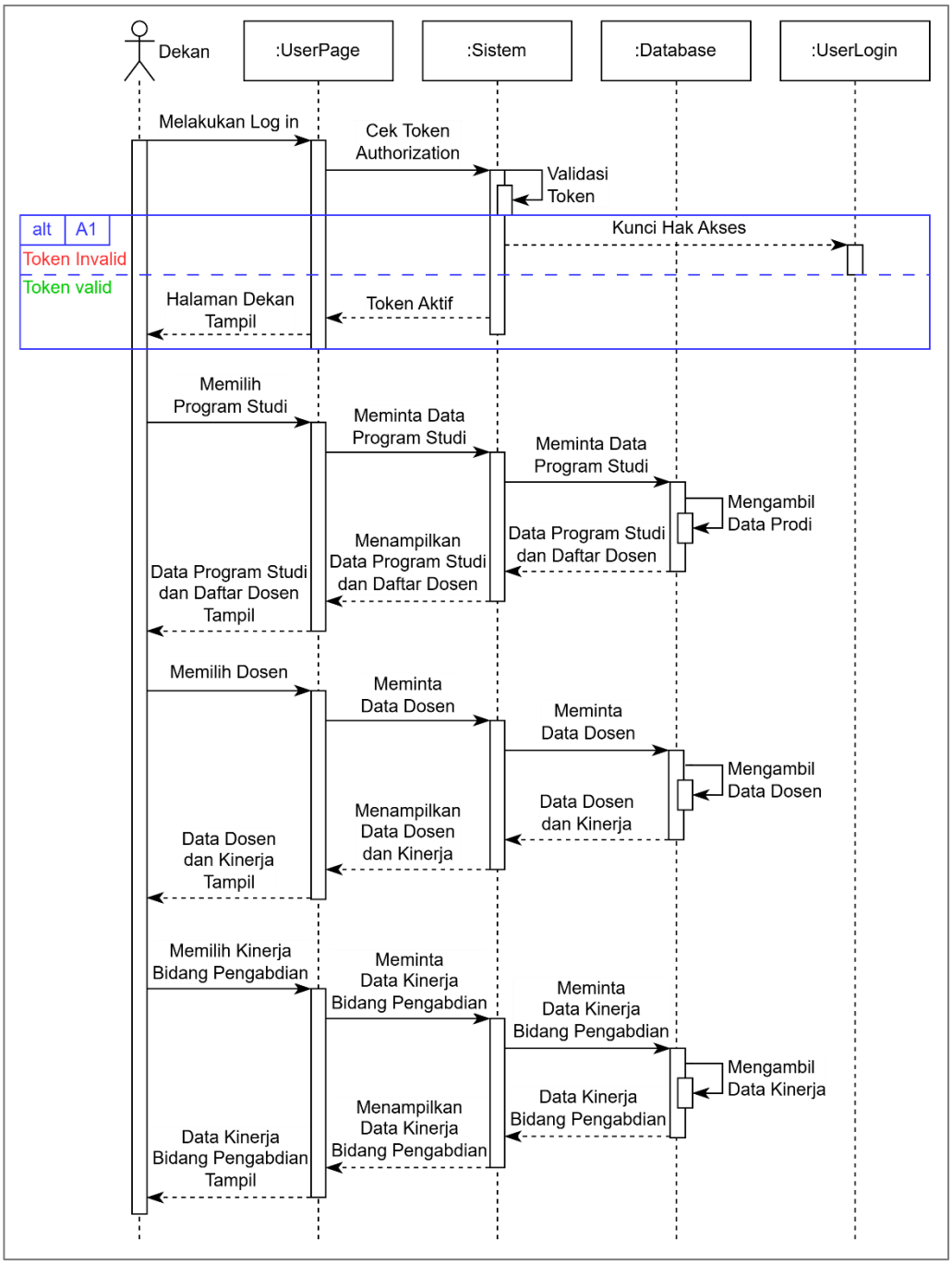
Gambar 14. *Sequence diagram* melakukan *log in*.



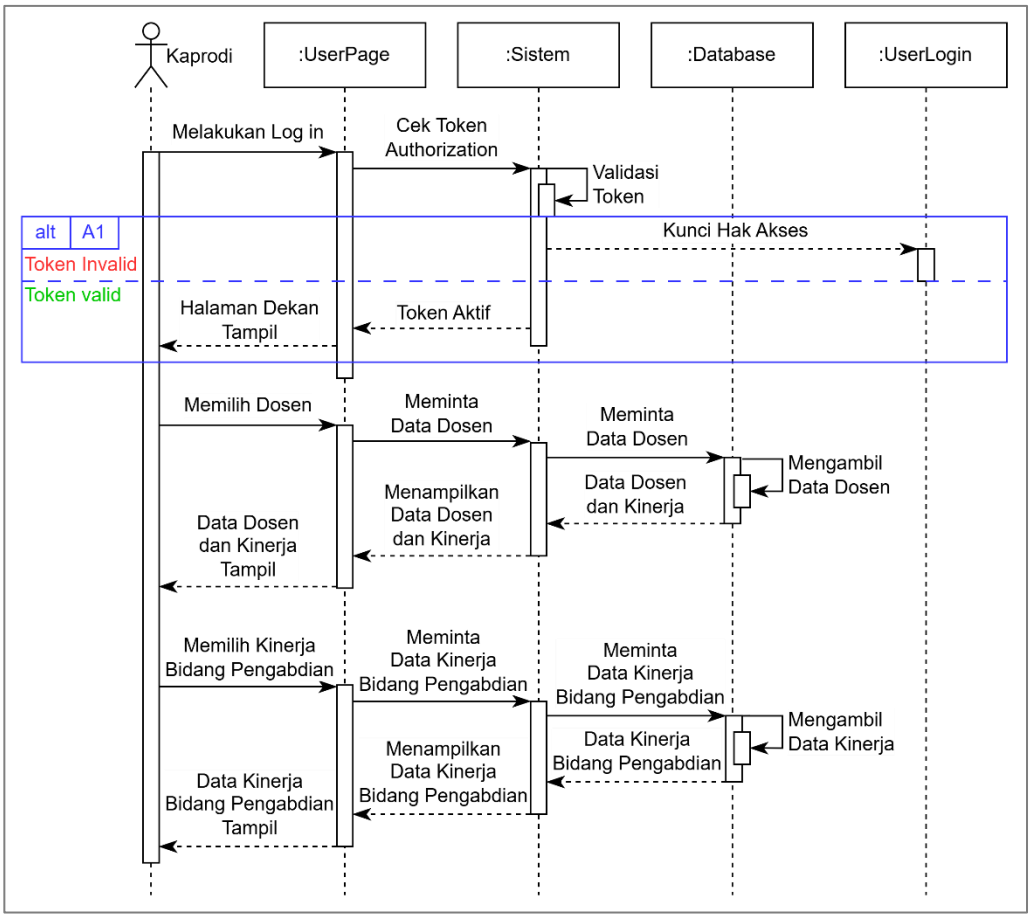
Gambar 15. *Sequence diagram* melakukan l*og out*.



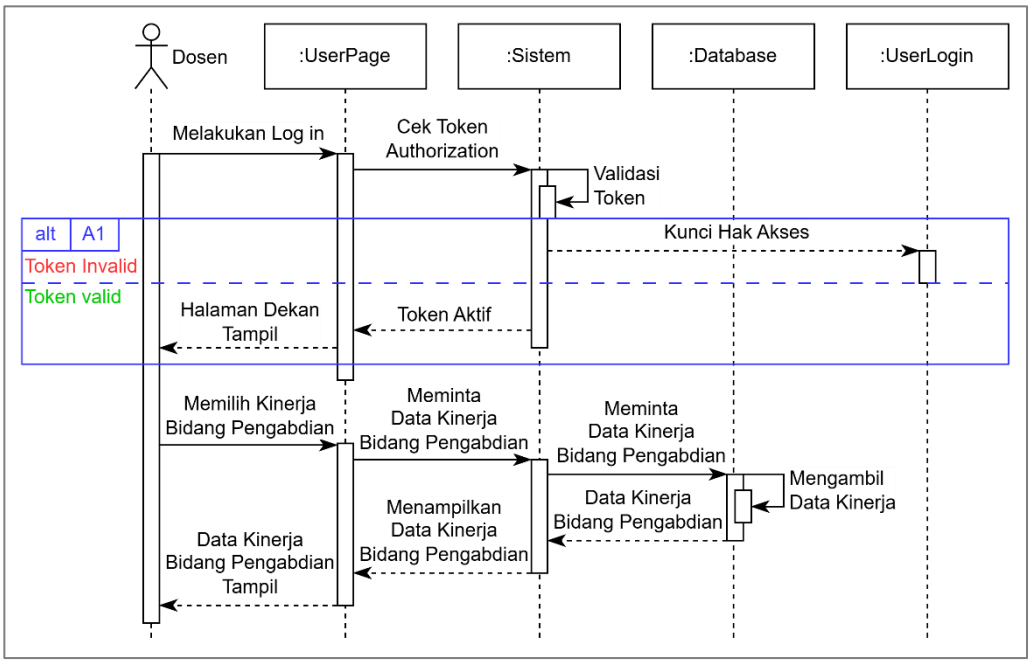
Gambar 16. *Sequence diagram* mengelola akun pengguna.



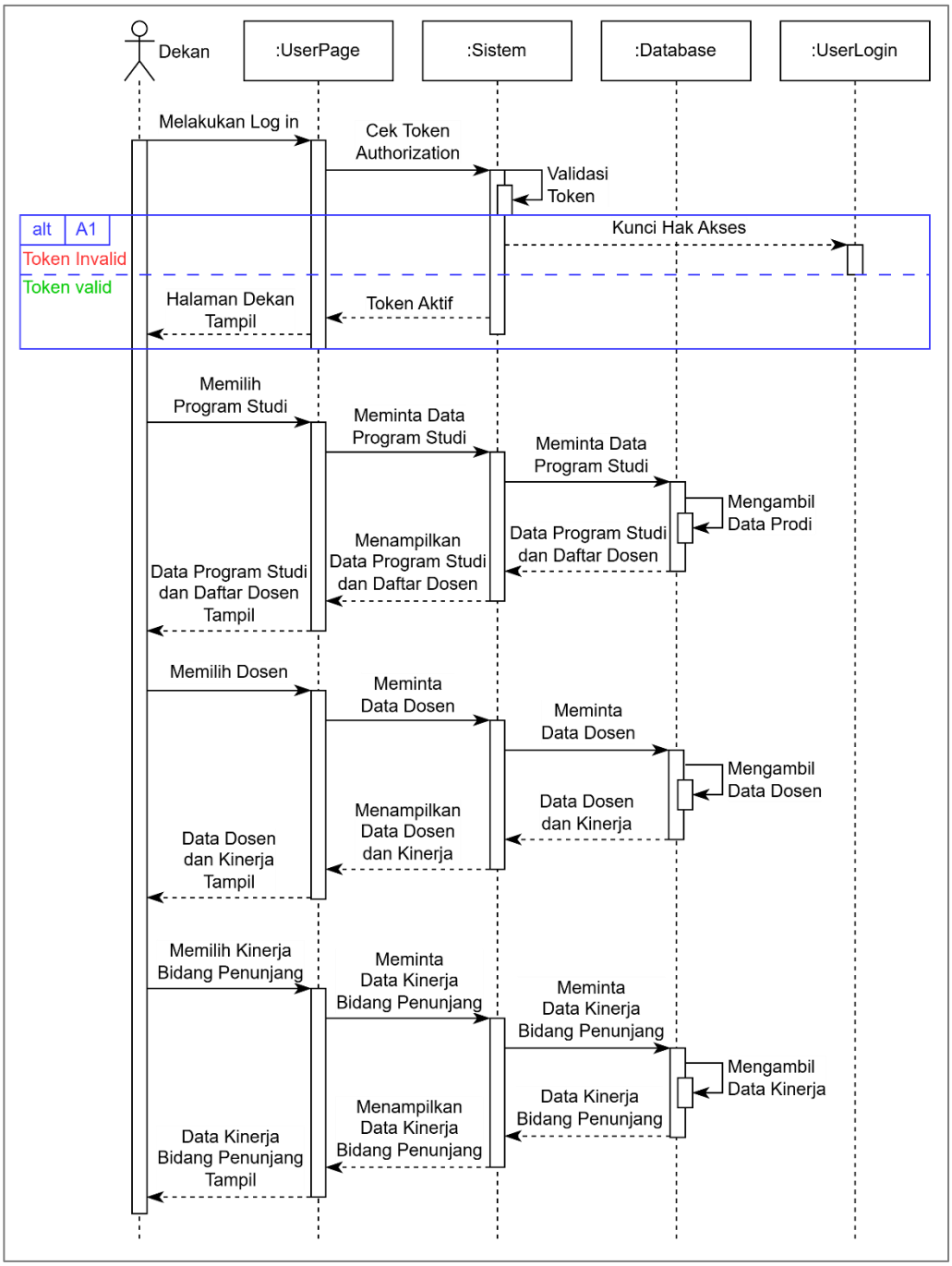
Gambar 17. *Sequence diagram* melihat kinerja bidang pengabdian tingkat fakultas.



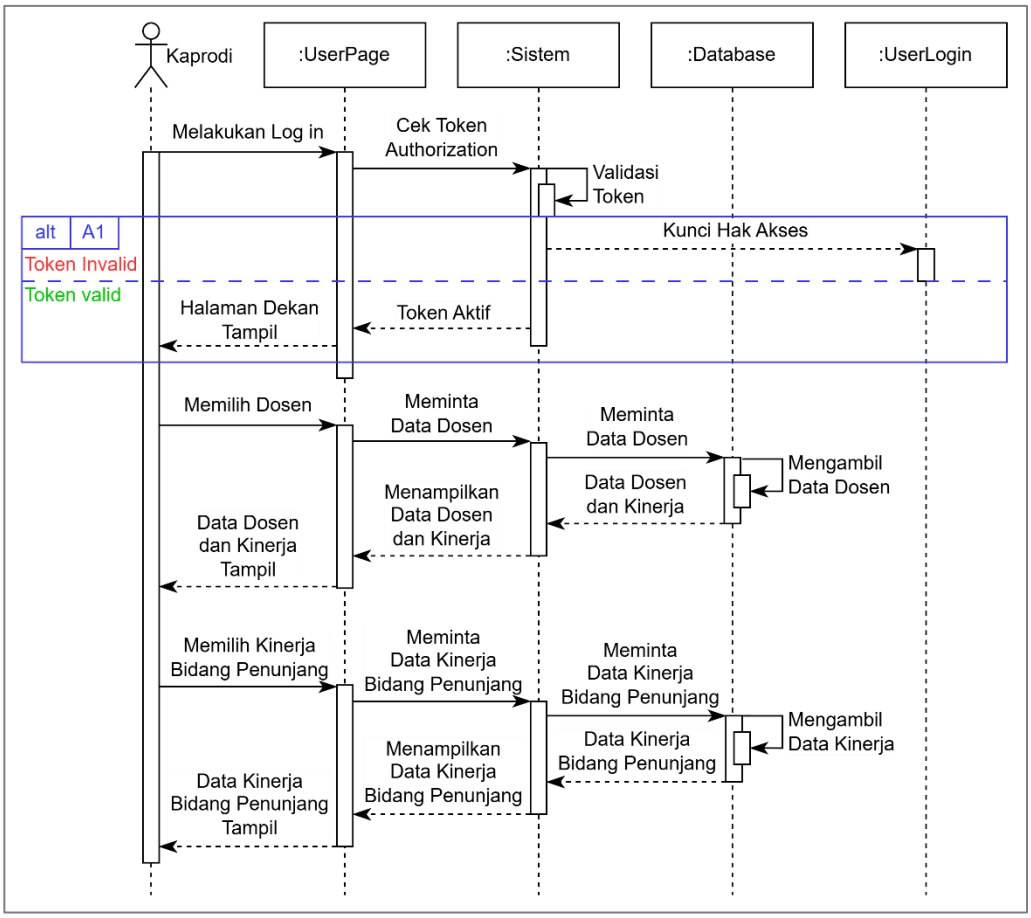
Gambar 18. *Sequence diagram* melihat kinerja bidang pengabdian tingkat program studi.



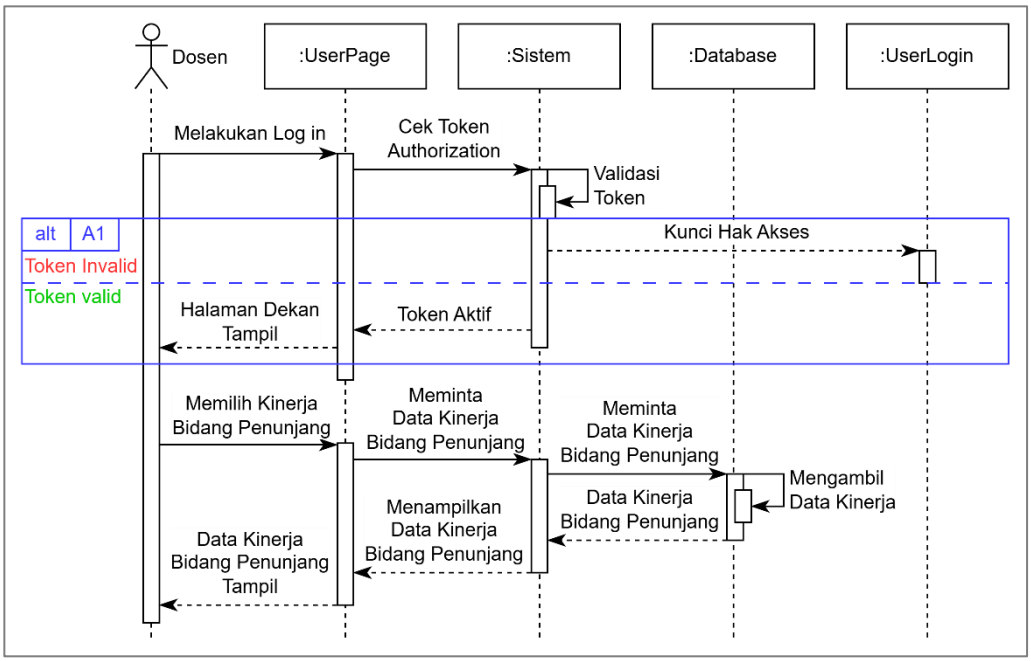
Gambar 19. *Sequence diagram* melihat kinerja bidang pengabdian.



Gambar 20. *Sequence diagram* melihat kinerja bidang penunjang tingkat fakultas.



Gambar 21. *Sequence diagram* melihat kinerja bidang penunjang tingkat program studi.

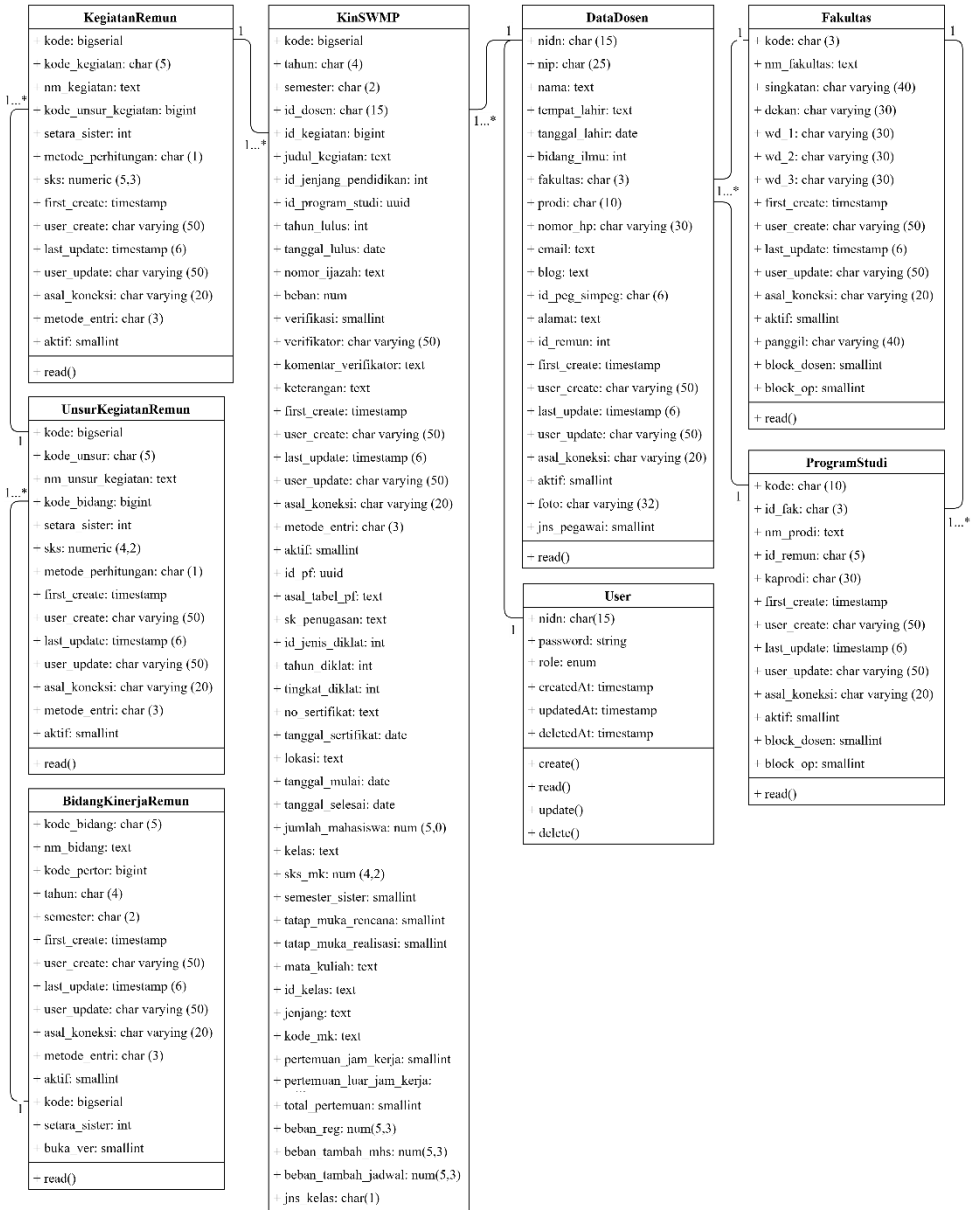


Gambar 22. *Sequence diagram* melihat kinerja bidang penunjang.

2. *Design* (Perancangan API)

Proses *design* bertujuan untuk memastikan integrasi data dan fungsi dapat berjalan secara optimal. Pada tahap ini, dilakukan perancangan *class diagram* untuk menggambarkan struktur objek dan relasinya, serta rancangan *endpoint* API untuk menentukan *client-server* berkomunikasi.

a. *Class* Diagram



Gambar 23. *Class diagram*.

Class diagram pada penelitian ini, yang digambarkan pada Gambar 23 di halaman sebelumnya, memiliki 7 kelas yang sudah tersedia pada sistem remunerasi. Terdapat juga 1 kelas tambahan yaitu "*User*" yang dirancang untuk mendukung penyesuaian dalam pengembangan sistem ini. Berikut adalah kelas-kelas yang akan digunakan pada pengembangan sistem.

a) Kelas Bidang Kinerja Remunerasi (BidangKinerjaRemun)
   Kelas ini berfungsi untuk menyimpan data mengenai kategori bidang kinerja. Jenis-jenis bidang yang tercatat meliputi bidang pendidikan, pelaksanaan pendidikan, penelitian, pengabdian, serta penunjang, yang menjadi dasar dalam proses evaluasi kinerja dosen.

b) Kelas Unsur Kegiatan Remunerasi (UnsurKegiatanRemun)
   Kelas ini digunakan untuk menyimpan data terkait unsur kegiatan yang menjadi bagian dari setiap bidang kinerja. Informasi dalam kelas ini mencakup jenis-jenis unsur kegiatan yang relevan dengan rubrik penilaian kinerja dosen.

c) Kelas Kegiatan Remunerasi (KegiatanRemun)
   Kelas ini berfungsi untuk mencatat data terkait jenis kegiatan yang dilakukan oleh dosen dalam setiap unsur kegiatan. Informasi yang tercatat dalam kelas ini meliputi nama kegiatan terkait.

d) Kelas Kinerja SWMP (KinSWMP)
   Kelas ini digunakan untuk menyimpan data yang lebih mendetail terkait kegiatan dosen. Data ini mencakup informasi spesifik seperti deskripsi kegiatan, waktu pelaksanaan, dan bukti pendukung lainnya.

e) Kelas Dosen (DataDosen)

Kelas ini bertugas menyimpan informasi personal dan profesional dosen yang menjadi bagian dari sistem remunerasi. Data yang dikelola dalam kelas ini mencakup nama lengkap, NIP, NIDN, serta afiliasi dosen dengan program studi dan fakultas tertentu.

f) Kelas Program Studi (ProgramStudi)

Kelas ini digunakan untuk menyimpan data terkait program studi yang ada di Universitas Lampung. Informasi yang tercatat dalam kelas ini meliputi nama program studi serta relasi ke fakultas terkait.

g) Kelas Fakultas (Fakultas)

Kelas ini berfungsi untuk mencatat informasi mengenai fakultas di Universitas Lampung, seperti nama fakultas, singkatannya, serta data lain yang mendukung.

h) Kelas Pengguna (*User*)

Kelas *User* menyimpan data pengguna sistem yang mencakup identitas, informasi autentikasi, serta perannya dalam sistem, seperti admin, dosen, kaprodi, atau dekan. Data ini digunakan untuk mengatur hak akses terhadap fitur dan data dalam sistem, sehingga hanya pengguna yang memiliki wewenang tertentu yang dapat melakukan tindakan spesifik.

Kelas-kelas tersebut dirancang untuk saling terhubung guna memastikan struktur data yang terstruktur dan konsisten. Dengan merancang *database* berbentuk *class diagram*, penggambaran relasi antar entitas yang jelas akan mendukung pengembangan sistem yang terintegrasi secara optimal. Selain itu, perancangan diagram ini juga bertujuan untuk meminimalkan redundansi data dan mempermudah perancangan *endpoint* dalam pengembangan API sistem ini.

b. API *Endpoint*

Perancangan API *endpoint* dilakukan untuk menentukan bagaimana setiap komponen sistem, seperti *server* dan *client*, akan saling berkomunikasi secara efisien melalui layanan API. Proses ini melibatkan penentuan struktur *endpoint*, metode HTTP dan parameter yang digunakan, serta format data *request* dan *response* untuk memastikan interaksi antar bagian dalam sistem dapat berjalan lancar dan sesuai dengan kebutuhan fungsional sistem.

Tabel 19. Spesifikasi API *endpoint* melakukan *log in*

Method		POST
Path		/api/auth/login
Description		Pengguna masuk dan mendapatkan token autentikasi dan otorisasi
Parameter		-
Request	***Header***	-
	Body	nidn*, password*

Response	***Success***	status code: 200 OK
		message: “Login successful”
		data: {
		token: string
		}

	Failed	status code: 401 Unauthorized message: “Please Provide NIDN and Password”
		status code: 401 Unauthorized message: “Invalid NIDN or password”

Tabel 19 merupakan spesifikasi API *endpoint* dari fungsi melakukan *log in*. Ketika melakukan *log in*, pengguna akan mengirimkan *request* berisi NIDN dan *password* melalui *endpoint*. Sistem kemudian akan memvalidasi kecocokan data yang tersimpan di *server*. Jika validasi berhasil, sistem akan memberikan *response* berupa token otorisasi berbasis JWT (*authorization token*). Token ini berfungsi untuk autentikasi yang memungkinkan pengguna dapat mengakses fitur dan data yang diizinkan sesuai dengan peran serta hak aksesnya.

Tabel 20. Rancangan API *endpoint* menampilkan data pengguna

Method		GET
Path		/api/users
Description		Admin dapat melihat data pengguna
Parameter		nidn, role, limit, sort, order, search
Request	***Header***	Authorization*: bearer {token}
	Body	-

Response	***Success***	status code: 200 OK
		message: "Get all user successful"
		data: [{
		nidn: char (15),
		role: enum,
		password: string,
		dataDosen: {
		nidn: char (15),
		nip : char (25),
		nama: text,
		fakultas: char (3),
		prodi: char (10),
		id_remun: int,
		foto: char varying (32)
		}
		}]

	Failed	status code: 404 Not Found message: "User Not Found"

Tabel 20 mendeskripsikan rancangan API *endpoint* dari fungsi menampilkan data pengguna. Admin dapat melihat data pengguna karena memiliki otoritas. Langkahnya, API akan mengirimkan *request* data seluruh pengguna yang terdaftar berdasarkan berbagai parameter, seperti *role*, nidn, dan lainnya, serta *sort* dan *search* untuk mempermudah dalam menampilkan data yang diinginkan. Jika data pengguna ditemukan, *database* akan mengembalikan *response* sukses beserta daftar pengguna berisi berbagai data yang terkait dengan akun pengguna. Sedangkan jika data tidak ditemukan, API akan memberikan *response* gagal dan data tidak berhasil ditampilkan.

Tabel 21. Rancangan API *endpoint* menambah pengguna

Method		POST
Path		/api/users
Description		Admin dapat menambah pengguna baru
Parameter		-
Request	***Header***	Authorization*: bearer {token}
	Body	nidn*, password*, confirmPassword*, role*

Response	***Success***	status code: 201 Created
		message: “Add user successful”
		data: {
		nidn: char (15),
		role: enum,
		password: string,
		createdAt: timestamp,
		updatedAt: timestamp,
		deletedAt: timestamp
		}

	Failed	status code: 400 Bad Request message: “NIDN was registered”
		status code: 400 Bad Request message: “Invalid Data Input”

Tabel 21 mendeskripsikan rancangan API *endpoint* dari fungsi menambah pengguna. Admin dapat mengelola data pengguna karena memiliki otoritas. Langkahnya, ketika ingin menambah pengguna baru, admin akan mengisi *form* berisi data pengguna yang ingin ditambahkan. Data masukan tersebut akan dikirim di dalam *request body* untuk selanjutnya diperiksa oleh sistem apakah data tersebut sudah ada dalam *server* atau belum. Jika belum ada dan tidak ada duplikasi data, proses penambahan data berhasil. Jika proses penambahan berhasil, API akan mengembalikan *response* sukses beserta informasi pengguna yang telah dimasukkan. Namun, jika terjadi kesalahan, seperti *form* tidak terisi dengan benar atau redundansi data, API akan mengembalikan *response* gagal menambahkan data dan admin akan diminta kembali untuk mengisi *form* sesuai ketentuan dari pesan kesalahan yang dijelaskan di dalamnya.

Tabel 22. Rancangan API *endpoint* mengubah data pengguna

Method		PUT
Path		/api/users?nidn={nidn}
Description		Admin dapat mengubah data pengguna
Parameter		nidn*
Request	***Header***	Authorization*: bearer {token}
	Body	role*, password, confirmPassword

Response	***Success***	status code: 200 OK
		message: "Update User Successful"
		data: {
		nidn: char (15),
		role: enum,
		password: string,
		createdAt: timestamp,
		updatedAt: timestamp,
		deletedAt: timestamp
		}

	Failed	status code: 400 Bad Request message: "Invalid Data Input"

Tabel 22 merupakan rancangan API *endpoint* dari fungsi mengubah data pengguna. Langkahnya, admin akan mengirimkan perubahan data melalui *request body*, kemudian sistem akan memeriksa data baru yang diterima. Jika berhasil dan tidak menyebabkan konflik seperti duplikasi, API akan mengembalikan *response* sukses dan data tersebut akan diperbarui. Jika terjadi kesalahan, API akan mengembalikan *response* gagal beserta pesan kesalahan untuk diperbaiki.

Tabel 23. Rancangan API *endpoint* menghapus pengguna

Method		DELETE
Path		/api/users?nidn={nidn}
Description		Admin dapat menghapus pengguna
Parameter		nidn*
Request	***Header***	Authorization*: bearer {token}
	Body	-
Response	***Success***	status code: 200 OK message: "User deleted successfully"

Tabel 23 adalah rancangan API *endpoint* menghapus pengguna. Admin yang memiliki otoritas dapat mengirimkan permintaan hapus pengguna berdasarkan nidn. Sebelum dihapus, sistem akan terlebih dahulu meminta konfirmasi admin terkait permintaannya. Jika terkonfirmasi, *server* akan mengembalikan *response* sukses dan pengguna berhasil dihapus. Namun jika format permintaan tidak divalidasi, maka penghapusan akun dibatalkan.

Tabel 24. Rancangan API *endpoint* menampilkan daftar dan data fakultas

Method		GET
Path		/api/fakultas
Description		Admin dan dekan dapat melihat fakultas
Parameter		fakultas, prodi, nidn, limit, offset, sort, order, search
Request	***Header***	Authorization*: bearer {token}
	Body	-
Response	***Success***	status code: 200 Ok message: “Get All Study Programs Successful” data: [{ kode: char (3), nm_fakultas: text, singkatan: char varying (40), dekan: char varying (30), }]
	Failed	status code: 403 Forbidden message: “Access Denied: not your faculty”
		status code: 404 Not Found message: “No data found for the specified faculty”

Tabel 24 merupakan rancangan API *endpoint* dari fungsi menampilkan daftar program studi. Dekan memiliki otoritas untuk melihat data fakultasnya, di mana sistem akan meminta *request* pada *server* berdasarkan fakultasnya. Jika sukses, *server* akan mengembalikan *response* berisi daftar program studi yang berada di dalam lingkup fakultas tersebut.

Tabel 25. Rancangan API *endpoint* menampilkan daftar dan data program studi

Method		GET
Path		/api/programstudi
Description		Admin, dekan, dan kaprodi dapat melihat program studi
Parameter		fakultas, prodi, nidn, limit, offset, sort, order, search
Request	***Header***	Authorization*: bearer {token}
	Body	-
Response	***Success***	status code: 200 Ok message: "Get All Lecturers Successful" data: [{ kode: char (10), id_fak: char (3), nm_prodi: text, id_remun: char (5), kaprodi: char (30), }]
	Failed	status code: 403 Forbidden message: "Access Denied: not your study program"
		status code: 404 Not Found message: "No data found for the specified study program"

Tabel 25 mendeskripsikan rancangan API *endpoint* dari fungsi menampilkan daftar dan data program studi. Dekan memiliki otoritas untuk melihat daftar program studi di bawah naungan fakultasnya, sedangkan kaprodi memiliki otoritas untuk dapat melihat daftar dosen yang berada di dalam lingkup program studinya. Langkahnya, sistem akan meminta *request* ke *server* berdasarkan parameter, seperti kode fakultas atau kode program studi terkait. Apabila permintaan valid dan data tersedia, API akan mengembalikan *response* sukses beserta data yang berisi informasi dari program studi. Namun jika permintaan tidak valid, seperti kode tidak ditemukan atau tidak sesuai dengan otoritasnya, API akan mengembalikan *response* gagal.

Tabel 26. Rancangan API *endpoint* melihat daftar dan data dosen

Method		GET
Path		/api/dosen
Description		Seluruh pengguna dapat melihat dosen
Parameter		fakultas, prodi, nidn, limit, offset, sort, order, search
Request	***Header***	Authorization*: bearer {token}
	Body	-
Response	***Success***	status code: 200 Ok message: "Successful" data: [{ nidn: char (15), nip: char (25), nama: text, tempat_lahir: text, tanggal_lahir: date, bidang_ilmu: int, fakultas: char (3), prodi: char (10), email: text, id_remun:int, foto: char varying(32) }]
	Failed	status code: 403 Forbidden nessage: "Access Denied: Back to your profile"
		status code: 404 Not Found message: "No data found for the specified lecturer"

Tabel 26 mendeskripsikan rancangan API *endpoint* dari fungsi menampilkan daftar dan data dosen. Admin, kaprodi, dan dekan memiliki otoritas untuk dapat melihat data dosen berdasarkan nidn. Begitu juga dengan dosen terkait, ia dapat melihat detail datanya sendiri. Langkahnya, sistem meminta *request* melalui API terkait data dan informasi dosen berdasarkan parameter yang terlibat. Apabila *request* valid, *server* akan mengirimkan *response* dari data yang diminta. Namun jika permintaan tidak valid, seperti kode tidak ditemukan atau tidak sesuai dengan otoritasnya, API akan mengirimkan *response* gagal.

Tabel 27. Rancangan API *endpoint* melihat data kinerja dosen bidang pengabdian

Method		GET
Path		/api/pengabdian
Description		Seluruh pengguna dapat melihat kinerja dosen bidang pengabdian
Parameter		fakultas, prodi, nidn, tahun, semester, limit, offset, sort, order
Request	***Header***	Authorization*: bearer {token}
	Body	-
Response	***Success***	status code: 200 Ok message: "Successful" data: [{ kode: bigserial, tahun: char (4), semester: char (2), id_kegiatan: bigint, judul_kegiatan: text, id_jenis_tugas: int, tanggal_mulai tugas: date, tanggal_selesai_tugas: date, nama_bimbingan: text, jabatan_fungsional_bimbingan: text, id_perguruan_tinggi: uuid, deskripsi_kegiatan: text, metode_pelaksanaan: text, jumlah_jam: bigint, judul: text, quartile: int, id_jenis_publikasi: int, tanggal: date, peran: smallint, urutan: smallint, jml_anggota: smallint, tkt_publikasi: smallint, corresponding_auhor: smallint, sinta: smallint, tkt_publikasi_jurnal: smallint, nomor_paten: text pemberi_paten: text, jenis_paten: smallint, id_afiliasi: uuid, id_jenis_skim: uuid, tanggal_sk_penugasan: date,

Tabel 27. (lanjutan)

Response	***Success***	id_kategori_pembicara: smallint, id_media_publikasi: uuid, peran_jurnal: text, status_jurnal: smallint, tkt_pengelola_jurnal_int: smallint, tkt_pengelola_jurnal_nas: smallint }]
	Failed	status code: 404 Not Found message: "Data not found"

Tabel 27 merupakan rancangan API *endpoint* untuk fungsi melihat kinerja dosen bidang pengabdian. Pengguna yang memiliki otoritas, seperti admin, kaprodi, dan dekan, dapat melihat detail data kinerja dosen berdasarkan fakultas dan program studi. Begitu juga dengan dosen terkait, ia dapat melihat detail data kinerjanya pribadi. Dengan menggunakan *endpoint*, sistem akan mengambil data berdasarkan parameter tertentu dan mengirimkan *request* pada data yang diminta. Jika permintaan valid, data yang dikembalikan akan mencakup detail kegiatan dari bidang pengabdian kinerja dosen, seperti judul kegiatan, tanggal pelaksanaan, deskripsi, serta informasi lainnya yang terkait dengan indikator dari BKD. Namun, jika permintaan tidak valid atau data terkait tidak ada, API akan mengirimkan *response* gagal atau kosong.

Tabel 28. Rancangan API *endpoint* melihat data kinerja dosen bidang penunjang

Method		GET
Path		/api/penunjang
Description		Seluruh pengguna dapat melihat kinerja dosen bidang penunjang
Parameter		fakultas, prodi, nidn, tahun, semester, limit, offset, sort, order
Request	***Header***	Authorization*: bearer {token}
	Body	-

Tabel 28. (lanjutan)

Response	***Success***	status code: 200 Ok message: “Successful” data: [{ kode: bigserial, tahun: char (4), semester: char (2), id_kegiatan: bigint, judul_kegiatan: text, id_jenis_kepanitiaan: int, instansi: text, peran_panitia: text, tingkat_panitia: int, klasifikasi_panitia: int, klasifikasi_anggota_panitia: int, tanggal_mulai_keanggotaan: date, tanggal_selesai_keanggotaan: date, instansi_profesi: text, peran_anggota_profesi: smallint, peran_pertemuan_ilmiah: smallint, id_tingkat_penghargaan: int, id_jenis_penghargaan: int, instansi_pemberi: text, tahun_penghargaan: int, jenis_terbitan_buku_pelajaran: smallint, peringkat_penghargaan: smallint, jenis_tim_penilai: smallint, id_sk: bigint }]
	Failed	status code: 404 Not Found message: “Data not found”

Tabel 28 merupakan rancangan API *endpoint* untuk fungsi melihat data kinerja dosen bidang penunjang. Sama seperti bidang pengabdian, API akan mengirimkan *request* ke *server* pada data yang diminta, dalam hal ini bidang penunjang. Jika permintaan valid, data yang dikembalikan akan mencakup detail kegiatan dari bidang penunjang, seperti judul kegiatan dan informasi lainnya yang terkait dengan indikator dari BKD. Namun, jika permintaan tidak valid atau data terkait tidak ada, API akan mengirimkan *response* gagal atau kosong.

3. *Develop* (Pengembangan API)

Proses inti dari penelitian ini berada pada proses pengembangan API atau *develop*. Dalam proses ini, akan direalisasikan rancangan-rancangan yang telah dibuat sebelumnya. Proses ini akan meliputi 3 tahap, yaitu penyusunan spesifikasi API menggunakan Postman untuk mendokumentasikan *endpoint* secara terstruktur, kemudian pengimplementasian API menggunakan Node.js, serta penerapan JWT untuk memastikan keamanan dan aksesibilitas data.

a. Menyusun Spesifikasi API

Untuk menyusun spesifikasi API, pengembang akan memanfaatkan aplikasi Postman. Spesifikasi ini akan berdasar pada panduan spesifikasi OpenAPI versi 3.0 (OAS3) untuk mendefinisikan struktur *endpoint* beserta visualisasinya.

b. Mengembangkan RESTful API Menggunakan Node.js

Pengembangan API ini akan menggunakan *server-side technology* Node.js yang berperan sebagai *runtime environment* serta Express.js sebagai *framework*-nya. *Database* yang digunakan ialah PostgreSQL. Selain itu, akan ada beberapa *library* tambahan dalam *package* Node.js yang membantu menyederhanakan pengembangan sistem.

c. Mengimplementasikan JWT

Untuk mengimplementasikan autentikasi dan otorisasi pengguna, akan dimanfaatkan sebuah *library* dari Node.js yaitu JWT “jsonwebtoken” untuk membuat dan memverifikasi mekanisme tokenisasi.

4. *Test* (Pengujian API)

Proses pengujian berperan penting dalam memastikan apakah API yang dikembangkan sudah berjalan sesuai dengan yang diharapkan. Pengujian ini dilakukan untuk mengevaluasi kualitas, keandalan, serta stabilitas sistem. Dalam penelitian ini, dilakukan beberapa jenis pengujian, yaitu pengujian fungsional, pengujian performa, dan pengujian keamanan.

a. Pengujian Fungsional

Pengujian fungsional bertujuan untuk memvalidasi apakah API berhasil memberikan *response* sesuai spesifikasi dan ketentuan yang telah dirancang. Proses uji fungsionalitas ini akan memanfaatkan Postman sebagai alat manajemen pengujian. Tabel 29 merupakan skenario-skenario yang akan diujikan dalam pengujian fungsional.

Tabel 29. Skenario pengujian fungsional

Endpoint	***Scenario Id***	***Test Scenario***
/api/auth/login	TS.01	Uji autentikasi pada *endpoint log in* dari kredensial yang valid/invalid
/api/fakultas	TS.02	Uji *endpoint* GET Fakultas dari otorisasi peran (Admin, Dekan) dan parameter yang valid/invalid
/api/programstudi	TS.03	Uji *endpoint* GET Program Studi dari otorisasi peran (Admin, Dekan, Kaprodi) dan parameter yang valid/invalid
/api/dosen	TS.04	Uji *endpoint* GET Dosen dari otorisasi peran (Admin, Dekan, Kaprodi, Dosen) dan parameter yang valid/invalid
/api/pengabdian	TS.05	Uji *endpoint* GET Kinerja Bidang Pengabdian dari otorisasi peran (Admin, Dekan, Kaprodi, Dosen) dan parameter yang valid/invalid

Tabel 29. (lanjutan)

Endpoint	*Scenario Id*	*Test Scenario*
/api/penunjang	TS.06	Uji *endpoint* GET Kinerja Bidang Penunjang dari otorisasi peran (Admin, Dekan, Kaprodi, Dosen) dan parameter yang valid/invalid
/api/users	TS.07	Uji *endpoint* GET, POST, PUT DELETE Pengguna dari otorisasi peran (Admin), parameter, dan *body* yang valid/invalid
Akses autentikasi dan otorisasi seluruh *endpoint*	TS.08	Uji penanganan autentikasi dan otorisasi pada semua akses *endpoint* dari token yang invalid

b. Pengujian Performa

Pengujian performa bertujuan untuk mengevaluasi *response time* dan *throughput* guna mengukur kemampuan API dalam menghadapi beban kerja tertentu. Dalam pengujian ini, digunakan sebuah aplikasi pengujian Apache JMeter yang berfungsi untuk mengamati kinerja API ketika menerima permintaan dalam jumlah yang besar, serta memastikan apakah API mampu menangani lonjakan permintaan secara bersamaan dengan mempertahankan kestabilan sistem.

c. Pengujian Keamanan

Pengujian keamanan bertujuan untuk mengidentifikasi potensi kerentanan yang mungkin terdapat dalam sistem, khususnya API yang dikembangkan ini. Proses ini memanfaatkan sebuah aplikasi pengujian OWASP ZAP yang berfungsi untuk melakukan pemindaian otomatis terhadap berbagai celah keamanan guna memastikan apakah API memiliki perlindungan yang memadai dan mampu menjaga integritas serta kerahasiaan data yang diintegrasikan.

5. *Deploy* (Penerapan API)

Proses penerapan atau *deploy* API adalah tahap di mana API yang telah dikembangkan dan diuji dapat diterapkan ke lingkungan produksi agar dapat digunakan oleh pengguna atau aplikasi lain.

a. *Deploy* API

Deploy API sementara akan dilakukan masih di dalam lingkup lokal menggunakan *localhost*. Dalam skenario ini, API akan terhubung ke *database* lokal PostgreSQL dan berjalan di perangkat masing-masing pengembang.

b. Dokumentasi API

Dokumentasi API adalah panduan yang menjelaskan cara kerja, penggunaan, dan integrasi API untuk pengembang yang ingin menggunakannya. Dokumen ini akan mencakup secara lengkap terkait autentikasi, otorisasi hak akses, *endpoint*, metode HTTP, *parameter*, serta bentuk *request* dan *response*. Dengan menerbitkan dokumentasi API ini, pengembang lain akan dengan mudah memahami dan menggunakan API ini, sehingga dapat meningkatkan efisiensi pengembangan serta kolaborasi dalam tim.

V. SIMPULAN DAN SARAN

5.1 Simpulan

Simpulan dari penelitian ini ialah pengembangan RESTful API pada sistem remunerasi dosen Universitas Lampung bidang pengabdian dan penunjang telah berhasil direalisasikan. API ini dilengkapi dengan autentikasi dan otorisasi berbasis JWT untuk menjamin keamanan pertukaran data. Selain itu, seluruh skenario pengujian telah menghasilkan *response* yang sesuai harapan. Hal ini membuktikan bahwa API mampu menjalankan fungsinya secara tepat, sehingga dapat digunakan dan dikembangkan lebih lanjut untuk mendukung berbagai proses penilaian kinerja, seperti akreditasi, evaluasi, maupun pelaporan institusional lainnya.

5.2 Saran

Saran dari penelitian ini yang dapat dijadikan sebagai usul, masukan, atau pembaruan untuk penelitian selanjutnya ialah:

1. Mengoptimalkan pemanfaatan data pengguna dengan mengintegrasikan sistem pada sumber data terpusat milik universitas untuk memungkinkan pengguna dapat masuk dengan skenario SSO (*Single Sign-On*).
2. Menyempurnakan mekanisme penggantian *password* pengguna yang memungkinkan pengguna untuk melakukan reset secara mandiri saat lupa *password* tanpa harus melalui bantuan administrator.
3. Meningkatkan penggunaan sistem *cache* yang lebih skalabel dan andal untuk mendukung performa aplikasi pada skala penggunaan yang lebih luas dan lingkungan produksi yang kompleks.

DAFTAR PUSTAKA

Adam, S. I., Moedjahedy, J. H., & Maramis, J. (2020). RESTful Web Service Implementation on Unklab Information System Using JSON Web Token (JWT). *2nd International Conference on Cybernetics and Intelligent System, ICORIS 2020*. https://doi.org/10.1109/ICORIS50180.2020.9320801

Armour, F., & Miller, G. (2000). *Advanced Use Case Modeling: Software Systems* (Vol. 1). Pearson Education. www.awprofessional.com/otseries.

Bell, D. (2004). UML Basics: The Sequence Diagram. *IBM*. http://www.ibm.com/developerworks/rational/library/3101.html

Bertolino, A., Fantechi, A., Gnesi, S., Lami, G., & Maccari, A. (2002). Use Case Description of Requirements for Product Lines. *International Workshop on Requirements Engineering for Product Lines*, 12–18.

Börger, E., Cavarra, A., & Riccobene, E. (2000). An ASM Semantics for UML Activity Diagrams. Dalam *Algebraic Methodology and Software Technology* (hlm. 293–308). Springer. https://doi.org/10.1007/3-540-45499-3_22

Bucko, A., Vishi, K., Krasniqi, B., & Rexha, B. (2023). Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History. *Computers*, *12*(4). https://doi.org/10.3390/computers12040078

Budiman, K., Putra, A. T., Alamsyah, Sugiharti, E., Muslim, M. A., & Arifudin, R. (2021). Implementation of ERP system functionalities for data acquisition based on API at the study program of Universities. *Journal of Physics: Conference Series*, *1918*(4). https://doi.org/10.1088/1742-6596/1918/4/042151

Chandrachood, A. (2021). Api First Development: A Modern Approach to Building Integrated Software Systems. *International Journal of Science and Research (IJSR)*, *10*(10), 1627–1629. https://doi.org/10.21275/SR24608141605

Da Costa, L. (2021). *Testing JavaScript Applications*. Manning Publications.

Dekawati, I. (2022). *Manajemen Pendidikan (Teori dan Praktik)* (1 ed.). Indonesia Emas Publisher.

Doglio, F. (2015). Pro REST API Development with Node.js. Dalam *Pro REST API Development with Node.js*. Apress Media. https://doi.org/10.1007/978-1-4842-0917-2

Dudjak, M., & Martinović, G. (2020). An API-first methodology for designing a microservice-based backend as a service platform. *Information Technology and Control*, *49*(2), 206–223. https://doi.org/10.5755/j01.itc.49.2.23757

Ehsan, A., Abuhaliqa, M. A. M. E., Catal, C., & Mishra, D. (2022). RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions. Dalam *Applied Sciences (Switzerland)* (Vol. 12, Nomor 9). MDPI. https://doi.org/10.3390/app12094369

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* [Dissertation]. University of California.

Halili, Emily. (2008). *Apache JMeter*. Packt Publishing.

Hendayun, M., Ginanjar, A., & Ihsan, Y. (2023). Analysis of Application Performance Testing Using Load Testing and Stress Testing Methods in API Service. *Junal Sisfotek Global*, *13*(1), 28. https://doi.org/10.38101/sisfotek.v13i1.2656

Husein, A. M., Simanjuntak, A. J., Sinaga, C. J., Tampubolon, M. M., & Situmorang, P. N. C. (2024). SIAKAD Mobile With API Service To Improve Academic Services. *Sinkron*, *8*(2), 641–650. https://doi.org/10.33395/sinkron.v8i2.13519

Kaniya, I. A., Paramitha, P., Made Wiharta, D., Made, I., & Suyadnya, A. (2022). Perancangan dan Implementasi RESTful API pada Sistem Informasi Manajemen Dosen Universitas Udayana. *Jurnal SPEKTRUM*, *9*(3), 15–23.

Kaur, M., Lata, S., & Kaur, H. (2022). A Review on Software Testing & Techniques. *International Journal of Research Publication and Reviews*, *3*(8), 2185–2186.

Kommey, B., Gyimah, F., Kponyo, J. J., & Andam-Akorful, S. A. (2022). A Web Based System for Easy and Secured Managing Process of University Accreditation Information. *Indonesian Journal of Computing, Engineering and Design (IJoCED)*, *4*(1), 17. https://doi.org/10.35806/ijoced.v4i1.240

Kore, P. P., Lohar, M. J., Surve, M. T., & Jadhav, S. (2022). API Testing Using Postman Tool. *International Journal for Research in Applied Science and Engineering Technology*, *10*(12), 841–843. https://doi.org/10.22214/ijraset.2022.48030

Krosnick, R. (2024). Postman Flows: A Visual Programming Tool for Building API-Powered Apps and Workflows. *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, 356–358. https://doi.org/10.1109/VL/HCC60511.2024.00047

Larsson, J., Åkermark, L., & Golubovic, D. (2021). *The Value of Implementing API-first Methodology When Developing APIs*.

Leuhery, F., & Nahumury, H. (2023). *Meningkatkan Kinerja Dosen Melalui Remunerasi dan Motivasi Berprestasi*. Deepublish Digital.

Mahindrakar, P., & Pujeri, U. (2020). Insights of JSON Web Token. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(6), 1707–1710. https://doi.org/10.35940/ijrte.F7689.038620

Maniraj, S. P., Sabapathy Ranganathan, C., & Sekar, S. (2024). Securing Web Applications with OWASP ZAP for Comprehensive Security Testing. Dalam *Int. J.Adv.Sig.Img.Sci* (Vol. 10, Nomor 2).

Mardikaningsih, R., & Darmawan, D. (2022). Tinjauan Tentang Kualitas Kehidupan Kerja, Kompensasi, Komitmen Organisasi Dan Kontribusi Terhadap Kinerja Dosen. *Jurnal Pendidikan dan Konseling (JPDK)*, *4*(6), 6511–6521.

Matthew, N., & Stones, R. (2005). *Beginning Databases with PostgreSQL: From Novice to Professional* (2 ed.). Apress Media.

Mcgill, M. J., Purchase, H. C., Colpoys, L., Mcgill, M., Carrington, D., & Britton, C. (2001). *UML Class Diagram Syntax: An Empirical Study of Comprehension*. https://www.researchgate.net/publication/221536273

Mowla, S., & Kolekar, S. V. (2020). Development and integration of E-learning services using rest APIs. *International Journal of Emerging Technologies in Learning*, *15*(4), 53–72. https://doi.org/10.3991/ijet.v15i04.11687

Pedoman Operasional BKD, Pub. L. No. 12/E/KPT/2021, Keputusan Direktur Jenderal Pendidikan Tinggi Kementerian Pendidikan dan Kebudayaan (2021).

Peyrott, S. E. (2024). *The JWT Handbook*.

Postman. (2024a). *API Lifecycle*. Postman, Inc. https://www.postman.com/api-platform/api-lifecycle/

Postman. (2024b). *Guide to API-first*. Postman, Inc. https://www.postman.com/api-first/

Postman. (2024c). *What is Postman?* Postman, Inc. https://www.postman.com/product/what-is-postman/

Prayogi, A. A., Niswar, M., Indrabayu, & Rijal, M. (2020). Design and Implementation of REST API for Academic Information System. *IOP Conference Series: Materials Science and Engineering*, *875*(1). https://doi.org/10.1088/1757-899X/875/1/012047

Ramakrishnan, R., & Gehrke, J. (2002). *Database Management System* (2 ed.). McGraw-Hill Inc. https://doi.org/10.5555/560733

Rumbaugh, J., Jacobson, I., & Booch, G. (1999). The Unified Modeling Language Reference Manual. *4th International Conference*.

Septama, H. D., Komarudin, M., Wintoro, P. B., Pratama, M., Yulianti, T., & Sulistiono, W. E. (2022). Implementation of One Data-based Lecturer Profile Information System for Key Performance Indicator Monitoring. *7th International Conference on Informatics and Computing, ICIC 2022*. https://doi.org/10.1109/ICIC56845.2022.10006928

Shukla, A. (2023). Modern JavaScript Frameworks and JavaScript's Future as a FullStack Programming Language. *Journal of Artificial Intelligence & Cloud Computing*, 1–5. https://doi.org/10.47363/JAICC/2023(2)144

Singh, P., & Gupta, V. (2020). JWT BASED AUTHENTICATION. *Skylark International Publication*, *1*(4). www.researchhub.org.in/research-hub

Sommerville, Ian. (2011). *Software engineering*. Pearson.

Soper, R., Torres, N. N., & Almoailu, A. (2023). *ZED ATTACK PROXY COOKBOOK : hacking tactics, techniques, and procedures for testing web applications and APIs*. PACKT PUBLISHING LIMITED.

Tuya, J., Suárez-Cabal, M. J., & De La Riva, C. (2006). A practical guide to SQL white-box testing. *ACM SIGPLAN Notices*, *41*(4), 36–41. https://doi.org/10.1145/1147214.1147221

Tzavaras, A., Mainas, N., & Petrakis, E. G. M. (2023). OpenAPI framework for the Web of Things. *Internet of Things (Netherlands)*, *21*. https://doi.org/10.1016/j.iot.2022.100675

Undang-Undang Nomor 14 Pasal 72 tentang Guru dan Dosen, Pub. L. No. 14, Kementerian Hukum dan Hak Asasi Manusia (2005).

Universitas Lampung. (2024). *Pedoman Rubrik Remunerasi Universitas Lampung*.

Utama, J. S., & Indriyanti, A. D. (2023). Pengamanan Restful API Web Service Menggunakan Json Web Token (Studi Kasus: Aplikasi Siakadu Mobile Unesa). *JEISBI*, *04*.

Wiyati, & Pradana, A. G. B. (2022). *Manajemen Sumber Daya Manusia: Kajian Teoritis Pengelolaan Pegawai Sektor Swasta & Aparatur Sipil Negara* (1 ed.). Media Nusa Creative (MNC Publishing).