PENGEMBANGAN *FRONT-END* ANGULAR PADA SISTEM INFORMASI MANAJEMEN KEPEGAWAIAN UNIVERSITAS ISLAM INDONESIA

Disusun Oleh:

N a m a : Abhirama Aji Mahardhika
NIM : 21523168

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN *FRONT-END* ANGULAR PADA SISTEM INFORMASI MANAJEMEN KEPEGAWAIAN UNIVERSITAS ISLAM INDONESIA

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Abhirama Aji Mahardhika

NIM : 21523168

Yogyakarta, 15 Januari 2026

Pembimbing,

(Feri Wijayanto, S.T., M.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN *FRONT-END* ANGULAR PADA SISTEM INFORMASI MANAJEMEN KEPEGAWAIAN UNIVERSITAS ISLAM INDONESIA

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 15 Januari 2026

Tim Penguji

Feri Wijayanto, S.T., M.T.

Anggota 1

Andhik Budi Cahyono, S.T., M.T.

Anggota 2

Dr. Nur Wijayaning Rahayu, S.Kom., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Abhirama Aji Mahardhika
NIM	:	21523168

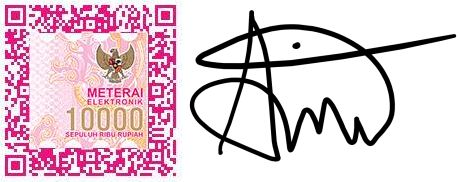
Tugas akhir dengan judul:

PENGEMBANGAN *FRONT-END* ANGULAR PADA SISTEM INFORMASI MANAJEMEN KEPEGAWAIAN UNIVERSITAS ISLAM INDONESIA

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apa pun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 15 Januari 2026



(Abhirama Aji Mahardhika)

HALAMAN PERSEMBAHAN

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat, karunia, dan hidayah-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir jalur magang ini dengan baik. Laporan akhir ini penulis persembahkan kepada Ayah tercinta, Hari Yuwono, yang senantiasa memberikan dukungan, doa, dan semangat dalam setiap tahap perjalanan pendidikan penulis, serta kepada almarhumah Ibu tercinta, Dwina Suratmi, yang semasa hidupnya telah memberikan kasih sayang, pendidikan, dan keteladanan yang menjadi sumber motivasi utama bagi penulis.

Persembahan ini juga penulis tunjukan kepada keluarga besar yang selalu memberikan doa dan dukungan, serta kepada diri penulis sendiri sebagai pengingat bahwa setiap proses membutuhkan kerja keras, kesabaran, dan ketekunan. Semoga ilmu dan pengalaman yang diperoleh selama pelaksanaan magang dan penyusunan laporan akhir ini dapat memberikan manfaat bagi penulis maupun bagi orang lain di masa mendatang.

HALAMAN MOTO

"Ilmu tanpa amal adalah kesia-siaan, dan amal tanpa ilmu adalah kebodohan."

(Imam Al-Ghazali)

"Jadilah anak muda yang produktif, sehingga menjadi pribadi yang profesional dengan tidak melupakan dua hal yaitu iman dan takwa."

(B.J. Habibie)

"I think it's possible for ordinary people to choose to be extraordinary."

(Elon Musk)

"The only way to do great work is to love what you do."

(Steve Jobs)

"It does not matter how slowly you go as long as you do not stop."

(Confucius)

"There is no greater weapon than a prepared mind."

(Zhuge Liang)

"Nothing is more intolerable than to have to admit to yourself your own errors."

(Ludwig van Beethoven)

"Hard work is worthless for those that don't believe in themselves."

(Uzumaki Naruto)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dengan menyebut nama Allah SWT Yang Maha Pengasih lagi Maha Penyayang. Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penyusunan laporan tugas akhir jalur magang yang berjudul "Pengembangan *Front-end* Angular pada Sistem Informasi Manajemen Kepegawaian Universitas Islam Indonesia". Laporan akhir ini disusun sebagai salah satu persyaratan kelulusan Program Studi Informatika – Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Pelaksanaan magang dan penyusunan laporan akhir ini bertujuan untuk memberikan pengalaman praktis kepada penulis dalam menerapkan ilmu pengetahuan yang diperoleh selama perkuliahan ke dalam lingkungan kerja profesional, khususnya pada bidang pengembangan *front-end* sistem informasi. Selama kegiatan magang di Badan Sistem Informasi (BSI) Universitas Islam Indonesia, penulis terlibat secara langsung dalam pengembangan antarmuka aplikasi kepegawaian pada portal UIIGateway yang berbasis *framework* Angular, serta berpartisipasi dalam proses pengembangan sistem yang menerapkan metode *Agile Scrum*.

Proses pelaksanaan magang dan penyusunan laporan akhir ini tentunya tidak terlepas dari berbagai kendala, baik dalam hal adaptasi terhadap lingkungan kerja, pemahaman terhadap struktur sistem yang kompleks, maupun penyesuaian dengan standar pengembangan perangkat lunak yang diterapkan di lingkungan BSI UII. Namun demikian, berkat bimbingan, arahan, serta dukungan dari berbagai pihak, seluruh proses tersebut dapat dilalui dengan baik.

Oleh karena itu, pada kesempatan ini penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT yang senantiasa memberikan rahmat, kesehatan, dan kemudahan kepada penulis selama proses pelaksanaan magang dan penyusunan laporan akhir ini.
2. Kedua orang tua penulis, Ayah Hari Yuwono yang senantiasa memberikan dukungan, doa, dan motivasi selama proses perkuliahan hingga penyusunan laporan akhir ini, serta almarhumah Ibu Dwina Suratmi yang semasa hidupnya telah memberikan kasih sayang, pendidikan, dan nilai-nilai ketekunan yang menjadi fondasi utama semangat penulis dalam menyelesaikan studi.
3. Universitas Islam Indonesia sebagai institusi pendidikan yang telah memberikan kesempatan kepada penulis untuk mengikuti program magang sebagai bagian dari tugas akhir.

4. Badan Sistem Informasi Universitas Islam Indonesia yang telah menerima penulis sebagai peserta magang dan memberikan pengalaman kerja yang berharga di bidang pengembangan sistem informasi.
5. Feri Wijayanto, S.T., M.T. selaku dosen pembimbing yang telah memberikan bimbingan, arahan, serta masukan yang sangat berarti dalam proses penyusunan laporan akhir ini.
6. Seluruh dosen Program Studi Informatika Universitas Islam Indonesia yang telah memberikan bekal ilmu pengetahuan dan wawasan selama penulis menempuh masa studi.
7. Seluruh tim dan staf di lingkungan BSI UII yang telah memberikan bimbingan teknis, dukungan, serta kesempatan bagi penulis untuk terlibat langsung dalam pengembangan sistem.
8. Rekan-rekan seperjuangan yang telah memberikan dukungan, semangat, serta pengalaman berharga selama masa perkuliahan dan penyusunan laporan akhir ini.
9. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang telah membantu dan mendukung penulis selama proses pelaksanaan magang dan penyusunan laporan akhir.

Penulis berharap laporan akhir ini dapat memberikan manfaat dan kontribusi, baik bagi pengembangan ilmu pengetahuan di bidang teknologi informasi, khususnya pengembangan *front-end*, maupun sebagai referensi bagi pihak-pihak yang berkepentingan. Penulis juga menyadari bahwa laporan akhir ini masih memiliki keterbatasan dan kekurangan. Oleh karena itu, kritik dan saran yang bersifat membangun sangat diharapkan demi penyempurnaan laporan ini di masa yang akan datang.

Yogyakarta, 15 Januari 2026

(Abhirama Aji Mahardhika)

SARI

Perkembangan teknologi informasi di lingkungan perguruan tinggi menuntut adanya sistem informasi manajemen sumber daya manusia yang handal dan efisien. Badan Sistem Informasi (BSI) Universitas Islam Indonesia (UII) menghadapi tantangan dalam pengelolaan data kepegawaian yang kompleks serta kebutuhan akan proses administrasi yang lebih otomatis. Sebagai solusi atas permasalahan tersebut, dikembangkanlah pengembangan *front-end* pada modul *Human Capital Management* (HCM) di portal UIIGateway. Sistem antarmuka ini dikembangkan menggunakan *framework* Angular yang menerapkan arsitektur berbasis komponen serta integrasi data melalui RESTful API.

Pengembangan sistem dilakukan dengan pendekatan metode *Agile Scrum* yang memungkinkan proses pengerjaan dilakukan secara iteratif dan responsif terhadap kebutuhan pengguna. Fitur-fitur utama yang berhasil dikembangkan meliputi antarmuka manajemen data pegawai yang konsisten, penerapan *saved state management*, komponen tabel penilaian dinamis, optimasi performa menggunakan teknik *lazy loading* dan sinkronisasi data *offset-based*, serta integrasi pembuatan dokumen digital otomatis menggunakan Jaspersoft Studio dan Docxtemplater. Laporan ini menyajikan proses pelaksanaan magang mulai dari pemahaman kebutuhan, perancangan komponen, hingga implementasi dan integrasi fitur. Hasil pengembangan ini diharapkan dapat meningkatkan efisiensi operasional layanan kepegawaian di lingkungan UII.

Kata kunci: *Angular, Front-end Development, Sistem Informasi Kepegawaian, Agile Scrum, Integrasi Dokumen Digital.*

GLOSARIUM

Agile Scrum	Kerangka kerja pengembangan perangkat lunak yang iteratif dan inkremental, fokus pada kolaborasi tim dan respons cepat terhadap perubahan.
Angular	*Framework* pengembangan aplikasi web berbasis TypeScript yang dikembangkan oleh Google, menggunakan konsep *Single Page Application* (SPA).
API	Antarmuka yang memungkinkan dua aplikasi perangkat lunak yang berbeda untuk saling berkomunikasi dan bertukar data.
Back-end	Bagian dari aplikasi web yang berjalan di sisi server, bertanggung jawab atas logika bisnis, pengolahan data, dan interaksi dengan basis data.
Docxtemplater	Pustaka untuk menghasilkan dokumen format .docx secara dinamis dengan mengisi *placeholder* dalam *template* menggunakan data JSON.
Boilerplate	*Template* standar sebuah proyek sesuai kriteria dari perusahaan tempat bekerja.
Front-end	Bagian dari aplikasi web yang berjalan di sisi klien (*browser*) dan berinteraksi langsung dengan pengguna.
Jaspersoft Studio	Perangkat lunak visual untuk merancang *template* laporan yang dapat digunakan untuk menghasilkan dokumen PDF atau format lain berdasarkan data sistem.
Lazy Loading	Pola desain optimasi di mana modul atau komponen aplikasi hanya dimuat ketika dibutuhkan, bukan sekaligus saat aplikasi pertama kali dibuka.
RESTful API	Gaya arsitektur layanan web yang menggunakan metode HTTP (GET, POST, PUT, DELETE) untuk komunikasi data antar sistem.
SPA	*Single Page Application* (SPA) adalah aplikasi web yang memuat satu halaman HTML tunggal dan memperbarui konten secara dinamis tanpa perlu memuat ulang keseluruhan halaman.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar belakang

Pesatnya perkembangan teknologi informasi saat ini telah mendorong organisasi untuk melakukan transformasi digital secara menyeluruh, terutama dalam optimalisasi manajemen modal manusia atau *Human Capital Management* (HCM) sebagai aset strategis organisasi. Di lingkungan perguruan tinggi, sistem informasi kepegawaian memegang peran vital dalam menjamin kelancaran operasional institusi serta mendukung pengambilan keputusan yang akurat. Badan Sistem Informasi Universitas Islam Indonesia (BSI UII) merespons kebutuhan ini melalui pengembangan UIIGateway, sebuah portal layanan terpadu yang mengintegrasikan berbagai kebutuhan akademik dan administratif. Dalam ekosistem ini, modul HCM menjadi komponen krusial yang mengelola data pegawai, portofolio, hingga dokumen kepegawaian, yang menuntut adanya antarmuka pengguna (*front-end*) yang modern dan modular untuk menangani kompleksitas data secara efisien.

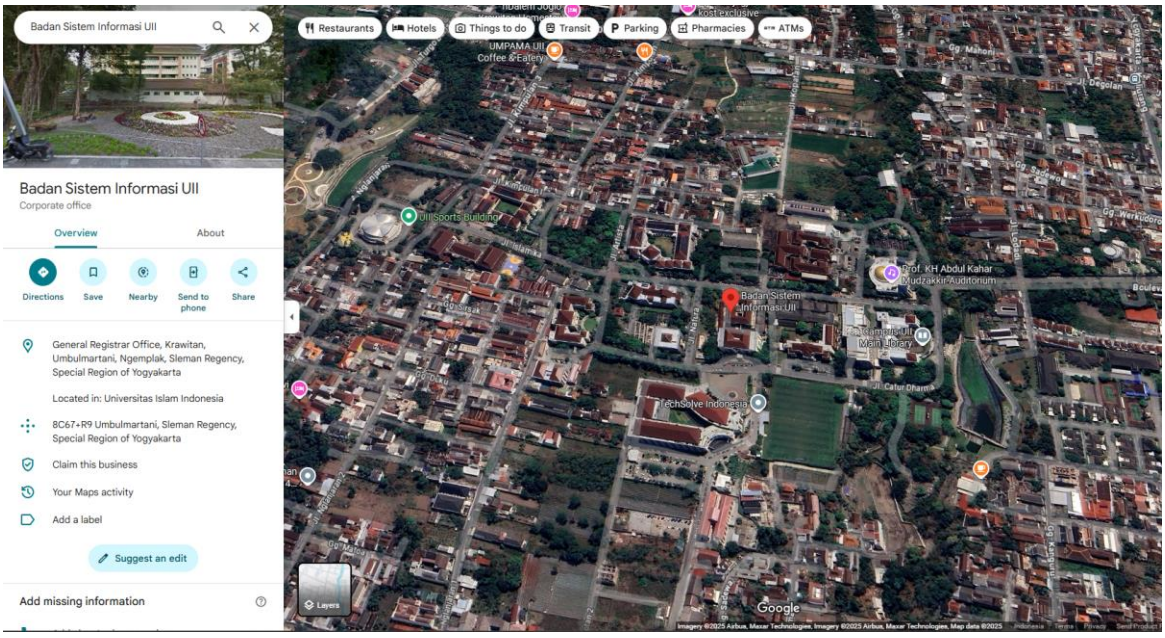
BSI UII merupakan unit kerja yang bertanggung jawab dalam perencanaan, pengembangan, serta pengelolaan sistem informasi terintegrasi di lingkungan Universitas Islam Indonesia. BSI UII berperan sebagai pengelola utama infrastruktur dan layanan teknologi informasi yang mendukung proses akademik, administrasi, serta operasional universitas. Dalam menjalankan perannya, BSI UII memastikan bahwa sistem informasi yang dikembangkan mampu berfungsi secara optimal, aman, dan berkelanjutan sesuai dengan kebutuhan institusi.

Secara fisik, BSI UII berkedudukan di Gedung H. GBPH Prabuningrat (Gedung Rektorat Lantai 4) Universitas Islam Indonesia yang berlokasi di Kampus Terpadu UII, Kabupaten Sleman, Daerah Istimewa Yogyakarta. Tampilan depan Gedung H. GBPH Prabuningrat sebagai lokasi kantor BSI UII ditunjukkan pada Gambar 1.1, sedangkan posisi geografis lokasi gedung berdasarkan peta digital ditampilkan pada Gambar 1.2. Lokasi BSI UII yang berada di pusat aktivitas manajerial universitas mendukung koordinasi lintas unit kerja serta memfasilitasi pengambilan keputusan yang terintegrasi.

Keberadaan BSI UII di lingkungan gedung rektorat juga memperkuat perannya sebagai penghubung antara kebutuhan pengguna di berbagai unit kerja dengan pengembangan sistem informasi universitas. Kondisi ini memungkinkan proses perencanaan dan pengembangan sistem dilakukan secara terkoordinasi, termasuk dalam pengelolaan dan pengembangan aplikasi

terintegrasi seperti UIIGateway. Dengan dukungan lokasi dan struktur organisasi yang terpusat, BSI UII mampu memastikan bahwa pengembangan dan pemeliharaan sistem informasi universitas berjalan selaras dengan visi strategis institusi.

Gambar 1.1 Tampak depan Gedung H. GBPH Prabuningrat



Gambar 1.2 Tangkapan layar pada Google Maps lokasi Gedung Rektorat

Dalam menjalankan tugas dan tanggung jawabnya, BSI UII berlandaskan pada visi untuk mendukung pencapaian tujuan strategis universitas melalui pemanfaatan sistem dan teknologi informasi yang efektif dan berkelanjutan. BSI UII berperan sebagai penyedia layanan teknologi informasi bagi seluruh *civitas academica*, termasuk mahasiswa, dosen, tenaga kependidikan, serta mitra eksternal, dengan menjamin ketersediaan layanan yang stabil, aman, dan andal. Bentuk pelayanan tersebut diwujudkan melalui pengelolaan infrastruktur jaringan kampus, pengembangan sistem akademik, serta aplikasi pendukung kegiatan administrasi dan proses pembelajaran.

Secara fungsional, BSI UII menjalankan tiga peran utama, yaitu melayani, mendampingi, dan mengakselerasi pengembangan teknologi informasi di lingkungan universitas. Peran melayani diwujudkan melalui penyediaan dan pemeliharaan layanan sistem informasi bagi unit-unit kerja di tingkat universitas, fakultas, hingga program studi. Peran mendampingi dijalankan dengan menjadi mitra diskusi dan pendukung teknis dalam pengembangan serta optimalisasi sistem dan teknologi informasi. Sementara itu, peran mengakselerasi berfokus pada upaya mendorong inovasi dan percepatan transformasi digital agar selaras dengan kebutuhan institusi dan perkembangan teknologi informasi (Badan Sistem Informasi Universitas Islam Indonesia, 2025).

UIIGateway dikembangkan sebagai gerbang utama (*single entry point*) bagi pengguna internal universitas untuk mengakses berbagai modul aplikasi sesuai dengan peran dan kewenangannya. Pada sisi kepegawaian, modul HCM di dalam UIIGateway mencakup pengelolaan data dosen, tenaga kependidikan, karyasiswa, serta berbagai proses administratif seperti evaluasi kinerja, perpanjangan kontrak, dan pembuatan dokumen kepegawaian. Kompleksitas proses bisnis tersebut menuntut dukungan antarmuka aplikasi yang konsisten, responsif, dan mampu beradaptasi terhadap perubahan kebutuhan organisasi.

Pada sisi teknis, pengembangan antarmuka UIIGateway memanfaatkan *framework* Angular sebagai teknologi utama *front-end*. Angular dipilih karena mendukung arsitektur berbasis komponen, pemisahan logika dan tampilan, serta integrasi yang terstruktur dengan layanan *back-end* melalui RESTful API. Meskipun *framework* Angular telah digunakan sebelumnya, seiring dengan bertambahnya modul dan kebutuhan sistem, masih ditemukan tantangan pada sisi *front-end*, seperti inkonsistensi tampilan antar modul, keterbatasan fleksibilitas komponen, serta penyesuaian mekanisme pemanggilan API akibat kebijakan keamanan sistem.

Selain itu, sistem informasi kepegawaian juga menuntut kemampuan dalam menghasilkan dokumen digital secara dinamis, seperti Surat Keputusan (SK) dan dokumen administratif lainnya. Proses ini memerlukan integrasi yang baik antara *front-end* dan *back-end*, baik melalui pemanfaatan *reporting tool* seperti Jaspersoft Studio maupun pustaka pengolahan dokumen seperti Docxtemplater. Perancangan *front-end* yang tepat menjadi faktor penting agar proses pemetaan data ke dalam dokumen dapat berjalan akurat dan efisien.

Berdasarkan kondisi tersebut, diperlukan pengembangan dan pengkajian lebih lanjut terhadap implementasi *front-end* berbasis Angular pada sistem informasi kepegawaian UIIGateway. Pengembangan ini tidak hanya berfokus pada aspek visual antarmuka, tetapi juga mencakup pengelolaan komponen, integrasi API, serta dukungan terhadap proses bisnis kepegawaian yang kompleks. Oleh karena itu, skripsi ini membahas pengembangan *front-end* Angular pada sistem informasi manajemen kepegawaian UII sebagai upaya untuk meningkatkan kualitas, konsistensi, dan efektivitas layanan kepegawaian.

1.2 Ruang Lingkup Magang

Kegiatan magang ini dilaksanakan di Badan Sistem Informasi Universitas Islam Indonesia (BSI UII) sebagai bagian dari pengembangan dan pemeliharaan sistem informasi terintegrasi universitas. Proyek yang menjadi fokus magang adalah pengembangan *front-end* pada portal UIIGateway, khususnya pada modul *Human Capital Management* (HCM). Pelaksanaan magang berlangsung selama kurang lebih sepuluh bulan, dimulai dari Januari hingga Oktober 2025, dan dilaksanakan secara langsung di lingkungan kerja BSI UII.

Proses pengembangan sistem dilakukan dengan pendekatan *Agile* menggunakan kerangka kerja *Scrum*. Pendekatan ini dipilih karena pengembangan *front-end* UIIGateway dilakukan secara bertahap dan berkelanjutan, menyesuaikan kebutuhan sistem yang telah berjalan serta masukan dari pengguna dan tim internal. Penerapan *Scrum* diwujudkan melalui pembagian pekerjaan dalam bentuk *backlog*, pelaksanaan iterasi pengembangan (sprint), diskusi rutin dengan tim pengembang, serta evaluasi berkala terhadap hasil implementasi antarmuka.

Tim pengembangan UIIGateway terdiri dari beberapa peran, antara lain *front-end developer*, *back-end developer*, UI/UX *designer*, serta *Quality Assurance* (QA). Dalam struktur tim tersebut, penulis berperan sebagai *front-end* developer yang bertanggung jawab pada pengembangan antarmuka pengguna berbasis *framework* Angular. Tugas utama meliputi perancangan dan implementasi komponen antarmuka, integrasi dengan RESTful API yang

disediakan oleh *back-end*, serta penyesuaian tampilan dan perilaku antarmuka agar konsisten dengan standar desain UIIGateway.

Ruang lingkup pengembangan *front-end* pada kegiatan magang ini mencakup modul kepegawaian yang terintegrasi dalam UIIGateway, yaitu modul UIIInsani, UIIKepegawaian, UIIPortofolio, dan UIIKaryasiswa. Pengembangan difokuskan pada peningkatan konsistensi tampilan antar modul, optimalisasi struktur komponen Angular, penerapan mekanisme *Role-Based Access Control* (RBAC) pada sisi antarmuka, serta integrasi fitur *generate* dokumen digital menggunakan Jaspersoft Studio dan Docxtemplater.

Laporan skripsi ini akan berfokus pada pembahasan pengembangan *front-end* Angular pada sistem informasi kepegawaian UIIGateway. Penulis tidak membahas secara mendalam pengembangan *back-end*, pengelolaan basis data, maupun infrastruktur sistem, karena aspek tersebut berada di luar tanggung jawab penulis selama pelaksanaan magang dan ditangani oleh tim terkait di BSI UII. Dengan pembatasan ruang lingkup ini, pembahasan diharapkan lebih terarah dan sesuai dengan kontribusi nyata penulis selama kegiatan magang berlangsung.

1.3 Tujuan

Tujuan utama dari pelaksanaan kegiatan magang dan penyusunan laporan ini adalah untuk memperoleh pengalaman praktis sekaligus menerapkan kompetensi di bidang pengembangan antarmuka pengguna melalui keterlibatan langsung dalam proyek pengembangan sistem informasi di lingkungan profesional. Secara lebih luas, kegiatan ini bertujuan untuk mengimplementasikan kerangka kerja pengembangan web modern guna membangun struktur antarmuka yang modular, terorganisir, dan memiliki fleksibilitas tinggi untuk dikembangkan di masa mendatang. Selain itu, upaya ini diarahkan untuk meningkatkan standar konsistensi visual serta kualitas pengalaman pengguna pada berbagai elemen sistem agar tercipta pola interaksi yang seragam, intuitif, dan mudah dipahami oleh seluruh pemangku kepentingan. Fokus pengembangan juga mencakup implementasi mekanisme integrasi antara sisi antarmuka dan layanan penyedia data secara efisien guna memastikan penyajian informasi berjalan secara akurat, responsif, serta mampu menjaga pemisahan tanggung jawab teknis yang jelas dalam sistem. Terakhir, pengembangan dilakukan untuk mendukung optimalisasi alur proses bisnis organisasi melalui penyediaan fitur-fitur pendukung antarmuka yang mampu meningkatkan efisiensi operasional serta mengurangi ketergantungan pada proses manual.

Selain itu, pengembangan ini bertujuan untuk menyelaraskan infrastruktur sistem informasi dengan arah transformasi digital organisasi secara menyeluruh guna mendukung

penyediaan layanan teknologi informasi yang stabil, aman, dan andal. Melalui penerapan standar keamanan yang lebih ketat pada sisi antarmuka, diharapkan sistem mampu memitigasi risiko kerentanan teknis sekaligus meningkatkan akuntabilitas proses pengelolaan data dalam lingkungan kerja profesional. Secara fungsional, integrasi berbagai fitur baru diarahkan untuk menciptakan ekosistem aplikasi yang kohesif, di mana setiap elemen sistem dapat berfungsi secara optimal dalam mendukung pencapaian tujuan strategis institusi melalui pemanfaatan teknologi informasi yang efektif dan berkelanjutan.

1.4 Manfaat

Pelaksanaan kegiatan pengembangan ini diharapkan dapat memberikan kontribusi positif, baik bagi penulis dalam aspek pengembangan diri maupun bagi instansi dalam aspek operasional sistem. Dari sisi pengembangan profesi, kegiatan ini bermanfaat untuk meningkatkan penguasaan teknis terhadap teknologi pengembangan *front-end* terkini, khususnya dalam penerapan arsitektur berbasis komponen, pengelolaan logika antarmuka yang kompleks, serta integrasi layanan data yang sesuai dengan standar industri. Bagi keberlanjutan sistem, penerapan standar desain yang seragam memberikan manfaat berupa peningkatan kenyamanan pengguna serta meminimalkan beban kognitif saat mengoperasikan berbagai fitur sistem yang berbeda. Kualitas integrasi data yang dihasilkan juga mendukung terciptanya sistem yang lebih andal, terstruktur, dan memiliki reliabilitas tinggi sehingga lebih mudah untuk diuji dan dipelihara secara berkelanjutan. Secara menyeluruh, hasil pengembangan fitur antarmuka ini memberikan manfaat strategis dalam mendukung efisiensi proses administratif organisasi melalui otomatisasi sistem, sehingga mampu mempercepat alur kerja serta meminimalkan potensi kesalahan administrasi yang dapat merugikan institusi.

Dari perspektif organisasional, pengembangan ini memberikan manfaat dalam memperkuat tata kelola sistem informasi yang transparan serta memfasilitasi pengambilan keputusan manajerial berbasis data yang lebih akurat melalui penyajian informasi yang terintegrasi secara sistemis. Keberadaan antarmuka yang modern dan responsif juga memberikan nilai tambah berupa peningkatan citra institusi melalui penyediaan layanan digital yang selaras dengan perkembangan teknologi informasi terkini. Lebih lanjut, standarisasi komponen yang diterapkan memberikan manfaat jangka panjang bagi tim pengembang dalam mempercepat siklus iterasi pengembangan fitur di masa mendatang, sehingga sistem tetap relevan dan adaptif dalam menghadapi perubahan kebutuhan operasional yang dinamis.

1.5 Sistematika Penulisan

Sistematika penulisan membahas inti dari setiap bab yang bertujuan untuk memudahkan pemahaman isi dan tujuan dari laporan akhir ini. Sistematika penulisan pada laporan akhir adalah sebagai berikut:

a. BAB I Pendahuluan

   Bab ini membahas mengenai latar belakang pelaksanaan magang, ruang lingkup pekerjaan yang dilakukan, tujuan yang ingin dicapai, manfaat dari kegiatan magang, serta sistematika penulisan laporan.

b. BAB II Landasan Teori dan Tinjauan Pustaka

   Bab ini membahas mengenai landasan teori yang relevan dengan topik magang, meliputi konsep sistem informasi kepegawaian, pengembangan *front-end* yang dilakukan dengan metode *Scrum* dan *Agile*. Teknologi yang digunakan untuk pengembangan adalah seperti *framework* Angular, RESTful API, dan alat bantu cetak dokumen digital seperti Jaspersoft Studio. Bagian akhir bab ini menyajikan tinjauan pustaka yang membandingkan penelitian ini dengan penelitian-penelitian terdahulu yang sejenis.

c. BAB III Pelaksanaan Magang

   Bab ini membahas mengenai pelaksanaan kegiatan magang secara menyeluruh, mulai dari gambaran umum instansi, lingkungan kerja, hingga proses teknis pengembangan sistem. Pembahasan mencakup perancangan arsitektur *front-end*, implementasi fitur-fitur utama pada modul HCM, serta integrasi dokumen digital.

d. BAB IV Refleksi Pelaksanaan Magang

   Bab ini membahas mengenai refleksi penulis terhadap proses pelaksanaan magang, kendala-kendala teknis maupun non-teknis yang dihadapi, strategi penyelesaian masalah, serta pembelajaran dan peningkatan kompetensi yang didapatkan selama terlibat dalam pengembangan UIIGateway.

e. BAB V Penutup

   Bab ini membahas mengenai kesimpulan dari seluruh rangkaian kegiatan magang dan pengembangan sistem yang telah dilakukan, serta saran-saran konstruktif untuk pengembangan sistem maupun pelaksanaan magang di masa mendatang.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 Sistem Informasi Kepegawaian

Sistem informasi kepegawaian merupakan bagian integral dari sistem informasi manajemen yang berfungsi untuk mengelola data sumber daya manusia secara terstruktur, terintegrasi, dan berbasis teknologi informasi. Menurut penelitian yang dilakukan oleh (Bahar dkk., 2023), sistem informasi kepegawaian tidak hanya berfungsi sebagai repositori data pegawai, tetapi juga sebagai alat strategis untuk mendukung pengambilan keputusan manajerial yang berbasis data. Sistem ini mencakup pengelolaan data pegawai yang komprehensif, mulai dari riwayat jabatan, pendidikan, penilaian kinerja, hingga administrasi kepegawaian lainnya yang mendukung operasional organisasi secara menyeluruh. Dalam konteks perguruan tinggi, implementasi sistem informasi kepegawaian menjadi semakin krusial mengingat kompleksitas data yang harus dikelola, termasuk data dosen, tenaga kependidikan, dan berbagai aspek administratif lainnya.

Pada Universitas Islam Indonesia (UII), sistem informasi kepegawaian dikembangkan dalam kerangka *Human Capital Management* (HCM) yang memandang pegawai sebagai aset strategis organisasi. Konsep HCM, sebagaimana dijelaskan oleh UKG (2024), merepresentasikan pendekatan holistik terhadap manajemen sumber daya manusia yang mengintegrasikan berbagai fungsi HR mulai dari rekrutmen, pengembangan, manajemen kinerja, hingga kompensasi dalam satu platform terpadu. (SAP, 2025) menambahkan bahwa HCM modern tidak hanya fokus pada aspek administratif, tetapi juga menggunakan analitik dan *workforce planning* untuk mendukung pengambilan keputusan strategis yang dapat meningkatkan produktivitas dan *engagement* karyawan. (Oracle, 2025) menekankan bahwa sistem HCM yang efektif harus mampu menyelaraskan strategi bisnis dengan manajemen talenta melalui integrasi yang *seamless* antara data kepegawaian, *payroll*, dan manajemen performa.

Dengan demikian, sistem kepegawaian di UII tidak hanya berfungsi sebagai penyimpan data, tetapi juga sebagai sarana integrasi layanan administratif, pelaporan, dan pendukung implementasi kebijakan institusi yang berbasis pada prinsip-prinsip *good governance*. Implementasi sistem informasi kepegawaian yang baik dapat meningkatkan efisiensi operasional, mengurangi redundansi data, dan memfasilitasi akses informasi yang cepat dan akurat bagi seluruh *stakeholder* institusi.

2.2 *Front-end Development*

Front-end development merupakan proses pengembangan antarmuka pengguna (*user interface*) yang berinteraksi langsung dengan pengguna akhir dalam sebuah aplikasi web. Fokus utama dari *front-end development* adalah menyajikan informasi secara visual, interaktif, dan responsif sehingga pengguna dapat mengakses serta memanfaatkan sistem secara efektif dan efisien. Menurut (Angela Gjorgievska, 2024), *front-end developer* tidak hanya bertanggung jawab mengimplementasikan desain visual, tetapi juga harus memahami prinsip-prinsip *UI/UX design* untuk menciptakan pengalaman pengguna yang optimal. (Anton Zubrytskyi, 2024) menambahkan bahwa *front-end development* memiliki dampak signifikan terhadap *user experience*, karena merupakan titik kontak pertama antara pengguna dengan sistem.

Front-end berperan sebagai penghubung antara pengguna dan sistem *back-end* melalui mekanisme komunikasi data yang umumnya menggunakan *Application Programming Interface* (API). (Asfak Ahmed, 2025) menekankan bahwa *front-end developer* yang memahami prinsip UI/UX dapat mengurangi *runtime error* hingga 40% dan menciptakan antarmuka yang lebih intuitif serta mudah digunakan. Dalam konteks pengembangan sistem informasi berskala besar, kualitas *front-end* sangat menentukan tingkat kegunaan (*usability*) dan pengalaman pengguna (*user experience*). Oleh karena itu, pengembangan *front-end* perlu memperhatikan beberapa aspek kritis seperti konsistensi tampilan, struktur navigasi yang jelas, dan performa aplikasi yang optimal, khususnya ketika sistem menangani data dalam jumlah besar seperti pada sistem informasi kepegawaian. *"Effective UI/UX design principles for front-end developers"* mencakup beberapa elemen penting. Menurut (Angela Gjorgievska, 2024) prinsip-prinsip ini meliputi:

- Meningkatkan navigasi dan aksesibilitas dengan mengimplementasikan arsitektur informasi yang logis dan mematuhi standar *Web Content Accessibility Guidelines* (WCAG).
- Memanfaatkan umpan balik dan interaksi mikro untuk meningkatkan *user engagement* melalui *visual cues* seperti *hover effects* dan *loading indicators*.
- Proses desain iteratif yang melibatkan umpan balik pengguna berkelanjutan dan *usability testing*.
- Memanfaatkan alat dan kerangka kerja modern seperti Figma untuk desain kolaboratif dan *frameworks* seperti React, Vue, atau Angular untuk membangun antarmuka responsif dan dinamis.

2.3 *Framework* Angular

Angular merupakan *framework front-end* berbasis TypeScript yang dikembangkan dan dipelihara oleh Google untuk membangun aplikasi web berskala besar dengan arsitektur yang kokoh dan mudah dipelihara. Sebagaimana dijelaskan dalam dokumentasi resmi Angular bahwa, Angular adalah sebuah *development platform* yang mencakup *component-based framework* untuk membangun aplikasi web yang *scalable*, dilengkapi dengan koleksi *library* yang terintegrasi dengan baik yang mencakup berbagai fitur seperti *routing*, *forms management*, *client-server communication*, dan lainnya. *Framework* ini mengadopsi arsitektur berbasis komponen yang memungkinkan pengembangan aplikasi secara modular, terstruktur, dan mudah dipelihara. Setiap komponen pada Angular terdiri dari logika (TypeScript class), tampilan (HTML *template*), dan gaya (CSS/SCSS) yang terpisah namun terhubung secara kohesif, sehingga mendukung pengelolaan kode yang lebih rapi dan terstruktur.

Hubungan Angular dengan TypeScript sangat fundamental dan tidak dapat dipisahkan. (Muzammil K, 2025) menjelaskan bahwa Angular menggunakan TypeScript sebagai bahasa pemrograman utama karena TypeScript menawarkan fitur-fitur penting untuk aplikasi perusahaan skala besar, seperti Keamanan tipe, antarmuka, generik, dan dukungan *refactoring* tingkat lanjut. TypeScript membantu menangkap *bugs* pada waktu kompilasi, meningkatkan produktivitas *developer* melalui *tooling* yang lebih baik (seperti *autocompletion* dan *navigation*), dan membuat *codebase* lebih *maintainable* dari waktu ke waktu. (Jacob, 2025) menambahkan bahwa penggunaan TypeScript dalam Angular dapat mengurangi *runtime errors* hingga 40% dan memberikan pengalaman pengembang yang lebih baik melalui fitur seperti *autocompletion* dan *navigation* yang lebih baik. Angular memanfaatkan TypeScript's *decorators* untuk mendefinisikan *components* (@Component), *modules* (@NgModule), *services* (@Injectable), dan *routing*, serta memanfaatkan *type system* untuk *dependency injection*, *generics* dalam RxJS, dan pemeriksaan tipe yang ketat di seluruh pohon komponen.

Keunggulan Angular yang menonjol antara lain adalah dukungan terhadap *two-way data binding* yang memfasilitasi sinkronisasi otomatis antara *model* dan *view*. Menurut (AngularMinds, 2024), *two-way data binding* dalam Angular memberikan cara untuk membagikan data antara *view* ke *source* dan sebaliknya secara bersamaan, sehingga ketika data berubah di *view* maka akan otomatis *update* di *source*, begitu pula sebaliknya. (GeeksforGeeks, 2025) menambahkan bahwa *two-way data binding* menggunakan *directive ngModel* yang merupakan kombinasi dari *property binding* ([]) dan *event binding* (()) dengan *syntax* yang

dikenal sebagai *"banana in a box"* [(ngModel)]. Fitur ini sangat berguna untuk *form handling* dan *interactive user interfaces*. Selain itu, Angular juga menyediakan *dependency injection* yang *powerful*, serta integrasi yang *seamless* dengan RESTful API untuk komunikasi *client-server*.

Framework ini juga menyediakan berbagai fitur optimasi seperti *lazy loading* dan *Ahead-of-Time* (AOT) *compilation* yang dapat meningkatkan performa aplikasi secara signifikan. Menurut penelitian yang dilakukan oleh jurnal (Setiawan & Fauzi, 2025), implementasi *lazy loading* pada Angular dapat meningkatkan skor performa dan mengurangi waktu pemuatan awal secara signifikan dibandingkan dengan *eager loading*. (Harikrishna Kundariya, 2024) menjelaskan bahwa *lazy loading* membantu dengan hanya melakukan pemuatan modul yang diperlukan untuk tahap pertama, membuat aplikasi menjadi *interactive* lebih cepat, yang sangat bermanfaat untuk aplikasi dengan pemuatan data awal yang besar. (Sukriti Srivastava, 2025) melaporkan bahwa dalam praktiknya, *enterprise app* yang mereka kerjakan mengalami penurunan waktu pemuatan awal dari 8.2 detik menjadi hanya 2.1 detik, dengan *main bundle* menyusut dari 4.3MB menjadi di bawah 1MB setelah implementasi *lazy loading* yang tepat.

Sementara itu, AOT *compilation* merupakan teknik yang mengompilasi Angular's *templates* dan *components* menjadi kode JavaScript yang siap digunakan di browser. selama proses *build*, yang menghasilkan waktu pemuatan awal yang lebih baik dan kinerja *rendering*. (Sukriti Srivastava, 2025) menjelaskan bahwa AOT *compilation* menghasilkan *bundle size* yang jauh lebih kecil dibandingkan dengan *Just-in-Time* (JIT) *compiler*, dan browser dapat langsung melakukan render aplikasi tanpa perlu melakukan kompilasi terlebih dahulu. (Abhishek Kesarwani, 2022) menekankan bahwa AOT *compiler* meningkatkan performa aplikasi secara dramatis dengan memanfaatkan *inline caching mechanism* dari JavaScript *Virtual Machines* (VM).

Oleh karena itu, Angular banyak digunakan dalam pengembangan sistem informasi enterprise, termasuk sistem informasi kepegawaian, karena kemampuannya dalam menangani persyaratan yang kompleks, menjaga kualitas kode, dan skala aplikasi seiring dengan pertumbuhan bisnis. Dalam konteks penelitian di Indonesia, (Kirana, 2023) telah membuktikan efektivitas penggunaan *directive* pada *framework* Angular untuk meningkatkan efisiensi pada sistem penerimaan mahasiswa baru di Universitas Islam Indonesia, di mana penggunaan *directive* memungkinkan pengembang untuk memanipulasi DOM dengan lebih efektif tanpa perlu menulis kode HTML yang repetitif.

2.4 Arsitektur Berbasis Komponen

Arsitektur berbasis komponen merupakan pendekatan pengembangan perangkat lunak yang membagi aplikasi ke dalam unit-unit kecil yang mandiri (*self-contained*), dan dapat digunakan kembali di berbagai bagian aplikasi. Setiap komponen memiliki tanggung jawab tertentu (*separation of concerns*) dan berinteraksi dengan komponen lain melalui antarmuka yang jelas dan terdefinisi dengan baik. Menurut (Das, 2024), arsitektur berbasis komponen mengatasi tantangan dalam pengembangan web dengan memperkenalkan konsep yang memecah UI menjadi bagian-bagian yang lebih kecil dan dapat dikelola, di mana setiap komponen merangkum logika, *state*, dan *presentation*, sehingga membuat kode lebih modular dan lebih mudah dikelola. Pendekatan ini tidak hanya meningkatkan *code reusability* tetapi juga membuat proses *testing*, *debugging*, dan *scaling applications* menjadi lebih mudah. (Das, 2024) menjelaskan beberapa prinsip fundamental dalam desain berbasis komponen:

- Dekomposisikan sistem menjadi unit-unit yang kohesif dan terbungkus.
- Setiap komponen memiliki antarmuka yang jelas dengan *port* yang disediakan dan dibutuhkan.
- Ketergantungan konkret di mana komponen bergantung pada kontrak, bukan implementasi.
- Dukungan untuk arsitektur ekstensi dan *plugin*.
- Komunikasi berbasis konektor melalui panggilan metode, pesan, peristiwa, atau aliran yang mendefinisikan bagaimana komponen berinteraksi.

(Maneshwar, 2025) menambahkan bahwa pendekatan ini menghasilkan sistem yang fleksibel, dapat diuji, dan lebih mudah untuk diskalakan karena perubahan pada satu komponen tidak akan mempengaruhi komponen lainnya selama kontrak antarmuka tetap konsisten.

Dalam Angular, arsitektur berbasis komponen memungkinkan pengembang untuk mengelola antarmuka kompleks dengan lebih terstruktur dan sistematik. Penerapan arsitektur ini sangat relevan pada UIIGateway yang terdiri dari banyak modul aplikasi, sehingga konsistensi tampilan dan perilaku antarmuka dapat dijaga meskipun modul dikembangkan secara terpisah oleh pengembang atau tim yang berbeda. (Nandaniya, 2019) menjelaskan bahwa arsitektur Angular berpusat pada *component* dan *service*, di mana setiap komponen merangkum *template*, gaya, dan logika, yang memungkinkan *developer* untuk membangun *Single Page Application* (SPA) yang kompleks dengan menyusun unit-unit yang lebih kecil dan mudah diuji dengan tanggung jawab yang jelas dan desain yang dapat digunakan kembali. (IT Consultis, 2024) menambahkan bahwa desain berbasis komponen berfungsi seperti blok bangunan atau

potongan LEGO untuk *website* atau aplikasi, di mana *developers* membuat blok bangunan yang mewakili berbagai bagian situs seperti *buttons*, *forms*, atau *navigation menus*, dan blok ini dapat digunakan kembali di seluruh *website*. Keuntungan utama dari arsitektur berbasis komponen meliputi:

- ***Reusability***, komponen dapat digunakan kembali di berbagai bagian aplikasi atau bahkan pada aplikasi berbeda, sehingga menghemat waktu dan tenaga.
- ***Maintainability***, komponen yang lebih kecil dan mandiri lebih mudah dirawat dan dilakukan *debug* karena sifatnya yang terisolasi.
- ***Scalability***, arsitektur berbasis komponen memudahkan mengubah skala aplikasi karena dapat menambahkan fitur baru tanpa memengaruhi fitur yang sudah ada.
- ***Consistency***, penggunaan komponen memastikan tampilan dan nuansa yang konsisten di seluruh aplikasi.
- ***Testability***, komponen individual dapat diuji secara terpisah, sehingga proses pengujian menjadi lebih efisien dan andal.

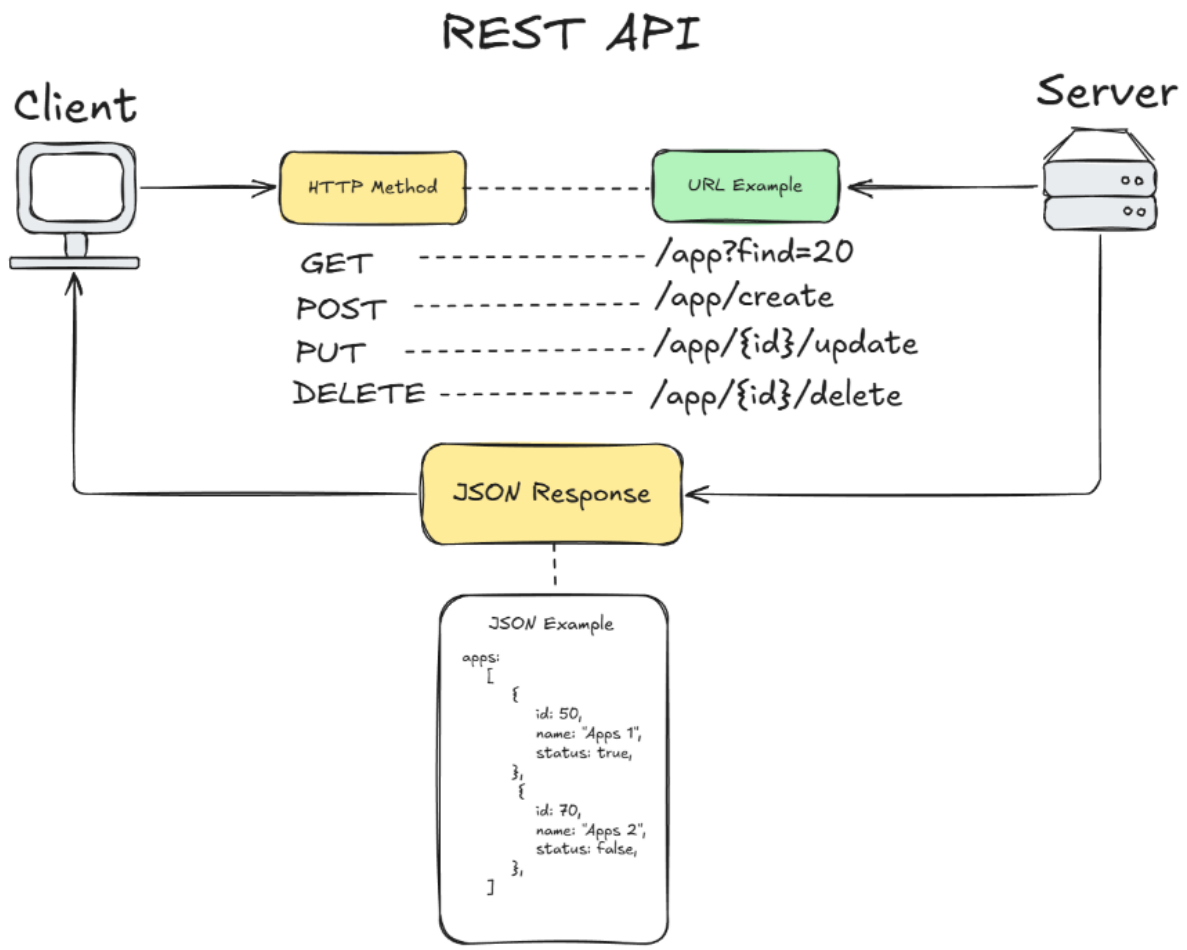
(Tecnovy, 2023) menambahkan bahwa software perlu berkembang, penambahan atau pengubahan komponen secara langsung tanpa perlu perombakan total, dan setiap komponen dapat diuji secara terpisah untuk memastikan setiap komponen bekerja dengan baik, dan mengarah kepada *software* yang *reliable*.

2.5 Mekanisme RESTful API

RESTful API merupakan gaya arsitektur layanan web yang menggunakan protokol *HTTP* sebagai media komunikasi antara *client* dan *server,* mengikuti prinsip-prinsip *Representational State Transfer* (REST). RESTful *Web Services* menyediakan prinsip-prinsip desain utama yang berfokus pada pengoptimalan kinerja dan skalabilitas dalam lingkungan akuisisi data seluler dan skala besar. RESTful API memanfaatkan metode HTTP seperti:

- **GET** (untuk mengambil data),
- **POST** (untuk mengirim data baru),
- **PUT** (untuk memperbarui data)
- **DELETE** (untuk menghapus data)

Metode tersebut digunakan untuk mengelola sumber daya (*resources*) yang direpresentasikan dalam bentuk format data seperti JSON. Contoh alur kerja RESTful API ditunjukkan pada Gambar 2.1



Gambar 2.1 Alur Kerja RESTful API

Penelitian yang dilakukan oleh (Neumann, Laranjeiro, & Bernardino, 2021) menganalisis 500 *public* REST *Web Service* APIs dan menemukan beberapa tren termasuk dukungan JSON yang luas dan dokumentasi yang dihasilkan perangkat lunak, namun pada saat yang sama juga keragaman layanan yang tinggi dengan perbedaan dalam kepatuhan terhadap praktik terbaik. Hasil penelitian menunjukkan bahwa hanya 0.8% dari *services* yang ketat mematuhi semua prinsip REST, namun 87.8% menggunakan REST-style URIs yang berorientasi pada sumber daya.

Pada UIIGateway, RESTful API digunakan sebagai media komunikasi antara *front-end* dan *back-end* dalam mengakses data kepegawaian. Penggunaan RESTful API memungkinkan pemisahan yang jelas (*clear separation of concerns*) antara logika bisnis pada server dan penyajian data pada sisi *front-end*, mengikuti prinsip *client-server architecture*. (Novianto & Munir, 2022) dalam penelitian mereka tentang implementasi RESTful API pada sistem informasi akademik perguruan tinggi menegaskan bahwa penggunaan RESTful API

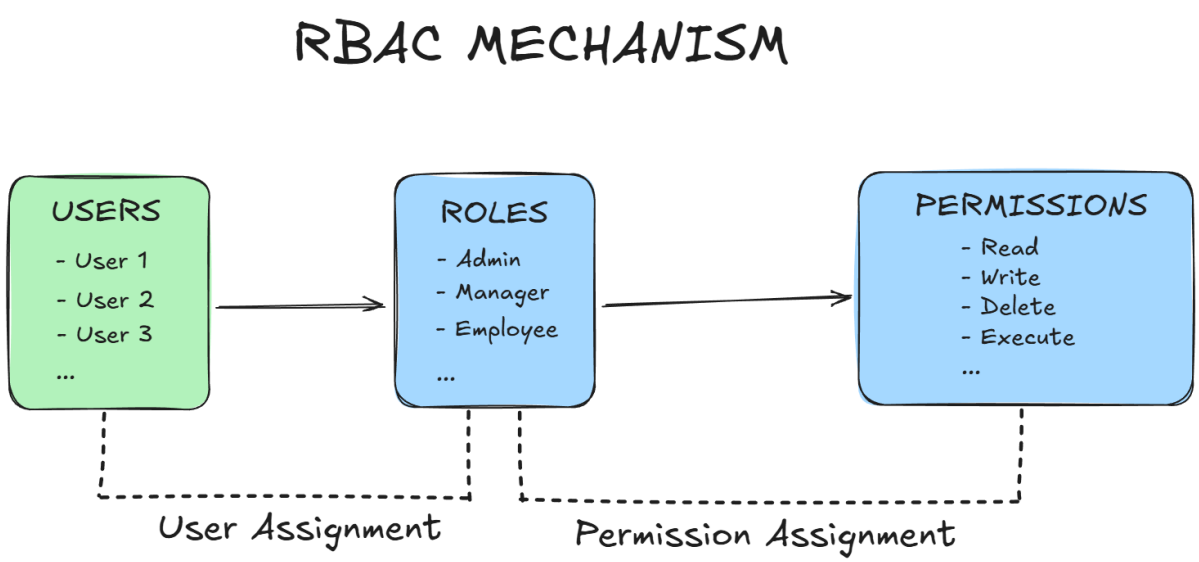
mendukung interoperabilitas antar modul sistem, meningkatkan fleksibilitas pengembangan, serta mempermudah integrasi antara *front-end* dan *back-end*. (Coblenz, Guo, Voozhian, & Foster, 2023) dalam *qualitative study* mereka tentang REST API *design* menemukan bahwa para partisipan akan mendapat manfaat dari alat yang memeriksa spesifikasi REST API terhadap kode *client* dan *server*, dan komunitas dapat mengembangkan pendekatan standar untuk menyampaikan pesan kesalahan yang lebih rinci.

Selain itu, RESTful API memudahkan integrasi antar sistem (*system integration*) serta mendukung pengembangan aplikasi yang bersifat modular dan terhubung secara longgar. Keuntungan lain dari RESTful API adalah *statelessness*, di mana setiap permintaan dari *client* ke *server* harus berisi semua informasi yang diperlukan untuk memahami dan memproses permintaan tersebut, tanpa server perlu menyimpan konteks klien antar permintaan. Hal ini membuat sistem lebih *scalable* dan lebih mudah dipelihara. RESTful API juga mendukung *caching*, yang dapat meningkatkan kinerja secara signifikan dengan mengurangi jumlah permintaan yang diproses oleh server.

2.6 Mekanisme *Role Based Access Control* (RBAC)

Role-Based Access Control (RBAC) merupakan mekanisme pengendalian akses sistem berdasarkan peran atau otoritas pengguna (Ghotbi & Fischer, 2012)**.** Dalam RBAC, setiap pengguna diberikan hak akses tertentu sesuai dengan perannya, sehingga hanya dapat mengakses fitur atau data yang sesuai dengan kewenangannya (Blundo & Cimato, 2012). Model RBAC telah menjadi standar dalam pengelolaan keamanan sistem informasi karena kemampuannya dalam menyederhanakan administrasi hak akses, meningkatkan fleksibilitas manajemen pengguna, serta memperkuat keamanan data melalui pemisahan tugas dan prinsip *least privilege* (Narasimman & Alsmadi, 2020)**.**

Konsep dasar RBAC melibatkan tiga elemen utama seperti, pengguna (*users*), peran (*roles*), dan hak akses (*permissions*) (Sabri, 2018). Pengguna diberikan satu atau lebih peran sesuai dengan posisi atau fungsi mereka dalam organisasi, sementara setiap peran memiliki seperangkat hak akses yang menentukan operasi apa saja yang boleh dilakukan terhadap sumber daya sistem (Xu & Gong, 2009). Dengan demikian, pengelolaan hak akses tidak dilakukan secara individual per pengguna, melainkan melalui peran yang dapat dikelola secara terpusat dan konsisten. Diagram konseptual cara kerja RBAC dapat dilihat pada Gambar 2.2, yang menggambarkan hubungan antara pengguna, peran, dan hak akses dalam sistem RBAC.



Gambar 2.2 Alur Kerja RBAC

Pada aplikasi berbasis web dengan banyak pengguna, model RBAC memberikan keleluasaan bagi pengembang untuk melakukan *customize* dan kontrol administratif terhadap elemen-elemen dengan tingkat granularitas yang berbeda, mulai dari akses ke modul aplikasi secara keseluruhan hingga akses ke elemen individual seperti kolom atau baris data tertentu (Ghotbi & Fischer, 2012). Pendekatan ini memisahkan logika otorisasi dari logika bisnis aplikasi, sehingga perubahan kebijakan akses dapat dilakukan tanpa harus mengubah kode aplikasi secara menyeluruh (Xu & Gong, 2009).

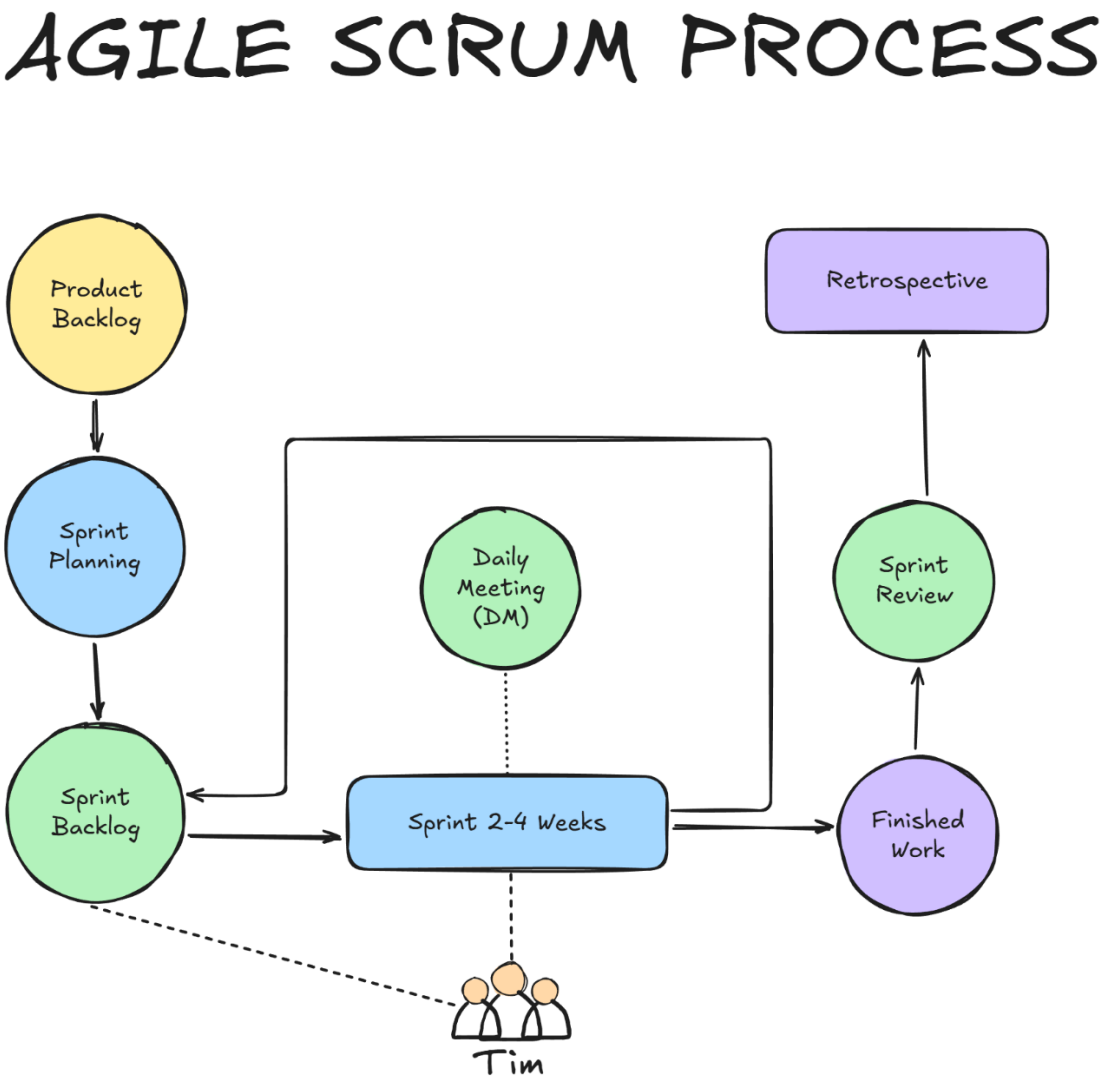
Pada UIIGateway, mekanisme RBAC diterapkan melalui pengelolaan kode menu (kd_menu) yang menentukan akses terhadap modul dan fitur tertentu. *Front-end* berperan dalam menyesuaikan tampilan antarmuka berdasarkan kd_menu yang dimiliki pengguna, sehingga satu modul kepegawaian dapat memiliki tampilan dan fungsi yang berbeda sesuai dengan peran pengguna tanpa mengubah logika *back-end*. Sebagai contoh, pengguna dengan peran admin super dapat mengakses seluruh data pegawai di semua unit kerja, sementara pengguna dengan peran admin fakultas hanya dapat mengakses data pegawai di fakultas yang menjadi kewenangannya. Mekanisme ini memastikan bahwa setiap pengguna hanya dapat melihat dan memanipulasi data sesuai dengan lingkup tanggung jawab dan otoritas yang telah ditetapkan oleh organisasi.

2.7 Metode *Agile* dan *Scrum*

Agile merupakan pendekatan pengembangan perangkat lunak yang menekankan fleksibilitas, kolaborasi, dan iterasi berkelanjutan (*continuous iteration*). Salah satu kerangka kerja *Agile* yang paling umum dan diadopsi secara luas digunakan adalah *Scrum*. (Peek, 2024) mendefinisikan metodologi *Agile Scrum* sebagai *project management system* yang mengandalkan pengembangan bertahap, di mana setiap iterasi terdiri dari *sprint* selama dua hingga empat minggu dengan tujuan setiap sprint adalah membangun fitur-fitur terpenting terlebih dahulu dan menghasilkan produk yang berpotensi siap dikirim. *Scrum* membagi proses pengembangan ke dalam siklus kerja pendek yang disebut sprint, di mana setiap sprint menghasilkan peningkatan fungsional (*functional increment*) pada sistem.

(Zhezherau, 2025) menjelaskan bahwa metodologi *Scrum* adalah *Agile framework* yang membantu tim dalam menangani proyek-proyek kompleks dengan bekerja dalam siklus pendek dan terfokus disebut sebagai *sprint*. *Framework* mendefinisikan peran yang jelas yaitu *Scrum Master, Product Owner, Developers*), menggunakan artefak bersama (*Product Backlog, Sprint Backlog, Increment*), dan mengandalkan acara rutin (*Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective*).

Penerapan *Scrum* memungkinkan tim pengembang untuk merespons perubahan kebutuhan secara cepat serta melakukan evaluasi berkala terhadap hasil pengembangan. Dalam konteks magang di BSI UII, metode *Scrum* digunakan untuk mengelola pengembangan *front-end* UIIGateway secara iteratif, dengan melibatkan diskusi rutin, perbaikan, serta kolaborasi lintas peran. Penelitian yang dilakukan oleh (Rahman dkk., 2025) menunjukkan bahwa pengembangan diorganisasikan ke dalam empat *sprint* iteratif, memungkinkan penyempurnaan berkelanjutan berdasarkan umpan balik pengguna. Alur kerja *agile* dan *scrum* dapat terlihat pada Gambar 2.3.



Gambar 2.3 Alur Kerja *Agile Scrum*

Framework Agile Scrum juga telah diterapkan dalam pengembangan *Human Resource Management System* (HRMS). Penelitian oleh (Fauzan & A B Dinariyana, 2024) menunjukkan bahwa penerapan metodologi *Design Thinking* memastikan solusi yang dikembangkan benar-benar berpusat pada pengguna, sementara metodologi *Agile Scrum* memungkinkan tim pengembang untuk bekerja secara iteratif dan adaptif, memastikan fleksibilitas dalam pengembangan dan kemampuan untuk merespons dengan cepat terhadap kebutuhan pengguna yang berubah dengan cepat. Hasil penelitian menunjukkan bahwa implementasi *Design Thinking* dan *Agile Scrum* dapat secara efektif mencapai transformasi digital dalam manajemen karyawan.

Dalam konteks efektivitas tim *Scrum*, penelitian oleh (Bülthoff & Maleshkova, 2014) melakukan analisis empiris menggunakan data dari perkiraan 5,000 *developers* dan 2,000

Scrum teams dengan survei yang dibuat khusus. Hasil penelitian menunjukkan kesesuaian yang sangat baik antara data empiris dan model teoretis (CFI = 0.959, RMSEA = 0.038, SRMR = 0.035), memungkinkan peneliti untuk:

- Mengusulkan dan memvalidasi teori yang dapat digeneralisasikan untuk tim *Scrum* yang efektif.
- Merumuskan rekomendasi yang jelas tentang bagaimana organisasi dapat lebih baik mendukung tim *Scrum*.
- Implementasi *Scrum* juga tidak lepas dari tantangan.

2.8 Tinjauan Pustaka

Pada subbab ini, penulis membahas mengenai beberapa hasil penelitian sebelumnya yang serupa atau mendekati dengan inti proyek yang menjadi topik laporan akhir ini supaya dapat dilihat persamaan dan perbedaannya. Terdapat sejumlah penelitian yang relevan dalam konteks pengembangan *front-end* sistem informasi dan integrasi dokumen, yang dapat menjadi acuan dalam pengembangan sistem ini.

Penelitian pertama dilakukan oleh Tony Wijaya pada tahun 2023 dengan judul *"Implementasi Framework Angular untuk Meningkatkan Performa Aplikasi Web Modern"*. Penelitian ini bertujuan untuk membuktikan bahwa penggunaan arsitektur *Single Page Application* (SPA) pada Angular dapat meningkatkan kecepatan akses pengguna. Teknologi yang digunakan meliputi Angular dan REST API dengan metode eksperimental. Hasilnya menunjukkan peningkatan signifikan pada *load time* dibandingkan aplikasi web konvensional (Wijaya, 2023).

Selanjutnya, penelitian oleh Heru Wijayanto Aripradono dan Kisusyenni Venessa pada tahun 2023 yang berjudul *"Pengembangan Front-End Website Sistem Keuangan UMKM di SMK Multistudi Highschool Batam Menggunakan Agile Scrum"*. Penelitian ini bertujuan menerapkan metode *Scrum* dalam pengembangan antarmuka sistem keuangan untuk UMKM. Menggunakan *library* berbasis komponen dan metode *Agile*, penelitian ini menyimpulkan bahwa pendekatan *Scrum* sangat efektif dalam mengakomodasi perubahan desain antarmuka yang dinamis selama proses iterasi pengembangan, serta meningkatkan kolaborasi antara tim pengembang dan pemilik produk (Aripradono & Venessa, 2023).

Penelitian ketiga oleh Alwan Alfian Setiawan dan Esa Fauzi pada tahun 2025 yang berjudul *"Analisis Komparatif Performa Implementasi Lazy Loading dan Code Splitting pada Framework React, Vue, dan Angular"*. Penelitian ini bertujuan mengukur efektivitas teknik

Lazy Loading dalam mengoptimalkan waktu muat awal aplikasi. Hasil penelitian menunjukkan bahwa meskipun Angular memiliki kompleksitas yang lebih tinggi dibanding *framework* lain, penerapan *lazy loading* yang tepat terbukti mampu menjaga stabilitas performa aplikasi berskala besar dengan memecah *bundle* aplikasi menjadi bagian-bagian yang lebih kecil (Setiawan & Fauzi, 2025).

Terakhir, penelitian oleh Amalia Nur Hasanah dan Hanson Prihantoro Putro pada tahun 2021 berjudul *"Implementasi JasperReports pada Sistem Informasi Manajemen"*. Penelitian ini berfokus pada integrasi fitur pelaporan otomatis dalam sistem informasi. Menggunakan teknologi Java dan JasperReports, penelitian ini berhasil mengembangkan modul yang memungkinkan sistem menghasilkan dokumen laporan dalam format PDF secara otomatis berdasarkan data yang tersimpan di basis data, sehingga mengurangi ketergantungan pada proses pembuatan laporan manual (Hasanah & Putro, 2021). Berikut adalah tabel perbandingan antara penelitian terdahulu dengan penelitian yang dilakukan penulis, sebagaimana disajikan pada Tabel 2.1

Tabel 2.1 Perbandingan Penelitian

No.	Peneliti (Tahun)	Judul Penelitian	Perbandingan dengan Penelitian Penulis
1.	Tony Wijaya (2023)	*"Implementasi Framework Angular untuk Meningkatkan Performa Aplikasi Web Modern"*	**Persamaan:** Sama-sama menggunakan *framework* Angular untuk pengembangan aplikasi web berbasis SPA. **Perbedaan:** Tony Wijaya berfokus pada eksperimen teknis performa umum, sedangkan penulis berfokus pada implementasi fitur spesifik modul kepegawaian (HCM) dan integrasi dokumen.

2.	Heru Wijayanto Aripradono, dan Kisusyenni Venessa (2023)	*"Pengembangan Front-End Website Sistem Keuangan UMKM di SMK Multistudi Highschool Batam Menggunakan Agile Scrum"*	**Persamaan:** Menggunakan metode pengembangan *Agile Scrum* untuk menangani kebutuhan antarmuka yang dinamis. **Perbedaan:** Penelitian tersebut berfokus pada sistem keuangan UMKM dengan teknologi umum, sedangkan penulis menerapkan Scrum pada sistem kepegawaian universitas yang kompleks dan terintegrasi.
3.	Alwan Alfian Setiawan, dan Esa Fauzi (2025)	*"Analisis Komparatif Performa Implementasi Lazy Loading dan Code Splitting pada Framework React, Vue, dan Angular"*	**Persamaan:** Membahas optimasi performa menggunakan teknik *Lazy Loading* pada Angular. **Perbedaan:** Alwan Alfian Setiawan, dan Esa Fauzi melakukan analisis komparatif antar *framework*, sedangkan penulis mengimplementasikan *lazy loading* yang dikombinasikan dengan sinkronisasi data parsial (*offset*) untuk kebutuhan riil aplikasi.
4.	Amalia Nur Hasanah, dan Hanson Prihantoro Putro (2021)	*"Implementasi JasperReports pada Sistem Informasi Manajemen"*	**Persamaan:** Mengembangkan fitur otomatisasi laporan/dokumen menggunakan JasperReports. **Perbedaan:** Hasanah, dan Hanson berfokus pada *generate* laporan dari sisi *back-end* Java murni, sedangkan penulis mengintegrasikan *preview* dan kontrol dokumen dari sisi *front-end* serta menambahkan Docxtemplater.

BAB III
PELAKSANAAN MAGANG

3.1 Gambaran Umum Pelaksanaan Magang

Pelaksanaan magang ini dilakukan di Badan Sistem Informasi (BSI) Universitas Islam Indonesia (UII) sebagai bagian dari pemenuhan tugas akhir jalur magang pada Program Studi Informatika Program Sarjana. Kegiatan magang berlangsung dalam rentang waktu Januari hingga Oktober, dengan penempatan pada tim pengembangan sistem informasi yang bertanggung jawab terhadap pengelolaan dan pengembangan portal UIIGateway.

UIIGateway merupakan portal layanan terpadu yang berfungsi sebagai pintu utama akses berbagai sistem akademik dan administratif di lingkungan UII. Dalam implementasinya, UIIGateway menggunakan pendekatan arsitektur berbasis *wrapper*, di mana setiap modul aplikasi berjalan secara terpisah namun disatukan dalam satu antarmuka utama. Kondisi ini menuntut pengembangan antarmuka *front-end* yang konsisten, modular, serta mampu beradaptasi dengan kebutuhan masing-masing modul.

Selama pelaksanaan magang, penulis berperan sebagai *Front-End Junior Staff* yang terlibat langsung dalam pengembangan dan pemeliharaan antarmuka pengguna berbasis Angular. Aktivitas utama yang dilakukan mencakup perancangan struktur komponen, implementasi fitur antarmuka, integrasi dengan RESTful API, serta optimalisasi performa dan pengalaman pengguna (*user experience*). Selain itu, penulis juga terlibat dalam proses diskusi teknis bersama tim *back-end* dan *UI/UX designer* guna memastikan kesesuaian antara kebutuhan bisnis, desain, dan implementasi teknis.

Kegiatan magang dilaksanakan dengan pola kerja profesional yang menyerupai lingkungan kerja industri, termasuk pembagian tugas berbasis *backlog*, diskusi rutin, serta evaluasi hasil pengembangan secara berkala. Setiap fitur yang dikembangkan tidak hanya diuji dari sisi fungsionalitas, tetapi juga ditinjau dari aspek konsistensi tampilan, performa, dan kemudahan penggunaan.

Hasil dari pelaksanaan magang ini berupa kontribusi nyata dalam pengembangan *front-end* modul-modul sistem kepegawaian pada UIIGateway, seperti UIIInsani, UIIKepegawaian, UIIPortofolio, dan UIIKaryasiswa. Kontribusi tersebut mencakup peningkatan kualitas antarmuka, pembuatan tabel khusus penilaian yang *reusable*, serta integrasi fitur pendukung seperti animasi pemuatan dinamis dan generasi dokumen digital. Dengan demikian,

pelaksanaan magang ini tidak hanya memberikan pengalaman praktis bagi penulis, tetapi juga memberikan nilai tambah terhadap pengembangan sistem informasi kepegawaian di lingkungan Universitas Islam Indonesia.

3.2 Lingkungan Kerja dan Tim Pengembangan

Pelaksanaan kegiatan magang dilaksanakan di lingkungan kerja Badan Sistem Informasi (BSI) Universitas Islam Indonesia, yang berperan sebagai unit pengelola dan pengembang sistem informasi terintegrasi di lingkungan universitas. Lingkungan kerja BSI UII bersifat profesional, kolaboratif, serta berorientasi pada pengembangan sistem informasi berskala institusi yang melibatkan banyak pemangku kepentingan, mulai dari *civitas academica* hingga unit administratif universitas.

Dalam pengembangan UIIGateway, khususnya pada modul-modul *Human Capital Management* (HCM), BSI UII menerapkan pola kerja berbasis tim lintas peran (*cross-functional team*). Setiap tim pengembangan terdiri dari beberapa peran utama yang saling berkolaborasi untuk memastikan sistem dapat dikembangkan secara terstruktur, berkelanjutan, dan sesuai dengan kebutuhan pengguna akhir.

3.2.1 Struktur Tim Pengembangan

Pelaksanaan magang di Badan Sistem Informasi Universitas Islam Indonesia (BSI UII) dilakukan dalam struktur organisasi proyek yang terdefinisi dengan jelas untuk memastikan kelancaran koordinasi dan pencapaian target pengembangan sistem. Struktur pengelolaan proyek pada pengembangan UIIGateway tidak menggunakan peran *Project Coordinator*, melainkan dikelola melalui peran *Project Manager* (PM) pada masing-masing tim pengembangan.

Setiap tim pengembangan aplikasi di BSI UII memiliki seorang Project Manager (PM) yang bertanggung jawab dalam perencanaan *sprint*, pembagian tugas, pemantauan progres pengembangan, serta koordinasi antar anggota tim. Anggota tim pengembangan umumnya terdiri dari *front-end developer* dan *back-end developer*, serta didukung oleh peran *UI/UX Designer*. *UI/UX Designer* tidak hanya bertugas dalam perancangan antarmuka dan pengalaman pengguna, tetapi juga melakukan pengujian pada lingkungan *staging*, menyusun *user manual*, serta menjalankan fungsi pengujian fungsional yang secara teknis setara dengan peran *Quality Assurance* (QA). Pada saat pelaksanaan magang, BSI UII belum memiliki tim QA khusus, sehingga aktivitas pengujian aplikasi dilakukan oleh *UI/UX Designer* sebagai

bagian dari proses validasi sistem. Selain itu, pada beberapa tim pengembangan terdapat peran *Software Engineer* yang merangkap tanggung jawab *front-end* dan *back-end*, namun pada tim pengembangan modul *Human Capital Management* (HCM) tidak terdapat peran tersebut, sehingga tanggung jawab pengembangan *front-end* dan *back-end* dijalankan oleh peran yang *terpisah*.

Seluruh PM berada di bawah pengawasan *Project Management Officer* (PMO), yang bertugas memastikan keselarasan pelaksanaan proyek dengan standar manajemen proyek yang diterapkan di lingkungan BSI UII. PMO berperan dalam mengoordinasikan jadwal lintas proyek, memantau capaian setiap tim, serta menjaga konsistensi penerapan metodologi pengembangan yang digunakan, khususnya kerangka kerja *Agile Scrum*.

Dalam pelaksanaan magang, penulis ditempatkan sebagai *Junior Staff Front-End Developer* pada tim pengembangan modul HCM di lingkungan UIIGateway. Penulis bekerja di bawah arahan langsung PM tim terkait serta berkolaborasi dengan *front-end developer* lain, *back-end developer*, dan *UI/UX Designer*. Kolaborasi ini memungkinkan penulis terlibat secara langsung dalam proses pengembangan sistem yang terstruktur, mulai dari perencanaan sprint, implementasi fitur antarmuka, integrasi API, hingga proses evaluasi dan penyesuaian antarmuka berdasarkan hasil pengujian pada lingkungan *staging*.

Struktur organisasi berbasis PM dan pengawasan PMO ini mendukung pelaksanaan pengembangan sistem yang terkontrol, terukur, dan selaras dengan kebutuhan institusi. Selain itu, model ini juga memberikan pengalaman kerja yang realistis bagi mahasiswa magang karena mencerminkan praktik manajemen proyek yang umum diterapkan pada pengembangan sistem informasi berskala besar.

3.2.2 Lingkungan dan Alat Pendukung Kerja

Lingkungan kerja pengembangan di Badan Sistem Informasi (BSI) Universitas Islam Indonesia didukung oleh berbagai alat dan platform kolaboratif yang digunakan untuk menunjang proses pengembangan sistem informasi secara terstruktur dan terintegrasi. Penggunaan alat-alat ini bertujuan untuk meningkatkan efektivitas koordinasi tim, menjaga kualitas kode, serta memastikan dokumentasi teknis dapat diakses dengan baik oleh seluruh anggota tim pengembang.

Dalam pengelolaan kode sumber, tim pengembang UIIGateway menggunakan GitLab sebagai *Version Control System* (VCS) utama. GitLab dimanfaatkan untuk mengelola repositori kode, mencatat riwayat perubahan, serta mendukung kolaborasi antar pengembang melalui

mekanisme *branch*, *merge request*, dan *code review*. Dengan penggunaan GitLab, proses pengembangan *front-end* dapat dilakukan secara paralel dan terkontrol, sehingga meminimalkan risiko konflik kode pada saat integrasi.

Untuk dokumentasi teknis, khususnya dokumentasi RESTful API yang dikembangkan oleh *back-end*, tim menggunakan platform Confluence. Dokumentasi ini mencakup deskripsi *endpoint* API, parameter yang digunakan, format respons data, serta contoh pemanggilan API. Keberadaan dokumentasi API di Confluence memudahkan *Front-End Developer* dalam melakukan integrasi API secara konsisten dan sesuai dengan kontrak layanan yang telah ditetapkan.

Pada aspek perancangan antarmuka pengguna, *UI/UX Designer* menggunakan Figma sebagai alat utama dalam menyusun desain antarmuka, *wireframe*, serta prototipe interaktif. Desain yang dihasilkan pada Figma menjadi acuan bagi *Front-End Developer* dalam mengimplementasikan tampilan antarmuka UIIGateway, sehingga konsistensi visual dan pengalaman pengguna antar modul dapat terjaga.

Manajemen proyek dan pengelolaan tugas pengembangan dilakukan menggunakan Jira dengan pendekatan sprint. Jira digunakan untuk mencatat *backlog* pengembangan, mendistribusikan tugas kepada anggota tim, serta memantau progres pekerjaan pada setiap siklus sprint. Dengan penggunaan Jira, proses pengembangan dapat dipantau secara transparan dan terukur, serta mendukung penerapan metode *Agile Scrum* secara efektif.

Selain alat-alat yang digunakan secara resmi oleh tim pengembang, penulis juga menggunakan PageSpeed Insights sebagai inisiatif pribadi untuk melakukan pengujian kualitas halaman *login* UIIGateway yang merupakan bagian dari *wrapper* sistem. Pengujian ini dilakukan untuk memperoleh gambaran awal terkait performa, aksesibilitas, dan kepatuhan terhadap praktik pengembangan web modern. Hasil pengujian tersebut dimanfaatkan sebagai bahan evaluasi tambahan dalam penelitian ini dan tidak merepresentasikan kebijakan pengujian resmi yang diterapkan oleh institusi.

Dengan dukungan lingkungan kerja yang terintegrasi serta pemanfaatan berbagai alat bantu pengembangan dan kolaborasi tersebut, proses pengembangan *front-end* UIIGateway dapat berjalan secara sistematis, terdokumentasi, dan selaras dengan kebutuhan organisasi.

3.2.3 Peran dan Kontribusi Penulis dalam Tim

Dalam struktur tim pengembangan UIIGateway, penulis berperan sebagai *Front-End Junior Staff* yang terlibat langsung dalam pengembangan dan penyempurnaan antarmuka

modul-modul kepegawaian. Kontribusi utama penulis meliputi implementasi komponen antarmuka berbasis Angular, penyesuaian tampilan antar modul, integrasi API, serta penerapan mekanisme *Role-Based Access Control (RBAC)* pada sisi *front-end*.

Selain itu, penulis juga terlibat dalam proses diskusi teknis, pengujian fitur, serta perbaikan *bug* berdasarkan masukan dari QA dan pengguna. Keterlibatan ini memberikan pengalaman praktis dalam bekerja pada sistem informasi berskala institusi serta meningkatkan pemahaman terhadap alur kerja pengembangan perangkat lunak profesional.

3.3 Arsitektur dan Struktur *Front-end* UIIGateway

Pengembangan *front-end* UIIGateway dirancang dengan arsitektur modular dan terstruktur untuk mendukung kebutuhan sistem informasi kepegawaian yang kompleks serta terus berkembang. UIIGateway berfungsi sebagai *wrapper* aplikasi utama, yang mencakup halaman *login* sebagai titik awal autentikasi pengguna sekaligus membungkus seluruh modul aplikasi di lingkungan Universitas Islam Indonesia. *Framework* Angular dipilih sebagai fondasi pengembangan *front-end* karena mendukung arsitektur berbasis komponen yang memudahkan pemisahan tanggung jawab antara tampilan, logika aplikasi, dan pengelolaan data.

Sebagai wrapper utama, UIIGateway bertanggung jawab dalam mengelola proses autentikasi pengguna, pemuatan menu berdasarkan hak akses, serta navigasi awal sebelum pengguna mengakses modul-modul aplikasi, termasuk modul kepegawaian. Setelah proses *login* berhasil, UIIGateway melakukan pemetaan hak akses pengguna dalam bentuk kode menu yang digunakan sebagai acuan untuk menentukan modul, fitur, dan komponen antarmuka yang dapat diakses oleh pengguna.

Modul-modul kepegawaian dikembangkan sebagai bagian dari *front-end* UIIGateway dan dimuat secara dinamis di dalam konteks *wrapper* tersebut. Setiap modul memiliki struktur komponen Angular tersendiri yang tetap mengikuti standar desain dan arsitektur UIIGateway, sehingga meskipun modul dikembangkan secara terpisah, antarmuka yang ditampilkan kepada pengguna tetap konsisten dan terintegrasi.

Arsitektur *front-end* UIIGateway secara umum dibagi ke dalam tiga lapisan utama, yaitu lapisan tampilan (*presentation layer*), lapisan logika aplikasi (*application logic layer*), dan lapisan layanan data (*data service layer*). Lapisan tampilan direpresentasikan oleh komponen Angular yang bertugas menampilkan antarmuka pengguna, menangani interaksi pengguna, serta mengelola *state* lokal komponen. Lapisan logika aplikasi diwujudkan melalui *service*

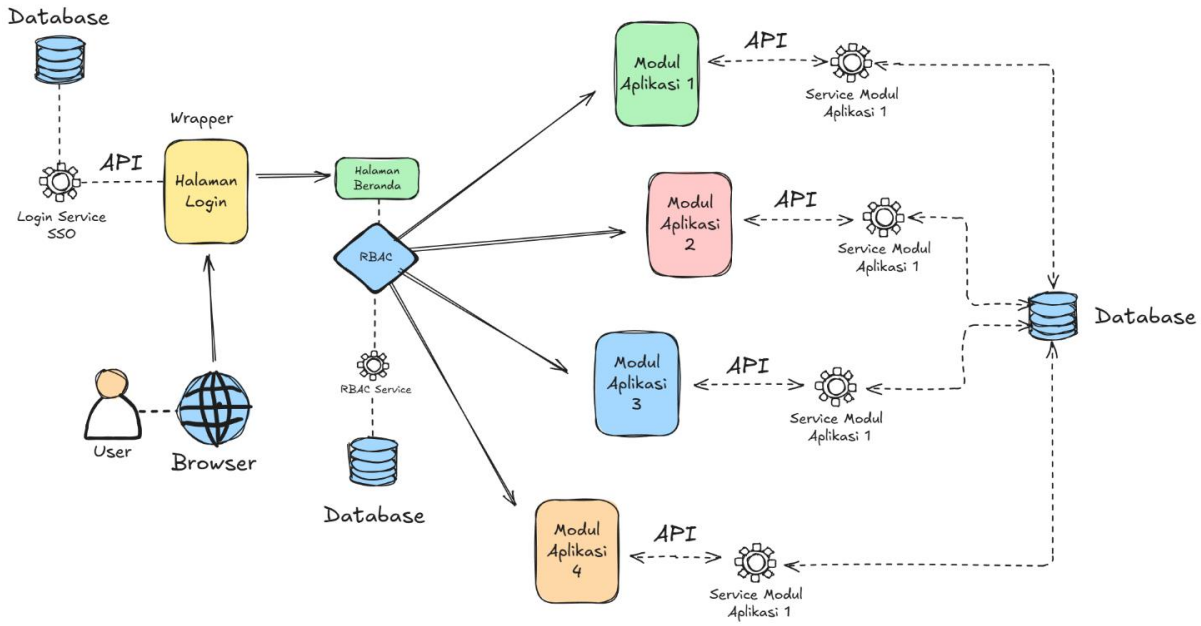
Angular yang menangani proses pengolahan data, pengaturan alur aplikasi, serta komunikasi antar komponen.

Lapisan layanan data berfungsi sebagai penghubung antara *front-end* UIIGateway dan *back-end* melalui RESTful API. Seluruh pemanggilan API dikonsolidasikan dalam *services* khusus yang disesuaikan dengan kebutuhan masing-masing modul. Pendekatan ini memudahkan pengelolaan perubahan kontrak API serta meningkatkan keterpisahan (*separation of concerns*) antara antarmuka dan sumber data.

Selain itu, UIIGateway menerapkan mekanisme *lazy loading* pada level modul Angular untuk meningkatkan performa aplikasi. Modul kepegawaian hanya akan dimuat ketika diakses oleh pengguna sesuai dengan hak aksesnya, sehingga dapat mengurangi beban pemuatan awal halaman *login* dan halaman beranda UIIGateway. Strategi ini sangat relevan mengingat UIIGateway berfungsi sebagai pintu masuk berbagai layanan digital dengan kompleksitas fitur yang tinggi.

Alur kerja pemuatan antarmuka *front-end* UIIGateway dimulai dari proses *login* pengguna, dilanjutkan dengan validasi autentikasi dan pengambilan data hak akses, kemudian dilanjutkan dengan pembentukan struktur menu, *routing*, serta pemuatan modul dan komponen antarmuka yang sesuai. Dengan pendekatan ini, UIIGateway mampu menyajikan tampilan dan fungsionalitas yang berbeda sesuai dengan peran pengguna, tanpa memerlukan perubahan pada logika *back-end*.

Ilustrasi arsitektur UIIGateway sebagai *wrapper* aplikasi beserta alur pemuatan modul ditunjukkan pada Gambar 3.1, yang memperlihatkan hubungan antara halaman *login*, modul-modul aplikasi, *service front-end*, serta *back-end* API. Untuk Gambar 3.2 menunjukkan struktur proyek Angular dari *boilerplate* (*template*) BSI UII.



Gambar 3.1 Arsitektur UIIGateway sebagai *Wrapper* Modul Aplikasi

```
.angular
deploy ---------------------------------------------- kumpulan konfigurasi deployment library
e2e
node_modules
projects
  └ uiigateway ------------------------------------- Wrapper UIIGateway
    └ {nama_aplikasi} ------------------------------ Nama modul aplikasi
      └ src
        └ lib
          └ modules ------------------------- Kumpulan komponen modul aplikasi
          └ services ------------------------ Kumpulan URL, dan method API yang akan dipakai
          └ shared -------------------------- Kumpulan komponen modul aplikasi yang dapat dipakai berulang kali
          └ {nama_aplikasi}.module.ts ------- Kumpulan definisi modul yang digunakan
          └ {nama_aplikasi}.routing.ts ------ Kumpulan tiap routes yang dipakai tiap komponen
        └ package.json ---------------------- Informasi nama modul aplikasi, versi modul aplikasi, author, dan contributor.
src ------------------------------------------------- Source core aplikasi seperti login page, navbar global, profil akun, dll.
CHANGELOG.md ---------------------------------------- List pembaruan atau perubahan berdasarkan versi modul aplikasi
README.md ------------------------------------------- File readme untuk repo di GitLab
package.json ---------------------------------------- Untuk nama proyek, script running proyek, dan kumpulan dependencies.
.gitlab-ci.yml -------------------------------------- Konfigurasi pipelines di GitLab
...
```

Gambar 3.2 Struktur Proyek Angular dari Boilerplate BSI

3.4 Implementasi dan Pengembangan Fitur *Front-end* pada Modul Kepegawaian

Pengembangan *front-end* pada UIIGateway dalam konteks modul kepegawaian dilakukan secara modular, mengikuti arsitektur *wrapper* yang telah ditetapkan. Setiap modul aplikasi

dikembangkan sebagai entitas terpisah namun terintegrasi dalam satu ekosistem UIIGateway. Pendekatan ini memungkinkan tim pengembang untuk bekerja secara paralel pada modul yang berbeda tanpa menimbulkan konflik kode, sekaligus memastikan konsistensi tampilan dan pengalaman pengguna di seluruh sistem.

Selama masa magang, penulis terlibat dalam pengembangan empat modul aplikasi utama dalam lingkup kepegawaian, yaitu UIIInsani, UIIPortofolio, UIIKepegawaian, dan UIIKaryasiswa. Masing-masing modul memiliki fungsi spesifik dalam mendukung proses administrasi kepegawaian, mulai dari pengelolaan data personal, portofolio pegawai, proses evaluasi kinerja, hingga manajemen program karyasiswa dosen. Pengembangan pada keempat modul tersebut mencakup beberapa aspek utama, yaitu:

a. Peningkatan konsistensi dan kualitas antarmuka pengguna yang akan diterapkan pada seluruh modul untuk memastikan keseragaman visual dan interaksi.
b. Pergantian cara pemanggilan API dari *path param* menjadi *query param* yang dilakukan pada seluruh modul sebagai peningkatan keamanan sistem.
c. Pengembangan komponen tabel penilaian dinamis yang *reusable* sebagai fitur khusus yang dikembangkan pada modul UIIKepegawaian.
d. Animasi pemuatan dinamis berbasis data offset sebagai fitur khusus yang dikembangkan pada modul UIIKaryasiswa.

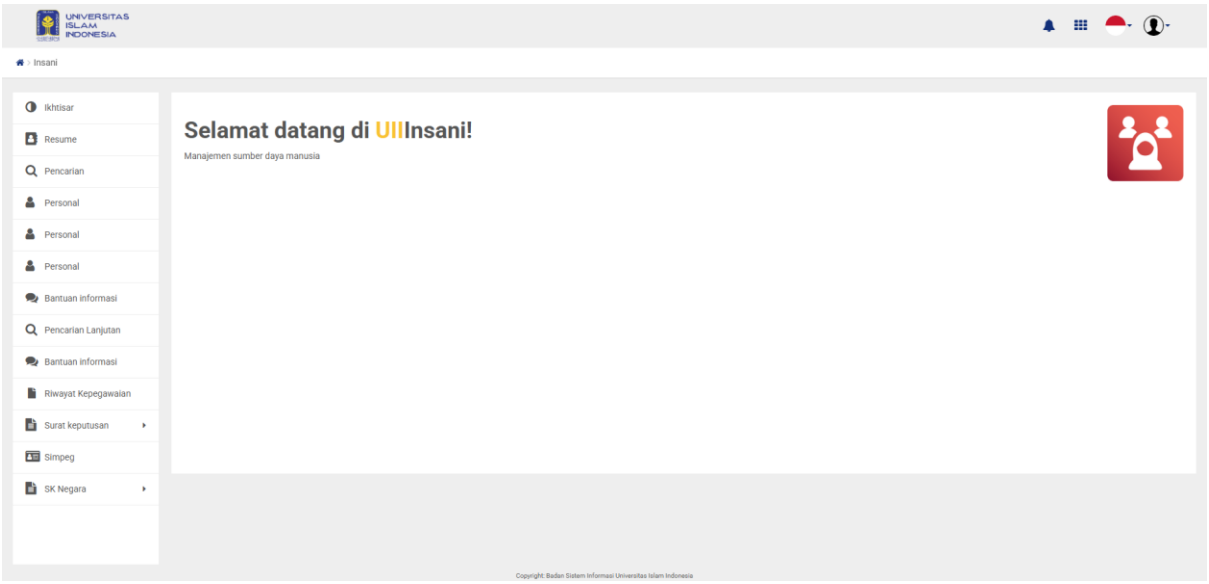
Pembahasan pada subbab berikut akan menjabarkan implementasi teknis pada masing-masing modul aplikasi secara rinci, mencakup konteks pengembangan, tantangan yang dihadapi, solusi yang diterapkan, serta hasil yang dicapai.

3.4.1 Penerapan *Saved State Management* pada Aplikasi Modul UIIInsani

Modul UIIInsani merupakan sistem informasi yang berfungsi sebagai pusat pengelolaan data personal pegawai di lingkungan Universitas Islam Indonesia. Modul ini mencakup pengelolaan identitas pegawai, data keluarga, riwayat pendidikan, riwayat jabatan, serta data administratif lainnya yang menjadi fondasi bagi proses kepegawaian universitas. Sebagai modul yang menyediakan data dasar kepegawaian, UIIInsani menjadi sumber referensi utama bagi modul lain seperti UIIKepegawaian, UIIPortofolio, dan UIIKaryasiswa.

Tampilan halaman depan modul UIIInsani dapat dilihat pada Gambar 3.3, yang menampilkan antarmuka pengelolaan data personal pegawai dengan navigasi yang intuitif dan komponen-komponen yang terstruktur dengan baik. Modul ini dirancang untuk memberikan

pengalaman pengguna yang mudah dipahami, dengan pemisahan yang jelas antara berbagai kategori data yang harus diisi oleh pegawai.



Gambar 3.3 Tampilan Halaman Awal Modul Aplikasi UIIInsani

Pada awal keterlibatan penulis dalam pengembangan modul ini, UIIInsani telah memiliki struktur dasar dan fungsionalitas inti. Namun, terdapat kebutuhan untuk meningkatkan konsistensi tampilan antarmuka serta memperkuat mekanisme keamanan dalam pemanggilan API. Oleh karena itu, pengembangan yang dilakukan penulis difokuskan pada dua aspek utama yaitu peningkatan kualitas antarmuka pengguna dan penyesuaian cara pemanggilan API untuk mendukung standar keamanan yang lebih ketat.

Peningkatan Konsistensi dan Kualitas Antarmuka Pengguna

Salah satu tantangan utama dalam pengembangan modul UIIInsani adalah kebutuhan untuk mengimplementasikan komponen *tabset* yang tidak hanya memiliki *styling* konsisten dengan design system UIIGateway, tetapi juga harus mendukung fitur *persistent tab state*. Fitur ini memiliki fungsi untuk menyimpan tab yang sedang aktif saat pengguna melakukan *refresh* halaman atau keluar masuk modul.

Meskipun UIIGateway telah menyediakan *library* komponen berbasis Angular bernama PilarNg dengan komponen *tabset* yang sudah memiliki *styling* sesuai dengan pedoman desain UIIGateway, komponen tersebut memiliki keterbatasan fungsional. Ketika pengguna melakukan refresh halaman atau menavigasi keluar dan masuk kembali ke modul, komponen

tabset akan selalu kembali ke tab pertama (*default state*), mengabaikan pilihan tab yang sebelumnya dipilih pengguna. Hal ini mengakibatkan pengalaman pengguna yang kurang memuaskan, terutama untuk pegawai yang memiliki data *form* yang panjang dan ingin melanjutkan *input* data pada tab yang sama setelah melakukan *refresh*.

Untuk mengatasi keterbatasan ini, penulis mengimplementasikan *state management* untuk *tabset* menggunakan URL *query parameter*. Pendekatan ini memanfaatkan *library* ngx-bootstrap tabset yang terintegrasi dengan Angular router untuk menyimpan pilihan tab yang aktif dalam URL *query parameter* activePage. Dengan cara ini, ketika pengguna melakukan *refresh*, Angular router akan secara otomatis membaca nilai activePage dari URL dan memulihkan tab yang sebelumnya dipilih.

Implementasi *Tabset Custom* dengan *Saved State Management*

Komponen *tabset custom* dikembangkan dengan mempertahankan *styling* yang mirip dengan PilarNg, sehingga tetap konsisten dengan desain sistem UIIGateway, namun dengan menambahkan logika untuk menyimpan dan memulihkan *state tab*. Komponen ini terdiri dari beberapa bagian utama, pertama adalah struktur kode HTML untuk komponen *tabset custom* seperti ditunjukkan pada Gambar 3.4.

```
<ng-container>
    <tabset #staticTabs [justified]="true">
        <ng-container>
            <tab [active]="activePage === 1" (selectTab)="tabSelect(1)">
                <ng-template tabHeading>
                    // Konten tabset 1 disini
                </ng-template>
            </tab>
            <tab [active]="activePage === 2" (selectTab)="tabSelect(2)">
                <ng-template tabHeading>
                    // Konten tabset 2 disini
                </ng-template>
            </tab>
            <tab [active]="activePage === 3" (selectTab)="tabSelect(3)">
                <ng-template tabHeading>
                    // Konten tabset 3 disini
                </ng-template>
            </tab>
            // dan tab seterusnya sesuai kebutuhan
        </ng-container>
    </tabset>
</ng-container>
```

Gambar 3.4 Struktur Kode HTML Komponen *Tabset Custom*

Kedua, adalah Komponen TypeScript mengelola *state tab* dengan membaca dan memperbarui URL *query parameter* activePage. Ketika komponen diinisialisasi, sistem

membaca nilai activePage dari URL. Ketika pengguna memilih tab, sistem memperbarui URL dengan nilai tab baru, seperti yang ditunjukkan di struktur kode pada Gambar 3.5.

```
import { TabsetComponent } from 'ngx-bootstrap/tabs';
@ViewChild('staticTabs', { static: false }) staticTabs?: TabsetComponent;
activePage : any = 1;
tabs: any = [];

const pageFromUrl = queryParams['activePage'] ? +queryParams['activePage'] : 1;
    if (this.activePage !== pageFromUrl) {
      this.activePage = pageFromUrl;
    }

public tabSelect(tabIndex: number) {
    if (this.activePage !== tabIndex) {
      this.activePage = tabIndex;
      this.router.navigate([], {
        relativeTo: this.routeActive,
        queryParams: { activePage: tabIndex },
        queryParamsHandling: 'merge',
        replaceUrl: true
      });
    }
  }
```

Gambar 3.5 Struktur Kode TypeScript *Tabset Custom*

Terakhir, adalah *styling* komponen *tabset* disesuaikan untuk mempertahankan *visual identity* UIIGateway dengan memodifikasi *styling* bawaan ngx-bootstrap *tabset*. Contoh *file* SCSS seperti ditunjukkan pada Gambar 3.6.

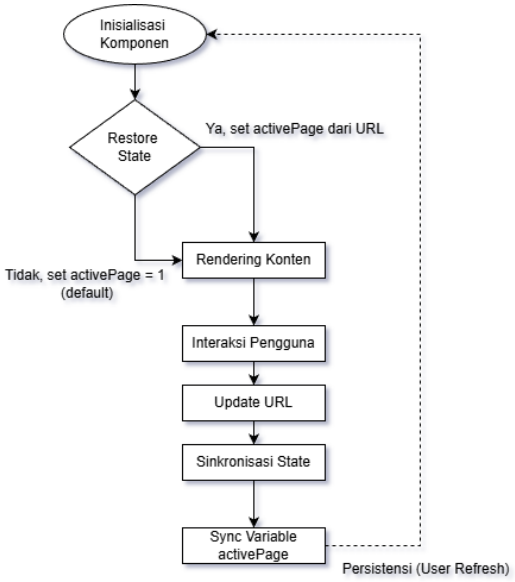
```
:host ::ng-deep .nav-tabs > li > a {
    border-bottom: solid #fbc02d;
}

:host ::ng-deep .nav-tabs.nav-justified > .active > a, .nav-tabs-justified >
.active, .nav-tabs-justified > .active > a:focus {
    color : #093697;
    border: none;
    border-bottom: solid 4px #093697;
}

:host ::ng-deep .nav > li > a:hover {
    background-color: transparent;
}
```

Gambar 3.6 Kode SCSS untuk *Styling Tabset Custom*

Proses kerja komponen *tabset* dengan *state management* dapat dilihat secara visual pada Gambar 3.7. Penjelasan rinci mengenai mekanisme alur kerja dari inisialisasi hingga persistensi *state* dijabarkan dalam Tabel 3.1.



Gambar 3.7 Diagram Alur Cara Kerja *Tabset Custom* dengan *State Management*

Tabel 3.1 Mekanisme Kerja *State Management Tabset* Berbasis URL

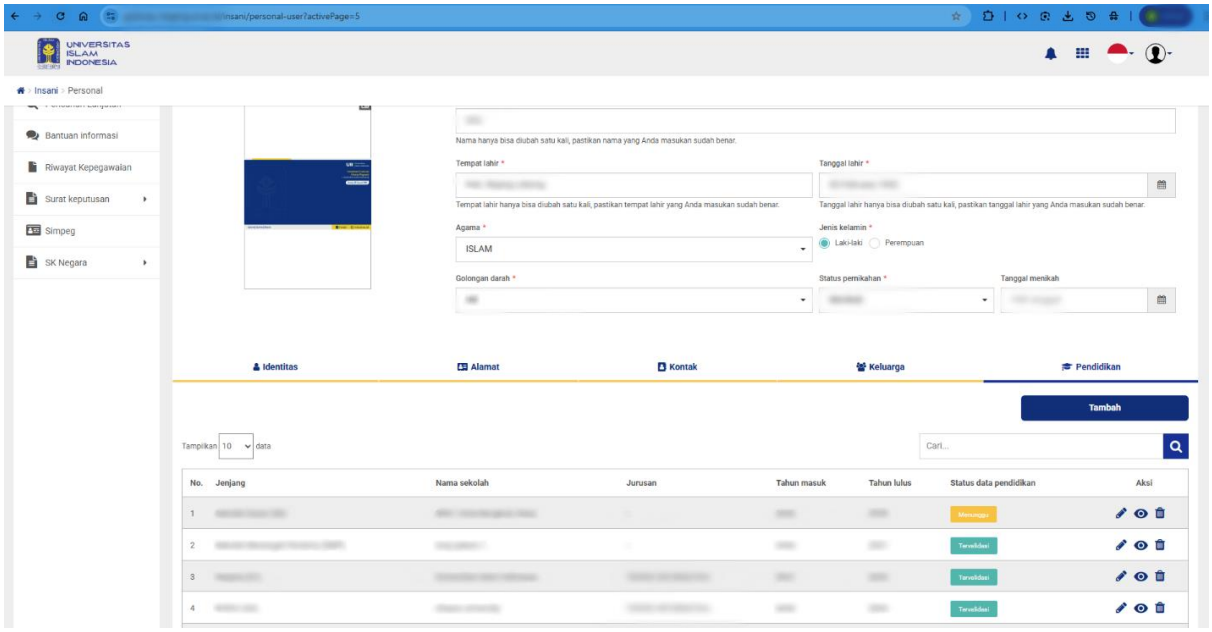
Tahapan	**Deskripsi Proses**
Inisialisasi Komponen	Saat komponen dimuat (ngOnInit), sistem melakukan *subscription* ke ActivatedRoute.queryParams untuk memantau perubahan pada URL.
Restore State	Sistem memeriksa keberadaan parameter activePage pada URL. Jika ditemukan, nilai tersebut digunakan untuk mengaktifkan tab terkait. Jika tidak, sistem menggunakan nilai *default* (Tab 1).
Interaksi Pengguna	Ketika pengguna memilih salah satu tab, *event binding* memicu pemanggilan fungsi tabSelect(index) dengan membawa indeks tab yang dipilih.
Update URL	Fungsi tabSelect() memperbarui URL menggunakan router.navigate() dengan opsi queryParamsHandling: 'merge'. Langkah ini hanya mengubah parameter URL tanpa memuat ulang halaman (*no reload*)
Sinkronisasi *State*	Perubahan pada URL dideteksi kembali oleh *subscription* (Langkah 1), yang secara otomatis memperbarui nilai variabel activePage di dalam komponen agar sinkron dengan URL.

Rendering Konten	Komponen ngx-bootstrap *merender* konten *tab* (Identitas/Alamat/dll) yang sesuai dengan nilai variabel activePage yang baru diperbarui.
Persistensi (*Refresh*)	Jika pengguna melakukan *refresh* halaman, browser memuat ulang URL yang sama (termasuk query parameter), sehingga Langkah 1 dan 2 otomatis berjalan kembali untuk memulihkan posisi tab terakhir.

Dengan mekanisme yang tertera pada tabel di atas, *state tab* tetap persisten melalui proses *refresh* halaman dan dapat dibagikan (*shareable*) melalui URL tanpa memerlukan penyimpanan lokal seperti sessionStorage atau localStorage. Pendekatan ini lebih sederhana karena memanfaatkan fitur bawaan Angular Router secara optimal.

Hasil Implementasi *Tabset Custom* dengan *State Management*

Hasil implementasi komponen *tabset* dengan *state management* URL *query parameter* dapat dilihat pada Gambar 3.8, yang menampilkan halaman *form* data personal modul UIIInsani dengan *tabset custom* yang sudah terintegrasi.



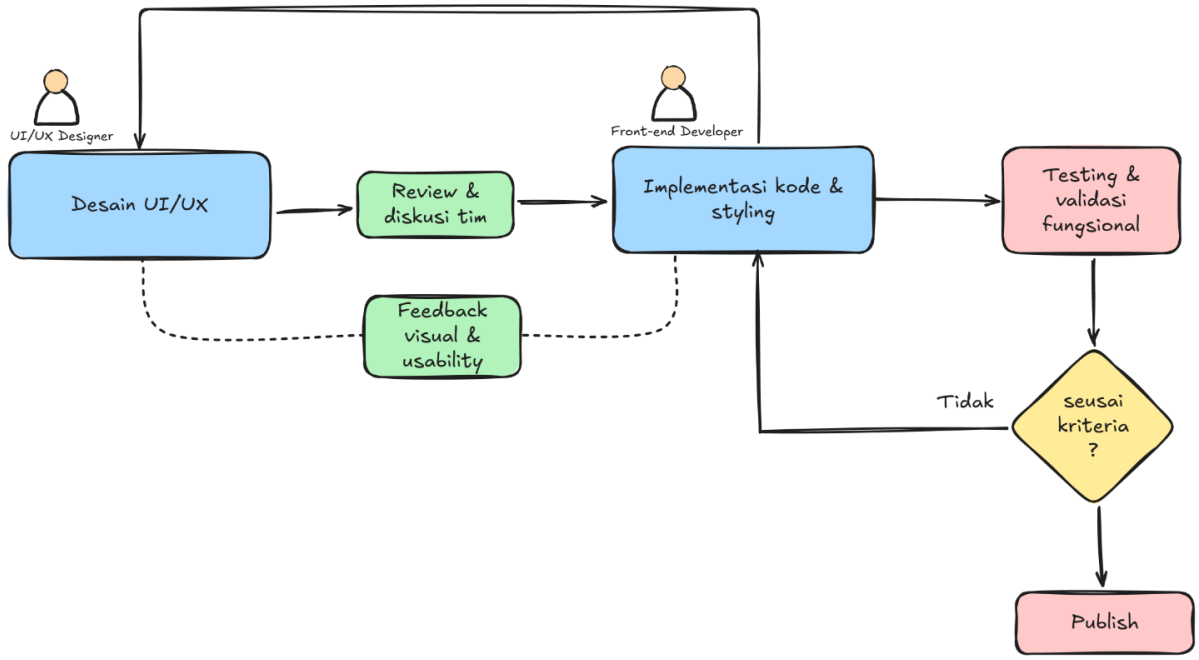
Gambar 3.8 Tampilan Halaman Formulir Personal dengan *Tabset Custom*

Tampilan tersebut menunjukkan antarmuka tab navigasi yang terdiri dari lima kategori utama, yaitu identitas, alamat, kontak, keluarga, dan pendidikan, yang didesain dengan gaya

visual konsisten mengikuti desain sistem UIIGateway. Setiap tab yang aktif ditandai dengan indikator visual yang jelas berupa warna *primary* dan garis bawah (*border-bottom*) yang tegas, memberikan petunjuk navigasi yang intuitif bagi pengguna. Area konten di bawah tab secara dinamis menampilkan *form fields* yang relevan sesuai dengan kategori yang dipilih. Secara fungsional, mekanisme persistensi URL juga terlihat bekerja dengan baik; ketika pengguna berpindah antar tab, URL pada address bar browser secara otomatis diperbarui dengan parameter activePage (misalnya ?activePage=2 untuk tab "Alamat"). Hal ini memastikan bahwa jika pengguna melakukan *refresh* halaman, sistem akan langsung memulihkan tampilan pada tab terakhir yang sedang diakses, sehingga meningkatkan efisiensi dan pengalaman pengguna secara signifikan.

Evaluasi Penerapan *Saved State Management*

Evaluasi hasil pengujian *blackbox* yang dilakukan menunjukkan bahwa implementasi *saved state management* pada modul UIIInsani berhasil mengatasi kendala kritis hilangnya posisi tab saat terjadi pemuatan ulang halaman, di mana penggunaan parameter pada URL kini berfungsi sebagai sumber kebenaran tunggal yang memastikan persistensi data dan memungkinkan fitur navigasi *back-forward* browser berjalan secara sinkron. Keberhasilan teknis ini tidak lepas dari proses kolaborasi yang dinamis antara penulis sebagai pengembang *frontend* dengan unit UI/UX, yang mencakup tahap sinkronisasi desain sistem agar komponen *custom* tetap memiliki estetika yang identik dengan standar visual PilarNg serta diskusi mendalam mengenai *user journey* untuk memastikan transisi antar kategori data tetap terasa mulus tanpa proses render ulang yang tidak perlu. Mekanisme koordinasi lintas unit yang melibatkan pertukaran *feedback* desain serta validasi fungsionalitas ini dapat dilihat secara sistematis pada Gambar 3.9. Selama menyelesaikan tugas magang ini, penulis memperoleh pelajaran berharga yang sangat teknis mengenai pengoptimalan fitur *native* Angular untuk menyelesaikan tantangan di dunia kerja nyata. Penulis belajar bahwa penggunaan *query parameter* sebagai mekanisme *saved state* merupakan solusi yang jauh lebih efisien dibandingkan penggunaan *local storage* untuk menjaga persistensi tab, karena memungkinkan status antarmuka dibagikan secara instan melalui tautan permanen.

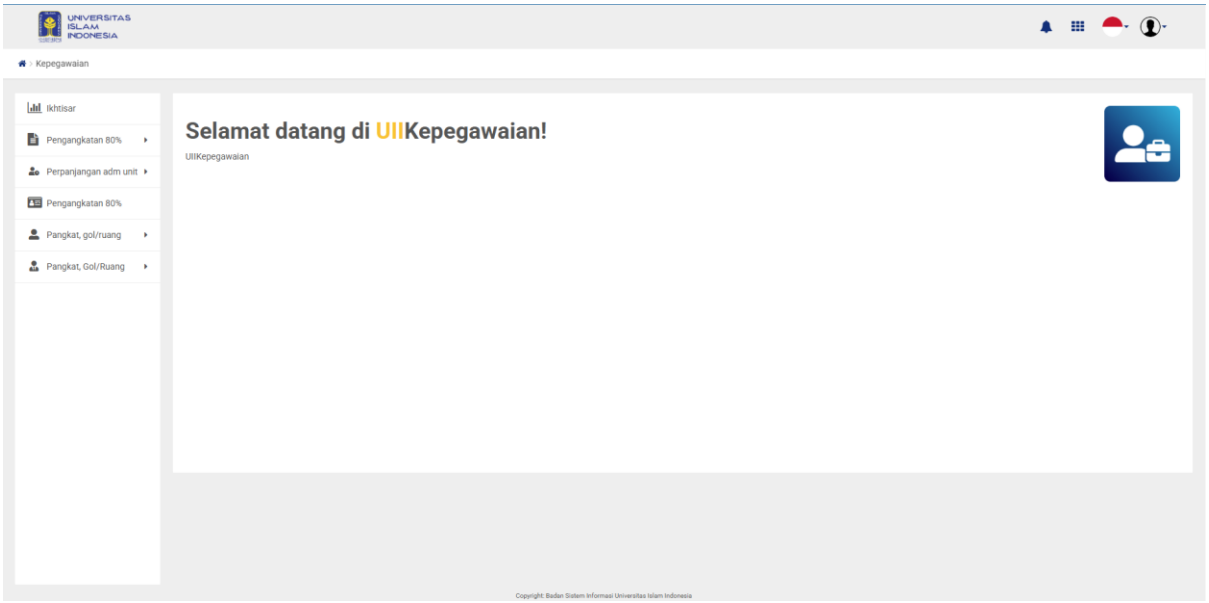


Gambar 3.9 Alur Kolaborasi *Front-end* dan *UI/UX Designer*

3.4.2 Penerapan Tabel Penilaian Dinamis pada Modul UIIKepegawaian

Modul UIIKepegawaian merupakan sistem informasi yang berfungsi untuk mengelola proses administratif kepegawaian di lingkungan Universitas Islam Indonesia, termasuk proses perpanjangan kontrak, kenaikan pangkat/golongan, kenaikan jabatan fungsional, serta evaluasi kinerja pegawai. Modul ini memiliki kompleksitas yang lebih tinggi dibandingkan modul lain karena melibatkan banyak aktor dengan hak akses yang berbeda, seperti admin unit, pimpinan unit, serta pegawai yang dinilai.

Tampilan halaman depan modul UIIKepegawaian dapat dilihat pada Gambar 3.10, yang menampilkan *dashboard* utama dengan menu-menu pengajuan perpanjangan kontrak, penilaian kinerja, dan manajemen dokumen kepegawaian lainnya. Kompleksitas modul ini tercermin dalam jumlah fitur dan alur bisnis yang harus didukung oleh antarmuka pengguna.



Gambar 3.10 Tampilan Halaman Awal Modul Aplikasi UIIKepegawaian

Hasil Implementasi Komponen Tabel Penilaian Dinamis yang *Reusable*

Pada modul UIIKepegawaian, penulis terlibat secara mendalam dalam pengembangan fitur baru serta penyempurnaan fitur yang sudah ada. Kontribusi utama penulis pada modul ini adalah pengembangan komponen tabel penilaian dinamis yang *reusable*, yang menjadi salah satu fitur kunci dalam proses evaluasi kinerja pegawai

Salah satu tantangan teknis yang dihadapi dalam pengembangan modul UIIKepegawaian adalah kebutuhan untuk menampilkan form penilaian yang kompleks dengan banyak indikator dan kolom nilai. Form penilaian ini harus dirancang agar mudah dipahami oleh pengguna, responsif pada berbagai ukuran layar, serta mampu menangani skenario penilaian dengan indikator yang berbeda-beda tergantung jenis kontrak pegawai.

Untuk mengatasi tantangan ini, penulis merancang dan mengembangkan komponen tabel penilaian dinamis yang diberi nama *table-grading-summary*. Komponen ini ditempatkan pada folder *shared* agar dapat diakses dan digunakan kembali (*reusable*) oleh modul lain yang membutuhkan fungsionalitas serupa, sehingga mendorong prinsip *Don't Repeat Yourself* (DRY) dalam pengembangan sistem.

Struktur *template* HTML komponen dirancang untuk menampilkan tabel penilaian dengan baris dinamis sesuai dengan indikator yang diberikan. Setiap baris tabel mencakup nama indikator, bobot nilai, serta kolom *input* untuk nilai yang diberikan oleh penilai. Contoh struktur kode *template* HTML komponen dapat dilihat pada Gambar 3.11, dan Gambar 3.12.

```
<thead>
   <tr>
      <th *ngFor="let header of columns">{{ header.name }}
      </th>
      <th *ngIf="rows[0]?.auth?.canCreate || rows[0]?.auth?.canUpdate"
         style="text-align: center;">Aksi
      </th>
   </tr>
</thead>
```

Gambar 3.11 Struktur Kode *Template* HTML Bagian Header Tabel

```
<tr class="summary-row">
   <td *ngFor="let column of columns">
      <span *ngIf="column.prop === 'unsur_dinilai'">Jumlah
      </span>
      <span *ngIf="column.prop === 'nilai'">{{ totalNilai }}
      </span>
      <span *ngIf="column.prop !== 'unsur_dinilai' && column.prop !== = 'nilai'">
      </span>
   </td>
   <td
     *ngIf="rows[0]?.auth?.canCreate || rows[0]?.auth?.canUpdate"
     class="summary-row">
   </td>
</tr>

<tr class="summary-row">
   <td *ngFor="let column of columns">
      <span *ngIf="column.prop === 'unsur_dinilai'">Nilai rata-rata
      </span>
      <span *ngIf="column.prop === 'nilai'">
            {{ averageNilai !== '-' ? averageNilai.toFixed(2) : '-' }}
      </span>
      <span *ngIf="column.prop === 'ket_nilai'">
            {{ averageKetNilai }}
      </span>
      <span *ngIf="column.prop !== 'unsur_dinilai'
                  && column.prop !== 'nilai'
                  && column.prop !== 'ket_nilai'">
      </span>
    </td>
    <td *ngIf="rows[0]?.auth?.canCreate || rows[0]?.auth?.canUpdate"
        class="summary-row">
    </td>
</tr>
```

Gambar 3.12 Tampilan Kode HTML untuk Jumlah dan Rata-rata Nilai

Template HTML pada Gambar 3.11 menunjukkan bahwa kolom dan baris tabel dibangun secara dinamis menggunakan *structural directive* Angular (*ngFor). Mekanisme ini memungkinkan jumlah indikator penilaian berubah-ubah sesuai konfigurasi sistem tanpa perlu perubahan pada struktur kode komponen. Selain itu, kolom aksi hanya ditampilkan apabila pengguna memiliki hak akses tertentu, yang ditentukan melalui properti otorisasi pada data baris. Logika komponen TypeScript mengelola perhitungan nilai total dan rata-rata secara *real-time* ketika pengguna mengubah nilai pada salah satu indikator. Komponen juga mengelola *state* penilaian, validasi *input*, serta pemicu *event* untuk modul induk. Contoh logika kode untuk

menghitung nilai akhir dapat dilihat pada Gambar 3.13 yang akan dimunculkan pada kode di Gambar 3.12.

```
calculateSummary() {
    const nilaiValues = this.rows.map(row => row.nilai);
    if (nilaiValues.includes(null)) {
      this.totalNilai = '-';
      this.averageNilai = '-';
      this.averageKetNilai = '-';
    } else {
      const nilaiNumbers = nilaiValues.map(value => parseFloat(value));
      this.totalNilai = nilaiNumbers.reduce((sum, value) => sum + value, 0);
            this.averageNilai   =   nilaiNumbers.length   ?   this.totalNilai   /
nilaiNumbers.length : 0;
      this.assignAverageKetNilai();
    }
  }
```

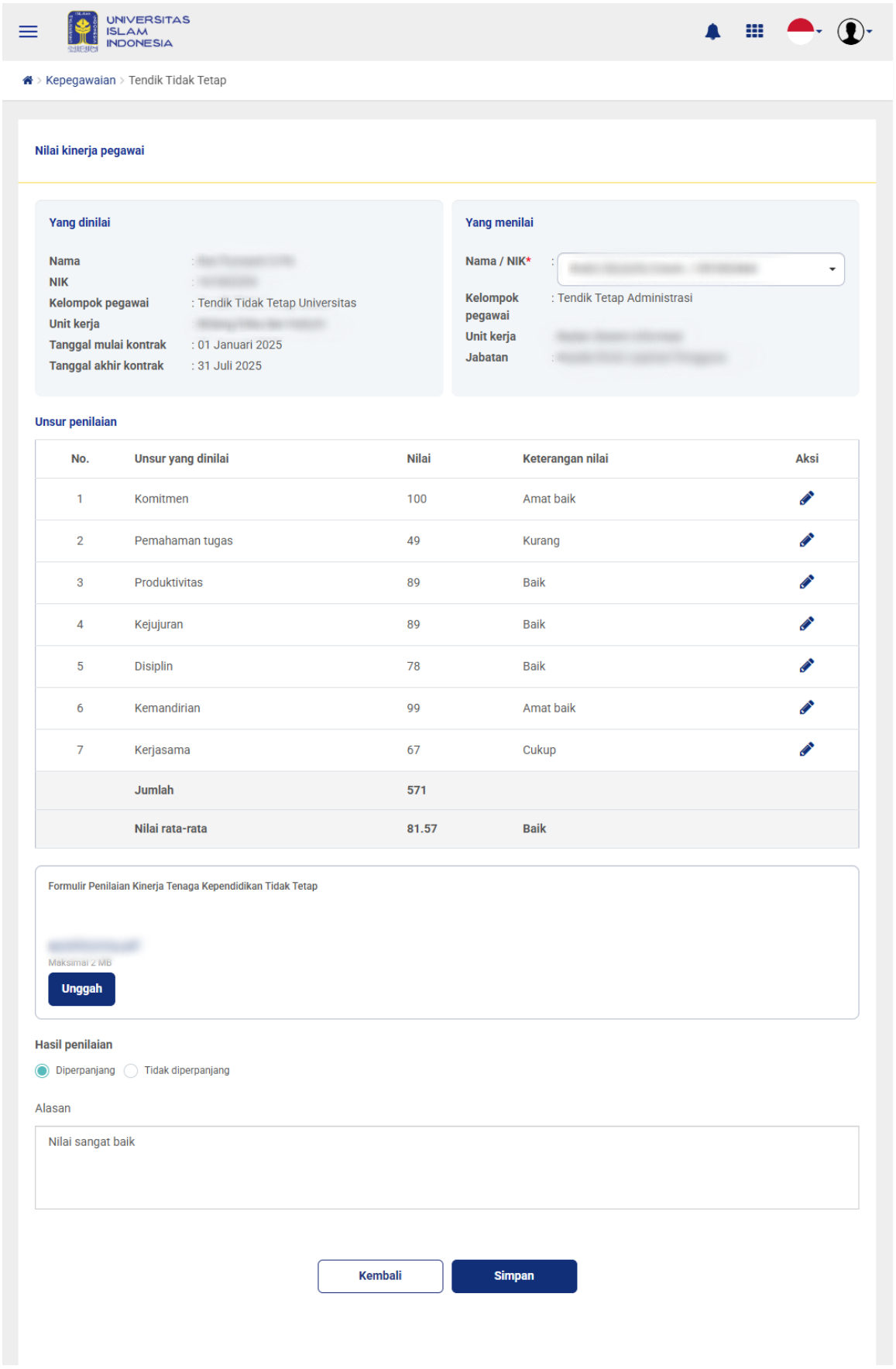
Gambar 3.13 Struktur Kode Perhitungan Nilai pada Komponen Tabel Penilaian

Kode pada Gambar 3.13 menunjukkan bahwa proses perhitungan nilai dilakukan secara lokal pada sisi *front-end* tanpa ketergantungan terhadap *back-end* selama proses pengisian berlangsung. Setiap perubahan nilai akan langsung memicu pembaruan nilai total dan rata-rata melalui mekanisme *data binding*, sehingga pengguna memperoleh umpan balik secara *real-time*.

Selain perhitungan numerik, komponen ini juga mengimplementasikan logika kategorisasi nilai (*grading description*) yang secara otomatis menetapkan keterangan kualitas nilai berdasarkan rentang tertentu. Pendekatan ini meningkatkan keterbacaan hasil penilaian dan mengurangi potensi kesalahan interpretasi oleh pengguna.

Penggunaan *responsive design* memastikan bahwa tabel penilaian tetap dapat digunakan dengan baik pada berbagai ukuran layar, termasuk perangkat tablet dan ponsel. Dengan demikian, komponen *table-grading-summary* tidak hanya memenuhi kebutuhan fungsional penilaian, tetapi juga mendukung konsistensi antarmuka dan kenyamanan pengguna di seluruh modul kepegawaian.

Untuk memperjelas implementasi komponen tabel penilaian dinamis secara nyata pada antarmuka pengguna, tampilan hasil *render* komponen *table-grading-summary* ditunjukkan pada Gambar 3.14. Gambar tersebut menampilkan tabel penilaian tenaga kependidikan tidak tetap yang digunakan dalam proses evaluasi perpanjangan kontrak, lengkap dengan kolom nilai yang dapat diedit, tombol aksi, serta baris ringkasan jumlah dan nilai rata-rata yang dihitung secara otomatis.



Gambar 3.14 Tampilan Komponen Tabel Penilaian Dinamis pada Modul Kepegawaian

Pada tampilan tersebut, setiap baris merepresentasikan unsur penilaian yang dikonfigurasi sebelumnya oleh sistem, sementara kolom nilai memungkinkan pengguna untuk melakukan *input* dan pembaruan data secara langsung. Perubahan nilai akan langsung memicu perhitungan ulang nilai total dan rata-rata pada bagian bawah tabel tanpa memerlukan pemanggilan ulang RESTful API, karena seluruh proses agregasi dilakukan pada sisi *front-end*.

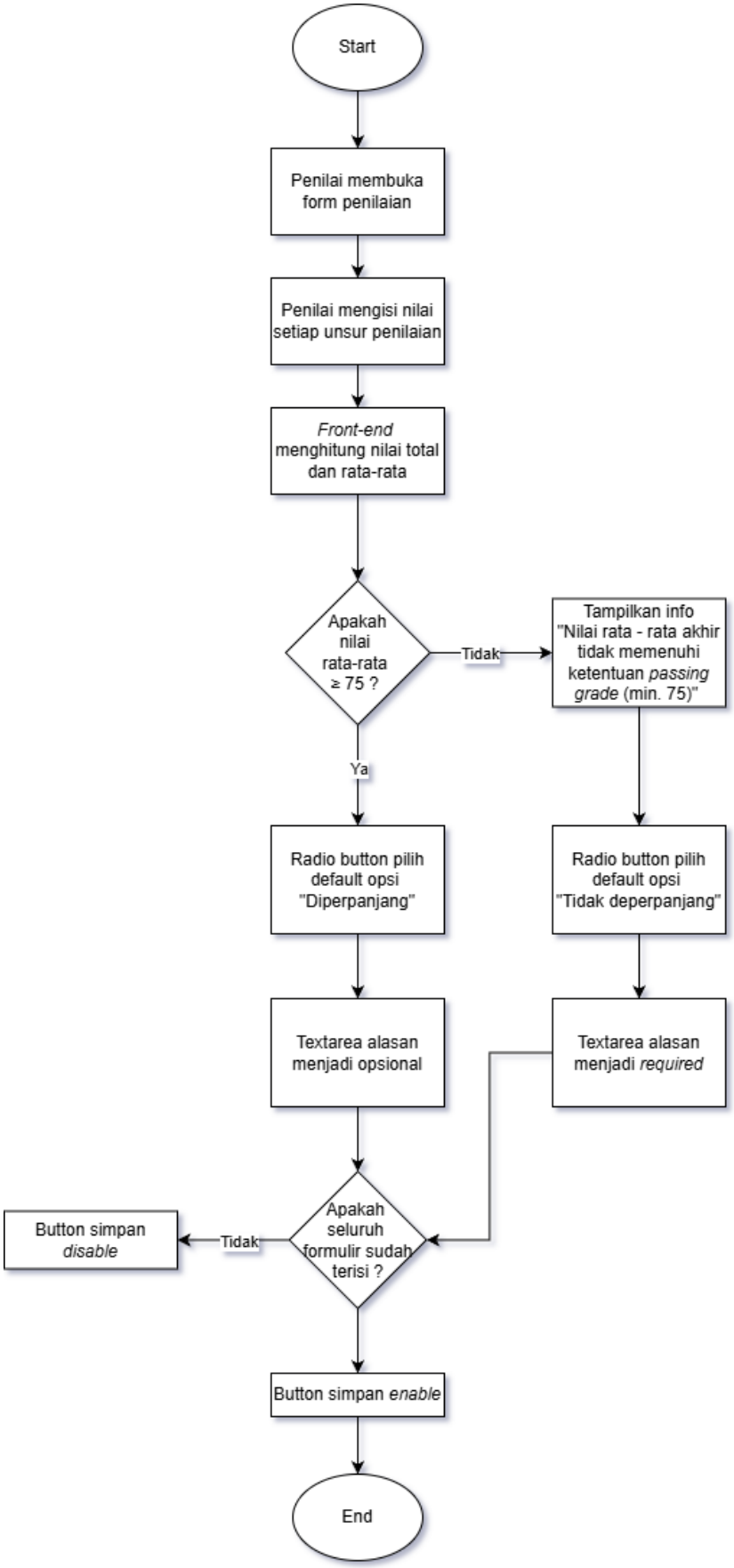
Untuk memastikan proses penilaian tenaga kependidikan berjalan sesuai dengan ketentuan bisnis yang berlaku, *front-end* UIIGateway menerapkan alur interaksi pengguna yang dilengkapi dengan validasi dinamis dan umpan balik visual secara langsung. Alur interaksi ini ditunjukkan pada Gambar 3.15, yang menggambarkan proses mulai dari pengisian nilai hingga penentuan hasil akhir penilaian.

Alur interaksi dimulai ketika penilai mengisi nilai pada setiap unsur penilaian melalui tabel penilaian dinamis. Setiap perubahan nilai secara otomatis memicu proses perhitungan ulang nilai total dan nilai rata-rata pada sisi *front-end* tanpa pemanggilan ulang API. Setelah seluruh unsur penilaian terisi, sistem melakukan evaluasi terhadap nilai rata-rata akhir.

Apabila nilai rata-rata akhir berada di bawah ambang batas ketentuan kelulusan (*passing grade*) sebesar 75, maka sistem menampilkan informasi peringatan berupa pesan *"Nilai rata-rata akhir tidak memenuhi ketentuan passing grade (min. 75)"*. Selain itu, pada kondisi ini sistem secara otomatis memilih (*default select*) opsi hasil penilaian "Tidak Diperpanjang" pada komponen *radio button*. Meskipun demikian, pengguna masih diberikan fleksibilitas untuk mengubah pilihan hasil penilaian menjadi "Diperpanjang" sesuai dengan kebijakan dan pertimbangan yang berlaku.

Selanjutnya, sistem menerapkan validasi kontekstual pada komponen isian alasan. Apabila hasil penilaian dipilih "Tidak Diperpanjang", maka kolom *textarea* alasan akan bersifat wajib (*required*). Sebaliknya, apabila hasil penilaian dipilih "Diperpanjang", kolom alasan bersifat opsional. Validasi ini diimplementasikan menggunakan mekanisme *FormGroup* pada Angular untuk memastikan konsistensi dan kelengkapan data sebelum disimpan.

Sebagai langkah pengendalian kualitas data, tombol simpan akan dinonaktifkan secara otomatis apabila terdapat unsur penilaian yang belum diisi, nilai yang tidak valid, atau kolom wajib yang belum dilengkapi sesuai dengan kondisi hasil penilaian. Dengan pendekatan ini, *front-end* berperan aktif dalam mencegah kesalahan *input* serta memastikan bahwa data yang dikirim ke *back-end* telah memenuhi aturan bisnis yang ditetapkan.



Gambar 3.15 Alur Interaksi Pengguna pada Proses Penilaian

Keberadaan baris ringkasan yang dihasilkan secara dinamis ini memberikan umpan balik instan kepada pengguna terkait hasil penilaian secara keseluruhan, sehingga meningkatkan transparansi dan efisiensi proses evaluasi. Tampilan visual komponen tetap mengikuti standar desain UIIGateway, baik dari sisi tipografi, warna, maupun jarak antar elemen, sehingga konsisten dengan modul-modul kepegawaian lainnya.

Secara keseluruhan, pengembangan komponen tabel penilaian yang *reusable* ini menunjukkan penerapan prinsip desain perangkat lunak yang baik pada sisi *front-end*, khususnya dalam hal modularitas, efisiensi pemrosesan data, dan pemisahan tanggung jawab antara antarmuka dan logika aplikasi.

Evaluasi Penerapan Tabel Penilaian Dinamis yang *Reusable*

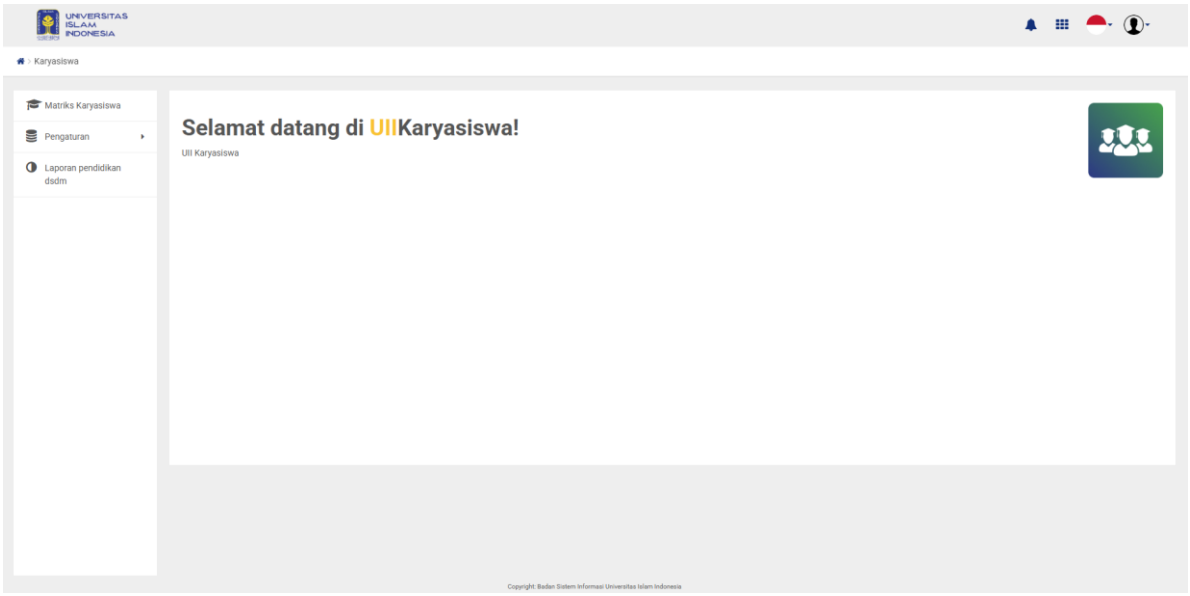
Evaluasi terhadap siklus pengembangan komponen tabel penilaian dinamis pada modul UIIKepegawaian mencakup alur kerja terintegrasi yang dimulai dari tahap permintaan, di mana kebutuhan akan standarisasi instrumen penilaian kinerja yang fleksibel diidentifikasi melalui koordinasi dengan *Business Analyst* dalam kerangka *Agile Scrum*. Tahap ini dilanjutkan ke proses implementasi teknis oleh penulis dengan mengembangkan komponen *shared* yang menggunakan *structural directive* Angular untuk merender struktur tabel secara dinamis berdasarkan konfigurasi API, serta diakhiri dengan tahap evaluasi melalui pengujian *blackbox* untuk memastikan akurasi kalkulasi nilai rata-rata dan keandalan validasi kondisional pada kolom alasan. Permintaan pembuatan tabel penilaian dinamis ini bermula dari *request* dari *stakeholder* yang diteruskan ke *UI/UX Designer*.

Pada tugas ini, penulis memperoleh pelajaran bahwa pengembangan komponen tabel penilaian dinamis memberikan pemahaman mendalam tentang *Component-Based Architecture*, di mana penulis berhasil mengimplementasikan logika perhitungan agregat dan kategorisasi nilai (*grading description*) secara *real-time* di sisi *front-end* untuk memberikan *instant feedback* kepada pengguna sebelum data divalidasi lebih lanjut oleh *backend*. Pengalaman ini mengasah kemampuan penulis dalam menerjemahkan aturan bisnis yang kaku, seperti ambang batas *passing grade* dan kewajiban pengisian alasan menjadi komponen antarmuka yang dinamis, responsif, dan siap digunakan kembali (*reusable*) oleh modul lain dalam aplikasi UIIGateway, sehingga memastikan integritas data tetap terjaga sejak dari interaksi pertama pengguna.

3.4.3 Penerapan Animasi Pemuatan Dinamis pada Modul UIIKaryasiswa

Modul UIIKaryasiswa merupakan sistem informasi yang berfungsi untuk mengelola program karyasiswa dosen di lingkungan Universitas Islam Indonesia, mulai dari proses pengajuan, seleksi, hingga pemantauan status studi dosen penerima karyasiswa. Modul ini berperan sebagai penghubung antara data kepegawaian dan data akademik dosen penerima beasiswa, sehingga memerlukan integrasi yang erat dengan modul lain dalam ekosistem UIIGateway, khususnya modul UIIInsani sebagai sumber data utama dosen.

Tampilan halaman depan modul UIIKaryasiswa dapat terlihat pada Gambar 3.16. Modul UIIKaryasiswa memiliki beberapa menu yang memiliki fitur untuk melihat daftar dosen beserta status pengajuan karyasiswa dalam bentuk tabel yang dilengkapi fitur pencarian dan filter berdasarkan kelompok pegawai maupun unit kerja. Antarmuka ini dirancang agar memudahkan administrator dan pengelola program karyasiswa dalam memantau seluruh proses pengajuan secara terpusat.



Gambar 3.16 Tampilan Halaman Awal Modul Aplikasi UIIKaryasiswa

Pada pengembangan aplikasi UIIGateway modul UIIPortofolio, salah satu tantangan yang dihadapi adalah proses sinkronisasi data dalam jumlah besar dari sistem eksternal, khususnya data dosen dari sistem Insani. Infrastruktur server yang digunakan di lingkungan BSI UII pada saat pelaksanaan magang belum mendukung mekanisme *Server-Sent Events* (SSE) maupun *real-time streaming* data secara berkelanjutan. Selain itu, penerapan SSE berpotensi menambah beban server apabila digunakan untuk proses sinkronisasi data berskala besar.

Sebagai solusi dari sisi *front-end*, diterapkan pendekatan sinkronisasi data bertahap menggunakan mekanisme *offset*, yang dikombinasikan dengan animasi pemuatan dinamis untuk memberikan umpan balik visual yang informatif kepada pengguna. Pendekatan ini bertujuan agar proses sinkronisasi tetap terasa responsif dan transparan, meskipun data diproses secara bertahap di belakang layar.

Proses sinkronisasi diawali dengan pengambilan total jumlah data dari *back-end* untuk menentukan batas akhir sinkronisasi. Setelah total data diketahui, *front-end* melakukan pemanggilan API secara berulang dengan parameter *limit* dan *offset*. Pada implementasi ini, nilai *limit* ditetapkan sebesar 200 data per permintaan, sedangkan nilai *offset* akan bertambah secara bertahap hingga seluruh data berhasil disinkronkan.

Untuk meningkatkan pengalaman pengguna, animasi progres tidak langsung meloncat mengikuti hasil respons API, melainkan digerakkan secara bertahap menggunakan *smooth progress animation*. Ketika *front-end* memulai pemanggilan *offset* pertama (*offset* 0), indikator progres akan bergerak perlahan hingga mendekati jumlah data yang diperkirakan akan diterima. Setelah respons API diterima, nilai progres diperbarui ke nilai aktual, kemudian dilanjutkan dengan pemanggilan *offset* berikutnya. Pola ini menciptakan ilusi proses yang terus berjalan tanpa jeda yang terasa janggal bagi pengguna.

Tahapan sinkronisasi data berbasis *offset* tersebut dapat dirangkum pada Tabel 3.2, yang menunjukkan hubungan antara nilai *offset*, jumlah data yang diterima, serta pergerakan animasi progres pada antarmuka pengguna.

Tabel 3.2 Tahapan Sinkronisasi Data Dosen Berbasis *Offset*

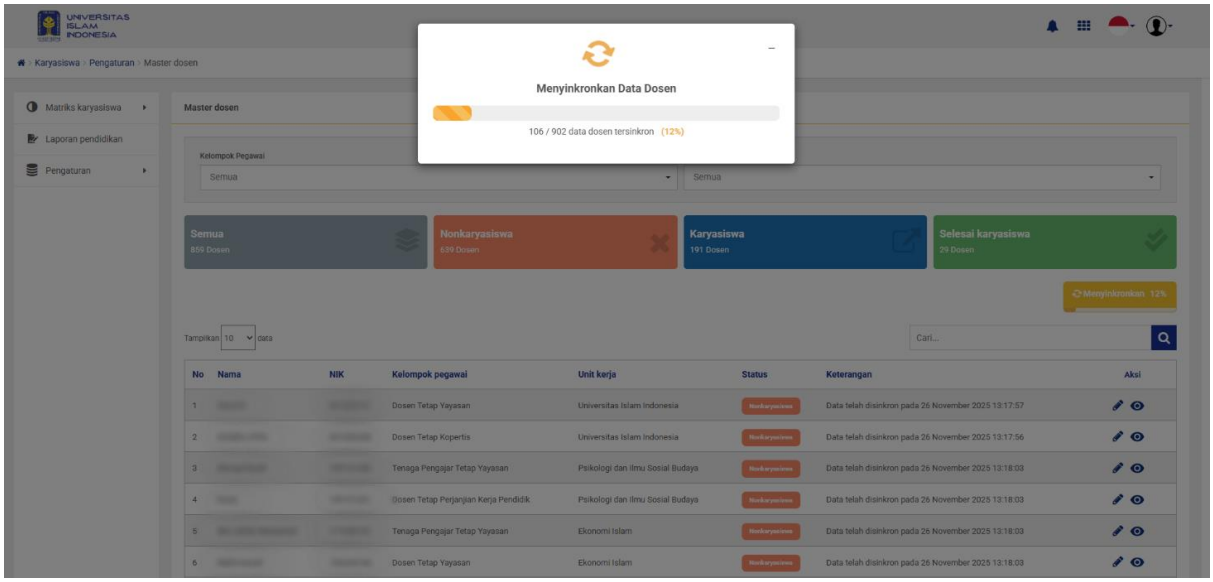
Tahap	***Offset***	***Limit***	**Jumlah Data Diterima**	***Progress* Animasi**
Pertama	0	200	200 data	0% -> ±25%
Kedua	200	200	200 data	±25% -> ±50%
ketiga	400	200	200 data	±50% -> ±75%
Keempat	600	200	200 data	±75% -> ±100%

Pergerakan animasi progres dihitung berdasarkan rasio antara jumlah data yang telah tersinkronisasi terhadap total data keseluruhan. Secara matematis, perhitungan persentase progres dapat dinyatakan dengan persamaan (3.1)

$$\text{Progres}\ (\%) = \frac{\text{Jumlah data tersinkronisasi}}{\text{Total data}} . 100\% \qquad (3.1)$$

Sebagai contoh, apabila total data dosen berjumlah 800 dan pada tahap pertama telah berhasil disinkronkan 200 data, maka persentase progres yang ditampilkan pada antarmuka adalah sebesar 25%. Nilai ini digunakan oleh front-end untuk mengatur lebar *progress bar* serta informasi persentase yang ditampilkan kepada pengguna.

Tampilan animasi pemuatan sinkronisasi data ditampilkan dalam bentuk *modal loader* yang berisi indikator animasi, *progress bar* dinamis, serta informasi jumlah data yang telah tersinkronisasi. Ilustrasi tampilan *modal loader* sinkronisasi data dosen ditunjukkan pada Gambar 3.17, yang menggambarkan kondisi proses sinkronisasi sedang berlangsung.



Gambar 3.17 Tampilan Animasi Pemuatan Data Dinamis Modul UIIKaryasiswa

Selain itu, implementasi logika sinkronisasi dan pengendalian animasi progres dilakukan sepenuhnya pada sisi *front-end* menggunakan *framework* Angular. Logika tersebut mencakup pengelolaan status sinkronisasi, pengaturan *offset*, penghitungan progres aktual, serta pengendalian animasi progres yang berjalan secara bertahap. Kode dari fungsi sinkronisasi data dan komponen *modal loader* ditunjukkan pada Gambar 3.18, dan Gambar 3.19, yang

memperlihatkan bagaimana *front-end* mengatur alur sinkronisasi tanpa bergantung pada mekanisme *real-time* dari server.

```
async onSyncInsani() {
    if (this.isSyncing) {
      this.syncModalRef = this.modalSvc.show(ModalSyncLoaderComponent, {
        initialState: {
          current: this.syncProgress.current,
          total: this.syncProgress.total,
          confirmStep: false // lewati konfirmasi jika sudah sinkronisasi
        },
        class: "modal-md",
        backdrop: "static",
        keyboard: false,
        ignoreBackdropClick: true,
      });
      return;
    }
    this.karyaSvc.get(KaryasiswaServiceType.MD_SYNC_INSANI, { limit: 1, offset: 0
}).subscribe({
      next: (response) => {
        const total = response.total || response.count || (response.data ?
response.data.length : 0);
        this.syncModalRef = this.modalSvc.show(ModalSyncLoaderComponent, {
          initialState: {
            current: 0,
            total: total,
            confirmStep: true,
            onConfirm: () => this.startSyncInsani(total)
          },
          class: "modal-md",
          backdrop: "static",
          keyboard: false,
          ignoreBackdropClick: true,
        });
      },
      error: () => {
        this.toastSvc.error("Gagal mengambil total data dosen dari insani.");
      }
    });
  }
  startSyncInsani(totalFromApi?: number) {
    this.isSyncing = true;
    const limit = 200;
    let offset = 0;
    let current = 0;
    let total = totalFromApi || 0;
    let smoothTimer: any = null;
    if (this.syncModalRef && this.syncModalRef.content) {
      this.syncModalRef.content.confirmStep = false;
      this.syncModalRef.content.current = 0;
      this.syncModalRef.content.total = total;
      this.syncProgress.total = total;
    }
```

Gambar 3.18 Kode Fungsi untuk Sinkronisasi Dinamis Tahap Awal

```
const startSmoothProgress = (target: number) => {
      if (smoothTimer) clearInterval(smoothTimer);
      smoothTimer = setInterval(() => {
        if (this.syncModalRef && this.syncModalRef.content) {
          // Hanya increment jika belum mencapai nilai real saat ini
          if (this.syncModalRef.content.current < target) {
            this.syncModalRef.content.current += 1;
            this.syncProgress.current = this.syncModalRef.content.current;
          }
        }
      }, 20); // kecepatan loading
    };

    const stopSmoothProgress = () => {
      if (smoothTimer) clearInterval(smoothTimer);
      smoothTimer = null;
    };

    const fetchBatch = () => {
      this.karyaSvc.get(KaryasiswaServiceType.MD_SYNC_INSANI,   {   limit,   offset
}).subscribe({
        next: (response) => {
          if (offset === 0) {
            total = response.total || response.count || response.data.length;
            this.syncModalRef.content.total = total;
            this.syncProgress.total = total;
          }
          // stop progress halus dan set ke nilai real
          stopSmoothProgress();
          current += response.data.length;
          this.syncModalRef.content.current = current;
          this.syncProgress.current = current;

          if (current < total && response.data.length > 0) {
            offset += limit;
            startSmoothProgress(current + limit > total ? total : current + limit);
            fetchBatch();
          } else {
            this.toastSvc.success(`Sinkron data dosen dari insani berhasil. Total
data tersinkronisasi: ${current} data.`);
            this.syncModalRef.hide();
            this.syncModalRef = null;
            this.isSyncing = false;
            stopSmoothProgress();
            this.retrievePersonalTable(this.page);
          }
        },
        error: () => {
          if (this.syncModalRef) this.syncModalRef.hide();
          this.syncModalRef = null;
          this.toastSvc.error("Sinkron data dosen dari insani gagal.");
          this.isSyncing = false;
          stopSmoothProgress();
        }
      });
    };
    startSmoothProgress(limit);
    fetchBatch();
  }
```

Gambar 3.19 Kode Fungsi untuk Sinkronisasi Dinamis untuk *Smooth Progress*

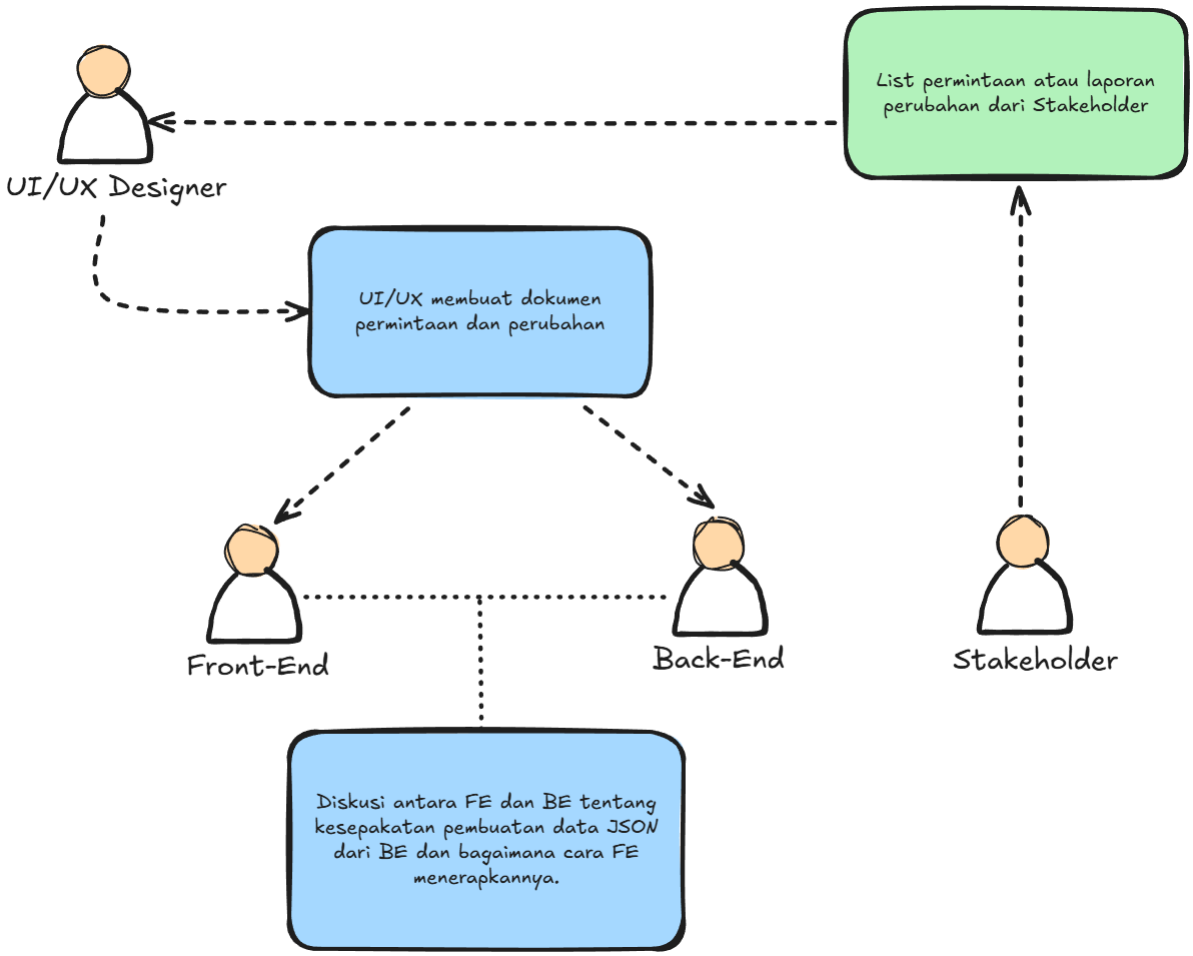
Dengan pendekatan animasi pemuatan dinamis berbasis data *offset* ini, proses sinkronisasi data dalam jumlah besar dapat disajikan secara lebih informatif dan ramah pengguna. Solusi ini

juga menunjukkan peran aktif *front-end* dalam mengelola pengalaman pengguna (*user experience*) meskipun terdapat keterbatasan pada sisi *back-end*. Pendekatan tersebut terbukti efektif dalam menjaga responsivitas antarmuka sekaligus memberikan transparansi proses sinkronisasi kepada pengguna sistem.

Evaluasi Penerapan Animasi Pemuatan Dinamis Berbasis Data *Offset*

Evaluasi terhadap implementasi animasi pemuatan dinamis pada modul UIIKaryasiswa menunjukkan bahwa transisi dari proses sinkronisasi yang semula bersifat pasif menjadi informatif telah meningkatkan kepuasan pengguna secara signifikan, terutama saat menangani ribuan data dosen yang memerlukan waktu pemrosesan lama. Implementasi ini didasari oleh permintaan dari *stakeholder* yang menekankan perlunya transparansi proses agar pengguna mendapatkan kepastian bahwa sistem benar-benar sedang bekerja dan tidak sedang mengalami kendala (*freeze*). Permintaan tersebut kemudian diteruskan kepada UI/UX Designer untuk merancang indikator pemuatan yang mampu memberikan umpan balik visual secara dinamis dan intuitif.

Sebagai tindak lanjut teknis, penulis selaku *Frontend Developer* (FE) berkolaborasi erat dengan pengembang *Backend* (BE) untuk menyepakati kontrak API yang menyediakan informasi *total_count* di awal pemanggilan. Data total tersebut sangat krusial bagi penulis untuk menghitung persentase progres secara akurat sebelum parameter *limit* membatasi jumlah data yang tampil di setiap *batch*. Koordinasi ini memastikan bahwa indikator progres tidak hanya sekadar animasi estetis, melainkan representasi data aktual yang memberikan kepastian kepada pengguna mengenai status sinkronisasi yang sedang berjalan. Alur koordinasi lintas unit yang melibatkan aspirasi *stakeholder*, rancangan UI/UX, hingga penyelarasan teknis FE dan BE ini dijabarkan secara sistematis pada Gambar 3.20.



Gambar 3.20 Alur Kolaborasi Antar Unit dalam Pengembangan Fitur Sinkronisasi

Pelajaran berharga yang penulis dapatkan secara spesifik dari tugas ini adalah pentingnya perancangan kontrak API yang berorientasi pada kebutuhan antarmuka (*UI-driven API design*), di mana penulis belajar untuk mengomunikasikan kebutuhan *metadata* pendukung seperti jumlah total data kepada tim *backend* demi menunjang fitur navigasi dan progres yang informatif sesuai arahan *stakeholder*. Selain itu, penulis memperoleh keahlian teknis dalam mengelola logika *asynchronous recursion* dan *interval management* menggunakan RxJS di Angular untuk menciptakan "ilusi" proses yang mulus melalui *smooth progress animation*, yang terbukti efektif dalam memitigasi kesan lambat akibat kendala *latency* jaringan atau beban pemrosesan server yang berat. Pengalaman ini memberikan wawasan mendalam bagi penulis bahwa seorang pengembang *frontend* di lingkungan BSI UII harus mampu menjadi jembatan kreatif antara keterbatasan infrastruktur teknis dengan ekspektasi kenyamanan pengguna, sehingga integritas data tetap terjaga tanpa mengorbankan pengalaman pengguna yang responsif dalam sistem informasi karyasiswa.

3.4.4 Migrasi Pola Pemanggilan API

Subbab ini menjelaskan kegiatan migrasi pola pemanggilan API dari *path parameter* ke *query parameter* yang diterapkan secara global pada seluruh modul aplikasi UIIGateway, termasuk modul-modul kepegawaian yang penulis kerjakan seperti UIIInsani, UIIPortofolio, UIIKepegawaian, dan UIIKaryasiswa. Migrasi ini bukan sekadar perubahan teknis pada satu modul, tetapi merupakan kebijakan arsitektural yang disepakati dan dijalankan bersama oleh Tim *Interoperability*, *Security Operation Center* (SOC), serta tim pengembang (*developer*) *front-end* dan *back-end*.

Pemicu utama perubahan pola pemanggilan API ini adalah insiden serangan siber yang terjadi pada UIIGateway pada bulan April 2025, yang menyoroti kelemahan pada beberapa *endpoint* yang masih menggunakan *path parameter* untuk membawa identitas sensitif, seperti UUID personal pegawai atau ID entitas tertentu. Pola URL dengan *path parameter* seperti (/personal/{uuid}/detail) dinilai lebih rentan terhadap upaya eksploitasi melalui teknik URL *manipulation* atau ID *enumeration*, terutama pada *endpoint* yang belum memiliki lapisan validasi dan otorisasi yang benar-benar ketat.

Sebagai respon terhadap temuan tersebut, Tim *Interoperability* bersama SOC melakukan kajian terhadap pola pemanggilan API yang sudah berjalan dan kemudian merekomendasikan standar baru, yaitu penggunaan *query parameter* untuk membawa nilai-nilai dinamis pada *endpoint*, misalnya menjadi (?uuid={uuid}) atau (?id={id}). Standar ini memudahkan penerapan lapisan keamanan tambahan di sisi *back-end*, seperti validasi parameter yang lebih eksplisit, *logging* terstruktur, serta penerapan aturan pada *Web Application Firewall* (WAF) yang lebih presisi terhadap parameter tertentu. Keputusan ini kemudian dilakukan normalisasi sebagai kebijakan teknis yang harus diikuti oleh seluruh modul aplikasi yang terhubung ke UIIGateway, sehingga migrasi pola pemanggilan API menjadi sebuah pekerjaan kolaboratif lintas tim, bukan hanya per modul atau per pengembang saja.

Implementasi migrasi ini membutuhkan koordinasi erat antara tim *back-end* dan *front-end*. Di sisi *back-end*, pengembang bertanggung jawab untuk :

- Mengubah definisi *endpoint* agar tidak lagi membaca nilai dinamis dari *path*, melainkan dari *query parameter*.
- Menyesuaikan *middleware*, mekanisme validasi, dan *logging* agar seluruh parameter penting diambil dari *query string*.
- Memperbarui dokumentasi atau API contract, sehingga tim front-end memiliki acuan yang jelas saat menyesuaikan pemanggilan API.

Di sisi *front-end*, penulis dan anggota tim *front-end* lainnya bertugas untuk :

- Mengidentifikasi semua pemanggilan API yang masih menggunakan pola *path parameter* pada berbagai modul (misalnya pemanggilan data personal, portofolio, kontrak, dan data dosen).
- Mengubah seluruh pemanggilan tersebut ke pola *query parameter* sesuai kontrak baru.
- Melakukan pengujian ulang pada fungsi dan menu terkait untuk memastikan tidak terjadi regresi fungsional setelah perubahan pola URL diterapkan.

Salah satu tantangan utama di sisi *front-end* adalah penanganan komponen yang digunakan bersama (*shared component*) untuk lebih dari satu menu atau peran pengguna. Pada modul UIIInsani, misalnya, satu komponen halaman digunakan baik oleh admin super maupun admin fakultas, sementara sebelum migrasi pembeda konteks peran tersebut ditempatkan pada bagian *path* URL API. Sebelum migrasi, pemanggilan API dapat dibedakan menggunakan variasi *path*, misalnya :

- v1/{aplikasi}/admin-super/{uuid}/{menu} /detail
- v1/{aplikasi}/admin-fakultas/{uuid}/{menu} /detail

Setelah migrasi, variasi ini tidak lagi ditempatkan pada *path*, melainkan dipindahkan menjadi kombinasi *query parameter*, seperti :

- v1/{aplikasi}/{menu} /detail?uuid=[uuid]&role=admin-super
- v1/{aplikasi}/{menu} /detail?uuid=[uuid]&role=admin-fakultas

Perubahan tersebut menuntut penyesuaian logika di sisi *front-end* untuk membentuk *query parameter* berdasarkan konteks peran atau menu aktif yang sebelumnya diwakili oleh *path*. Selain itu, komponen yang dipakai ulang harus diuji kembali dengan berbagai kombinasi peran dan menu untuk memastikan bahwa tidak terjadi kebingungan konteks maupun kegagalan otorisasi setelah pola parameter diubah. Untuk perubahan API dengan metode HTTP seperti GET, penyesuaian dilakukan dengan cara mengganti konstruksi URL yang semula menggabungkan nilai dinamis langsung ke dalam *path* menjadi pengisian nilai tersebut pada bagian *query* string. Pada modul UIIInsani, contoh perubahan kode dari sisi *front-end* untuk pemanggilan API pada *file* TypeScript ditunjukkan pada Gambar 3.21 dan Gambar 3.22, yang masing-masing memperlihatkan perbandingan implementasi sebelum dan sesudah migrasi pola pemanggilan API.

```
retrievePersonal(uuid) {

const param = '/' + uuid + '/data-pribadi';
this.insaniSvc.getParam(InsaniServiceType.IAS_PERSONAL_SEARCH,
param).subscribe(response =>
 {
    this.dataUserCv.dataPersonalForm = response.data;
    this.isLoading = false;
 });

}
```

Gambar 3.21 Kode Pemanggilan API dengan *Path Parameter*

```
retrievePersonal(uuid) {

const param = '/data-pribadi?uuid_personal=' + uuid;
this.insaniSvc.getParam(InsaniServiceType.IAS_PERSONAL_SEARCH,
param).subscribe(response =>
 {
    this.dataUserCv.dataPersonalForm = response.data;
    this.isLoading = false;
 });

}
```

Gambar 3.22 Kode Pemanggilan API dengan *Query Parameter*

Perubahan pola pemanggilan API pada berbagai modul dirangkum dalam Tabel 3.3, sehingga perbedaan antara pola lama dan pola baru dapat dilihat dengan jelas untuk berbagai kebutuhan fungsional.

Tabel 3.3 Perbandingan Pola Pemanggilan API Sebelum dan Sesudah Migrasi

No.	**Modul**	**Kebutuhan / Fungsi**	**Sebelum (*Path Parameter)***	**Sesudah (*Query Parameter)***
1	UIIInsani	Mengambil data personal pegawai	/{uuid}/data-prbadi	/data-prbadi?uuid_personal ={uuid}
2	UIIPortfolio	Mengambil data personal portofolio	/super-admin/{nik}/detail	/detail?nik={nik}&role=super-admin

3	UIIKepegawaian	Mengambil data personal kepegawaian	/{uuid}/grading	/grading?uuid_pegawai={uuid}
4	UIIKaryasiswa	Mengambil data dosen yang akan karyasiswa	/{nik}/detail-dosen	/detail-dosen?nik={nik}

Tabel 3.3 menunjukkan bahwa migrasi tidak hanya mengubah format URL, tetapi juga menstandarkan cara pengiriman parameter di seluruh modul, sehingga memudahkan pengelolaan keamanan, *logging*, dan dokumentasi API secara terpusat.

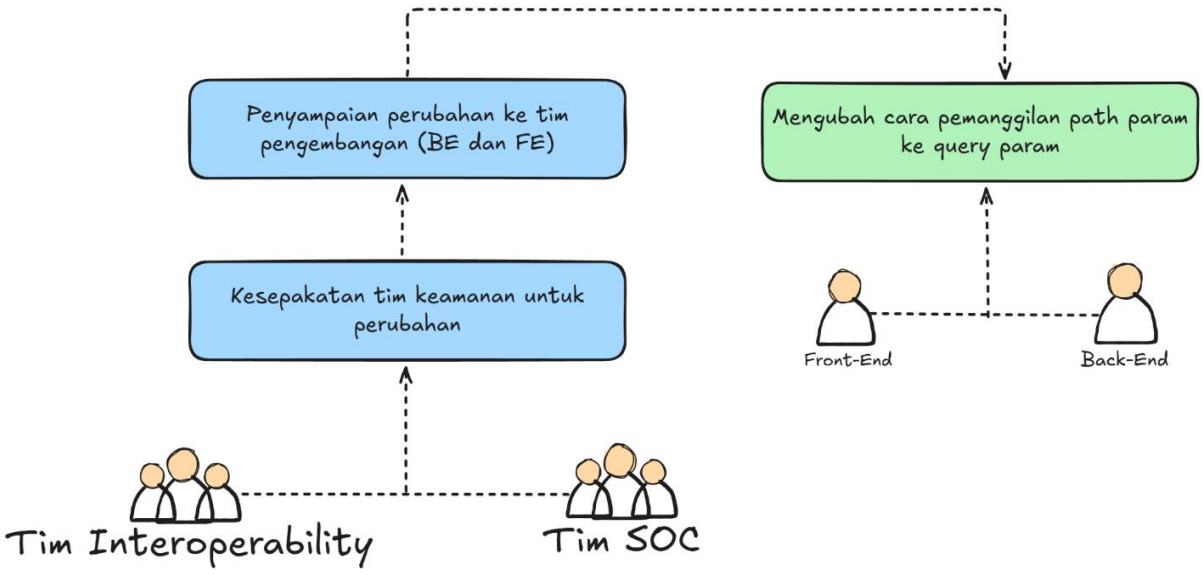
Untuk meminimalkan risiko *error* dan regresi fungsional, penulis menerapkan strategi pengujian berlapis setelah migrasi pola pemanggilan API dilakukan. Pengujian dilakukan pada beberapa tingkat:

- Pengujian per komponen, dimana dipastikan setiap komponen yang memanggil API lama telah dikonversi ke pola *query parameter* dan tetap mengembalikan data yang benar (misalnya data personal, daftar portofolio, daftar pengajuan kontrak, dan data dosen karyasiswa).
- Pengujian per alur bisnis, dimana menjalankan alur lengkap seperti menampilkan detail pegawai, mengisi dan menyimpan *form* penilaian, serta menjalankan sinkronisasi data dosen, untuk memastikan tidak ada langkah yang terputus akibat perubahan URL *endpoint*.
- Pengujian lintas peran pengguna, dimana khusus pada komponen yang digunakan oleh lebih dari satu jenis peran (admin super, admin fakultas, admin unit), pengujian dilakukan menggunakan akun yang berbeda untuk memastikan setiap peran tetap mengakses data sesuai hak aksesnya setelah pola parameter diubah.

Melalui kombinasi perubahan arsitektural di sisi *back-end* dan penyesuaian di sisi *front-end* yang diuji secara menyeluruh, migrasi pola pemanggilan API dari *path parameter* ke *query parameter* berhasil diterapkan tanpa mengganggu operasi harian pengguna, sekaligus meningkatkan standar keamanan dan konsistensi integrasi di seluruh ekosistem UIIGateway.

Evaluasi Penerapan *Query Parameter* dari *Path Parameter*

Evaluasi terhadap proses migrasi arsitektural ini menunjukkan bahwa transisi pola pemanggilan API dari *path parameter* ke *query parameter* telah berhasil meningkatkan postur keamanan sistem secara signifikan dengan menutup celah manipulasi URL dan *ID enumeration* yang menjadi temuan kritis pasca insiden siber April 2025. Alur siklus pekerjaan ini dimulai dari tahap *request*, di mana tim *Security Operation Center* (SOC) dan Tim *Interoperability* mengeluarkan mandat keamanan global yang mewajibkan normalisasi seluruh *endpoint* dinamis untuk memfasilitasi filter *Web Application Firewall* (WAF) yang lebih presisi. Tahap ini kemudian dilanjutkan ke proses implementasi teknis oleh penulis di sisi *frontend*, yang melibatkan pemetaan ulang (*mapping*) seluruh fungsi servis pada modul UIIInsani hingga UIIKaryasiswa, serta melakukan *refactor* pada komponen bersama (*shared components*) agar mampu membentuk *query string* secara dinamis berdasarkan konteks peran pengguna seperti admin super, dan admin unit. Proses ini diakhiri dengan tahap evaluasi melalui pengujian regresi berlapis yang memastikan bahwa perubahan struktur URL tidak mengganggu integritas data maupun otorisasi akses pada fitur-fitur krusial. Koordinasi lintas unit yang melibatkan tim keamanan, pengembang *backend*, dan *frontend* dalam menjalankan standarisasi ini dipaparkan secara sistematis pada Gambar 3.23.



Gambar 3.23 Alur Kolaborasi Keamanan dan Migrasi API

Pelajaran berharga yang penulis dapatkan secara spesifik dari tugas migrasi global ini adalah pentingnya kesadaran akan keamanan informasi (*security awareness*) dalam setiap baris kode

yang ditulis, di mana pemilihan pola pengiriman parameter ternyata memiliki dampak besar pada kemudahan pemantauan *logging* dan efektivitas pertahanan lapis pertama di sisi server. Penulis belajar mengelola *refactor* kode skala besar secara terstruktur, di mana konsistensi penamaan parameter seperti uuid_personal atau uuid_pegawai menjadi kunci utama dalam menjaga interoperabilitas antar modul dalam ekosistem UIIGateway. Pengalaman ini juga mengasah ketelitian penulis dalam melakukan pengujian lintas peran (*cross-role testing*), memberikan pemahaman mendalam bahwa sebuah perubahan arsitektural yang terlihat sederhana di sisi URL menuntut penyesuaian logika *frontend* yang kompleks untuk memastikan setiap kategori pengguna tetap mendapatkan hak akses yang tepat tanpa terjadi kebocoran data.

3.4.5 Evaluasi Dampak Implementasi terhadap Pengalaman Pengguna

Evaluasi terhadap implementasi *front-end pada* modul kepegawaian UIIGateway dilakukan untuk menilai dampaknya terhadap pengalaman pengguna selama berinteraksi dengan sistem. Fokus evaluasi mencakup kemudahan penggunaan, konsistensi antarmuka, responsivitas aplikasi, kejelasan umpan balik sistem, serta dampak perubahan pola pemanggilan API terhadap kenyamanan dan rasa aman pengguna.

Peningkatan konsistensi antarmuka yang telah diterapkan, seperti penyelarasan gaya komponen, penggunaan skema warna yang seragam dengan identitas visual UIIGateway, serta penyesuaian tipografi dan *spacing*, memberikan dampak positif terhadap kemudahan navigasi. Pengguna dapat berpindah antar halaman dan modul tanpa mengalami perubahan tampilan yang kontras, sehingga beban kognitif berkurang dan pengguna lebih cepat beradaptasi dengan berbagai fitur dalam ekosistem kepegawaian UIIGateway.

Dari sisi responsivitas, penerapan teknik pemuatan bertahap seperti *lazy loading* pada level modul dan komponen membantu mengurangi waktu muat awal aplikasi. UIIGateway sebagai *wrapper* hanya memuat modul sesuai hak akses pengguna, sehingga halaman *login* dan halaman beranda dapat ditampilkan lebih cepat meskipun sistem memiliki kompleksitas tinggi. Pendekatan ini membuat interaksi awal pengguna dengan sistem terasa lebih ringan dan tidak mengintimidasi, terutama bagi pengguna yang mengakses sistem secara rutin dalam konteks pekerjaan harian.

Implementasi animasi pemuatan dinamis berbasis data *offset* pada modul UIIKaryasiswa berdampak signifikan terhadap persepsi pengguna terhadap performa sistem. Pada proses sinkronisasi data dosen dalam jumlah besar, antarmuka tidak lagi terlihat seolah membeku, melainkan menampilkan progres sinkronisasi secara bertahap disertai informasi jumlah data

yang telah tersinkron. Pengguna dapat memahami bahwa proses masih berjalan dan memperkirakan durasi yang dibutuhkan, sehingga kecenderungan untuk melakukan *refresh* berulang atau mengulangi aksi sinkronisasi dapat diminimalkan.

Pada komponen tabel penilaian kinerja di modul UIIKepegawaian, penerapan validasi *input*, perhitungan nilai otomatis, dan pembaruan tampilan berdasarkan kondisi data memberikan pengalaman interaksi yang lebih intuitif. Penilai dapat langsung melihat total nilai dan rata-rata tanpa menghitung manual, serta memahami konsekuensi logis dari nilai yang dimasukkan, misalnya perubahan status hasil penilaian atau kewajiban mengisi keterangan tambahan. Hal ini membantu mengurangi kesalahan *input* sekaligus memastikan data yang tersimpan tetap selaras dengan aturan bisnis yang berlaku.

Migrasi pola pemanggilan API dari *path parameter* ke *query parameter* juga memberi dampak tidak langsung terhadap pengalaman pengguna melalui peningkatan aspek keamanan dan stabilitas sistem. Meskipun perubahan ini tidak selalu terlihat di lapisan antarmuka, sistem yang lebih aman dan konsisten mengurangi risiko gangguan layanan akibat insiden keamanan, sehingga pengguna dapat menggunakan modul-modul kepegawaian dengan lebih tenang dan percaya bahwa data mereka dikelola secara bertanggung jawab.

Secara keseluruhan, rangkaian implementasi *front-end* yang dilakukan pada modul-modul kepegawaian UIIGateway berkontribusi nyata terhadap peningkatan kualitas pengalaman pengguna. Penekanan pada konsistensi desain, efisiensi pemuatan data, kejelasan umpan balik sistem, dan peningkatan standar keamanan menunjukkan bahwa peran pengembangan *front-end* tidak hanya sebatas menyajikan tampilan, tetapi juga berperan penting dalam memastikan sistem dapat digunakan secara efektif, nyaman, dan sesuai dengan kebutuhan operasional pengguna.

3.5 Integrasi *Generate* Dokumen Digital Secara Dinamis

Bagian ini membahas implementasi integrasi proses *generate* dokumen digital secara dinamis pada UIIGateway sebagai bagian dari pengembangan *front-end* modul kepegawaian. Proses *generate* dokumen merupakan salah satu alur bisnis penting dalam sistem kepegawaian, khususnya dalam pembuatan dokumen administratif seperti Surat Keputusan (SK) dan dokumen pendukung lainnya.

Integrasi dilakukan dengan memanfaatkan dua pendekatan teknologi, yaitu Jaspersoft Studio dan Docxtemplater, yang masing-masing memiliki karakteristik dan mekanisme pemrosesan yang berbeda. Pada implementasi ini, *front-end* UIIGateway berperan aktif dalam

penyusunan dan pengelolaan *template* dokumen, pemetaan variabel dinamis, serta pemicu proses *generate* dokumen melalui RESTful API. Pembahasan pada subbab ini difokuskan pada

1. Peran *front-end* dalam penyusunan *template* dokumen digital.
2. Alur integrasi *generate* dokumen menggunakan Jaspersoft Studio.
3. Alur integrasi *generate* dokumen menggunakan Docxtemplater.
4. Perbedaan pendekatan dan implikasi implementasi terhadap fleksibilitas antarmuka.

Dengan pendekatan ini, proses *generate* dokumen dapat dilakukan secara dinamis dan terintegrasi langsung dari antarmuka UIIGateway tanpa memerlukan interaksi manual di sisi pengguna, sehingga meningkatkan efisiensi dan konsistensi proses administrasi kepegawaian.

3.5.1 Integrasi *Generate* Dokumen Digital Menggunakan Jaspersoft Studio

Integrasi proses *generate* dokumen digital menggunakan Jaspersoft Studio pada UIIGateway diterapkan untuk mendukung kebutuhan pembuatan dokumen kepegawaian yang bersifat formal dan terstruktur, seperti Surat Keputusan (SK), surat penugasan, dan dokumen administratif lainnya. Jaspersoft Studio dipilih karena kemampuannya dalam menghasilkan dokumen dengan format yang konsisten serta mendukung pemetaan data berbasis *query* basis data secara langsung.

Pada implementasi ini, *front-end* UIIGateway berperan aktif dalam penyusunan dan konfigurasi *template* dokumen, sementara *back-end* berfungsi sebagai penyedia data dan eksekutor proses *generate* laporan. Pemisahan peran ini memungkinkan *front-end* memiliki kendali terhadap struktur tampilan dokumen tanpa harus melakukan perubahan langsung pada logika *back-end*.

Proses integrasi dimulai dari perancangan *template* dokumen oleh *front-end* menggunakan Jaspersoft Studio. *Template* tersebut dirancang dengan memperhatikan format dokumen resmi universitas, seperti tata letak *header, footer*, penempatan logo, format tanggal, serta struktur isi dokumen. *Front-end* kemudian menentukan variabel-variabel yang akan diisi secara dinamis berdasarkan data kepegawaian, seperti nama pegawai, nomor induk, jabatan, unit kerja, dan periode berlaku dokumen.

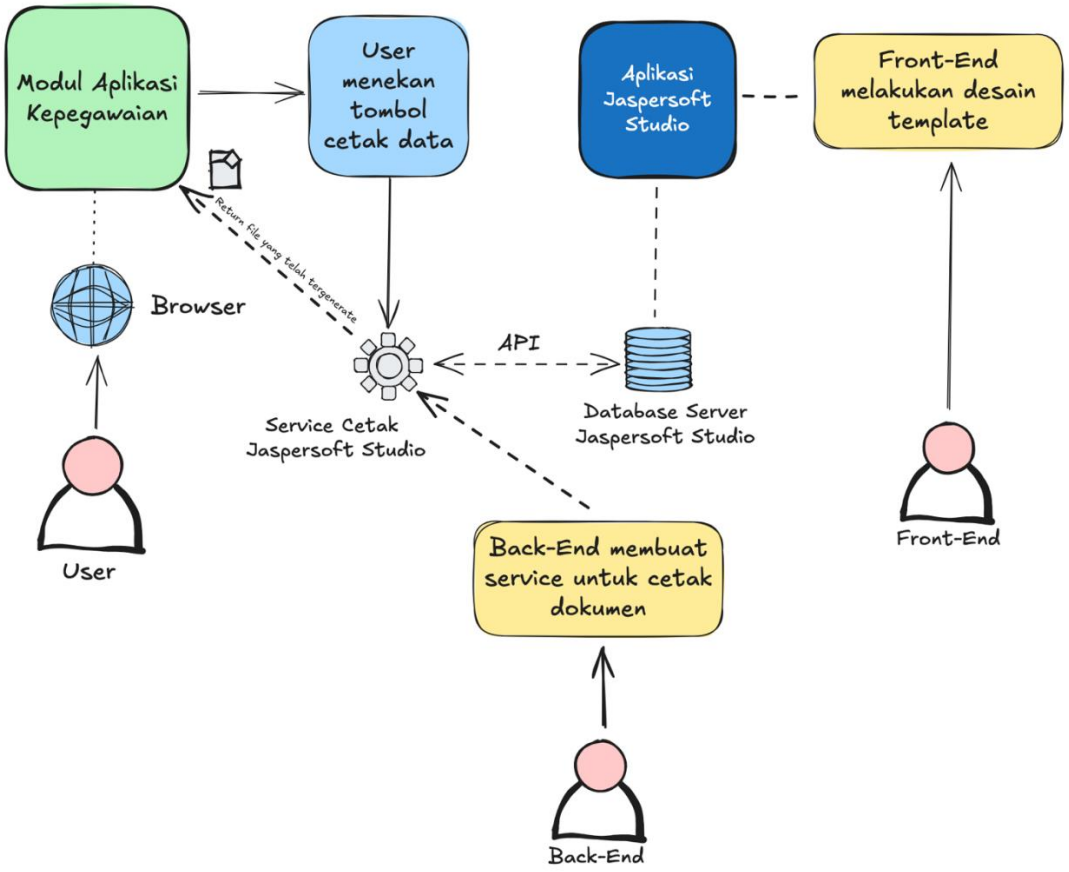
Untuk memastikan kesesuaian data yang digunakan dalam *template*, *front-end* berkoordinasi dengan *back-end* dalam menentukan *query* SQL yang diperlukan untuk mengambil data dari basis data sistem kepegawaian. *Query* tersebut disusun oleh *back-end*

sesuai dengan kebutuhan data yang telah didefinisikan pada *template* Jaspersoft Studio. Dengan pendekatan ini, *front-end* dapat memastikan bahwa setiap *field* pada *template* memiliki pasangan data yang sesuai dan konsisten.

Setelah *template* dan *query* data siap, *back-end* menyediakan *endpoint* RESTful API khusus untuk proses *generate* dokumen Jaspersoft Studio. *Endpoint* ini bertugas menerima parameter yang diperlukan dari *front-end*, mengeksekusi proses *generate* laporan berdasarkan *template* yang telah ditentukan, serta mengembalikan hasil *generate* dokumen dalam bentuk *binary large object* (blob), umumnya berupa file PDF.

Front-end UIIGateway kemudian memicu proses *generate* dokumen melalui pemanggilan API tersebut berdasarkan interaksi pengguna, misalnya ketika pengguna menekan tombol *Generate* Dokumen pada antarmuka. Setelah *back-end* mengembalikan respon blob, *front-end* menangani proses unduh atau pratinjau dokumen secara langsung pada sisi klien tanpa memerlukan pemrosesan tambahan dari pengguna.

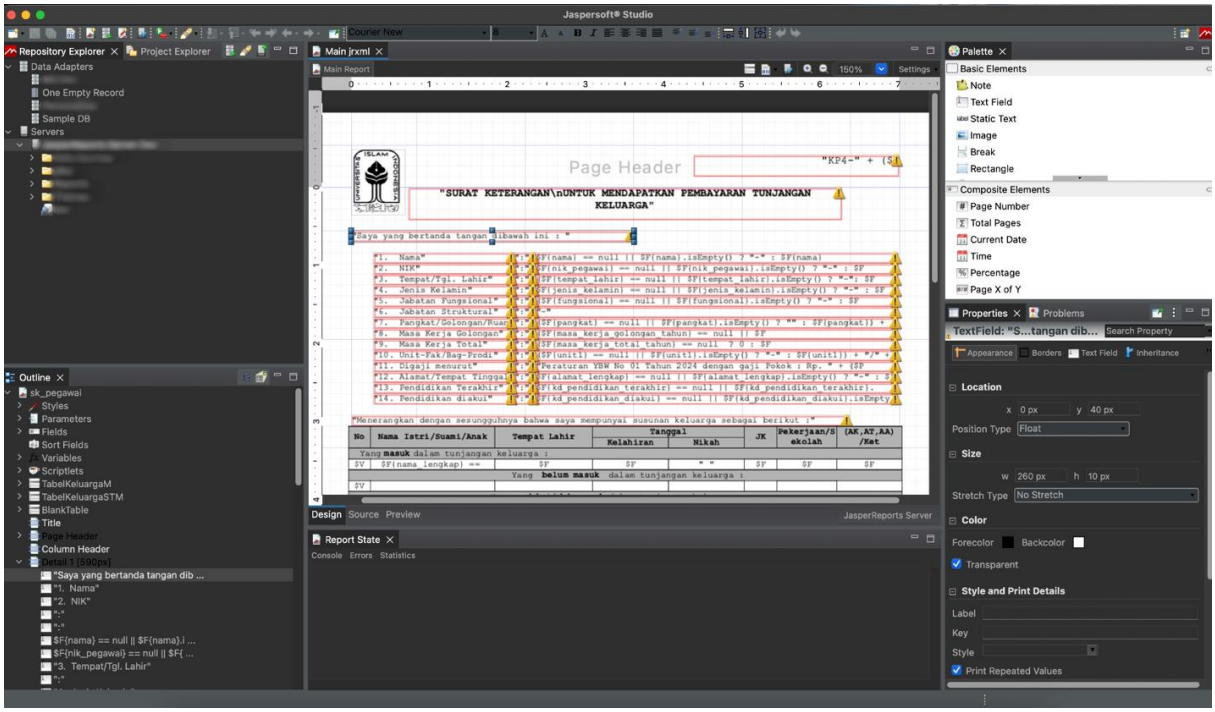
Alur proses *generate* dokumen digital menggunakan Jaspersoft Studio secara keseluruhan ditunjukkan pada Gambar 3.24, yang menggambarkan tahapan mulai dari penyusunan *template* oleh *front-end*, pengambilan data melalui *back-end*, hingga dokumen berhasil dihasilkan dan ditampilkan kepada pengguna.



Gambar 3.24 Alur Proses Cetak Dokumen Jaspersoft Studio

Pendekatan ini memberikan beberapa keuntungan dalam pengembangan sistem. Pertama, *front-end* memiliki fleksibilitas tinggi dalam mengatur tampilan dokumen tanpa harus melakukan perubahan berulang pada *back-end*. Kedua, konsistensi format dokumen dapat dijaga karena *template* dikelola secara terpusat. Ketiga, beban pemrosesan dokumen tetap berada pada *back-end*, sehingga *front-end* hanya berperan sebagai pemicu proses dan penyaji hasil.

Sebagai pendukung proses perancangan *template* dokumen, Jaspersoft Studio menyediakan antarmuka visual yang memudahkan *front-end* dalam menyusun struktur laporan secara terperinci. Tampilan lingkungan kerja Jaspersoft Studio yang digunakan dalam penelitian ini ditunjukkan pada Gambar 3.25, yang memperlihatkan area desain laporan (*design view*), struktur elemen laporan, serta konfigurasi parameter dan *field* data yang akan diisi secara dinamis. Melalui antarmuka ini, *front-end* dapat memastikan bahwa tata letak dokumen telah sesuai dengan format resmi universitas sebelum digunakan pada proses *generate* dokumen.



Gambar 3.25 *Screenshot* Tampilan Jaspersoft Studio

Selain itu, contoh *file template* dokumen Jaspersoft Studio yang berhasil di *generate* dapat dilihat pada Gambar 3.26. *File template* tersebut berisi definisi elemen laporan seperti parameter *input*, *field* data hasil *query*, serta pengaturan *layout* halaman yang menjadi acuan *back-end* dalam mengeksekusi proses *generate* dokumen. Keberadaan *file template* ini menjadi penghubung antara struktur tampilan yang dirancang oleh *front-end* dan data yang disediakan oleh *back-end*, sehingga proses *generate* dokumen dapat berjalan secara konsisten dan terstandar.

KP4-PEG.TETAP 2026

SURAT KETERANGAN
UNTUK MENDAPATKAN PEMBAYARAN TUNJANGAN KELUARGA

Saya yang bertanda tangan dibawah ini :

1. Nama : Abhirama Aji M
2. NIK : 255230103
3. Tempat/Tgl. Lahir : Kota Yogyakarta/2003-03-16
4. Jenis Kelamin : L
5. Jabatan Fungsional : -(TMT: -)
6. Jabatan Struktural : -
7. Pangkat/Golongan/Ruang : ,
8. Masa Kerja Golongan : 0 tahun 0 bulan (TMT: -)
9. Masa Kerja Total : 0 Th 0 Bl (per: -)
10. Unit-Fak/Bag-Prodi : -/Informatika S1
11. Digaji menurut : Peraturan YBW No 01 Tahun 2024 dengan gaji Pokok : Rp. 0,00
12. Alamat/Tempat Tinggal : -
13. Pendidikan Terakhir : S2
14. Pendidikan diakui : S2

Menerangkan dengan sesungguhnya bahwa saya mempunyai susunan keluarga sebagai berikut :

No	Nama Istri/Suami/Anak	Tempat Lahir	Tanggal		JK	Pekerjaan/Sekolah	(AK,AT,AA)/Ket
			Kelahiran	Nikah			
Yang **masuk** dalam tunjangan keluarga :							
1							
Yang **belum masuk** dalam tunjangan keluarga :							
1							
Yang **sudah tidak masuk** dalam tunjangan keluarga :							
1							
Keterangan :		• **Suami/Istri bekerja di UII (sebagai pegawai tetap), Tunjangan di tanggung oleh Suami** • **Jumlah anak maksimal yang masuk dalam tunjangan adalah 3 orang anak**					

Jumlah keluarga yang masuk dalam tunjangan : Anak = 0 orang, Suami/Istri = 0 orang, Lainnya = 0 orang, Total = 0 orang.

Keterangan ini saya buat dengan sesungguhnya dan apabila keterangan ini ternyata tidak benar (palsu) saya bersedia dituntut di muka pengadilan berdasarkan Undang-undang yang berlaku, dan bersedia mengembalikan semua penghasilan yang telah saya terima yang seharusnya bukan menjadi hak saya.

Yogyakarta,_____

(Abhirama Aji M, S.Kom., M.Kom.)

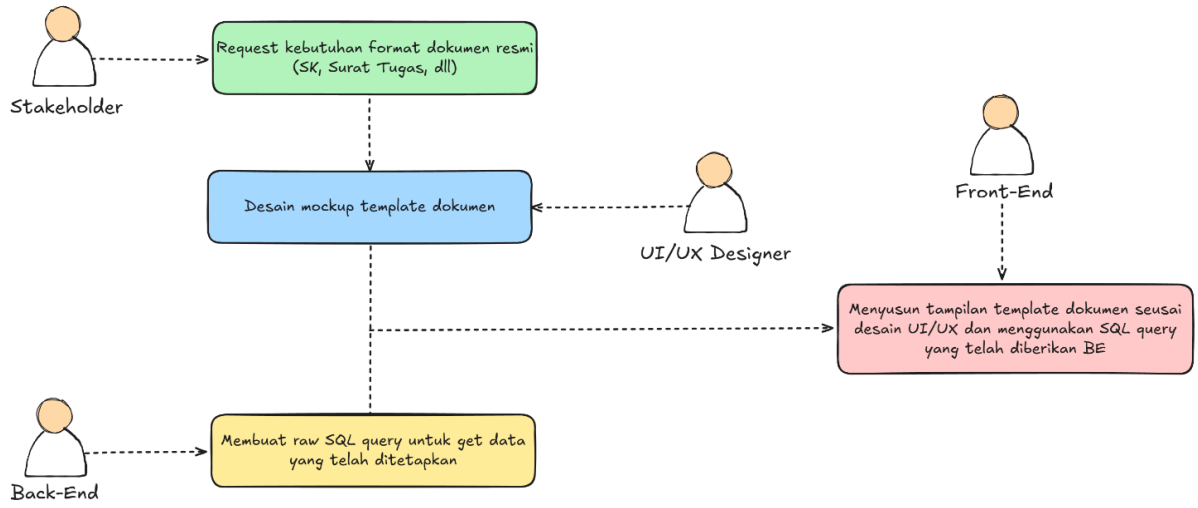
Catatan :
- **Perubahan data tanggungan (Anak/Istri) harap menyertakan lampirannya (Akte Kelahiran/Surat Nikah) atau menyertakan Fotocopy kartu Keluarga yang terbaru**
- **Anak yang berumur *diatas 21 Tahun dan dibawah 25 Tahun* yang belum menikah, masih menjadi tanggungan dan masih sekolah/kuliah, harap menyertakan fotocopy keterangan masih sekolah/ kuliah atau fotocopy KTM yang masih berlaku.**
- AK = Anak Kandung
- AT = Anak Tiri
- AA = Anak Angkat

Gambar 3.26 Contoh Dokumen Hasil *Generate* JasperSoft Studio

Dengan integrasi Jaspersoft Studio ini, UIIGateway mampu menyediakan layanan *generate* dokumen digital yang andal, konsisten, dan sesuai dengan standar administrasi universitas, sekaligus mendukung efisiensi kerja pengguna dalam pengelolaan dokumen kepegawaian.

Evaluasi Integrasi *Generate* Dokumen Digital Menggunakan Jaspersoft Studio

Evaluasi terhadap proses integrasi Jaspersoft Studio pada ekosistem UIIGateway menunjukkan bahwa pemisahan tanggung jawab antara perancangan visual dokumen di sisi *frontend* dan pemrosesan data di sisi *backend* telah meningkatkan efisiensi produksi dokumen administratif universitas secara signifikan. Siklus pengembangan fitur ini diawali dari tahap *request* oleh *stakeholder* (bagian kepegawaian) yang memerlukan standarisasi format dokumen resmi guna meminimalisasi kesalahan input manual, yang kemudian diterjemahkan oleh *UI/UX Designer* ke dalam rancangan antarmuka pratinjau dokumen agar selaras dengan pengalaman pengguna di modul lainnya. Pada tahap implementasi, penulis selaku *Front-end Developer* melakukan perancangan *template* .jrxml secara presisi di Jaspersoft Studio, yang mencakup pengaturan tata letak dinamis dan pemetaan variabel, sembari berkoordinasi intensif dengan *backend* untuk penyelarasan *query* SQL agar data yang ditarik dari basis data sinkron dengan *field* yang didefinisikan pada *template*. Tahap *evaluasi* akhir dilakukan melalui pengujian fungsional untuk memastikan bahwa dokumen PDF yang dihasilkan memiliki integritas visual yang konsisten, tidak mengalami regresi tata letak pada berbagai panjang data, serta dapat diunduh secara lancar sebagai *binary large object* (blob), adapun mekanisme koordinasi lintas unit dalam merancang hingga mengeksekusi dokumen digital ini dipaparkan secara sistematis pada Gambar 3.27.



Gambar 3.27 Alur Kolaborasi dalam Integrasi Jaspersoft

Pelajaran berharga yang penulis dapatkan secara spesifik dari tugas ini adalah penguasaan kompetensi baru dalam merancang laporan berbasis XML menggunakan Jaspersoft Studio,

yang menuntut ketelitian tinggi dalam menyelaraskan struktur elemen visual dengan parameter data dinamis. Penulis belajar bahwa kolaborasi teknis antara FE dan BE dalam menentukan kontrak *query* SQL sangat krusial untuk menghindari ketidakcocokan tipe data yang dapat menyebabkan kegagalan proses *generate* di sisi server. Selain itu, penulis memperoleh wawasan teknis mengenai penanganan *file stream* pada sisi klien, di mana pengelolaan memori saat melakukan *preview* dokumen PDF berukuran besar menjadi aspek penting untuk menjaga performa peramban tetap optimal. Pengalaman ini memperkuat pemahaman penulis bahwa peran pengembang *frontend* di BSI UII tidak hanya terbatas pada estetika aplikasi web, tetapi juga mencakup standarisasi dokumen legal universitas melalui penggabungan kemahiran desain visual dan logika integrasi data yang kompleks.

3.5.2 Integrasi *Generate* Dokumen Digital Menggunakan Docxtemplater

Selain menggunakan Jaspersoft Studio, UIIGateway juga mengimplementasikan mekanisme *generate* dokumen digital menggunakan Docxtemplater sebagai solusi alternatif dalam pembuatan dokumen kepegawaian. Penerapan Docxtemplater dilatarbelakangi oleh keterbatasan yang ditemukan pada Jaspersoft Studio, khususnya terkait perbedaan margin dan *spacing* pada hasil cetak dokumen yang terkadang tidak sepenuhnya konsisten dengan tampilan desain pada mode *design view*. Kondisi tersebut berpotensi menimbulkan ketidaksesuaian format dokumen resmi apabila digunakan secara langsung tanpa penyesuaian tambahan.

Docxtemplater dipilih karena kemampuannya dalam mempertahankan format dokumen sesuai dengan *template* Microsoft Word (.docx) secara presisi. Pada Docxtemplater, margin, jarak antar paragraf, tabel, dan elemen *layout* lainnya akan mengikuti sepenuhnya desain yang telah ditentukan pada *template* dokumen, sehingga risiko pergeseran tata letak dapat diminimalkan. Namun demikian, Docxtemplater memiliki keterbatasan dalam format keluaran, yaitu hanya menghasilkan *file* .docx. Oleh karena itu, *back-end* UIIGateway menyediakan layanan tambahan untuk melakukan konversi dokumen .docx ke format lain yang dibutuhkan pengguna, seperti PDF.

Berbeda dengan Jaspersoft Studio yang proses *generate* dokumennya sepenuhnya dilakukan di sisi *back-end*, integrasi Docxtemplater ditempatkan pada sisi *front-end* UIIGateway sebagai *library* berbasis TypeScript. Dalam skema ini, *front-end* terlebih dahulu melakukan pemanggilan RESTful API ke *back-end* untuk memperoleh data kepegawaian yang diperlukan. Data tersebut kemudian dipetakan dan diolah langsung oleh *front-end* ke dalam

template dokumen .docx menggunakan Docxtemplater, sehingga proses pengisian konten dokumen dilakukan secara dinamis di sisi klien.

Template dokumen Docxtemplater dirancang menggunakan format variabel berbasis kurung kurawal tunggal, seperti {nama_pegawai}, {no_sk}, dan {jabatan}, yang akan digantikan secara otomatis dengan nilai data dari hasil respon API. Selain variabel statis, Docxtemplater juga mendukung fitur templating lanjutan berupa perulangan (looping) dan kondisi logika (conditional rendering). Contoh penerapan perulangan data objek dituliskan dalam format {#users}{nama_user}{/users}, yang memungkinkan penulisan daftar data secara dinamis di dalam dokumen, seperti daftar anggota tim atau riwayat penugasan.

Selain itu, Docxtemplater menyediakan dukungan kondisi bersyarat menggunakan sintaks {#if kondisi} dan {#else}, misalnya {#if user.isVerified} untuk menampilkan bagian tertentu dari dokumen hanya apabila kondisi terpenuhi. Fitur ini sangat berguna dalam pembuatan dokumen kepegawaian yang memiliki variasi isi, seperti perbedaan konten berdasarkan status pegawai, hasil evaluasi, atau jenis keputusan administratif yang diterbitkan.

Alur proses *generate* dokumen menggunakan Docxtemplater dimulai dari interaksi pengguna pada antarmuka UIIGateway, dilanjutkan dengan pemanggilan API *back-end* untuk mengambil data yang diperlukan. Data tersebut kemudian diproses oleh *front-end* dan dipetakan ke dalam *template* .docx menggunakan Docxtemplater. Setelah dokumen berhasil dihasilkan, file dapat langsung diunduh oleh pengguna atau dikirim kembali ke *back-end* untuk dikonversi ke format lain sesuai kebutuhan. Ilustrasi contoh *template* dokumen .docx dengan variabel Docxtemplater serta alur proses *generate* dokumen ditampilkan pada Gambar 3.28 dan Gambar 3.29.

Lampiran Surat Keputusan Pengurus Yayasan Badan Wakaf UII
Nomor [redacted], tanggal 04 Agustus 2023 M / 17 Muharam 1445 H
Tentang : Kenaikan Jabatan Fungsional Dosen Tetap Ull

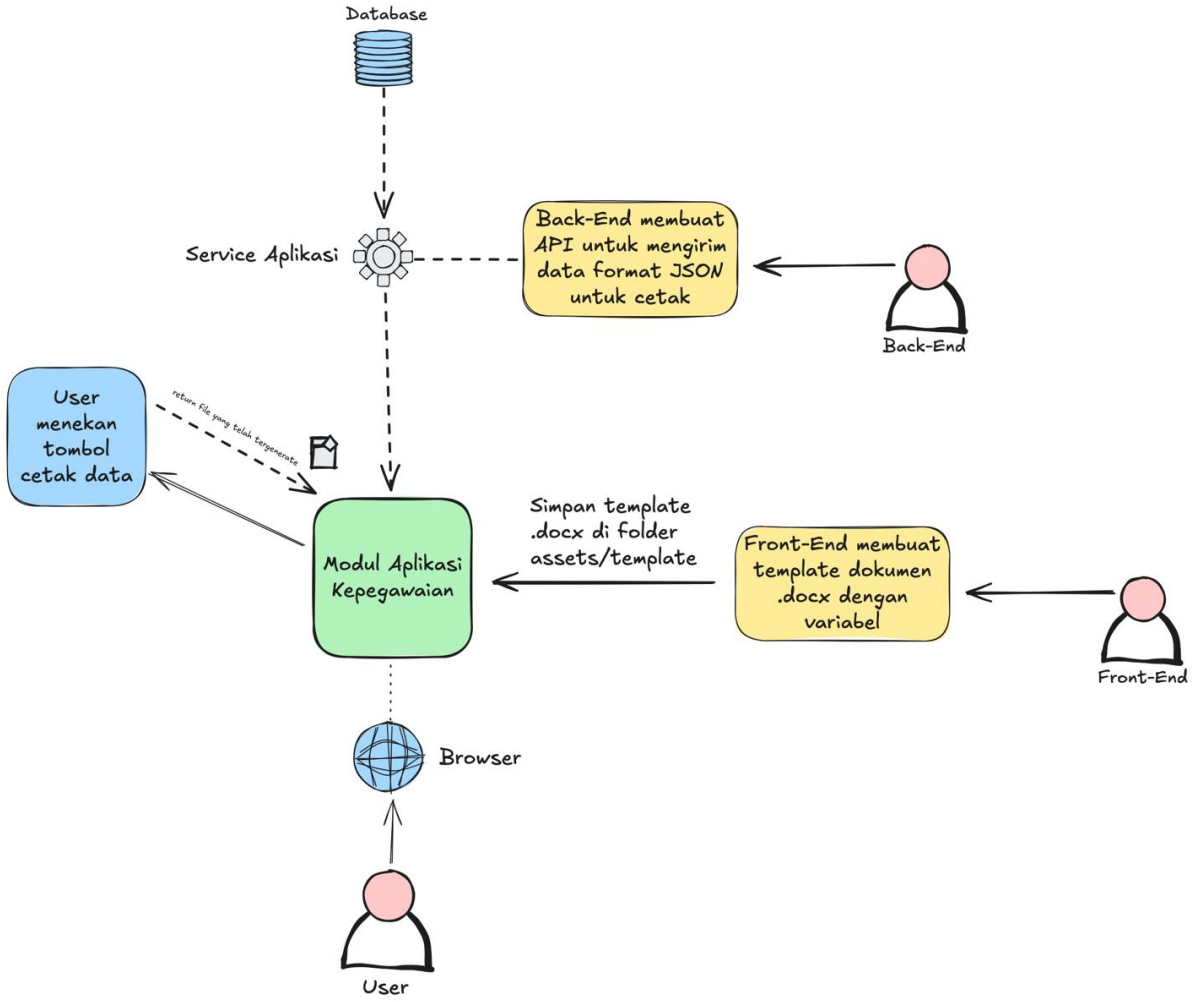
1.	Nama	: {nama_gelar}
2.	NIP / NIK	: {nip_nik}
3.	Tempat dan tanggal lahir	: {tempat_tgl_lahir}
LAMA		
4.	Jabatan Fungsional	: {jabatan_fungsional_lama}
5.	Angka Kredit	:{angka_kredit_lama}
6.	Tunjangan Fungsional	: {tunjangan_fungsional_lama}
BARU		
7.	Jabatan Fungsional	: **{jabatan_fungsional_baru}**
8.	Angka Kredit	: :{angka_kredit_baru}
9	Tunjangan Fungsional	: {tunjangan_fungsional_baru}
10.	Terhitung Mulai Tanggal (TMT)	: {tmt_jabatan_fungsional_baru}
11.	Status	: {kelompok_pegawai}
12.	Keterangan	:

Ditetapkan di : Yogyakarta
{tgl_terbit_sk}
Pada tanggal : ----------
{tgl_terbit_sk_hijri}

Pengurus Yayasan Badan Wakaf UII
Ketua Umum,

{ketua_umum}

Gambar 3.28 Contoh *Template* Dokumen Docxtemplater



Gambar 3.29 Alur Proses *Generate* Dokumen Docxtemplater

Pendekatan ini memberikan beberapa keuntungan dalam pengembangan sistem. Pertama, *front-end* memiliki kendali penuh terhadap struktur dan isi dokumen tanpa ketergantungan langsung pada logika *generate* di *back-end*. Kedua, konsistensi tata letak dokumen dapat dijaga sesuai dengan desain awal *template* Word. Ketiga, fleksibilitas dalam pengelolaan variasi konten dokumen meningkat berkat dukungan logika *templating* seperti perulangan dan kondisi.

Dengan mengombinasikan Jaspersoft Studio dan Docxtemplater, UIIGateway mampu menyediakan mekanisme *generate* dokumen digital yang adaptif dan sesuai dengan kebutuhan operasional kepegawaian. Jaspersoft Studio digunakan untuk dokumen yang membutuhkan integrasi kuat dengan basis data dan format laporan terstruktur, sedangkan Docxtemplater dimanfaatkan untuk dokumen yang menuntut presisi tata letak serta fleksibilitas pengelolaan konten pada sisi *front-end*.

Evaluasi Integrasi *Generate* Dokumen Digital Menggunakan Docxtemplater

Evaluasi terhadap penerapan Docxtemplater pada ekosistem UIIGateway menunjukkan bahwa solusi ini berhasil menutupi celah teknis yang ada pada Jaspersoft Studio, khususnya terkait presisi tata letak (*layout*) dan margin pada dokumen format .docx. Siklus pengembangan fitur ini dimulai dari tahap *request*, di mana *stakeholder* memerlukan dokumen yang memiliki fleksibilitas tinggi dalam penyuntingan manual pasca-cetak namun tetap mempertahankan standar format Microsoft Word. Tahap implementasi dilakukan sepenuhnya di sisi *front-end* menggunakan *library* TypeScript, di mana penulis merancang *tagging* variabel, logika perulangan (*looping*), serta kondisi (*conditional rendering*) langsung pada *template* Word. Tahap evaluasi dilakukan dengan memvalidasi keakuratan hasil pemetaan data dari API ke dalam dokumen serta memastikan proses konversi dari .docx ke PDF di sisi *backend* berjalan tanpa merusak struktur elemen visual

Pelajaran berharga yang penulis dapatkan secara spesifik dari tugas ini adalah pemahaman mendalam mengenai *Client-Side Document Processing*. Penulis belajar bahwa memindahkan beban kerja *generation* dokumen ke sisi *front-end* memberikan kebebasan penuh dalam mengatur logika tampilan tanpa perlu sering mengubah *codebase* di sisi *backend*. Secara teknis, penulis mengasah kemahiran dalam mengelola struktur data JSON yang kompleks untuk dipetakan ke dalam format XML pada dokumen Word, serta menangani skenario logika kondisional yang rumit—seperti menampilkan paragraf tertentu hanya jika pegawai memiliki status khusus. Pengalaman ini juga memberikan wawasan tentang pentingnya memilih *tool* yang tepat sesuai kebutuhan: Jaspersoft untuk laporan data masif dan terstruktur, serta Docxtemplater untuk dokumen yang mengutamakan presisi tata letak dan fleksibilitas konten di sisi pengguna.

3.6 Pengujian dan Validasi Fungsional *Front-end* UIIGateway

Pengujian dan validasi fungsional *front-end* UIIGateway dilakukan untuk memastikan bahwa seluruh fitur antarmuka yang dikembangkan dapat berjalan sesuai dengan kebutuhan fungsional sistem serta mampu berinteraksi dengan layanan *back-end* secara benar dan konsisten. Tahapan pengujian ini menjadi bagian penting dalam proses pengembangan karena UIIGateway berperan sebagai *wrapper* utama yang menjadi titik awal akses pengguna terhadap seluruh modul aplikasi, termasuk modul kepegawaian.

Pengujian fungsional dilakukan secara berlapis dan melibatkan lebih dari satu peran dalam tim pengembangan. Pada tahap awal, pengujian dilakukan oleh *front-end* developer sebagai

bagian dari proses pengembangan harian. Pengujian ini bertujuan untuk memastikan bahwa fitur yang diimplementasikan telah sesuai dengan spesifikasi kebutuhan, bebas dari kesalahan logika antarmuka, serta mampu menangani berbagai skenario interaksi pengguna secara benar.

Pengujian oleh *front-end* developer difokuskan pada pendekatan *black-box testing*, di mana validasi dilakukan berdasarkan keluaran sistem dan respons antarmuka terhadap *input* pengguna, tanpa menguji implementasi internal *back-end* secara langsung. Ruang lingkup pengujian meliputi proses autentikasi pengguna pada halaman login UIIGateway, pemuatan menu dan modul berdasarkan hak akses, pengelolaan data kepegawaian, validasi *form*, serta mekanisme sinkronisasi data berbasis *offset*.

Setelah pengujian awal oleh *front-end* developer dinyatakan selesai, aplikasi selanjutnya diuji kembali oleh *UI/UX Designer* pada *environment* staging. Pada konteks ini, *UI/UX Designer* tidak hanya berperan dalam perancangan antarmuka, tetapi juga menjalankan fungsi pengujian sebagai *quality control* terhadap aspek kegunaan (*usability*), konsistensi visual, serta kesesuaian implementasi antarmuka dengan desain yang telah ditetapkan. Peran ini menjadi penting mengingat pada saat pelaksanaan magang, BSI UII belum memiliki tim *Quality Assurance* (QA) khusus.

Pengujian oleh *UI/UX Designer* mencakup validasi alur interaksi pengguna, keterbacaan informasi, konsistensi warna dan tipografi, serta kejelasan umpan balik visual yang ditampilkan sistem. Selain itu, *UI/UX Designer* juga melakukan simulasi penggunaan aplikasi dari sudut pandang pengguna akhir untuk mengidentifikasi potensi kendala atau ketidaksesuaian antarmuka sebelum aplikasi dirilis ke lingkungan produksi.

Selain pengujian fungsional, dilakukan pula pengujian kualitas antarmuka pada halaman *login* UIIGateway sebagai bagian dari *wrapper* aplikasi. Pengujian ini dilakukan menggunakan PageSpeed Insights sebagai inisiatif mandiri dari pengembang *front-end* untuk mengevaluasi performa, aksesibilitas, dan kepatuhan terhadap praktik terbaik pengembangan web modern. Pengujian difokuskan pada halaman *login* karena halaman tersebut merupakan titik awal pemuatan aplikasi dan memiliki pengaruh langsung terhadap persepsi awal pengguna terhadap sistem.

Seluruh hasil pengujian digunakan sebagai dasar evaluasi terhadap kualitas implementasi *front-end* UIIGateway. Temuan yang diperoleh dari pengujian oleh *front-end* developer maupun *UI/UX Designer* dimanfaatkan untuk melakukan perbaikan antarmuka, penyempurnaan alur interaksi pengguna, serta optimalisasi performa pemuatan aplikasi. Dengan pendekatan ini, proses pengujian dan validasi fungsional menjadi bagian integral dari

siklus pengembangan iteratif yang selaras dengan metode *Agile Scrum* yang diterapkan selama pelaksanaan magang.

3.6.1 Pengujian Kualitas Antarmuka Halaman *Login* UIIGateway

Selain pengujian fungsional, dilakukan pula pengujian kualitas antarmuka pada halaman *login* UIIGateway untuk mengevaluasi aspek performa, aksesibilitas, dan kepatuhan terhadap standar pengembangan web modern. Pengujian ini dilakukan menggunakan PageSpeed Insights yang mengintegrasikan Lighthouse sebagai alat analisis.

Penggunaan PageSpeed Insights dalam penelitian ini merupakan inisiatif mandiri dari *front-end developer* sebagai bagian dari upaya peningkatan kualitas antarmuka, dan bukan merupakan kebijakan wajib pengujian sistem secara keseluruhan di BSI UII. Pengujian difokuskan pada halaman *login* karena halaman tersebut merupakan komponen *wrapper* yang selalu dimuat pertama kali oleh pengguna, sehingga memiliki pengaruh langsung terhadap persepsi awal terhadap sistem UIIGateway.

Pengujian dilakukan dengan mengakses halaman *login* UIIGateway melalui peramban web pada kondisi jaringan normal. Metrik yang dianalisis meliputi *Performance*, *Accessibility*, *Best Practices*, dan *SEO*. Hasil pengujian menunjukkan bahwa halaman *login* UIIGateway telah memenuhi standar kualitas antarmuka yang baik, khususnya pada aspek kepatuhan praktik terbaik pengembangan web dan aksesibilitas. Ringkasan hasil pengujian kualitas antarmuka halaman *login* UIIGateway menggunakan PageSpeed Insights disajikan pada Tabel 3.4.

Tabel 3.4 Metrik Performa Hasil Pengujian Halaman *Login*

Kategori	**Skor**	**Metrik Utama**	**Nilai**
Performance	78	*First Contentful Paint* (FCP)	1.1 detik
Accessibility	91	*Largest Contentful Paint* (LCP)	2.5 detik
Best Practices	100	*Total Blocking Time* (TBT)	120 ms
SEO	83	*Cumulative Layout Shift* (CLS)	0.01

Hasil pengujian ini menunjukkan bahwa implementasi *front-end* UIIGateway sebagai *wrapper* aplikasi telah memperhatikan aspek kualitas teknis dan pengalaman pengguna sejak tahap awal pemuatan sistem. Meskipun demikian, hasil pengujian juga menjadi bahan evaluasi untuk perbaikan lanjutan, khususnya terkait optimasi performa pemuatan aset statis dan pengelolaan sumber daya *front-end*.

Pelajaran berharga yang penulis dapatkan secara spesifik dari pengujian mandiri ini adalah pentingnya peran proaktif seorang *Frontend Developer* dalam menjaga kualitas produk melalui *self-audit* secara berkala. Penulis belajar bahwa kualitas sebuah sistem tidak hanya diukur dari keberhasilan fungsionalitasnya, tetapi juga dari kemudahannya diakses oleh berbagai kalangan (*accessibility*) dan kecepatan respons antarmuka (*performance*). Melalui analisis metrik *Core Web Vitals*, penulis memperoleh pemahaman mendalam tentang bagaimana setiap milidetik pada *Total Blocking Time* (TBT) dapat memengaruhi pengalaman pengguna. Pengalaman ini membentuk pola pikir penulis untuk selalu mengintegrasikan langkah-langkah optimasi dan kepatuhan standar web sebagai bagian dari alur kerja pengembangan, guna memastikan bahwa **UIIGateway** memberikan kesan pertama yang profesional dan andal bagi seluruh civitas akademika sejak tahap login.

3.6.2 Proses *Deploy* dan Pembaruan Versi Modul *Front-end* UIIGateway

Proses *deploy* dan pembaruan versi modul *front-end* UIIGateway dilakukan oleh *front-end* sebagai bagian dari siklus pengembangan dan pengujian fungsional. Dalam konteks magang, pemagang atau *Junior Staff* berperan dalam melakukan pembaruan versi modul aplikasi ketika terdapat perubahan fitur atau perbaikan *bug*, namun tetap berada di bawah supervisi *Front-End Developer* tetap. Alur *deploy* ini memastikan bahwa setiap perubahan kode yang dilakukan pada modul *Human Capital Management* (HCM) seperti UIIInsani, UIIKepegawaian, UIIPortofolio, dan UIIKaryasiswa dapat diuji terlebih dahulu pada *environment* pengembangan sebelum diintegrasikan ke *environment* yang lebih tinggi.

Sebelum proses *deploy* dilakukan, setiap perubahan kode wajib didokumentasikan pada berkas CHANGELOG.md yang menjadi catatan resmi riwayat rilis modul. Berkas ini memuat daftar versi rilis beserta tanggal dan ringkasan perubahan yang dilakukan, misalnya penambahan fitur baru, perbaikan *bug*, maupun penyesuaian perilaku antarmuka. Contoh entri pada CHANGELOG.md terlihat pada Gambar 3.30.

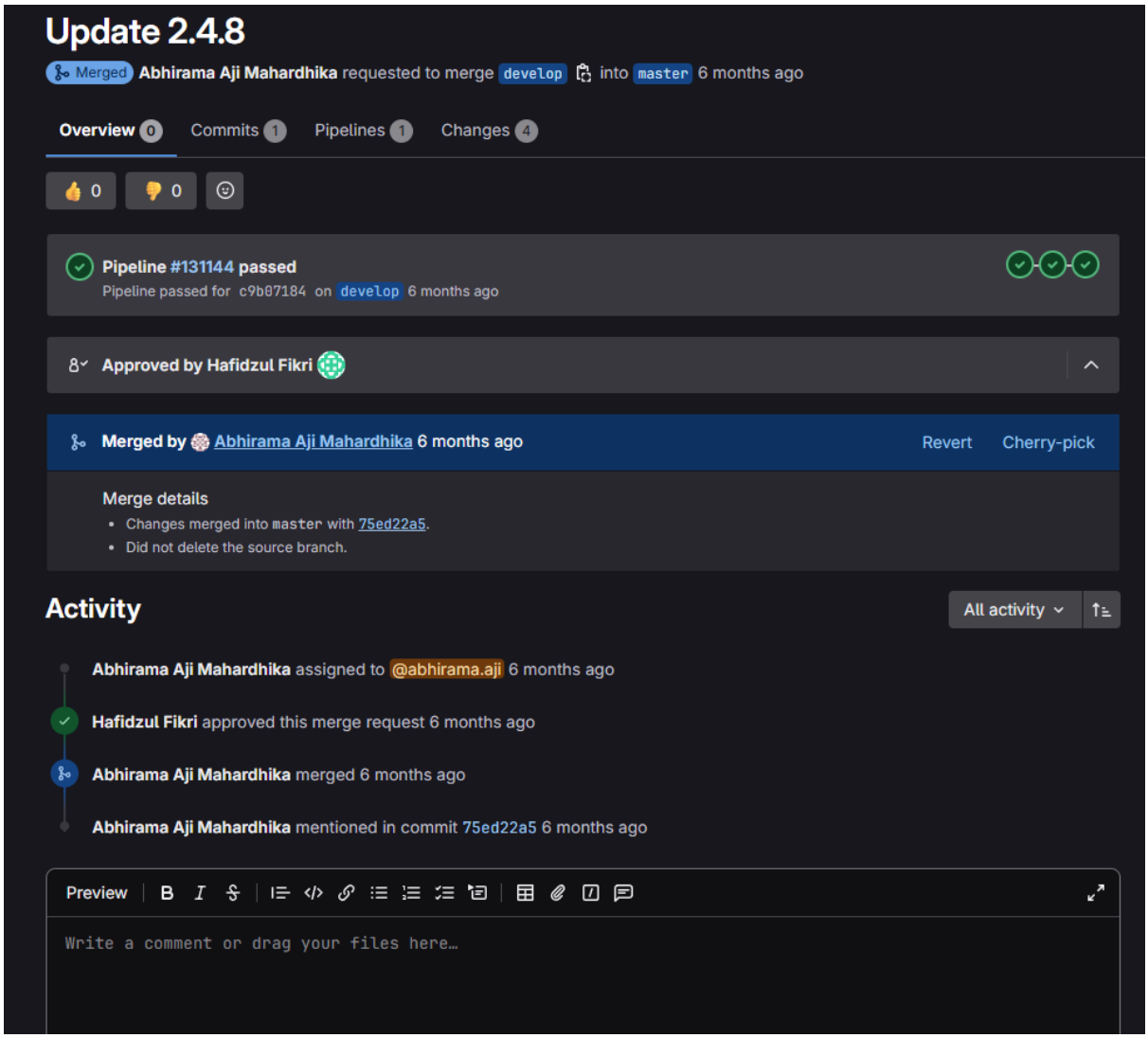
```
# Release
## 2.2.53 (2025-07-07)
- Fix insani personal table button auth
## 2.2.52 (2025-06-24)
- Add property advance search (rumpun ilmu, pohon ilmu, cabang ilmu, bidang ilmu)
- Add property export pdf advance search
- Fix property name to export pdf advance search
```

Gambar 3.30 Entri *History* Perubahan

Pencatatan seperti ini memudahkan tim untuk melacak perubahan yang terjadi pada setiap versi dan mengaitkan versi tertentu dengan fitur atau perbaikan yang diimplementasikan. Selain itu, dokumentasi CHANGELOG juga menjadi referensi penting ketika terjadi isu di *production*, karena pengembang dapat menelusuri perubahan apa saja yang masuk pada versi tertentu.

Deploy modul *front-end* diawali dengan melakukan pembaruan kode pada *branch* pengembangan (misalnya *branch feature* atau *develop*) di *repository* GitLab, kemudian menjalankan pipeline CI/CD yang telah dikonfigurasi oleh tim. Ketika seluruh tahapan *pipeline*, seperti instalasi dependensi, *build*, dan pengujian otomatis, selesai dengan status sukses, perubahan tersebut diajukan dalam bentuk *merge request* ke *branch* utama (master). Pada tahap ini, pemagang atau *Junior Staff* umumnya diberikan hak akses sebagai *Developer*, bukan *Maintainer*, sehingga tidak dapat menyetujui *merge* secara mandiri. Kode yang diajukan harus

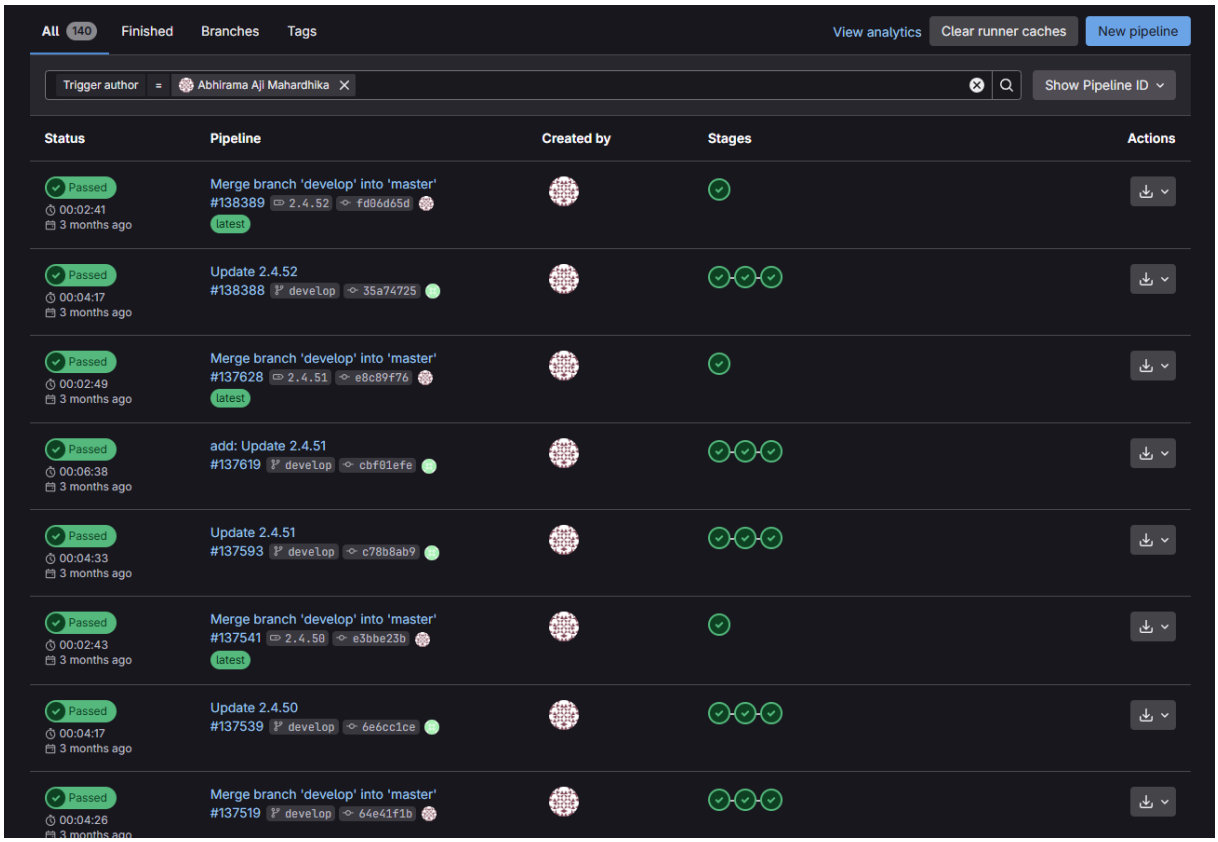
terlebih dahulu dilakukan *review* oleh mentor atau anggota tetap tim *front-end* yang memiliki hak akses lebih tinggi, untuk memastikan kualitas implementasi dan kesesuaian dengan standar proyek. Contoh tampilan *merge request* dan status *pipeline* di GitLab dapat dilihat pada Gambar 3.31.



Gambar 3.31 Tangkapan Layar *Merge Request*

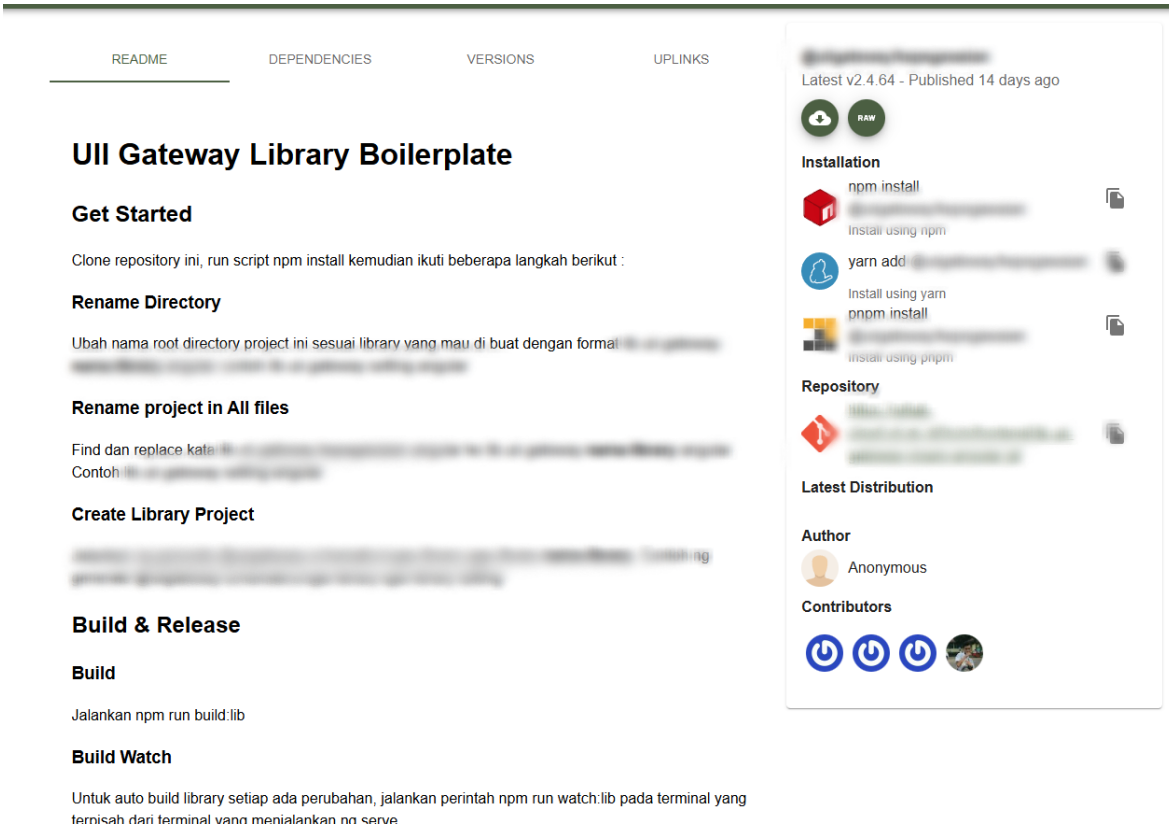
Setelah perubahan disetujui dan berhasil *merge* ke *branch* master, langkah berikutnya adalah pembuatan *tag* versi baru pada *repository* modul aplikasi. Pemberian *tag* ini mengacu pada versi yang sudah dicatat di CHANGELOG.md sehingga konsisten antara dokumentasi dan artefak rilis. Pembuatan tag akan memicu *pipeline* tambahan yang melakukan proses *build* dan *deploy* modul ke *environment library* (lib), yaitu *environment* khusus yang berfungsi mirip dengan *environment* pengembangan, tetapi hanya memuat modul aplikasi terkait. *Environment* ini memungkinkan tim untuk menguji modul secara terpisah sebelum modul tersebut digunakan

oleh *wrapper* UIIGateway. Status keberhasilan *pipeline deploy* dan informasi *tag* versi terbaru dapat diamati melalui antarmuka *GitLab* maupun halaman *monitoring* internal. Gambar 3.32 menunjukkan contoh daftar *tag* versi beserta status *pipeline* yang menyertainya.



Gambar 3.32 Tangkapan Layar *Status Pipeline*

Meskipun modul telah berhasil *deploy* pada *environment library*, perubahan tersebut belum otomatis muncul pada *environment* UIIGateway lainnya seperti *develop*, *staging*, maupun *production*. Agar modul versi terbaru digunakan oleh UIIGateway, perlu dilakukan pembaruan versi *library* pada berkas package.json milik *repository wrapper* UIIGateway. Pembaruan ini dilakukan dengan mengganti nomor versi modul aplikasi pada *dependency* yang relevan sesuai dengan *tag* versi yang telah sukses *deploy* di *environment library*. Sebelum melakukan perubahan, *front-end* developer memastikan terlebih dahulu bahwa versi tersebut sudah terdaftar dan dapat diakses melalui halaman internal yang menampilkan daftar seluruh *library* modul beserta informasi versinya. Contoh tampilan *library* modul aplikasi sespesifik dan versi yang digunakan ditunjukkan pada Gambar 3.33.

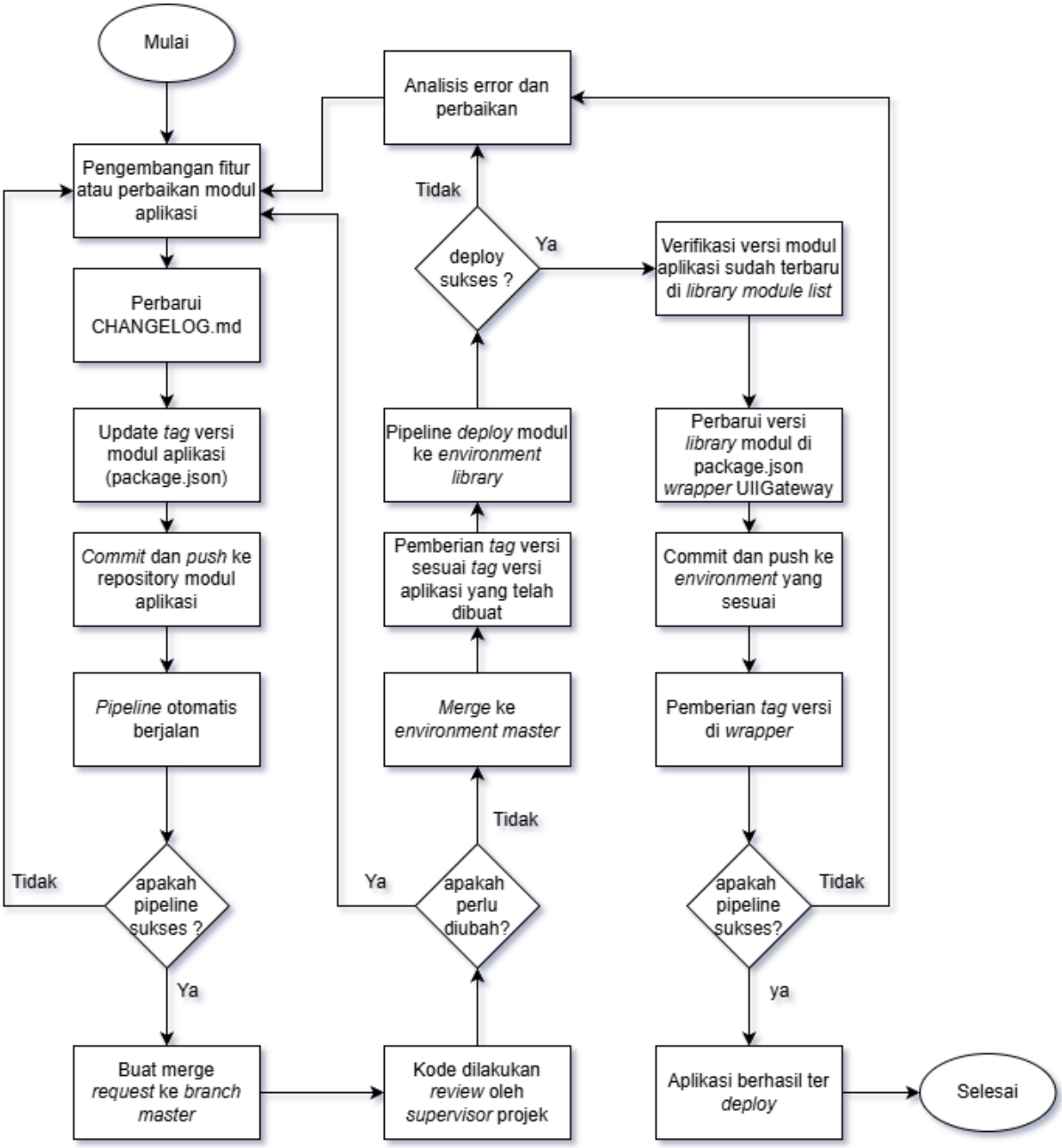


Gambar 3.33 Tangkapan Layar *Library* Modul Aplikasi Spesifik

Setelah package.json *wrapper* UIIGateway diperbarui, proses serupa kembali dilakukan yaitu, perubahan dikomit, diajukan melalui *merge request*, dilakukan *review*, kemudian dijalankan *pipeline* CI/CD untuk membangun dan mendistribusikan versi baru UIIGateway ke *environment develop*, *staging*, maupun *production* sesuai kebijakan tim. Secara keseluruhan, alur proses *deploy* dan pembaruan versi modul *front-end* yang dilakukan oleh tim dapat digambarkan dalam diagram alur pada Gambar 3.34 yang menunjukkan hubungan antara perubahan kode, pembaruan CHANGELOG, pembuatan *tag* versi, *deploy* ke *environment library*, pembaruan *dependency* pada *wrapper*, hingga rilis ke *environment* akhir. Pendekatan bertahap ini memastikan bahwa setiap rilis modul *front-end* dapat dilacak melalui versi yang jelas, diuji secara terpisah pada *environment library*, dan hanya diintegrasikan ke UIIGateway setelah mendapatkan persetujuan dari pengembang senior. Dengan demikian, proses *deploy* dan pembaruan versi yang dilakukan oleh tim *front-end* tidak hanya menjamin kelancaran integrasi teknis, tetapi juga mendukung praktik pengembangan perangkat lunak yang terkontrol, terdokumentasi, dan dapat diaudit.

Pelajaran berharga yang penulis dapatkan secara spesifik dari tugas manajemen rilis ini adalah pemahaman mendalam mengenai *DevOps Culture* dan pentingnya disiplin versi

(*versioning discipline*) dalam proyek skala besar. Penulis belajar bahwa setiap perubahan sekecil apa pun harus memiliki jejak audit yang jelas melalui CHANGELOG.md dan penomoran tag yang konsisten untuk memudahkan proses *rollback* jika terjadi kendala. Secara teknis, penulis mengasah kemahiran dalam mengelola ekosistem GitLab CI/CD, mulai dari memahami *pipeline status* hingga manajemen dependensi pada package.json di dalam arsitektur sistem yang terpisah antara modul dan *wrapper*. Pengalaman di BSI UII ini menyadarkan penulis bahwa kualitas perangkat lunak tidak hanya ditentukan oleh logika pemrograman yang benar, tetapi juga oleh ketatnya prosedur integrasi dan kolaborasi profesional antara pengembang junior dan senior dalam menjaga integritas kode di setiap lingkungan rilis.



Gambar 3.34 Diagram Alur Proses *Deploy* Aplikasi

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Refleksi Proses Pengembangan *Front-end* UIIGateway

Pelaksanaan magang pada Badan Sistem Informasi Universitas Islam Indonesia (BSI UII) memberikan pengalaman komprehensif bagi penulis dalam memahami pengembangan *front-end* pada sistem informasi berskala institusional. UIIGateway sebagai portal layanan terpadu universitas memiliki karakteristik sistem yang kompleks, terintegrasi dengan berbagai modul aplikasi, serta digunakan oleh pengguna dengan latar belakang dan kebutuhan yang beragam. Kondisi tersebut menuntut proses pengembangan *front-end* yang tidak hanya berorientasi pada aspek teknis, tetapi juga mempertimbangkan stabilitas sistem, konsistensi antarmuka, serta keberlanjutan pengembangan jangka panjang.

Berbeda dengan pengembangan aplikasi dalam konteks akademik yang umumnya bersifat simulatif, pengembangan *front-end* UIIGateway dilakukan pada sistem produksi yang telah berjalan dan memiliki dependensi dengan modul-modul lain. Hal ini menuntut penulis untuk memahami terlebih dahulu struktur arsitektur aplikasi, alur bisnis sistem, serta standar pengembangan yang telah ditetapkan oleh BSI UII sebelum melakukan implementasi fitur baru atau modifikasi antarmuka. Setiap perubahan yang dilakukan harus dipastikan tidak mengganggu fungsionalitas sistem lain yang saling terintegrasi.

Dalam proses pengembangan, penulis menyadari bahwa peran *front-end developer* dalam sistem terintegrasi tidak dapat dipisahkan dari pemahaman konteks *back-end* dan alur data. Meskipun fokus utama berada pada sisi antarmuka, setiap implementasi komponen dan fitur harus mempertimbangkan bagaimana data diperoleh melalui RESTful API, bagaimana hak akses pengguna diterapkan, serta bagaimana respons sistem ditampilkan secara informatif dan konsisten. Pemahaman ini menjadi krusial mengingat UIIGateway berfungsi sebagai *wrapper* aplikasi yang membungkus berbagai modul layanan universitas.

Penggunaan *framework* Angular memberikan pengalaman berharga terkait penerapan arsitektur berbasis komponen dalam skala besar. Penulis belajar bahwa pengelolaan struktur folder, pemisahan antara komponen tampilan dan *service*, serta penerapan *lazy loading* pada level modul sangat berpengaruh terhadap performa dan *maintainability* aplikasi. Dalam konteks UIIGateway, pendekatan ini memungkinkan modul kepegawaian dikembangkan secara

terpisah namun tetap terintegrasi dengan *wrapper* utama, sehingga konsistensi sistem dapat tetap terjaga.

Selain aspek arsitektur, refleksi juga dilakukan terhadap proses implementasi antarmuka pengguna. Penulis terlibat dalam upaya penyelarasan tampilan antar modul kepegawaian, baik dari sisi tipografi, warna, maupun hierarki visual. Proses ini menuntut ketelitian tinggi karena perubahan kecil pada *style* atau komponen dapat berdampak pada pengalaman pengguna secara keseluruhan. Penulis memahami bahwa konsistensi antarmuka merupakan faktor penting dalam meningkatkan *usability*, terutama pada sistem yang digunakan secara rutin oleh pegawai universitas.

Dari sisi kolaborasi tim, proses pengembangan *front-end* UIIGateway memberikan pembelajaran mengenai pentingnya komunikasi lintas peran. Penulis bekerja di bawah arahan *Project Manager* serta berkoordinasi dengan *back-end* developer dan *UI/UX Designer*. Diskusi rutin dilakukan untuk menyamakan persepsi terkait kebutuhan fitur, desain antarmuka, serta hasil pengujian sebelum fitur dirilis ke lingkungan *staging* atau *production*. Melalui proses ini, penulis menyadari bahwa keberhasilan pengembangan sistem tidak hanya ditentukan oleh kemampuan individu, tetapi juga oleh efektivitas kerja tim.

Pengujian antarmuka menjadi bagian penting dalam proses pengembangan. Meskipun tidak terdapat tim *Quality Assurance* khusus pada saat pelaksanaan magang, peran pengujian dilakukan secara kolaboratif antara *front-end developer* dan *UI/UX Designer*. Penulis melakukan pengujian fungsional dasar untuk memastikan komponen berjalan sesuai kebutuhan, sementara *UI/UX Designer* melakukan evaluasi lanjutan terkait kesesuaian desain, kenyamanan penggunaan, serta kejelasan informasi yang disampaikan kepada pengguna. Proses ini memperkuat pemahaman penulis mengenai pentingnya *quality control* dalam pengembangan perangkat lunak.

Selain itu, pengembangan fitur-fitur tertentu, seperti animasi pemuatan dinamis dan sinkronisasi data bertahap berbasis *offset*, memberikan pengalaman dalam merancang solusi *front-end* yang adaptif terhadap keterbatasan sistem *back-end*. Penulis belajar bahwa solusi teknis yang efektif tidak selalu bergantung pada teknologi canggih, tetapi pada kemampuan menyesuaikan pendekatan dengan kondisi infrastruktur yang tersedia. Pendekatan ini membantu menjaga performa sistem sekaligus memberikan pengalaman pengguna yang lebih baik.

Secara keseluruhan, refleksi terhadap proses pengembangan *front-end* UIIGateway menunjukkan bahwa pengalaman magang ini tidak hanya meningkatkan kemampuan teknis

penulis, tetapi juga membentuk pola pikir profesional dalam pengembangan perangkat lunak. Penulis belajar untuk bekerja secara sistematis, memperhatikan detail, serta bertanggung jawab terhadap setiap perubahan yang dilakukan pada sistem. Pengalaman ini menjadi bekal penting bagi penulis dalam menghadapi tantangan pengembangan sistem informasi di dunia kerja profesional di masa mendatang.

4.2 Tantangan dan Strategi Penyelesaian Selama Pelaksanaan Magang

Selama pelaksanaan magang pada Badan Sistem Informasi Universitas Islam Indonesia, penulis menghadapi berbagai tantangan yang bersifat teknis maupun non-teknis. Tantangan tersebut muncul seiring dengan kompleksitas sistem UIIGateway sebagai portal layanan terpadu universitas yang telah digunakan secara aktif oleh *civitas academica*. Berbeda dengan proyek pengembangan baru, pengembangan *front-end* UIIGateway dilakukan pada sistem yang telah berjalan dan memiliki ketergantungan tinggi antar modul, sehingga setiap perubahan harus dilakukan secara hati-hati dan terencana.

4.2.1 Tantangan Pemahaman Arsitektur Sistem yang Kompleks

Salah satu tantangan awal yang dihadapi penulis adalah memahami arsitektur *front-end* UIIGateway sebagai *wrapper* aplikasi. UIIGateway tidak hanya berfungsi sebagai halaman *login*, tetapi juga bertanggung jawab dalam membungkus berbagai modul aplikasi dengan mekanisme pemuatan dinamis dan pengaturan hak akses pengguna. Struktur proyek Angular yang terdiri dari banyak modul, komponen, dan *service* memerlukan waktu adaptasi agar penulis dapat memahami alur kerja aplikasi secara menyeluruh.

Strategi yang dilakukan untuk mengatasi tantangan ini adalah dengan mempelajari dokumentasi internal proyek, menelusuri struktur kode secara bertahap, serta melakukan diskusi intensif dengan *Project Manager* dan anggota tim pengembang lainnya. Penulis juga memanfaatkan pendekatan *learning by doing* dengan mengerjakan *task* berskala kecil terlebih dahulu sebelum menangani fitur yang lebih kompleks. Pendekatan ini membantu penulis memahami hubungan antar komponen, *service*, serta alur komunikasi dengan *back-end* API secara lebih mendalam.

4.2.2 Tantangan Integrasi *Front-end* dengan *Back-end* API

Tantangan berikutnya berkaitan dengan integrasi *front-end* dengan *back-end* API yang telah dikembangkan sebelumnya. Tidak seluruh *endpoint* API memiliki dokumentasi yang

lengkap atau seragam, sehingga penulis perlu memahami struktur respons API melalui pengujian langsung dan komunikasi dengan *back-end* developer. Selain itu, perubahan kebutuhan bisnis terkadang menyebabkan adanya penyesuaian kontrak API yang berdampak pada sisi *front-end*.

Untuk mengatasi hal tersebut, penulis menerapkan strategi komunikasi aktif dengan *back-end* developer dan melakukan pengujian API menggunakan alat bantu seperti Postman sebelum proses integrasi ke dalam komponen Angular. Pendekatan ini membantu mengurangi kesalahan integrasi serta memastikan data yang ditampilkan pada antarmuka sesuai dengan kebutuhan fungsional sistem. Penulis juga membiasakan diri untuk menangani kondisi *error* dan validasi data pada sisi *front-end* agar sistem tetap stabil meskipun terjadi anomali pada respons API.

4.2.3 Tantangan Penyesuaian Antarmuka dengan Standar *UI/UX*

Penyesuaian antarmuka pengguna menjadi tantangan tersendiri karena UIIGateway harus mempertahankan konsistensi visual antar modul. Meskipun UIIGateway telah menggunakan *library* PilarNg sebagai dasar komponen, beberapa kebutuhan desain tidak dapat sepenuhnya dipenuhi oleh *template* standar yang tersedia. Hal ini menuntut penulis untuk melakukan kustomisasi komponen tanpa merusak struktur inti *library*.

Strategi yang diterapkan adalah dengan memanfaatkan teknik *styling* berbasis *scoped style* pada Angular, seperti penggunaan (:host) dan (::ng-deep), sehingga perubahan *style* hanya berdampak pada komponen tertentu. Selain itu, setiap perubahan desain selalu dikonsultasikan dengan *UI/UX Designer* untuk memastikan kesesuaian dengan pedoman desain UIIGateway. Melalui proses ini, penulis belajar bahwa pengembangan antarmuka tidak hanya berfokus pada aspek estetika, tetapi juga harus memperhatikan konsistensi, aksesibilitas, dan kenyamanan pengguna.

4.2.4 Tantangan Pengelolaan Data dalam Jumlah Besar

Pada beberapa modul kepegawaian, penulis dihadapkan pada tantangan pengelolaan data dalam jumlah besar, khususnya pada proses sinkronisasi data dosen dari sistem eksternal. Keterbatasan infrastruktur *back-end* yang tidak mendukung mekanisme *Server-Sent Events* (SSE) mendorong perlunya solusi alternatif pada sisi *front-end* agar pengguna tetap memperoleh umpan balik visual selama proses sinkronisasi berlangsung.

Sebagai strategi penyelesaian, penulis menerapkan pendekatan sinkronisasi data bertahap berbasis *offset* yang dikombinasikan dengan animasi pemuatan dinamis. Pendekatan ini

memungkinkan *front-end* menampilkan progres sinkronisasi secara bertahap meskipun proses pengambilan data dilakukan secara tersegmentasi. Dengan strategi tersebut, pengalaman pengguna dapat ditingkatkan tanpa menambah beban berlebih pada sisi server.

4.2.5 Tantangan Pengujian dan Validasi Fungsional

Pengujian sistem juga menjadi tantangan tersendiri karena pada saat pelaksanaan magang belum terdapat tim *Quality Assurance* khusus. Proses pengujian dilakukan secara kolaboratif antara *front-end* developer dan *UI/UX Designer*, dengan fokus pada pengujian fungsional dan kesesuaian tampilan antarmuka. Kondisi ini menuntut penulis untuk lebih teliti dalam melakukan pengujian mandiri sebelum fitur diserahkan untuk evaluasi lanjutan.

Strategi yang dilakukan adalah dengan menerapkan pengujian bertahap pada setiap fitur yang dikembangkan, mulai dari pengujian unit sederhana pada komponen, pengujian integrasi API, hingga pengujian tampilan pada berbagai skenario penggunaan. Penulis juga memanfaatkan alat bantu seperti PageSpeed Insights sebagai inisiatif pribadi untuk mengevaluasi kualitas antarmuka halaman *login* UIIGateway, khususnya dari aspek performa dan aksesibilitas.

4.2.6 Tantangan Manajemen Waktu dan Prioritas Pekerjaan

Selain tantangan teknis, penulis juga menghadapi tantangan dalam manajemen waktu dan prioritas pekerjaan. Pengembangan sistem dilakukan dalam kerangka kerja *Agile Scrum*, sehingga penulis harus menyesuaikan diri dengan ritme sprint, tenggat waktu, serta perubahan kebutuhan yang dapat terjadi sewaktu-waktu. Beberapa *task* memiliki tingkat urgensi tinggi karena berkaitan langsung dengan kebutuhan operasional pengguna.

Untuk mengatasi hal ini, penulis menerapkan strategi pengelolaan waktu yang lebih terstruktur dengan memanfaatkan *tools* manajemen proyek seperti Jira. Setiap tugas dicatat dan diprioritaskan berdasarkan tingkat urgensi dan kompleksitas. Diskusi rutin dengan *Project Manager* juga membantu penulis dalam menentukan fokus pekerjaan serta memastikan bahwa target sprint dapat tercapai dengan baik.

4.3 Pembelajaran dan Peningkatan Kompetensi Penulis

Selama pelaksanaan magang di Badan Sistem Informasi Universitas Islam Indonesia (BSI UII), penulis memperoleh berbagai pembelajaran yang berkontribusi secara signifikan terhadap peningkatan kompetensi, baik dari aspek teknis maupun non-teknis. Kegiatan magang tidak

hanya berfokus pada penerapan teori yang diperoleh selama perkuliahan, tetapi juga memberikan pengalaman nyata dalam pengembangan sistem informasi berskala institusi.

Dari sisi teknis, penulis mengalami peningkatan kemampuan dalam pengembangan *front-end* berbasis *framework* Angular. Penulis terlibat langsung dalam pengembangan modul kepegawaian pada UIIGateway, termasuk perancangan komponen antarmuka, pengelolaan *state* aplikasi, implementasi *reactive form*, serta integrasi dengan RESTful API. Selain itu, penulis juga mempelajari penerapan arsitektur modular, *lazy loading*, serta pemanfaatan *service* Angular untuk menjaga keterpisahan antara logika aplikasi dan tampilan antarmuka. Pengalaman ini memperdalam pemahaman penulis terhadap praktik pengembangan *front-end* yang terstruktur, efisien, dan mudah dipelihara.

Selain kompetensi teknis, penulis juga memperoleh pembelajaran dalam aspek non-teknis yang tidak kalah penting. Selama magang, penulis terbiasa bekerja dalam lingkungan tim pengembangan dengan pembagian peran yang jelas serta alur kerja yang terkoordinasi. Penulis belajar berkomunikasi secara efektif dengan anggota tim, menyampaikan progres pekerjaan, serta menerima masukan dan evaluasi dari atasan maupun rekan kerja. Pembelajaran ini melatih penulis dalam manajemen waktu, tanggung jawab terhadap tugas, serta kemampuan beradaptasi dengan dinamika kerja di lingkungan profesional.

Manfaat lain yang diperoleh penulis selama magang adalah pemahaman terhadap standar kerja dan budaya organisasi di BSI UII. Penulis menjadi lebih memahami pentingnya dokumentasi, konsistensi desain antarmuka, serta kepatuhan terhadap standar pengembangan sistem yang telah ditetapkan. Pengalaman ini memberikan gambaran nyata mengenai ekspektasi dunia kerja terhadap seorang pengembang perangkat lunak, khususnya dalam konteks pengembangan sistem informasi institusional.

Pada masa akhir pelaksanaan magang, tepatnya pada bulan Oktober, BSI UII menyelenggarakan ujian kompetensi sebagai bagian dari proses seleksi penerbitan Surat Pegawai Kontrak (SPK). Penulis mengikuti ujian kompetensi tersebut sebagai bentuk evaluasi terhadap kemampuan dan pengalaman yang telah diperoleh selama menjalani program magang. Ujian ini mencakup penilaian terhadap kompetensi teknis, pemahaman sistem, serta kesiapan penulis untuk berkontribusi lebih lanjut dalam lingkungan kerja BSI UII.

Berdasarkan hasil seleksi tersebut, penulis dinyatakan lulus ujian kompetensi dan resmi menerima Surat Pegawai Kontrak (SPK). Dengan diterbitkannya SPK tersebut, status penulis berubah dari peserta magang menjadi pegawai kontrak di BSI UII. Perubahan status ini

menyebabkan masa kontrak magang penulis berakhir lebih awal pada bulan Oktober, meskipun sebelumnya direncanakan berlangsung hingga bulan Desember.

Keberhasilan penulis dalam memperoleh SPK menjadi indikator bahwa proses pembelajaran selama magang berjalan secara efektif dan relevan dengan kebutuhan institusi. Pengalaman langsung dalam pengembangan *front-end* UIIGateway, keterlibatan dalam tim pengembangan, serta pemahaman terhadap alur kerja dan budaya organisasi memberikan bekal yang kuat bagi penulis untuk memenuhi standar kompetensi yang ditetapkan. Dengan demikian, program magang yang dijalani tidak hanya berfungsi sebagai sarana pembelajaran akademik, tetapi juga sebagai jembatan menuju kesiapan profesional di dunia kerja.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan pelaksanaan magang dan pengembangan *front-end* pada sistem informasi manajemen kepegawaian UIIGateway, dapat disimpulkan bahwa penerapan *framework* Angular dengan arsitektur berbasis komponen telah berhasil menciptakan antarmuka yang modular, *reusable*, dan konsisten pada modul-modul kepegawaian. Arsitektur berbasis komponen memungkinkan setiap fungsi antarmuka dibuat secara terpisah namun saling terhubung, sehingga memudahkan pemeliharaan kode, mempercepat pengembangan fitur baru, dan meningkatkan kolaborasi antar tim pengembang. Standar desain UI yang diterapkan secara konsisten di semua modul juga memberikan pengalaman pengguna yang seragam dan intuitif, mengurangi kurva pembelajaran pengguna baru, serta meningkatkan kepuasan pengguna terhadap sistem secara keseluruhan. Dengan pendekatan modular ini, sistem menjadi lebih skalabel dan siap menghadapi pertambahan modul serta perubahan kebutuhan bisnis di masa depan.

Implementasi teknik *lazy loading* pada level modul dan mekanisme sinkronisasi data berbasis *offset* telah terbukti secara signifikan meningkatkan performa aplikasi dan pengalaman pengguna. Teknik *lazy loading* memungkinkan aplikasi hanya memuat modul yang diperlukan pengguna pada saat itu, bukan seluruh modul sekaligus saat aplikasi pertama kali dibuka, sehingga mengurangi ukuran *bundle* awal dan mempercepat waktu akses awal. Sementara itu, sinkronisasi data berbasis *offset* memungkinkan sistem untuk menampilkan data kepegawaian dalam jumlah besar secara bertahap, menghindari beban berlebihan pada sisi klien maupun server, dan memberikan pengalaman *browsing* yang lebih responsif. Pengujian performa menunjukkan bahwa pengguna dapat mengakses halaman dan memuat data dengan waktu tunggu yang jauh lebih singkat dibandingkan dengan metode pemuatan konvensional, meningkatkan produktivitas pengguna, dan mengurangi tingkat kegagalan akses data.

Integrasi *front-end* dengan *reporting tools* (Jaspersoft Studio dan Docxtemplater) melalui RESTful API telah berhasil mengotomatisasi proses pembuatan dokumen administratif seperti Surat Keputusan (SK) dan dokumen kepegawaian lainnya secara dinamis dan *real-time*. Sebelum adanya fitur ini, pembuatan dokumen dilakukan secara manual dengan melibatkan banyak proses copy-paste dan editing, yang memakan waktu serta rentan terhadap kesalahan

data dan inkonsistensi format. Dengan sistem *otomasi* yang terintegrasi, pengguna dapat menghasilkan dokumen berkualitas tinggi hanya dengan beberapa klik, meningkatkan efisiensi administratif, mempercepat waktu penerbitan dokumen, dan meminimalkan kesalahan penulisan data. Fitur *preview* dokumen sebelum mencetak juga memberikan fleksibilitas kepada pengguna untuk memverifikasi isi dokumen, sehingga mengurangi risiko kesalahan dokumen yang diterbitkan. Metode *Agile Scrum* yang diterapkan selama magang juga terbukti sangat mendukung proses pengembangan yang dinamis, kolaboratif, dan responsif terhadap perubahan kebutuhan. Komunikasi rutin melalui *Daily Standup*, *Sprint Planning*, *Sprint Review*, dan *Retrospective* memastikan setiap fitur yang dikembangkan selaras dengan kebutuhan pengguna, dapat langsung diuji validitasnya, dan tim tetap fokus pada pencapaian target iterasi. Kombinasi semua elemen ini *framework* Angular yang solid, optimasi performa yang efektif, integrasi dokumen otomatis, dan metodologi Scrum yang adaptif—telah menghasilkan sistem informasi kepegawaian yang berkualitas tinggi, efisien, dan siap memenuhi kebutuhan operasional UII saat ini maupun di masa depan.

5.2 Saran

Format Untuk pengembangan sistem yang lebih optimal, *robust*, dan berkelanjutan di masa mendatang, penulis menyarankan beberapa peningkatan teknis dan fungsional yang perlu dipertimbangkan oleh tim pengembang dan manajemen. Pertama, disarankan untuk mengimplementasikan *Automated Testing* baik pada tingkat Unit Testing maupun *End-to-End* (E2E) Testing menggunakan *framework* seperti Jasmine/Karma atau Cypress untuk memastikan setiap perubahan kode tidak menimbulkan regresi pada fitur yang sudah berjalan dengan baik. Penambahan *automated testing* akan meningkatkan kualitas rilis, mempercepat proses regresi testing, dan memberikan kepercayaan diri yang lebih tinggi kepada tim saat melakukan *update sistem*. Kedua, seiring dengan bertambahnya kompleksitas pengelolaan data di sisi klien, penggunaan *library state management* yang terstruktur seperti NgRx atau Akita sangat disarankan untuk mengelola aliran data antar komponen dengan lebih efisien, terukur, dan mudah di-debug. *State management* yang baik akan memudahkan pelacakan perubahan data, mengurangi bug yang sulit diidentifikasi, dan meningkatkan performa aplikasi secara keseluruhan.

Ketiga, pengembangan selanjutnya sebaiknya lebih memperhatikan aspek aksesibilitas dengan mengadopsi standar *Web Content Accessibility Guidelines* (WCAG) agar sistem dapat diakses dengan baik oleh seluruh pengguna, termasuk pengguna dengan kebutuhan khusus

seperti gangguan penglihatan, pendengaran, atau motorik. Implementasi aksesibilitas yang baik tidak hanya dari aspek kemanusiaan, tetapi juga memperluas jangkauan pengguna dan memenuhi persyaratan regulasi yang semakin ketat di banyak negara.

Terakhir, penguatan keamanan *front-end* seperti *input validation* yang ketat, proteksi terhadap *Cross-Site Scripting* (XSS), dan *Cross-Site Request Forgery* (CSRF) perlu diperkuat untuk menjaga integritas data pengguna dan mencegah penyalahgunaan sistem. Keamanan aplikasi web bukan hanya tanggung jawab *back-end*, tetapi juga memerlukan implementasi kontrol keamanan yang solid di sisi *front-end*.

DAFTAR PUSTAKA

Abhishek Kesarwani. (2022, September 16). Ahead-of-time (AOT) Compilation. Diambil 24 Desember 2025, dari Halodoc Blog website: https://blogs.halodoc.io/ahead-of-time-compilation/

Angela Gjorgievska. (2024, Juni 11). Effective UI/UX Design Principles for Front-End Developers. Diambil 24 Desember 2025, dari Keitaro website: https://www.keitaro.com/insights/2024/06/11/effective-ui-ux-design-principles-for-front-end-developers/

AngularMinds. (2024, April 21). How to Implement Two-Way Data Binding in Angular. Diambil 24 Desember 2025, dari AngularMinds website: https://www.angularminds.com/blog/how-to-implement-two-way-data-binding-in-angular

Anton Zubrytskyi. (2024, Agustus 16). The Impact of UX in Front-End Development in 2026. Diambil 24 Desember 2025, dari Elitex website: https://elitex.systems/blog/the-impact-of-ux-in-front-end-development

Aripradono, H. W., & Venessa, K. (2023). Pengembangan Front-End Website Sistem Keuangan UMKM di SMK Multistudi Highschool Batam Menggunakan Agile Scrum. *Madani: Jurnal Pengabdian Masyarakat dan Kewirausahaan*, *1*(2), 90–107. https://doi.org/10.37253/madani.v2i1.7442

Asfak Ahmed. (2025, September 12). Why Front-End Developers Should Understand UI/UX Design. Diambil 24 Desember 2025, dari FreeCodeCamp website: https://www.freecodecamp.org/news/why-front-end-developers-should-understand-uiux-design/

Badan Sistem Informasi Universitas Islam Indonesia. (2025, September 5). Sekilas Tentang BSI. Diambil 5 September 2025, dari https://bsi.uii.ac.id/sekilas-bsi/

Bahar, M. M., Nurwahid, M. S., Putra, S. A., Parenreng, J. M., Wahid, A., & Irmawati. (2023). PERANCANGAN SISTEM INFORMASI MANAJEMEN KEPEGAWAIAN (SIMPEG) BERBASIS WEB PADA UNIVERSITAS NEGERI MAKASSAR. *Journal of Embedded Systems, Security and Intelligent Systems*, *2*(1), 1–6. Diambil dari http://journal.unm.ac.id/index.php/JESSI/article/view/414

Blundo, C., & Cimato, S. (2012). *Constrained Role Mining*. Diambil dari https://arxiv.org/abs/1203.3744

Bülthoff, F., & Maleshkova, M. (2014). *RESTful or RESTless – Current State of Today's Top Web APIs*. https://doi.org/10.1007/978-3-319-11955-7_6

Coblenz, M., Guo, W., Voozhian, K., & Foster, J. S. (2023). A Qualitative Study of REST API Design and Specification Practices. *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 148–156. IEEE. https://doi.org/10.1109/VL-HCC57772.2023.00025

Das, A. (2024). The Future of Component-Based Architecture in Web Development. Diambil 24 Desember 2025, dari PixelFree Studio website: https://blog.pixelfreestudio.com/the-future-of-component-based-architecture-in-web-development/#respond

Fauzan, F., & A B Dinariyana, A. (2024). Integrating Design Thinking and Agile Scrum Methodology In Human Resources Management System Development: Digital Transformation In Employee Management Case Study on PT Derma Konsep Estetika. *Dinasti International Journal of Education Management And Social Science*, *5*(6), 2182–2199. https://doi.org/10.38035/dijemss.v5i6.3058

GeeksforGeeks. (2025, Juli 23). How to achieve Two-Way Data Binding in Angular with ngModel ? Diambil 24 Desember 2025, dari GeeksforGeeks website: https://www.geeksforgeeks.org/angular-js/how-to-achieve-two-way-data-binding-in-angular-with-ngmodel/

Ghotbi, S. H., & Fischer, B. (2012). A declarative fine-grained role-based access control model and mechanism for the web application domain. *ICSOFT 2012 - Proceedings of the 7th International Conference on Software Paradigm Trends*, 80–91.

Harikrishna Kundariya. (2024, Agustus 21). Mastering Lazy Loading in Angular: A Comprehensive Guide. Diambil 24 Desember 2025, dari You.fun website: https://yon.fun/angular-lazy-loading/

Hasanah, A. N., & Putro, H. P. (2021). Implementasi JasperReports pada Sistem Informasi Manajemen. *AUTOMATA*, *2*(1).

IT Consultis. (2024). 5 Ways Component-Based Design Empowers Web Development. Diambil 24 Desember 2025, dari IT Consultis website: https://it-consultis.com/insights/component-based-design/

Jacob. (2025, September 3). What Is TypeScript in Angular? Unlocking Its Power in 2025. Diambil 24 Desember 2025, dari StackInterface website: https://stackinterface.com/what-is-typescript-in-angular/

Kirana, W. K. C. (2023). Pemanfaatan Directive pada Framework Angular untuk Pengembangan Website Penerimaan Mahasiswa Baru. *AUTOMATA*, *4*(2). Diambil dari https://journal.uii.ac.id/AUTOMATA/article/view/28845

Maneshwar, A. (2025). Component-Based Design in Software Architecture. Diambil 24 Desember 2025, dari Dev.to website: https://dev.to/lovestaco/component-based-design-in-software-architecture-pbf

Muzammil K. (2025, Juni 24). Key Difference Between Angular and Typescript. Diambil 24 Desember 2025, dari Aalpha website: https://www.aalpha.net/articles/key-difference-between-angular-and-typescript/

Nandaniya, H. (2019). Component-Based Architecture: Principles, Benefits & Examples. Diambil 24 Desember 2025, dari Maruti Techlabs website: https://marutitech.com/guide-to-component-based-architecture/

Narasimman, R., & Alsmadi, I. (2020). *RBAC for Healthcare-Infrastructure and data storage*. Diambil dari https://arxiv.org/abs/2010.11096

Neumann, A., Laranjeiro, N., & Bernardino, J. (2021). An Analysis of Public REST Web Service APIs. *IEEE Transactions on Services Computing*, *14*(4), 957–970. https://doi.org/10.1109/TSC.2018.2847344

Novianto, M. A., & Munir, S. (2022). Analisis dan Implementasi Restful API guna Pengembangan Sistem Informasi Akademik pada Perguruan Tinggi. *Jurnal Informatika Terpadu*, *8*(1), 47–61. https://doi.org/10.54914/jit.v8i1.409

Oracle. (2025). What is human capital management (HCM)? Diambil 24 Desember 2025, dari Oracle website: https://www.oracle.com/human-capital-management/what-is-hcm/

Peek, S. (2024, Juni 24). What Is Agile Scrum Methodology? Diambil 24 Desember 2025, dari Business News Daily website: https://www.businessnewsdaily.com/4987-what-is-agile-scrum-methodology.html

Rahman, S. F., Ataullah, W. H., Yasirandi, R., Wibowo, J. A., Insan, I. M., & Makky, M. A. (2025). Frontend Design and Implementation of a Research Dashboard for the PPM Directorate Using the Agile Scrum Methodology. *2025 3rd International Conference on Software Engineering and Information Technology (ICoSEIT)*, 1–6. https://doi.org/10.1109/ICoSEIT67010.2025.11290847

Sabri, K. E. (2018). An Algebraic Model to Analyze Role-Based Access Control Policies. *Modern Applied Science*, *12*(10), 50. https://doi.org/10.5539/mas.v12n10p50

SAP. (2025). What is human capital management (HCM)? Diambil 24 Desember 2025, dari https://www.sap.com/products/hcm/what-is-human-capital-management.html

Setiawan, A. A., & Fauzi, E. (2025a). ANALISIS KOMPARATIF PERFORMA IMPLEMENTASI LAZY LOADING DAN CODE SPLITTING PADA FRAMEWORK REACT, VUE, DAN ANGULAR BERDASARKAN SKOR LIGHTHOUSE. *INTECOMS: Journal of Information Technology and Computer Science*, *8*(3), 921–930. https://doi.org/10.31539/intecoms.v8i3.15847

Setiawan, A. A., & Fauzi, E. (2025b). ANALISIS KOMPARATIF PERFORMA IMPLEMENTASI LAZY LOADING DAN CODE SPLITTING PADA FRAMEWORK REACT, VUE, DAN ANGULAR BERDASARKAN SKOR LIGHTHOUSE. *INTECOMS: Journal of Information Technology and Computer Science*, *8*(3), 921–930. https://doi.org/10.31539/intecoms.v8i3.15847

Sukriti Srivastava. (2025, Agustus 1). Mastering Angular Lazy Loading: Boosting App Performance and Load Speed. Diambil 24 Desember 2025, dari MetaDesign Solutions website: https://metadesignsolutions.com/mastering-angular-lazy-loading-boosting-app-performance-and-load-speed/

Tecnovy. (2023, Agustus). What is Component-Based Software Architecture? Diambil 24 Desember 2025, dari Tecnovy website: https://tecnovy.com/en/component-software-architecture-guide

Wijaya, T. (2023). Implementasi Framework Angular untuk Meningkatkan Performa Aplikasi Web Modern. *Prosiding CORISINDO 2023*. Diambil dari https://ojs.stmikpontianak.ac.id/corisindo/article/view/4

Xu, C., & Gong, S. (2009). Formal Description for an Object-Oriented Role-based Access Control Model. *Computer and Information Science*, *2*(2). https://doi.org/10.5539/cis.v2n2p68

Zhezherau, A. (2025, September 11). Scrum methodology: The leading Agile framework. Diambil 24 Desember 2025, dari Wrike website: https://www.wrike.com/scrum-guide/scrum-methodology/

LAMPIRAN