PENGEMBANGAN SISTEM MONITORING PROYEK BERBASIS WEB MENGGUNAKAN SPRING BOOT DAN METODE *ROLE-BASED ACCESS CONTROL*

Disusun Oleh:

N a m a : Rizqdwan Dhuhakbar Hendyutama
NIM : 21523100

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN SISTEM MONITORING PROYEK BERBASIS WEB MENGGUNAKAN SPRING BOOT DAN METODE *ROLE-BASED ACCESS CONTROL*

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Rizqdwan Dhuhakbar Hendyutama
NIM : 21523100

Yogyakarta, 19 Agustus 2025

Pembimbing,

(Ari Sujarwo, S.Kom., M.I.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN SISTEM MONITORING PROYEK BERBASIS WEB MENGGUNAKAN SPRING BOOT DAN METODE *ROLE-BASED ACCESS CONTROL*

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, 19 Agustus 2025

Tim Penguji

Ari Sujarwo, S.Kom., M.I.T. _____

Anggota 1

Erika Ramadhani, S.T., M.Eng. _____

Anggota 2

Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D. _____

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	:	Rizqdwan Dhuhakbar Hendyutama
NIM	:	21523100

Tugas akhir dengan judul:

PENGEMBANGAN SISTEM MONITORING PROYEK BERBASIS WEB MENGGUNAKAN SPRING BOOT DAN METODE *ROLE-BASED ACCESS CONTROL*

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 apapun.

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

Yogyakarta, 19 Agustus 2025

(Rizqdwan Dhuhakbar Hendyutama)

HALAMAN PERSEMBAHAN

Segala puji dan syukur saya panjatkan kepada Allah SWT, atas segala rahmat, karunia, dan hidayah-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir ini. Laporan akhir ini penulis persembahkan kepada kedua orang tua tercinta, Ayah dan Ibu yang senantiasa memberikan dukungan, arahan, dan doa di sepanjang perjalanan penulis. Kepada Simbah dan kedua eyang penulis, yang selalu mendoakan pada setiap langkah penulis, memberikan motivasi dan menjadi pengingat dalam setiap perjalanan hidup penulis. Terakhir, penulis juga persembahkan laporan akhir ini untuk diri sendiri, sebagai bukti bahwa pencapaian tidak akan didapat tanpa adanya kerja keras, kesabaran dan tekad yang kuat. Semoga ilmu yang didapatkan dapat bermanfaat bagi diri sendiri dan orang lain.

HALAMAN MOTO

“Allah tidak membebani seseorang melainkan sesuai dengan kemampuannya”

(QS Al-Baqarah: 286)

“If you want something you’ve never had, you must be willing to do something you’ve never done.”

(Thomas Jefferson)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dengan menyebut nama Allah SWT yang Maha pengasih dan Maha Penyayang. Segala puji dan syukur bagi Allah SWT yang telah melimpahkan Rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan penyusunan laporan tugas akhir jalur magang yang berjudul "Pengembangan Sistem Monitoring Proyek Berbasis Web Menggunakan Spring Boot dan Metode *Role-Based Access Control*". Penulisan laporan akhir ini bertujuan untuk memenuhi persyaratan kelulusan dari penjaluran magang di Program Studi Informatika – Program Sarjana, Fakultas Teknologi Industri, Universitas Islam Indonesia. Proses penulisan dan penyusunan laporan akhir ini tentunya tidak akan berjalan lancar tanpa bantuan, kebaikan, serta dukungan dari berbagai pihak. Dengan itu, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT. Yang senantiasa memberikan rahmat, petunjuk dan kekuatan kepada penulis selama proses penulisan dan penyusunan laporan akhir ini.
2. Kedua orang tua, Hendriasto Hartoyo dan Yuli Astuti, yang senantiasa memberikan dukungan, arahan, dan doa di sepanjang perjalanan penulis. Terima kasih atas kasih sayang tak terhingga, nasihat dan dukungan yang telah diberikan semasa pengerjaan laporan akhir maupun perkuliahan.
3. Simbah, Sumiyati, dan kedua eyang, Hartoyo Subagyo dan Sri Endah, yang selalu mendoakan, memotivasi dan menjadi pengingat dalam setiap perjalanan hidup penulis.
4. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
5. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika dan Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika – Program Sarjana.
6. Bapak Ari Sujarwo, S.Kom., M.I.T., selaku dosen pembimbing yang telah memberikan bimbingan dan arahan terkait penulisan laporan akhir.
7. Bapak Hanson Prihantoro Putro, S.T., M.T., selaku *supervisor* magang dari kampus yang telah memberikan arahan dan motivasi dalam proses magang.
8. Bapak dan Ibu dosen Program Studi Informatika, yang berjasa dalam memberikan berbagai ilmu dan nasehat yang bermanfaat kepada penulis semasa kuliah.

9. PT Bank Syariah Indonesia, yang telah memberikan kesempatan kepada penulis dalam program magang untuk dapat merasakan pengalaman langsung dunia kerja pada lingkup IT perbankan syariah.
10. Bapak Taufiq Galang, selaku *supervisor* magang di PT Bank Syariah Indonesia yang telah membantu dan membimbing penulis dalam mengikuti proses magang.
11. Bapak Abdul Tholib dan Bapak Arif Sabarudin, selaku mentor dalam departemen WHA di PT Bank Syariah Indonesia yang telah baik hati membagikan pengetahuan dan pengalamannya selama pelaksanaan magang, serta memberikan arahan selama pengembangan proyek.
12. Kucing kesayangan, Miming, Oren, Timbul, Ciko, Cilo, Gendon, Bobo, Jenggo, Bebe, Simbok, dan Nemo, yang senantiasa menemani perjalanan penulis.
13. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang telah memberikan cerita, pengalaman, dukungan, motivasi pada penulis dalam perjalanan perkuliahan dan selama proses penyusunan laporan akhir ini.

Terima kasih kepada semua pihak yang telah berkontribusi. Penulis berharap laporan akhir ini memberikan manfaat dan kegunaan untuk di masa mendatang. Penulis juga menyadari bahwa laporan ini masih jauh dari sempurna, oleh karena itu, kritik dan saran yang membangun sangat diharapkan.

Yogyakarta, 19 Agustus 2025

(Rizqdwan Dhuhakbar Hendyutama)

SARI

Perkembangan teknologi informasi dalam industri perbankan mendorong kebutuhan akan sistem yang mampu meningkatkan efisiensi dan transparansi dalam pengelolaan proyek. Departemen WHA (*IT Wholesale & Office Automation Development*) di bawah naungan *IT & Digital Development Group* (IDG) di PT Bank Syariah Indonesia Tbk menghadapi tantangan dalam memantau dan mengelola proyek-proyek teknologi secara terstruktur dan *real-time*. Sebagai solusi atas permasalahan tersebut, dikembangkanlah sistem monitoring proyek berbasis web yang disesuaikan dengan kebutuhan internal. Sistem dikembangkan menggunakan *framework* Spring Boot, serta integrasi basis data Microsoft SQL Server. Sistem ini juga mengadopsi arsitektur RESTful API, menggunakan JSON Web Token (JWT) sebagai metode autentikasi, dan menerapkan mekanisme keamanan *Role-Based Access Control* (RBAC). Sistem dirancang dengan empat peran utama pengguna (*Project Admin, Department Head,* PIC*,* dan *Team Member*), yang masing-masing memiliki hak akses berbeda sesuai dengan tanggung jawabnya. Pengembangan sistem dilakukan dengan pendekatan metode *Semi-Agile* yang menggabungkan unsur dokumentasi sistematis dari *Waterfall* serta fleksibilitas dan iterasi dari *Agile*. Fitur-fitur utama yang berhasil dikembangkan meliputi manajemen proyek, aplikasi, pengguna, dokumen dan *server installation package*, yang seluruhnya diakses melalui *endpoint* REST API. Laporan ini menyajikan proses pengembangan mulai dari perencanaan sistem, desain arsitektur hingga implementasi. Ke depanya, sistem ini berpotensi untuk diimplementasikan pada skala lebih luas di bawah naungan IDG.

Kata kunci: Monitoring proyek, Spring Boot, REST API, RBAC, *Semi-Agile*.

GLOSARIUM

Agile	metode pengembangan sistem yang memiliki prinsip fleksibilitas dan iterasi dalam pengembangan.
Backlog	daftar fitur, perbaikan, atau tugas yang menjadi acuan pengembangan sistem dalam metode *Semi-Agile*.
Department Head	pengguna yang bertangung jawab dalam memimpin dan mengelola suatu departemen.
Endpoint	titik akses pada sistem yang digunakan untuk menerima dan mengembalikan respon melalui protok HTTP.
Entity	representasi dari struktur tabel dalam basis data yang digunakan sebagai model dalam pemrograman berbasis objek.
Project Admin	pengguna yang memiliki semua hak akses pada sistem dan mewakili departemen untuk mengelola proyek.
Person In Charge	pengguna yang memiliki hak akses tertentu pada sistem dan bertanggung jawab melakukan manajemen proyek yang ditugaskan.
RBAC	mekanisme kontrol akses yang memberikan hak pengguna berdasarkan peran atau jabatan tertentu dalam sistem.
Semi-Agile	metode pengembangan sistem yang menggabungkan pendekatan dari prinsip model *waterfall* dan *agile*.
Spring Boot	*framework* berbasis Java yang digunakan untuk mempercepat proses pengembangan aplikasi dengan konfigurasi minimal dan integrasi otomatis berbagai komponen.
Team Member	pengguna yang memiliki hak akses terbatas pada proyek yang ditugaskan.
Waterfall	metode pengembangan perangkat lunak yang memiliki prinsip tahapan berurutan dari perencanaan awal hingga akhir pengembangan tanpa pengulangan.

DAFTAR ISI

DAFTAR TABEL

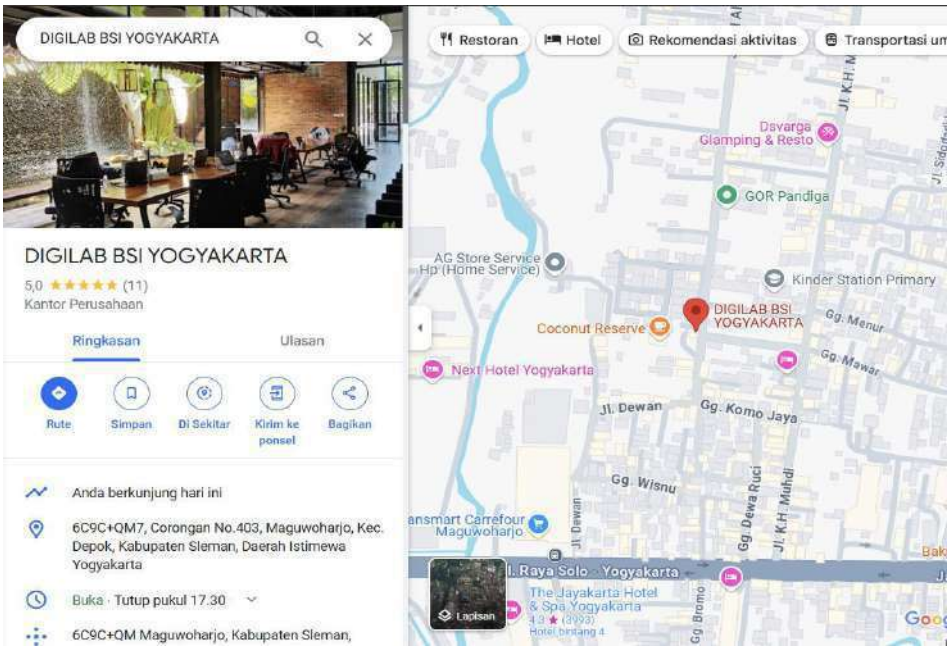
DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah menjadi unsur utama dalam transformasi industri perbankan, termasuk dalam sistem operasional dan layanan kepada nasabah. PT Bank Syariah Indonesia Tbk atau biasa disingkat BSI merupakan Bank Syariah terbesar di Indonesia berkomitmen untuk menyediakan layanan keuangan syariah yang inovatif dan relevan bagi perkembangan zaman. Dalam mendukung kebutuhan tersebut, BSI memiliki unit kerja bernama *IT & Digital Development Group* (IDG) yang berada di bawah Direktorat *Information Technology*. IDG memiliki fokus utama dalam pengembangan solusi digital untuk memenuhi kebutuhan internal dan eksternal perusahaan. Dalam hal tersebut, IDG didukung oleh Digilab BSI, sebuah *software house* internal yang didirikan oleh BSI. Digilab BSI didirikan di beberapa daerah, salah satunya di Yogyakarta. Digilab BSI berperan dalam mempercepat proses inovasi dengan menghasilkan produk-produk teknologi berkualitas tinggi untuk mendukung kebutuhan organisasi. Dalam operasionalnya, Digilab BSI mengintegrasikan keahlian internal perusahaan dengan kolaborasi pihak eksternal, sehingga mampu menghadirkan solusi yang adaptif terhadap perkembangan teknologi dan kebutuhan bisnis. Kantor Digilab Yogyakarta terletak di Corongan No.403, Maguwoharjo, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta. Tampak depan kantor ditampilkan pada Gambar 1.1, sedangkan lokasi ditunjukkan pada Gambar 1.2.

Gambar 1.1 Tampak depan kantor Digilab BSI



Gambar 1.2 Tangkapan layar pada Google Maps Lokasi kantor Digilab BSI

IDG juga menaungi beberapa departemen yang memiliki tanggung jawab strategis dalam pengembangan sistem dan layanan berbasis teknologi. Salah satunya adalah Departemen WHA (*IT Wholesale & Office Automation Development*), tempat penulis melaksanakan program magang. WHA berfokus pada pengembangan sistem otomasi kantor dan layanan IT untuk kebutuhan internal skala besar. Dalam praktik operasionalnya, WHA mengelola puluhan aplikasi dan menangani berbagai proyek teknologi yang bergerak secara dinamis. Hal ini menuntut adanya tata kelola proyek yang efisien serta transparansi informasi antar tim dan *Departmen Head* (DH). Namun dalam implementasinya, WHA menghadapi kendala dalam hal manajemen proyek, pelacakan progres aplikasi, dan transparansi status pengembangan. Pengelolaan proyek dilakukan secara terpisah, belum terdokumentasi secara terintegrasi, serta belum memiliki sistem sentralisasi yang mampu menggambarkan kondisi asli dari seluruh aktivitas pengembangan. Akibatnya, proses monitoring proyek menjadi kurang efisien, dan pengambilan keputusan pun memerlukan waktu lebih lama karena keterbatasan data yang tersedia secara *real-time*.

Dalam merespon permasalahan tersebut, WHA mengembangkan solusi berupa pengembangan sistem monitoring proyek berbasis web, yang dirancang khusus untuk mendukung kebutuhan internal departemen dalam memantau dan mengelola aplikasi serta proyek yang sedang berjalan. Sistem ini memiliki fungsi seperti Jira, namun dengan cakupan yang lebih spesifik dan disesuaikan dengan kebutuhan internal WHA. Melalui sistem ini, DH dapat secara langsung melakukan pengawasan terhadap aktivitas tim, alur pengembangan aplikasi, serta status proyek yang ditangani oleh departemen.

Dalam program magang ini, penulis ditempatkan pada Departemen WHA (*IT Wholesale & Office Automation Development*) untuk proyek baru dengan peran sebagai *Back-end Developer*. Fokus proyek yang dikerjakan adalah pengembangan sistem berbasis web untuk memonitoring proyek pada setiap departemen. Teknologi yang digunakan dalam proyek ini meliputi Spring Boot untuk *backend*, Vite React untuk *front-end*, serta *Microsoft SQL Server* sebagai *database*.

Secara keseluruhan, sistem monitoring proyek ini bertujuan untuk meningkatkan efisiensi, transparansi, dan koordinasi antar tim dalam setiap proyek yang berada di bawah grup IDG. Dengan adanya sistem ini, diharapkan proses pengawasan, pelaporan, dan pengambilan keputusan dapat dilakukan lebih cepat, akurat, dan terstruktur sehingga mendukung pencapaian tujuan strategis perusahaan secara keseluruhan.

1.2 Ruang Lingkup

Pengembangan sistem monitoring proyek berbasis web merupakan proyek baru yang diinisiasi oleh Departemen WHA (*IT Wholesale & Office Automation Development*). Proyek dikembangkan selama kurang lebih lima bulan, dimulai pada bulan Oktober 2024 sampai dengan Februari 2025. Proses pengembangan proyek dilakukan dengan pendekatan yang mengarah pada metode *Semi-Agile*, di mana proses tidak sepenuhnya mengikuti metode Scrum, namun tetap mengadopsi prinsip-prinsip iteratif dan kolaboratif, seperti pembagian tugas berdasarkan *backlog*, sesi diskusi rutin untuk evaluasi dan perencanaan, serta penyesuaian berkelanjutan terhadap kebutuhan departemen.

Tim pengembangan terdiri dari dua *developer*, yang mencakup satu *front-end developer* dan satu *back-end developer*, serta didukung oleh tiga *Quality Assurance* (QA). *Front-end developer* bertanggung jawab mengembangkan antarmuka pengguna dan mengintegrasikan API *back-end* sesuai dengan kebutuhan setiap fitur. *Back-end developer* bertanggung jawab untuk merancang, membangun, dan menyediakan API menggunakan *framework* Spring Boot, serta memastikan API dapat diimplementasikan dengan baik oleh *front-end*. Tim QA melakukan pengujian fungsional untuk memastikan setiap fitur sesuai dengan spesifikasi skenario dan melaporkan temuan terkait *bug* atau kendala pada sistem lainnya.

Laporan ini akan berfokus pada pembahasan mengenai proyek pengembangan sistem monitoring proyek berbasis web menggunakan *framework* Spring Boot dan metode RBAC. Penulis sebagai *back-end developer* berkontribusi pada keseluruhan pengembangan proyek antara lain mengembangkan fitur autentikasi, fitur *project management*, fitur *application management*, *user management*, *department management,* dan *group management*.

1.3 Tujuan

Tujuan dari proyek pengembangan sistem monitoring proyek berbasis web yang dikerjakan penulis adalah untuk menyediakan solusi digital yang memudahkan insan BSI dalam memantau, mengelola dan melakukan manajemen proyek pada setiap departemen yang di bawah naungan grup IDG (*IT Development Group*). Sistem ini bertujuan untuk meningkatkan efisiensi, transparansi, dan koordinasi antar tim serta memastikan keberhasilan implementasi proyek sesuai dengan tujuan strategis perusahaan.

1.4 Manfaat

Dengan adanya sistem monitoring proyek berbasis web yang rencananya akan diterapkan awal pada setiap departemen di bawah naungan grup IDG (*IT Development Group*), maka diharapkan dapat mempermudah khususnya *Department Head* dalam memantau, mengelola dan melakukan manajemen setiap proyek. Proyek ini juga memiliki beberapa manfaat sebagai berikut:

a. Meningkatkan efisiensi dalam pengambilan keputusan

   Sistem ini dirancang untuk mempermudah akses informasi secara visual dan terstruktur termasuk status pada proyek atau aplikasi yang sedang dikembangkan pada departemen terkait.

b. Mempermudah transparansi dan koordinasi

   Dengan informasi proyek yang terstruktur dan berfokus pada tim yang terkait, sehingga dapat memudahkan antar tim dalam satu departemen maupun berbeda departemen dalam berkolaborasi.

c. Menyediakan informasi data proyek sebagai dasar evaluasi kinerja

   Sistem ini juga berfungsi sebagai data historis proyek secara sistematis yang mencakup informasi mengenai progres, waktu pelaksanaan, serta hasil akhir dari setiap proyek. Informasi ini dapat digunakan sebagai dasar evaluasi terhadap kinerja tim maupun efektivitas pelaksanaan proyek dan juga menjadi acuan dalam perancangan proyek ke depan.

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 penjelasan umum terkait perusahaan tempat magang, serta ruang lingkup magang, tujuan, manfaat, dan sistematika penulisan.

b. BAB II: Landasan Teori dan Tinjauan Pustaka

   Bab ini membahas mengenai penjelasan teknologi dengan dukungan teori-teori dan penggunaannya yang berkaitan dengan pengembangan proyek, seperti *framework* Spring Boot, REST API, SQL Server, arsitektur RBAC, dan metode pengembangan *Semi-Agile*. Diakhiri dengan tinjauan pustaka yang membahas

persamaan dan perbedaan topik laporan dengan hasil penelitian-penelitian sebelumnya yang serupa atau mendekati.

c. BAB III: Pelaksanaan Magang

   Bab ini membahas mengenai kegiatan dan pelaksanaan magang pada proses pengembangan proyek yang diawali dengan manajemen proyek, penerapan metode pengembangan *Semi-Agile*, dan proses implementasi proyek.

d. BAB IV: Refleksi Pelaksanaan Magang

   Bab ini membahas mengenai refleksi dari proses dan dampak serta pembelajaran yang didapatkan selama pelaksanaan magang.

e. BAB V: Penutup

   Bab ini membahas mengenai kesimpulan dari keseluruhan pembahasan yang telah dilakukan pada setiap bab sebelumnya, serta saran yang disampaikan untuk ke depannya.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

Industri perbankan menjadi salah satu sektor dengan kompleksitas dan sensitivitas tinggi dalam pengelolaan data dengan menuntut penerapan sistem informasi yang andal, aman, dan terstruktur. Untuk mendukung operasional internal, termasuk pemantauan proyek, institusi perbankan secara umum mengadopsi teknologi-teknologi yang telah teruji dari segi stabilitas, skalabilitas, dan keamanan. Salah satunya adalah penggunaan *framework* Spring Boot berbasis Java yang banyak digunakan karena mendukung pengembangan sistem berskala besar, sedangkan pada sistem manajemen basis data seperti Microsoft SQL Server dan Oracle dipilih karena keandalan dan tingkat keamanannya yang tinggi. Praktik umum lain yang diterapkan dalam sistem perbankan adalah penggunaan pendekatan *Role-Based Access Control* (RBAC) untuk mengelola hak akses pengguna secara tepat, serta implementasi arsitektur REST API guna mendukung integrasi antarsistem.

Pada penelitian ini, pengembangan sistem dilakukan selama kegiatan magang penulis di PT Bank Syariah Indonesia Tbk (BSI), dengan mengikuti standar dan kebijakan teknologi yang berlaku di lingkungan perusahaan, termasuk kewajiban penggunaan bahasa pemrograman Java dengan *framework* Spring Boot serta Microsoft SQL Server sebagai basis data. Metode pengembangan yang digunakan mengacu pada pendekatan *Semi-Agile* untuk memberikan fleksibilitas dalam proses iteratif.

BSI sebagai bank syariah terbesar di Indonesia memiliki kekuatan pada infrastruktur teknologi yang solid, sumber daya manusia yang kompeten, serta budaya kerja berbasis nilai AKHLAK (Amanah, Kompeten, Harmonis, Loyal, Adaptif, dan Kolaborasi). Budaya ini mendorong terciptanya kolaborasi lintas tim dan adaptasi yang cepat terhadap perubahan kebutuhan. Di sisi lain, skala organisasi yang besar mengharuskan setiap proses adopsi teknologi baru harus melalui prosedur berlapis dan kepatuhan ketat terhadap regulasi keamanan data.

Faktor ekternal turut memberikan pengaruh signifikan. Tren digitalisasi perbankan dan perkembangan teknologi menghadirkan peluang untuk mengadopsi platform dan metode pengembangan modern, sementara persaingan dengan bank konvensional dan perusahaan teknologi finansial menuntut percepatan inovasi yang tetap aman dan terkontrol. Pertimbangan ini menjadi landasan strategis dalam pemilihan teknologi, di mana *framework* Spring Boot dipilih karena kemampuannya mendukung pengembangan sistem yang modular, skalabel, dan

mudah diintegrasikan, sedangkan Microsoft SQL Server menawarkan keandalan, kemanan, dan kepatuhan yang sesuai dengan standar industri perbankan.

Oleh karena itu, bab ini akan menguraikan teori-teori yang relevan sebagai dasar pengembangan sistem, meliputi *framework* Spring Boot, REST API, SQL Server, RBAC, serta metode pengembangan *Semi-Agile*.

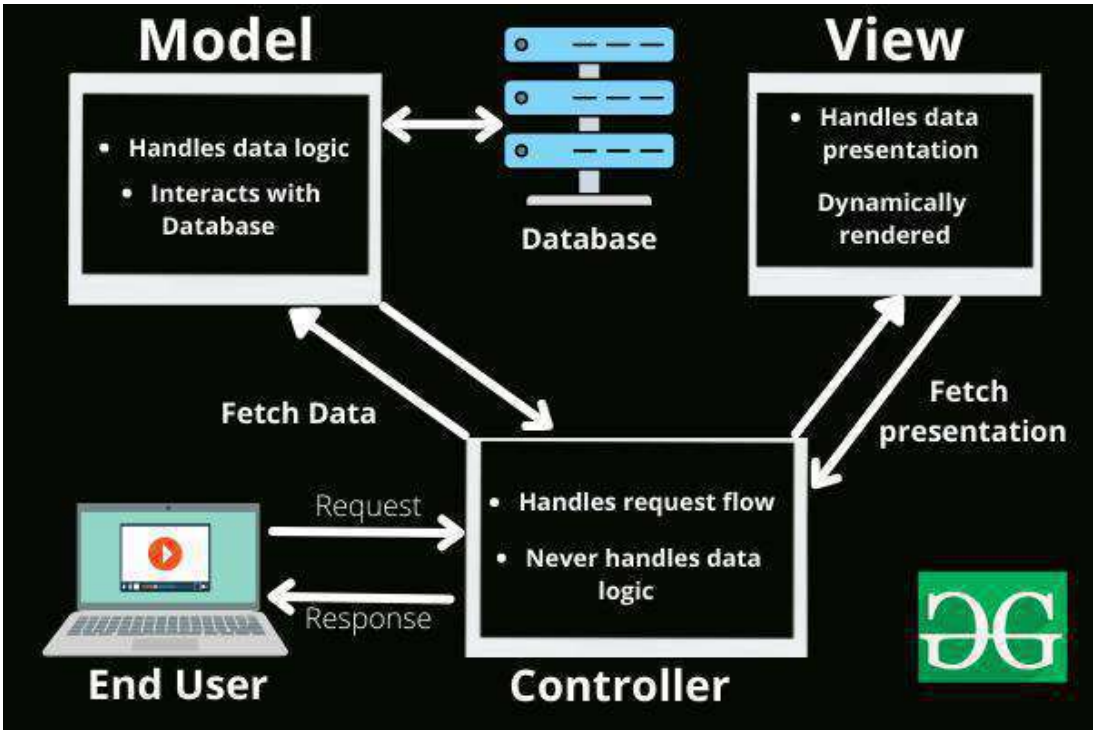
2.1 *Framework* Pengembangan Sistem dengan Spring Boot

Pemilihan teknologi dalam pengembangan sistem informasi perbankan tidak hanya didasarkan pertimbangan teknis, tetapi juga pada nilai strategis yang mendukung efisiensi proses pengembangan dan percepatan inovasi. Salah satu *framework* yang secara luas diadopsi dalam industri perangkat lunak adalah Spring Boot. Spring Boot merupakan kerangka kerja aplikasi web berbasis Java yang bersifat *open-source*, yang menekankan pada kemampuan *reusability* dan *pluggability* untuk mendukung pengembangan yang cepat. Dengan demikian, *framework* ini membantu menyederhanakan proses pengembangan aplikasi yang kompleks menjadi lebih efisien dan terstruktur (Luan, 2021).

Sebagai salah satu kerangka kerja berbasis Java yang populer, Spring Boot menawarkan kemudahan dalam pengembangan aplikasi skala besar dengan struktur modular yang dapat dikustomisasi. Dalam konteks perbankan, keunggulan ini menyediakan tingkat adaptabilitas yang tinggi terhadap proses bisnis kompleks dan terus berkembang. Spring Boot juga dikenal sebagai kerangka kerja yang kuat karena kemampuannya dalam menyederhanakan pengaturan dan konfigurasi aplikasi, sehingga membebaskan pengembang dari tanggung jawab untuk mengkhawatirkan pengaturan infrastruktur (Aggarwal & Pandit, 2023). Spring Boot mendukung *auto-configuration* melalui anotasi @SpringBootApplication, yang akan secara otomatis memindai dan menginisialisasi *bean* yang dibutuhkan, serta mengatur *dependency* sesuai konteks aplikasi.

Lebih jauh, menurut buku *Spring in Action, Sixth Edition* oleh Walls (2022), Spring Boot juga menyediakan berbagai fitur tambahan yang mendukung efisiensi operasional seperti *actuator*, pengaturan properti lingkungan yang fleksibel, dan dukungan pengujian tambahan yang meningkatkan efisiensi pengembangan aplikasi. Spring Boot juga menawarkan model pemrograman alternatif melalui skrip *Groovy* yang disebut Spring Boot CLI (*Command-Line Interface*). Dengan Spring Boot CLI, memungkinkan pengembang untuk dapat menulis seluruh aplikasi sebagai kumpulan skrip *Groovy* dan menjalankannya dari *command line*.

Selain itu, Spring Boot menerapkan pola arsitektur *Model-View-Controller* (MVC) sebagai pendekatan utama dalam pengembangan web. Lapisan *Controller* menerima *request* pengguna melalui protokol HTTP, kemudian meneruskan ke *Service* sebagai logika bisnis, dan hasilnya dikelola oleh *Model* untuk ditampilkan oleh *View.* Pola ini menjaga pemisahan tanggung jawab antarkomponen agar sistem mudah dikembangkan dan dipelihara. Gambar 2.1 berikut menggambarkan alur kerja dari pola arsitektur MVC tersebut dalam konteks interaksi antara pengguna, *controller*, *model*, dan *view*.



Gambar 2.1 Alur Arsitektur MVC
(Sumber: GeeksforGeeks, 2024)

Melalui arsitektur ini, setiap komponen memiliki tanggung jawab spesifik yang terpisah, sehingga memudahkan dalam pengelolaan kode, pengujian unit secara terpisah, serta fleksibel dalam pengembangan lebih lanjut, khususnya pada sistem informasi berskala besar seperti sistem perbankan.

Untuk mendukung manajemen komponen dalam pengembangan aplikasi, Spring Boot menerapkan prinsip *Inversion of Control* (IoC) melalui mekanisme *Dependency Injection* (DI). Dengan konsep ini, objek-objek atau komponen tidak perlu dibuat secara manual oleh pengembang, melainkan akan secara otomatis ditambahkan oleh Spring sesuai kebutuhan. Hal

ini dilakukan melalui anotasi seperti @Autowired, yang memungkinkan kelas-kelas seperti *controller* dan *service* menerima instansiasi dependensinya tanpa pengaturan eksplisit. Selain itu, Spring Boot menyediakan integrasi yang kuat dengan sistem basis data melalui Spring Data JPA. Dengan mengimplementasikan *JpaRepository*, pengembang tidak perlu menuliskan *query* SQL secara eksplisit untuk operasi dasar seperti *storage, read, update,* atau *delete* data. *Hibernate* ORM digunakan untuk melakukan operasi terhadap *database*, dan telah terintegrasi melalui Spring Data JPA yang menyederhanakan proses interaksi dengan *database*. Spring Boot secara otomatis membentuk perintah *query* berdasarkan nama *method* dan *entity* yang digunakan. Dukungan terhadap ORM (*Object Relational Mapping*) seperti *Hibernate* juga memastikan bahwa interaksi dengan *database* berlangsung secara efisien dan konsisten, sesuai kebutuhan sistem informasi perbankan yang menuntut integritas data tinggi.

Gambar 2.2 Contoh implementasi *JpaRepository* dalam Spring Boot
(Sumber: Juarez Junior, 2023)

Pada Gambar 2.2 memperlihatkan contoh deklarasi *interface EmployeeRepository* yang meng-*extend JpaRepository*, di mana *entity* yang digunakan adalah *Employee* dengan tipe ID berupa *Long*. Melalui pendekatan ini, pengembang dapat langsung menggunakan fungsi bawaan seperti *findAll(), save(), deleteById()* tanpa perlu membuat implementasi manual.

2.2 Integrasi Proyek Perbankan dengan REST API

REST (*Representational State Transfer*) adalah arsitektur API (*Application Programming Interface*) yang memungkinkan komunikasi *client-server* untuk aplikasi web melalui protokol HTTP (Prayogi et al., 2020). API merupakan antarmuka untuk penghubung komunikasi antar aplikasi atau sistem yang memungkinkan pertukaran data

secara efisien. Dalam pengembangan aplikasi web modern, API berperan penting dalam integrasi berbagai fitur dan layanan, khususnya ketika sistem terdiri dari banyak komponen yang berjalan secara independen.

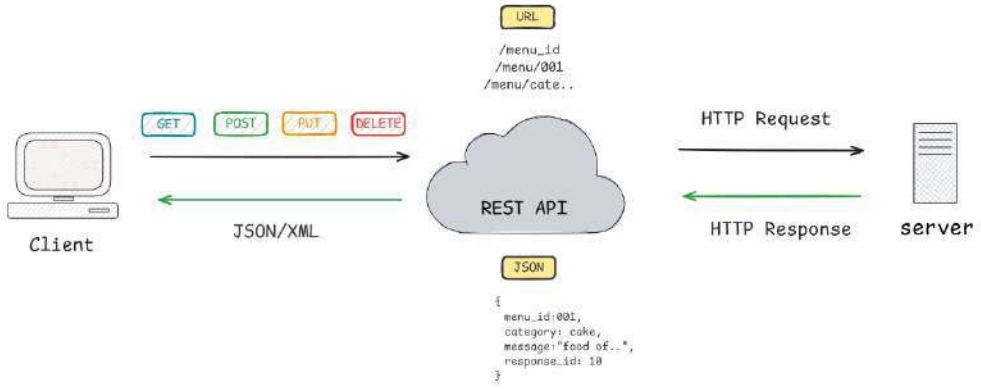
Implementasi API yang mengikuti prinsip dan aturan REST disebut RESTful API. RESTful API dirancang agar dapat mengakses dan memanipulasi *resource* melalui serangkaian *endpoint* menggunakan prinsip metode HTTP seperti **GET**, **POST**, **PUT**, dan **DELETE**. Keempat metode ini sejalan dengan prinsip operasi *Create*, *Read*, *Update*, dan *Delete* (CRUD) yaitu:

- **POST** digunakan untuk membuat (*create*) *resource* baru
- **GET** digunakan untuk mengambil (*read*) data dari *server*
- **PUT** digunakan untuk memperbarui (*update*) *resource* yang ada
- **DELETE** digunakan untuk menghapus (*delete*) *resource* tertentu

Prinsip CRUD ini sejalan dengan fungsi dasar dalam pengelolaan data seperti *INSERT, SELECT, UPDATE,* dan *DELETE* yang umum digunakan pada sistem penyimpanan berbasis SQL (Prayogi et al., 2020). Dengan mengacu pada prinsip CRUD, pengoperasian RESTful API menjadi lebih terstruktur dan mudah dipahami, baik dalam konteks integrasi sistem maupun pengembangan fitur internal.

REST API dinilai cocok untuk mendukung pengembangan aplikasi terdistribusi karena bersifat ringan dan mudah diimplementasikan, khususnya dalam ekosistem web dan mobile (Suzanti et al., 2020). Dalam praktiknya, *client* akan mengirimkan *request* ke *web-services* melalui RESTful atau REST API dari suatu *endpoint,* kemudian *server* akan memberikan *response* dengan data yang dibutuhkan dalam format umum, seperti JSON atau XML. Komunikasi ini dilakukan menggunakan protokol HTTP. Dengan struktur *request* yang dapat berisi elemen-elemen seperti *query* parameter, *headers*, *request body* tergantung dari spesifikasi *endpoint* yang diakses. Setiap *request* yang dikirim akan menghasilkan *response* dengan data serta kode status HTTP (seperti 200 OK, 201 Created, 400 Bad Request, atau 404 Not Found) yang memberikan informasi terkait keberhasilan atau kegagalan proses. Pendekatan ini memberikan fleksibilitas dan kejelasan dalam pengelolaan dan integrasi *resource* antar komponen sistem.

Dalam pengembangan sistem monitoring proyek berbasis web, pendekatan REST API digunakan sebagai pondasi komunikasi antara *front-end* dan *back-end*. Pemilihan ini mempertimbangkan kebutuhan sistem yang harus mendukung integrasi modular serta mampu menangani permintaan data dari berbagai antarmuka secara efisien. RESTful API memungkinkan *front-end* untuk berinteraksi secara langsung dengan layanan *back-end* melalui *endpoint-endpoint* yang telah didefinisikan, sekaligus menjaga struktur komunikasi agar tetap sederhana dan terstandarisasi.



Gambar 2.3 Alur komunikasi REST API

Gambar 2.3 menunjukkan bagaimana REST API bekerja sebagai perantara antara *client* dan *server* mealui protokol HTTP. *Client* mengirimkan HTTP *request* menggunakan metode seperti GET, POST, PUT, dan DELETE ke REST API, yang kemudian diteruskan ke *server* untuk diproses. *Response* dari *server* dikirim kembali ke *client* dalam format umum seperti JSON atau XML. Skema ini menggambarkan integrasi modular dan komunikasi data terstruktur yang mendukung arsitektur sistem monitoring berbasis web.

2.3 Infrastruktur Basis Data pada Industri Perbankan

Infrastruktur basis data merupakan komponen penting dalam sistem informasi modern. Basis data tidak hanya berfungsi sebagai tempat penyimpanan informasi, tetapi juga sebagai fondasi bagi proses transaksi, pelaporan, dan pengambilan keputusan yang berbasis data. Untuk memungkinkan *user* dapat berinteraksi dengan basis data, diperlukan perantara berupa perangkat lunak pengelola basis data yang dikenal dengan istilah *Database Management System* (DBMS). Sebagaimana dijelaskan dalam buku *Sistem Basis Data* oleh Tri Rachmadi

(2020), DBMS adalah perantara untuk *user* dengan *database*, sehingga interaksi antara *user* dengan *database* memerlukan bahasa pemrograman tertentu. Bahasa pemrograman tersebut umumnya terdiri dari dua jenis yaitu:

- ***Data Definition Language* (DDL).** Digunakan untuk membuat tabel baru, memuat indeks, ataupun mengubah tabel.
- ***Data Manipulation Language* (DML).** Digunakan untuk memanipulasi dan mengambil data dari *database*, seperti manambahkan data baru, menghapus data, atau mengubah data.

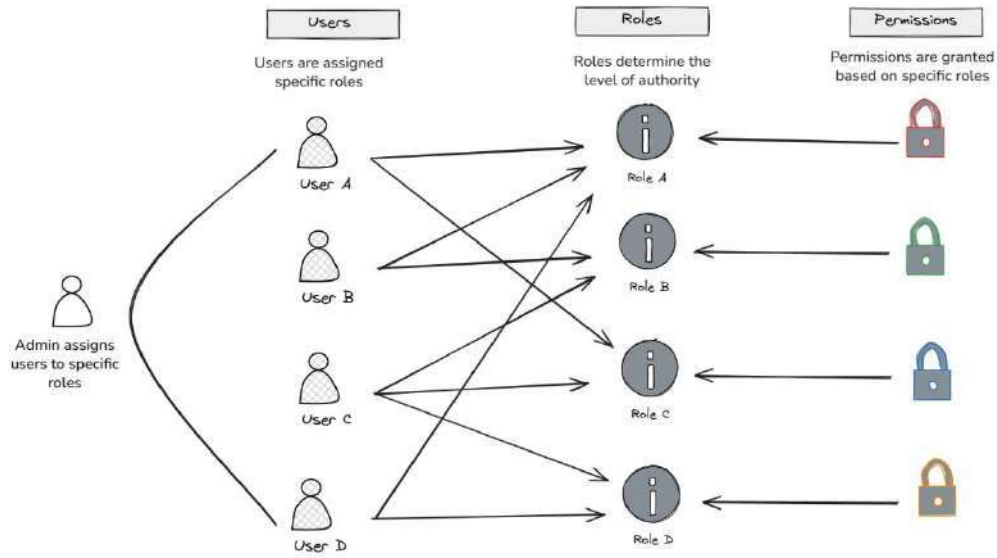
Dalam praktiknya, sistem informasi berskala besar seperti sektor perbankan lebih banyak mengandalkan *Relational Database Management System* (RDBMS). Penggunaan RDBMS menjadi pilihan utama karena kemampuannya dalam menjamin konsitensi data (*ACID compliance*) dengan mendukung transaksi yang kompleks, serta menyediakan kontrol akses yang granular. *Relational database* mengikuti prinsip ACID (*availability, consistency, isolation,* dan *durability*) dan dirancang untuk menangani data terstruktur (Khan et al., 2023). RDBMS dirancang untuk menangani data terstruktur, sehingga sangat sesuai dengan kebutuhan sistem yang mengandalkan skema data yang terorganisir dan relasional.

Salah satu RDBMS yang banyak digunakan dalam lingkungan enterprise adalah Microsoft SQL Server. Microsoft SQL Server merupakan sistem manajemen basis data relasional yang terhubung ke SQL Server *instance* atau *database*, dan berkomunikasi menggunakan Transact-SQL (T-SQL) (Microsoft Learn, 2025). Nevarez (2022) menjelaskan bahwa di dalam SQL Server terdapat dua komponen utama, yaitu *storage engine* dan *relational engine*, yang masing-masing bertugas untuk mengelola transfer data dari disk ke memori serta merancang dan menjalankan rencana eksekusi *query* dengan optimal.

Dalam pengembangan sistem monitoring proyek, SQL Server dipilih sebagai fondasi basis data karena memenuhi kebutuhan keamanan, stabilitas, dan skalabilitas yang tinggi dalam pengelolaan data internal. Pemilihan ini juga disesuaikan dengan ekosistem teknologi yang telah diterapkan secara luas di lingkungan perusahaan, sehingga mempermudah proses integrasi antar sistem dan menjaga konsistensi standar pengembangan. Selain itu, SQL Server mendukung integrasi dengan teknologi *back-end* seperti Spring Boot melalui berbagai konektor JDBC, serta menyediakan fitur-fitur tambahan seperti *stored procedures, indexing,* dan *role-level security* yang sesuai dengan standar tata kelola data di lingkungan perbankan.

2.4 Penguatan Tata Kelola Akses Informasi melalui *Role-based Access Control*

Kebutuhan dalam mengontrol akses terhadap informasi dan data menjadi aspek yang sangat penting dalam menjaga keamanan serta integritas sistem, terutama pada sistem yang menangani entitas pengguna dan data sensitif. Salah satu pendekatan yang secara luas diterapkan dalam pengaturan akses adalah *Role-based Access Control* (RBAC), yakni model kontrol akses yang membatasi akses ke sumber daya dalam sistem berdasarkan peran yang dimiliki oleh pengguna (Penggalih & Silmina, 2025). Penerapan ini memungkinkan organisasi untuk mengelola izin akses dengan lebih sistematis dan efisien.



Gambar 2.4 Struktur mekanisme *Role-Based Access Control* (RBAC)

Penggunaan model arsitektur keamanan RBAC pada sistem, menyediakan kontrol yang manajemen dan pengelolaan yang fleksibel menggunakan mekanisme *dual privilege mapping* yaitu pengguna (*user*) ke peran (*role*), dan dari peran (*role*) ke hak akses (*permissions*). Arsitektur sistem RBAC yang ditunjukkan pada Gambar 2.4 menggambarkan bagaimana sistem ini bekerja dengan titik fokus pada peran admin sebagai pengontrol utama dalam menetapkkan hak akses dan informasi pada pengguna. Setiap peran memiliki tingkatan otorisasi tertentu yang menentukan hak akses pengguna terhadap fitur atau data dalam sistem. Model ini menyediakan kerangka kerja keamanan yang fleksibel, terutama dalam lingkungan sistem dengan struktur organisasi hierarkis.

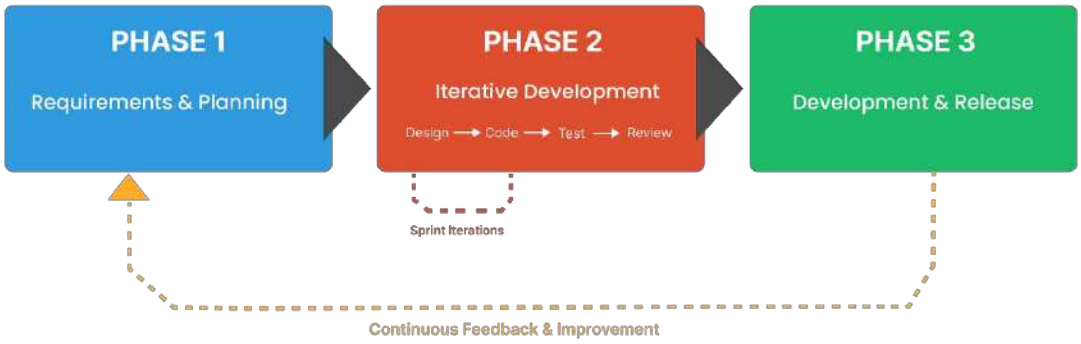
Dalam penerapan pada pengembangan sistem monitoring proyek, RBAC digunakan untuk mengakomodasi struktur kewenangan yang ada di dalam organisasi. Setiap pengguna

sistem seperti, *Project Admin, Department Head,* PIC*,* maupun *Team Member*, diberikan hak akses yang berbeda-beda sesuai dengan peran dan tanggung jawabnya terhadap proyek. Pendekatan ini membantu memastikan bahwa hanya individu yang relevan yang dapat melihat atau mengelola data tertentu, selaras dengan prinsip *least privilege*. Dengan adanya struktur kontrol akses yang berbasis peran ini, sistem menjadi lebih mudah dipelihara dari segi keamanan, lebih mudah diaudit, dan lebih aman terhadap potensi penyalahgunaan akses. RBAC bukan hanya berperan sebagai lapisan keamanan, namun juga sebagai bagian dari tata kelola sistem yang mendukung efisiensi dan kejelasan fungsi masing-masing pengguna dalam siklus hidup proyek.

2.5 Metode pengembangan sistem *Semi-Agile*

Metode *Semi-Agile* merupakan gabungan metode antara *Waterfall* dan *Agile* yang dikenal dengan *Semi-Agile* atau *Hybrid.* Pendekatan dari metode yang digunakan dari keduanya adalah adanya perencanaan dan dokumentasi yang terstruktur yang masuk dalam unsur model *Waterfall* dan juga penekanan fleksibilitas, iterasi, dan kolaborasi dalam pengembangan yang ada pada *Agile*. Metode ini termasuk bagian dari model kerangka kerja *System Development Life Cycle* (SLDC). SDLC atau *System Development Life Cycle* merupakan metode umum yang dipakai untuk mengembangkan sistem informasi (Wibowo & Idris, 2025). Metode ini memungkinkan tim untuk beradaptasi dengan perubahan dan memperbaiki produk secara efektif (Saleh & Paputungan, 2024). Pada penerapan dari metode *Semi-Agile* dalam pengembangan sistem ini dibagi menjadi tiga fase utama sebagaimana ditampilkan pada Gambar 2.5. Visualisasi siklus tersebut merupakan hasil adaptasi dari pendekatan *Semi-Agile*

yang disesuaikan dengan kebutuhan tim pengembang dalam proyek ini, guna mencerminkan alur kerja yang terstruktur namun tetap fleksibel.



Gambar 2.5 *Semi-Agile Lifecycle*

A. *Requirements & Planning*

Pada fase ini dilakukan perumusan kebutuhan sistem serta penyusunan rencana kerja proyek. Hal ini melibatkan sejumlah proses untuk mengumpulkan persyaratan sesuai dengan persyaratan dan tuntutan pengguna dan pemangku kepentingan produk perangkat lunak (Romindo et al., 2023). Proses tersebut dapat diperoleh dari dokumen, diskusi, dan observasi untuk lebih memahami kebutuhan sistem yang akan dikembangkan. Dilanjutkan dengan penyusunan rencana kerja atau *timeline* serta pembagian peran dalam tim. Fase ini menghasilkan pemahaman yang *solid* dan menyeluruh mengenai tujuan, ruang lingkup, serta spesifikasi teknis dari sistem yang akan dikembangkan.

B. *Iterative Development*

Fase ini mengadopsi prinsip-prinsip *agile* dengan proses pengembangan dilakukan melalui beberapa siklus pendek atau *sprint*. Setiap *sprint* terdiri dari tahapan desain, pengkodean, pengujian, dan evaluasi. Pendekatan ini memungkinkan kegiatan spesifikasi, pengembangan, dan validasi di serangkaian versi, dengan setiap versi menambahkan fungsionalitas ke versi sebelumnya (Dawis et al., 2023). Penggunaan teknologi *back-end* berupa bahasa pemrograman Java, *framework* Spring Boot, dan *database* Microsoft SQL Server pada sistem mengikuti ekosistem perusahaan akan menghasilkan fitur-fitur yang sesuai dengan kebutuhan dan perencanaan awal. Fokusnya fase ini akan menghasilkan fitur secara bertahap dan memungkinkan *feedback* cepat dari pengguna untuk peningkatan kualitas sistem.

C. *Development & Release*

Fase terakhir dalam siklus yang berfokus pada penyelesaian akhir sistem dan proses rilis ke lingkungan produksi, namun saat ini sistem belum masuk ke lingkungan produksi. Aktivitas utama pada tahap ini meliputi integrasi komponen, validasi akhir terhadap fitur yang dikembangkan, serta distribusi sistem kepada pengguna. Selain sistem yang fungsional dan fitur yang sesuai dengan kebutuhan, fase ini juga menghasilkan dokumentasi sebagai bagian proyek.

Penerapan metode *Semi-Agile* dalam proyek ini juga merefleksikan karakteristik lingkungan kerja di industri perbankan. Metode *Agile* murni kerap sulit diadopsi sepenuhnya karena adanya regulasi ketat, kebutuhan dokumentasi yang komprehensif, serta proses persetujuan berlapis yang wajib dipenuhi sebelum sebuah fitur dapat diimplementasikan. Kondisi ini menuntut adanya keseimbangan antara kepatuhan prosedural dan kemampuan beradaptasi terhadap perubahan kebutuhan. Dalam konteks proyek ini, pendekatan dengan metode *Semi-Agile* tidak ditetapkan secara formal oleh mentor, melainkan terbentuk secara alami dari pola kerja tim dalam mengembangkan proyek tersebut. Kombinasi antara perencanaan dan dokumentasi yang rapi dengan siklus iteratif dan kolaborasi intensif memingkinkan tim untuk tetap mematuhi standar perusahaan sekaligus responsif terhadap masukan dan dinamika selama proses pengembangan. Pendekatan ini menjadikan *Semi-Agile* sebagai model yang selaras dengan kebutuhan proyek, baik dari sisi teknis maupun kepatuhan terhadap kebijakan perusahaan.

2.6 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. Berikut hasil penelitian tersebut yang dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka Penelitian

No	Judul	Tujuan	Teknologi	Metode	Hasil
1.	Pengembangan Sistem Monitoring My School dengan menggunakan NetBeans (Trista & Wisdariah, 2021)	Untuk memberikan solusi manajerial yang lebih efisien dalam pengelolaan data siswa dan aktivitas sekolah, serta proses pemantauan kehadiran, penilaian akademik dan laporan kegiatan lainnya (Trista & Wisdariah, 2021).	Java, Spring Boot, NetBeans IDE, MySQL	Waterfall	Sistem dikembangkan dengan fitur pemantau kehadiran, nilai akademik, serta laporan kegiatan lainnya, yang membantu peningkatan efisiensi pengelolaan data dan monitoring di sekolah, serta memudahkan akses bagi guru, kepala sekolah, dan orang tua siswa (Trista & Wisdariah, 2021).
2.	*Implementation of Online Network File Management System based on Spring Boot and Vue* (Lai, 2025)	Untuk mengembangkan aplikasi ringan yang memungkinkan pengguna melakukan *login, registration, file upload,* serta pencatatan informasi file seperi *upload time* dan *user roles* yang ditujukan untuk penggunaan personal maupun bisnis skala kecil (Lai, 2025).	Spring Boot, Vue, MySQL, RESTful API, JWT, RBAC	Iterative	Sistem berhasil mengimplementasikan fitur *login*, *file operations*, serta kontrol akses menggunakan *Role-Based Access Control* (RBAC) untuk dua peran pengguna (administrator dan *general user*). Sistem berjalan dengan baik, mendukung kebutuhan *file management* yang aman dan sederhana (Lai, 2025).

3.	*Access Management Solution to Corporate Service* (Hyytiäinen, 2024)	Untuk mengevaluasi sistem kontrol akses layanan internal perusahaan dan merancang solusi baru yang lebih aman dan fleksibel berdasarkan kebutuhan pengguna dan regulasi (Hyytiäinen, 2024).	Java, Spring Framework, RBAC, PostgreSQL, REST API, LDAP, JWT	System Development	Penerapan *Role-Based Access Control* (RBAC) pada tingkat *endpoint* REST API memungkinkan pengelolaan hak akses yang lebih terstruktur sesuai peran pengguna. Solusi ini memperbaiki kekurangan sistem sebelumnya yang rawan kompleksitas dan sulit dipelihara, sehingga lebih selaras dengan kebutuhan otorisasi layanan korporat (Hyytiäinen, 2024)
4.	*Extension of an Enterprise Web Application for Top-Management Reporting: A Modular Approach to Web Application Development* (Torredimare, 2024)	Untuk meningkatkan efisiensi pelaporan manajemen atas dengan menambahkan modul fungsional ke aplikasi web SmartSpend (Torredimare, 2024)	Spring Boot, SQL Server, REST API, Freemarker	Modular Web Development	Penggunaan Spring Boot dengan arsitektur modular memungkinkan pengembangan fitur pelaporan manajemen yang terintegrasi dan fleksibel menggantikan proses manual berbasis Excel. Integrasi dengan SQL Server memungkinkan akses data pengadaan secara *real-time*. (Torredimare, 2024).
5.	*Streamlining Project Management: Enchancing Efficiency and Progress Tracking Through a CRM Website* (Nguyen, 2024)	Untuk mengembangkan sistem CRM berbasis web yang terintegrasi dengan fitur manajemen proyek untuk meningkatkan efisiensi kerja, alokasi tugas, dan pelacakan progress proyek secara *real-time* (Nguyen, 2024).	Spring Boot, Java, React, Redux, MySQL, JWT, RBAC	System Development	Sistem ini berhasil mengintegrasikan CRM dan fitur manajemen proyek dalam satu platform. Penerapan RBAC dan JWT memungkinkan otorisasi akses yang terstruktur, sedangkan integrasi REST API dengan Spring Boot mempercepat pelacakan progres

					proyek secara *real-time* (Nguyen, 2024).

Berdasarkan kajian pusataka dari penelitian yang telah dipaparkan pada Tabel 2.1, dapat disimpulkan bahwa seluruh penelitian terdahulu memiliki fokus utama dalam pengembangan sistem informasi berbasis web menekankan pada efisiensi manajemen data, pengguna *framework* Spring Boot sebagai inti pengembangan *back-end*, serta penerapan basis data relasional seperti MySQL, SQL Server, maupun PostgresSQL. Beberapa penelitian juga menerapkan pendekatan REST API dalam komunikasi antar komponen sistem serta konsep *Role-Based Access Control* (RBAC) sebagai mekanisme pengelolaan hak akses pengguna. Penelitian-penelitian tersebut dikembangkan dengan beragam pendekatan teknologi, baik dalam konteks sistem informasi sekolah, manajemen file, kontrol akses layanan internal, pelaporan manajemen, maupun sistem manajemen proyek.

Secara umum, terdapat kesamaan antara kelima penelitian tersebut dan laporan akhir ini, antara lain:

- Seluruh sistem dikembangkan dengan mengggunakan *framework* Spring Boot.
- Mayoritas penelitian mengimplementasikan REST API dan *database* relasional sebagai infrastruktur utama penyimpanan dan distribusi data.
- Tiga dari kelima penelitian menerapkan konsep *Role-Based Access Control* (RBAC) dalam mengelola hak akses pengguna, yang mendukung keamanan dan struktur otorisasi sistem.
- Fokus sistem cendrung pada peningkatan efisiensi, visibilitas informasi, dan otomatisasi proses, yang juga menjadi tujuan dari sistem monitoring proyek yang dikembangkan dalam laporan akhir ini.

Adapun perbedaan antara masing-masing penelitian dan laporan akhir ini meliputi:

- Penelitian Trista & Wisdariah (2021) berfokus pada sistem monitoring akademik sekolah, sedangkan laporan akhir ini berada dalam konteks enterprise untuk pengawasan progres proyek antar divisi.
- Penelitian Lai (2025) dan Hyytiäinen (2024) lebih menekankan pada pengelolaan fille digital dan otorisasi layanan internal, bukan monitoring progres aktivitas tim proyek secara menyeluruh.
- (Torredimare, 2024) berfokus pada pengembangan modul pelaporan manajemen dalam sistem pengadaan, sedangkan laporan ini lebih menekankan pengelolaan proyek dari sisi divisi IT secara terstruktur.
- Penelitian Nguyen (2024) mengintegrasikan CRM dan manajemen proyek, namun tidak menjelaskan secara detail penerapan manajemen progres pada lingkungan organisasi yang terdistribusi secara hierarkis.
- Metode yang digunakan pada beberapa penelitian juga bervariasi, seperti pendekatan iteratif, *modular development*, dan pengembangan sistem secara umum tanpa eksplisit menyebut model *waterfall* atau *agile*.

BAB III
PELAKSANAAN MAGANG

3.1 Aktivitas Magang

Penulis mengikuti proses program BSI IT *Digilab Intership* yang diselenggarakan oleh BSI sebagai program kerja sama magang kampus merdeka dengan beberapa Universitas yang berfokus dan tergabung pada unit kerja *IT & Digital Development Group* (IDG) yang dilaksanakan secara *on*-site pada Digilab BSI yang berada di Yogyakarta. Program ini bertujuan untuk memberikan peluang pembelajaran, pengembangan keterampilan, dan pengalaman praktis bagi talenta muda yang tertarik pada dunia teknologi dan pengembangan perangkat lunak, sekaligus mendukung upaya pada Bank BSI dalam membangun ekosistem digital yang berkelanjutan.

Pada program magang ini, ada beberapa aktivitas atau tahapan yang harus diikuti. Aktivitas yang penulis lalui pada proses magang seperti berikut:

1. **Induction & General Banking**

   Kegiatan ini berfokus pada pengenalan lingkungan magang, perkenalan antar petinggi dan pengurus program serta adanya sesi pemaparan materi dasar perbankan secara umum dan syariah. Kegiatan ini berlangsung selama dua hari yang dilaksanakan di BSI KC Yogyakarta.

2. **Branch X**

   Pada kegiatan ini peserta magang melakukan *On the Job Training* (OJT) di berbagai Kantor Cabang Pembantu (KCP) di area Yogyakarta yang sudah ditentukan. Fokus pada kegiatan ini adalah untuk memberikan pengalaman secara langsung terkait operasional bisnis BSI serta dapat mengidentifikasi dan mengeksplorasi *IT Issues Takeaways*. Kegiatan ini berlangsung selama tiga hari yang dilaksanakan sesuai dengan KCP yang telah ditentukan.

3. **IT Fundamental & Specialist**

   Rangkaian kegiatan yang berfokus dengan teknologi yang digunakan Bank Syariah Indonesia (BSI) yang disampaikan langsung oleh ahli dari perwakilan kantor pusat BSI. Peserta magang akan mengikuti pemaparan dua modul IT yang masing-masing modul dipaparkan selama 1 minggu yang dilaksanakan di BSI KC Yogyakarta dan BSI UMKM Center Yogyakarta.

4. **On The Job Assignment Project**

Pada kegiatan ini peserta magang akan terlibat langsung dalam pengembangan proyek yang akan dipaparkan oleh beberapa departemen, yang mencakup penjelasan umum mengenai tugas masing-masing departemen dalam organisasi serta proyek yang akan diberikan kepada peserta magang. Peserta magang akan memilih beberapa departemen tersebut sesuai dengan minat dan kebutuhan pengembangan diri. Selanjutnya, peserta akan mengimplementasikan teknologi IT yang digunakan BSI. Kegiatan ini dilakukan selama lima bulan yang dilaksanakan di Digilab BSI Yogyakarta.

Departemen-departemen yang terlibat dalam program ini tergabung dalam satu kesatuan grup bernama *IT & Digital Development Group* (IDG). Grup ini terdiri dari tujuh departemen yang memiliki peran penting dalam pengembangan teknologi dan sistem informasi di BSI. Adapun daftar departemen adalah sebagai berikut:

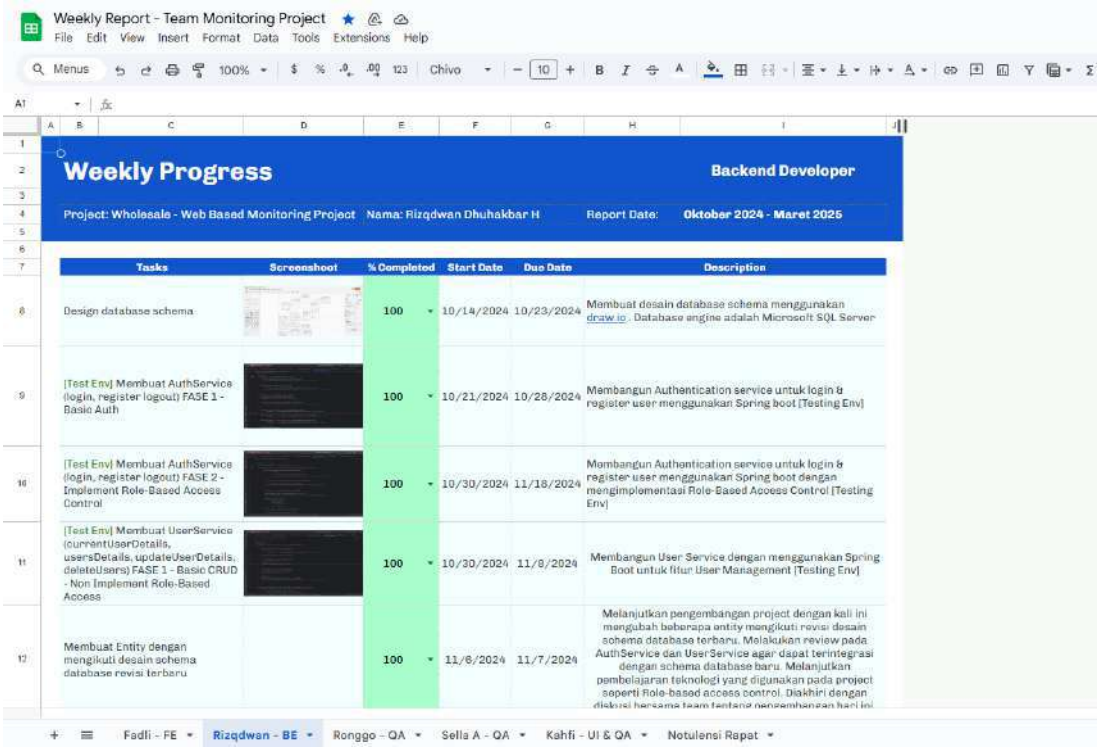
- WHA (*IT Wholesale & Office Automation Development*)
- CBS (*IT Core Banking Development*),
- PIN (*IT Payment & Integration Development*)
- FCG (*IT Financing & Collection Development*)
- SBP (*IT Service Banking Platform*)
- CBP (*IT Customer Banking Platform*)
- MIS (*IT Management Information System*)

Dalam pelaksanaan program magang ini, penulis ditempatkan pada departemen WHA (*IT Wholesale & Office Automation Development*) dengan peran sebagai *Back-end Developer.* Penulis berkesempatan untuk berkontribusi dalam pengembangan sistem monitoring proyek berbasis web, yang digunakan untuk mempermudah dalam mengawasi proyek-proyek internal maupun eksternal antar departemen dalam lingkup IDG. Selama proses magang, penulis mendapatkan bimbingan dari dua mentor teknis di departemen WHA yakni Bapak Abdul Tholib dan Bapak Arif Sabarudin serta arahan dari *field supervisor* yaitu Bapak Taufiq Galang. Proses pengembangan sistem monitoring proyek berbasis web dilaksanakan selama kurang lebih 5 bulan. Sistem dikembangkan dengan menggunakan metodologi *Semi-Agile* yang menggabungkan unsur-unsur dari model *Waterfall* seperti perencanaan dan dokumentasi terstruktur serta dari unsur *Agile* seperti fleksibilitas, iterasi, dan kolaborasi tim.

3.2 Manajemen Proyek

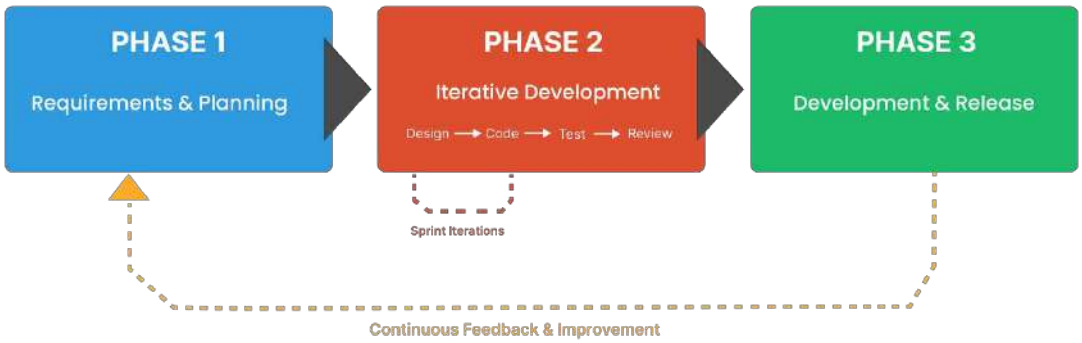
Pengembangan sistem monitoring proyek berbasis web dikembangkan menggunakan metodologi *Semi-Agile*. Penggunaan metode *Semi-Agile* sebagai pendekatan dalam manajemen proyek dipilih karena memberikan keseimbangan antara struktur perencanaan dan fleksibilitas dalam proses pengembangan. Metode ini menggabungkan prinsip-prinsip dari metode *Agile* seperti iterasi dan umpan balik cepat serta *Waterfall* seperti perencanaan dan dokumentasi. Selama proses pengembangan, tim secara rutin mengadakan *weekly meeting* secara daring melalui Microsoft Teams sebagai sarana evaluasi dan pemantauan perkembangan proyek. Biasanya tim akan melakukan diskusi singkat setelah *weekly meeting* untuk membahas hasil pertemuan dan melakukan perencanaan *task* ke depan. Microsoft Teams menjadi media komunikasi utama dengan mentor dan *field supervisor* selama masa pengembangan proyek. Selain pertemuan daring, mentor juga beberapa kali melakukan kunjungan langsung ke Digilab BSI untuk meninjau kemajuan pengembangan proyek serta mendiskusikan kendala yang dihadapai. Meskipun menerapkan prinsip *Semi-Agile*, proses pengembangan proyek tidak sepenuhnya mengikuti kerangka kerja seperti *Scrum*. Proses yang dijalankan bersifat fleksibel dan menyesuaikan dengan pola kerja tim, namun tetap mempertahankan prinsip-prinsip iteratif dan kolaborasi dalam metode tersebut.

Dalam pengembangan proyek ini tim menggunakan Microsoft Excel sebagai *tools* manajemen proyek. *Tools* manajemen proyek seperti Jira atau Trello tidak diadopsi secara menyeluruh karena beberapa anggota tim belum terbiasa dan membutuhkan waktu adaptasi yang cukup lama. Namun, penulis memanfaatkan Trello secara mandiri sebagai alat bantu dalam melakukan notulensi selama pengembangan. Penggunaan Microsoft Excel selama pengembangan proyek berlangsung dikostumisasi sedemikian rupa yakni untuk mencatat berbagai aspek manajemen proyek, termasuk notulensi rapat minggunan, dokumentasi pengembangan, serta pelacakan progres masing-masing anggota tim. Gambar 3.1 menampilkan salah satu lembar kerja Excel yang memuat *weekly report* dari penulis sebagai *back-end developer* yang mencakup kolom *tasks*, *screenshot* pekerjaan, *presentase* penyelesaian, tanggal mulai dan akhir, serta deskripsi tugas yang dikerjakan. Pendekatan ini dinilai cukup efektif dalam mengatur ritme kerja dan membantu mentor dalam memantau kontribusi setiap anggota tim.



Gambar 3.1 Penerapan *tools* Excel pada proyek

Untuk mendukung efektivitas proses pengembangan selama magang, tim mengadaptasi proses kerja pada pengembangan proyek ke dalam tiga fase utama berdasarkan pendekatan *Semi-Agile*. Tiga fase ini terdiri dari: *Requirements & Planning*, *Iterative Development*, dan *Development & Release*, sebagaimana ditunjukkan pada Gambar 3.2.



Gambar 3.2 Fase pengembangan pada *Semi-Agile*

Diagram ini menggambarkan alur proses pengembanan sistem yang diterapkan selama pelaksanaan proyek, termasuk mekanisme iterasi serta umpan balik berkelanjutan yang

menjadi ciri khas dari pendekatan *Semi-Agile*. Detail dari setiap fase pengembangan proyek akan dijelaskan lebih lanjut pada Subbab 3.3 mengenai Pelaksanaan Proyek.

3.3 Pelaksanaan Proyek

Pada pengembangan proyek monitoring proyek berbasis web ini penulis bertanggung jawab pada bagian *back-end*. Subsubbab 3.3.1 akan menjelaskan tentang perencanaan proyek yang masuk dalam tahapan pada *Semi-Agile* yakni *Requirements & Planning*. Pada Subsubbab 3.3.2 dilanjutkan dengan penjelasan tentang pengembangan proyek yang masuk dalam tahap *Iterative Development*, dan *Development & Release*.

3.3.1 Perencanaan Proyek

Sistem ini dikembangkan untuk memenuhi kebutuhan internal dari departemen WHA (*IT Wholesale & Office Automation Development*) yang bertujuan untuk mempermudah kepala departemen dalam memantau, mengelola dan melakukan manajemen berbagai proyek yang berjalan. Tahapan awal proyek dimulai dengan perencanaan sistem secara menyeluruh, termasuk pemilihan teknologi dan perumusan alur proses bisnis yang akan diterapkan. Diskusi awal dilakukan secara langsung bersama mentor di Digilab BSI, dan diskusi selanjutnya akan diadakan secara daring melalui *Microsoft Teams*. Tim mendapatkan rancangan awal prototipe berupa desain figma dari sistem yang akan dikembangkan selama diskusi awal bersama mentor. Sistem ini direncanakan akan memiliki empat peran (*role*) yang masing-masing peran akan memiliki hak akses tersendiri. Peran yang ada pada proyek ini yakni

- *Project Admin*

  Memiliki akses penuh terhadap semua fitur dan service pada sistem. *Project Admin* akan mengelola administrasi serta melakukan manajemen proyek pada departemen yang telah ditentukan.
- *Department Head*

  Memiliki akses untuk melakukan penugasan proyek, memperbarui informasi proyek, serta melakukan pengawasan terhadap jalannya proyek.
- *Person In Charge* (PIC)

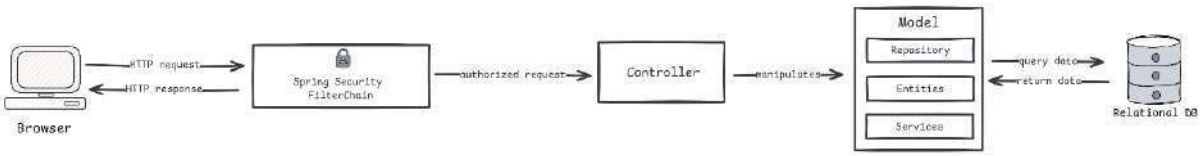
  Memiliki akses untuk melakukan penugasan proyek, memperbarui status proyek, mengelola kemajuan aktivitas proyek dan menambahkan anggota tim yang terlibat dalam proyek.
- *Team Member*

Memiliki akses terbatas yaitu hanya dapat melihat semua proyek pada departemen terkait dan proyek yang ditugaskan.

Dari sisi teknologi, sistem ini dikembangkan menggunakan bahasa pemrograman Java dengan *framework* Spring Boot sebagai *basis* nya dan menggunakan Vite React sebagai *interface build*-nya. Untuk *database* menggunakan Microsoft SQL Server menyesuikan dengan ekosistem teknologi yang digunakan pada lingkungan BSI. Sistem ini akan mencakup fitur-fitur utama seperti *management departmen*, *applications*, *projects*, *users*, serta penugasan proyek kepada pengguna. Seluruh fitur dan layanan tersebut menerapkan mekanisme *Role-based Access Control* (RBAC), untuk mengatur akses pengguna sesuai dengan peran yang dimiliki.

3.3.1.1 Arsitektur Sistem Back-end

Sistem monitoring proyek berbasis web dikembangkan dengan menggunakan *framework* Spring Boot. Sistem ini menggunakan pola arsitektur *Model-View-Controller* (MVC) yang bersifat *monolith*. Pada sisi *back-end*, arsitektur ini diimplementasi tanpa komponen *View*, karena antarmuka dikembangkan secara terpisah menggunakan teknologi *front-end* (*client-side-rendering*). Dengan demikian, komponen *View* direpresentasikan melalui HTTP *response* yang mengirimkan data dari sistem dalam format JSON.



Gambar 3.3 Alur Arsitektur Sistem

Arsitektur pada sistem monitoring proyek berbasis web diimplementasikan seperti pada Gambar 3.3. Sistem ini mendukung pengembangan RESTful Web Service berbasis Spring MVC dengan penggunaan *dependency* spring-boot-starter-web, sekaligus menyediakan *embedded container* Apache Tomcat untuk memudahkan pengembangan dan penyebaran aplikasi. Arsitektur sistem terdiri dari beberapa komponen utama yaitu:

- **Browser**

  Sebagai pihak *client* yang mengirimkan *request* melalui protokol HTTP.

- **Spring Security FilterChain**

sebagai lapisan pengaman yang memfilter *request* dan memastikan hanya pengguna yang terautentikasi dan memiliki otorisasi yang valid yang dapat melanjutkan akses ke *endpoint*

- **Controller**

  Sebagai lapisan untuk menerima *request* dari *client* dan mengelola alur logika sistem, kemudian meneruskan ke lapisan layanan (*service*).

- **Model**

  Lapisan ini terdiri dari tiga bagian, yaitu *Services*, *Entities,* dan *Repository* dengan menangani fungsi yang berbeda. Komponen *Services* berfungsi sebagai penghubung antara *Controller* dan *Repository,* serta bertanggung jawab dalam mengatur alur bisnis dan pemanggilan data. *Entities* merepresentasikan struktur data sesuai dengan tabel dalam basis data, sementara *Repository* digunakan untuk penghubung antara layanan dan *database* dalam berinteraksi langsung pendekatan *object-relational mapping* (ORM) melalui *Hibernate* dan mekanisme *query* dari Spring Data JPA.

- **Relational Database**

  Sebagai basis data tempat penyimpanan data permanen yang diakses oleh lapisan *repository*.

3.3.2 Pengembangan Proyek

Dalam fase pengembangan proyek ini, pendekatan iteratif dilakukan dengan merujuk pada metode *Semi-Agile* yang telah dijelaskan sebelumnya. Untuk mengelola kebutuhan sistem dan memetakan alur pengembangan, penulis bersama tim menggunakan *backlog* sebagai acuan utama. Setiap *backlog* disusun berdasarkan kebutuhan pengguna dan diskusi dengan mentor dari departemen WHA. *Backlog* ini berisi deskripsi singkat fitur atau layanan, beserta *acceptance criteria* yang dijadikan tolak ukur keberhasilan implementasinya. Fitur dan layanan yang penulis kerjakan antara lain fitur autentikasi dan registrasi, fitur *Dashboard*, fitur *Project management*, fitur *user management*, fitur *app management*, fitur *group management*, dan fitur *department management*.

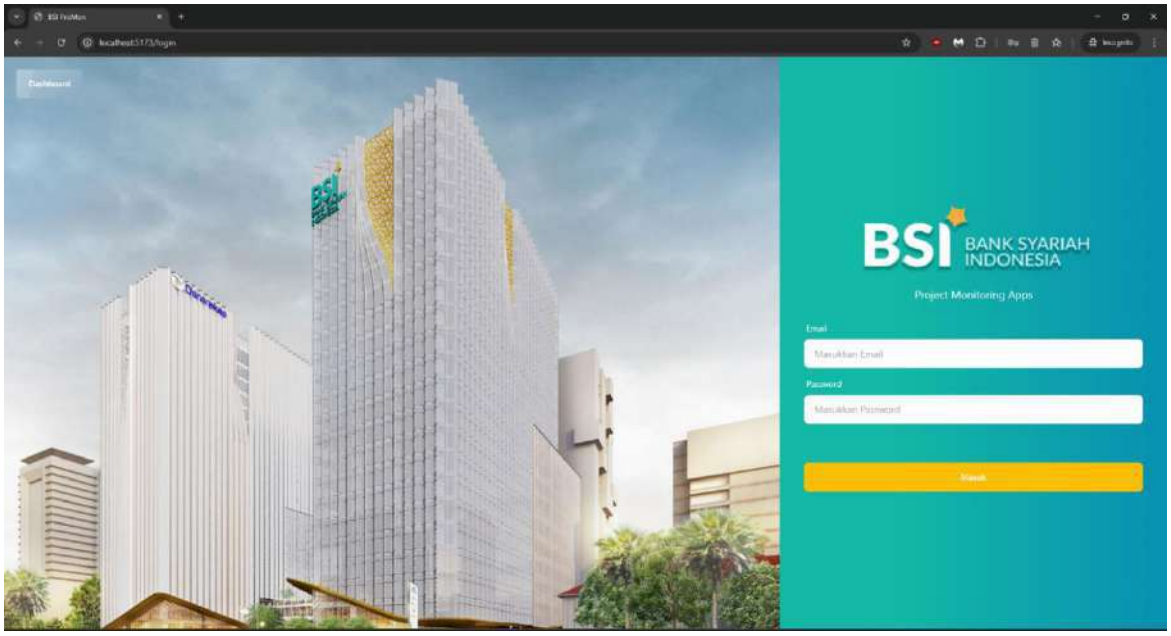
3.3.2.1 Fitur autentikasi dan registrasi

Tabel 3.1 *Backlog* fitur autentikasi dan registrasi

Backlog	*Acceptance Criteria*
Sebagai insan BSI dapat melakukan autentikasi pada sistem monitoring proyek berbasis web	Memasukkan *email* dan *password* sebagai kredensial
	Email menjadi *username* bagi *user*
	Terdapat validasi *email*
	Terdapat validasi pada *password* minimal delapan karakter, simbol, dan angka
	Mekanisme *login* menggunakan JWT
Sebagai *Project Admin* dapat melakukan registrasi pada sistem monitoring proyek berbasis web	Hanya dapat melakukan registrasi pengguna untuk departemen sendiri
	Melakukan input data dengan mengirimkan nama lengkap, *email*, *role user*, posisi atau jabatan, departemen, *password*, dan konfirmasi *password*

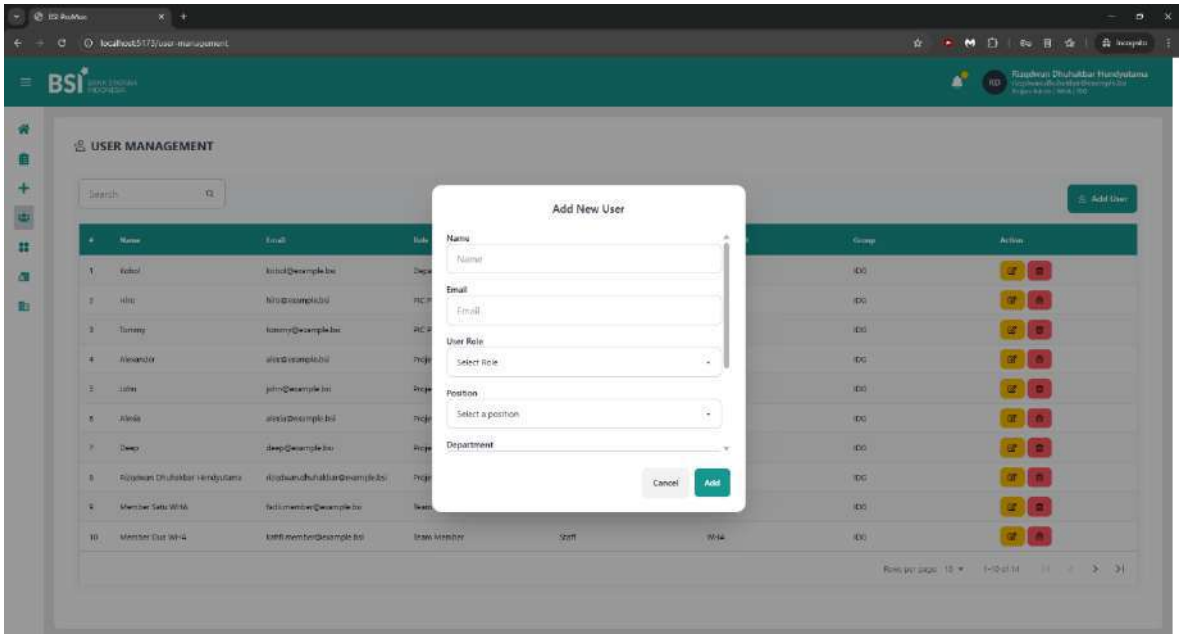
Pada fitur autentikasi, *user* diharuskan memasukkan *email* dan *password* sebagai ketentuan kredensial. Pada sistem ini dirancang untuk *email* menjadi *username* juga bagi *user*. Untuk bagian autentikasi, sistem menggunakan JWT sebagai mekanisme untuk memverifikasi identitas pengguna.

Sebagaimana yang disajikan pada Tabel 3.1, *backlog* menunjukkan penerapan implementasi fitur autentikasi dan registrasi. Pada fitur registrasi, hanya *user* dengan peran (*role*) *Project Admin* yang dapat mengakses dan melakukan penambahan data pengguna baru dengan menginputkan data sesuai *acceptance criteria* pada sistem sesuai dengan departemen yang terkait.



Gambar 3.4 Halaman autentikasi

Tampilan dari halaman *login* pada sistem sebagai fitur autentikasi bagi *user* ditunjukkan pada Gambar 3.4. Halaman ini menjadi antarmuka awal bagi *user* sebelum masuk ke dalam sistem. *User* perlu memasukkan *email* dan *password* sesuai dengan data yang didaftarkan pada sistem. Sistem akan mengecek sebagai kredensial untuk mengidentifikasi pengguna terkait peran (*role*), tanggung jawab, dan departemn terkait di dalam sistem.



Gambar 3.5 Tampilan halaman registrasi pada fitur *user management*

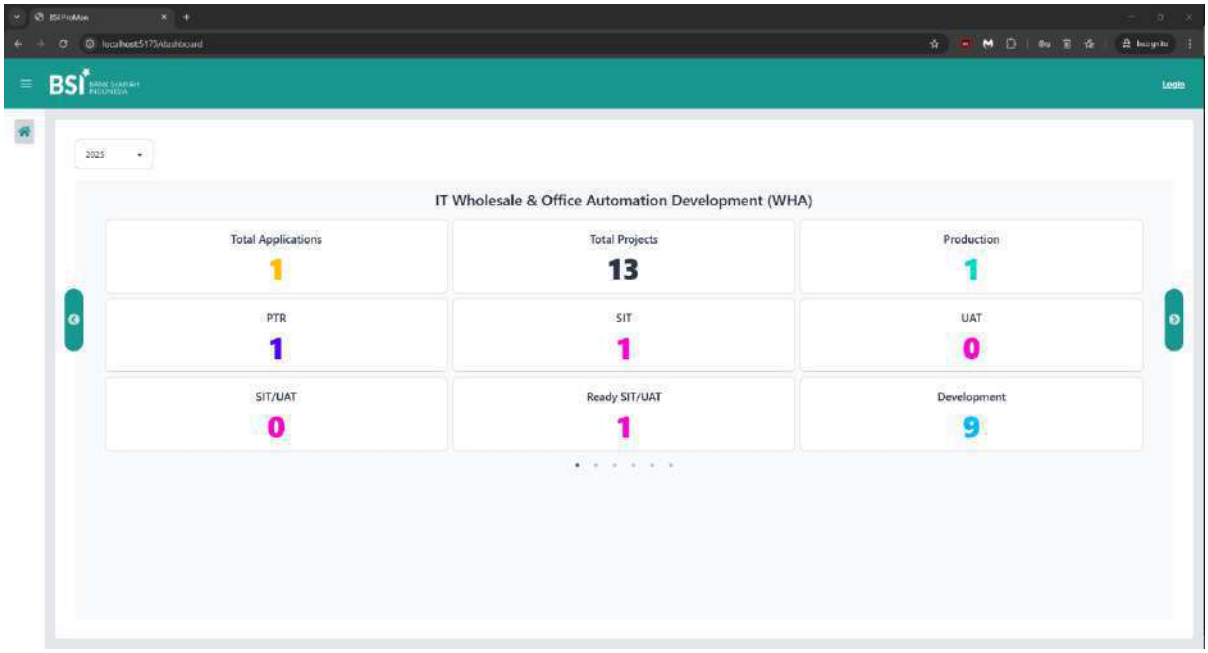
Gambar 3.5 menampilkan halaman registrasi yang terdapat pada fitur *user management*. Fitur registrasi ini hanya dapat diakses oleh peran (*role*) *Project Admin*. Pengguna yang belum memiliki akun pada sistem wajib meminta dan memberitahu *Project Admin* dari departemen sendiri agar dapat dibuatkan akun didalam sistem. *Project Admin* akan memasukkan data diri yang diperlukan meliputi nama lengkap, *email*, *role*, posisi/jabatan, departemen, serta *password* yang akan digunakan. Setelah itu informasi akun yakni *email* dan *password* diberikan kepada pengguna untuk selanjutnya dapat melakukan autentikasi pada halaman *login* seperti pada Gambar 3.4.

3.3.2.2 Fitur *Dashboard*

Tabel 3.2 *Backlog* fitur *Dashboard*

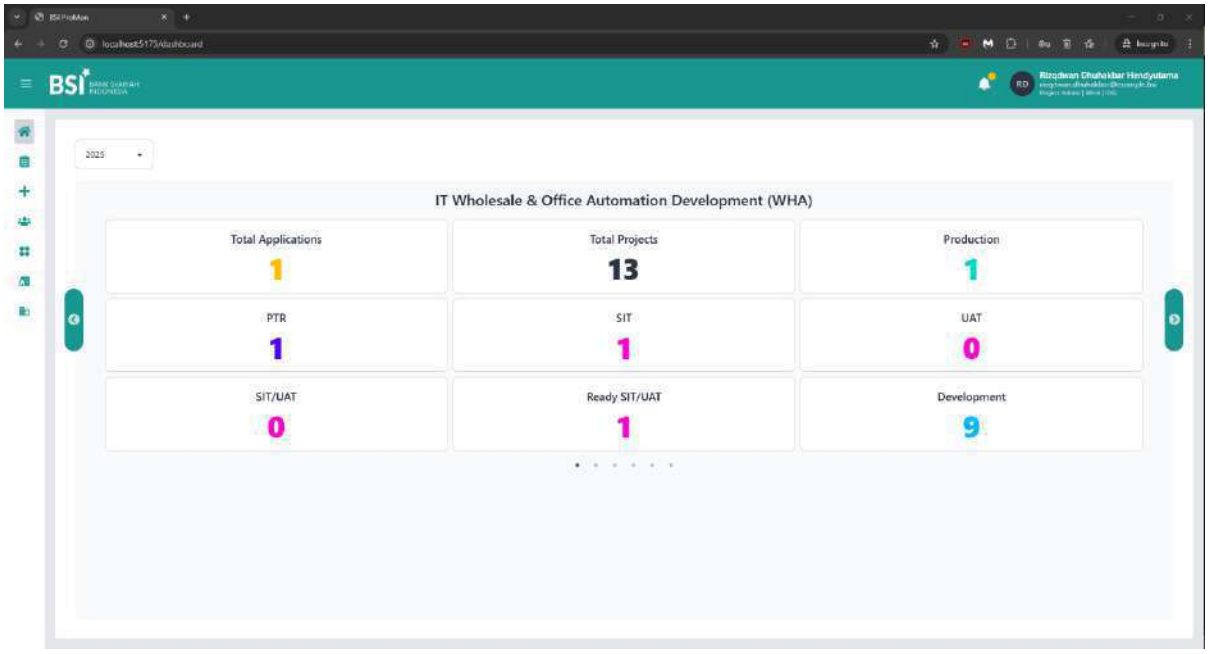
Backlog	*Acceptance Criteria*
Sebagai insan BSI yang melakukan *login* dapat mengakses tampilan *dashboard* pada sistem monitoring proyek berbasis web	Dapat melihat data statistik dari beberapa departemen
	Dapat melakukan filter berdasarkan tahun pada data statistik dari beberapa departemen
	Menampilkan data statistik beberapa departemen
	Dapat melihat fitur dan layanan lain sesuai dengan peran (*role)*
Sebagai insan BSI non-login dapat mengakses tampilan *dashboard* pada sistem monitoring proyek berbasis web	Dapat melihat data statistik dari beberapa departemen
	Dapat melakukan filter berdasarkan tahun pada data statistik dari beberapa departemen
	Menampilkan data statistik beberapa departemen

Tabel 3.2. untuk fitur *Dashboard* dirancang dapat diakses oleh semua pengguna, baik itu pengguna yang melakukan *login* atau autentikasi maupun yang tidak melakukan *login* atau non-login. Perbedaan dari kedua hal tersebut hanya terletak pada tampilan fitur tambahan yang ditampilkan pada pengguna yang melakukan autentikasi. Sistem akan langsung memberikan akses fitur atau layanan kepada pengguna sesuai dengan peran dan tanggung jawabnya.



Gambar 3.6 Tampilan halaman *Dashboard* dari pengguna non-login

Gambar 3.6 menampilkan halaman *Dashboard* dari pengguna yang tidak melakukan *login*. Halaman ini menyajikan rekapitulasi data proyek berdasarkan status (seperti *Development*, SIT/UAT, *Production*), jumlah aplikasi, serta distribusi proyek berdasarkan tahun. Pengguna juga melakukan filter untuk menampilkan rekapitulasi data proyek berdasarkan tahun. *Dashboard* ini berfungsi sebagai ringkasan umum yang memberikan gambaran awal bagi pengguna mengenai jumlah dan penyebaran proyek yang sedang berjalan dari beberapa departemen.



Gambar 3.7 Tampilan halaman *Dashboard* dari pengguna yang sudah *login*

Gambar 3.7 menampilkan halaman *dashboard* dari sisi pengguna yang melakukan *login* dengan peran sebagai *Project Admin*. Tampilan pada halaman ini masih sama dengan menyajikan rekapitulasi data proyek berdasarkan status (seperti *Development*, SIT/UAT, *Production*), jumlah aplikasi, serta distribusi proyek berdasarkan tahun. Pengguna juga bisa melakukan filter untuk menampilkan rekapitulasi data proyek berdasarkan tahun. *Dashboard* ini berfungsi sebagai ringkasan umum yang memberikan gambaran awal bagi pengguna mengenai gambaran proyek yang sedang berjalan dari beberapa departemen.

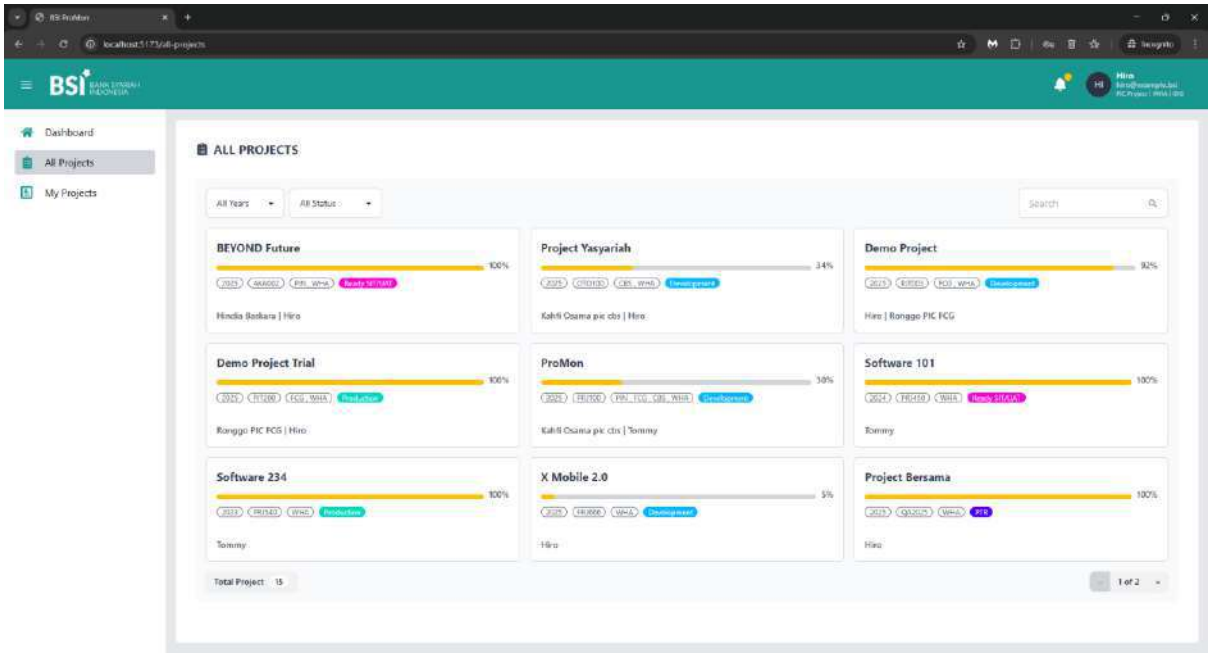
3.3.2.3 Fitur *All Projects* dan *My Projects*

Tabel 3.3 *Backlog* fitur *All Projects* dan *My Projects*

Backlog	*Acceptance Criteria*
Sebagai insan BSI dapat mengakses tampilan *all projects* pada sistem monitoring proyek berbasis web	Dapat melihat semua proyek dari departemen sendiri dan kolaborasi lain
	Dapat melakukan filter proyek berdasarkan tahun dan status pada daftar proyek departemen
	Dapat melakukan pencarian berdasarkan nama proyek
Sebagai insan BSI dapat mengakses tampilan *my projects* pada sistem monitoring proyek berbasis web	Dapat melihat semua proyek yang ditugaskan dari departemen sendiri ataupun kolaborasi lain

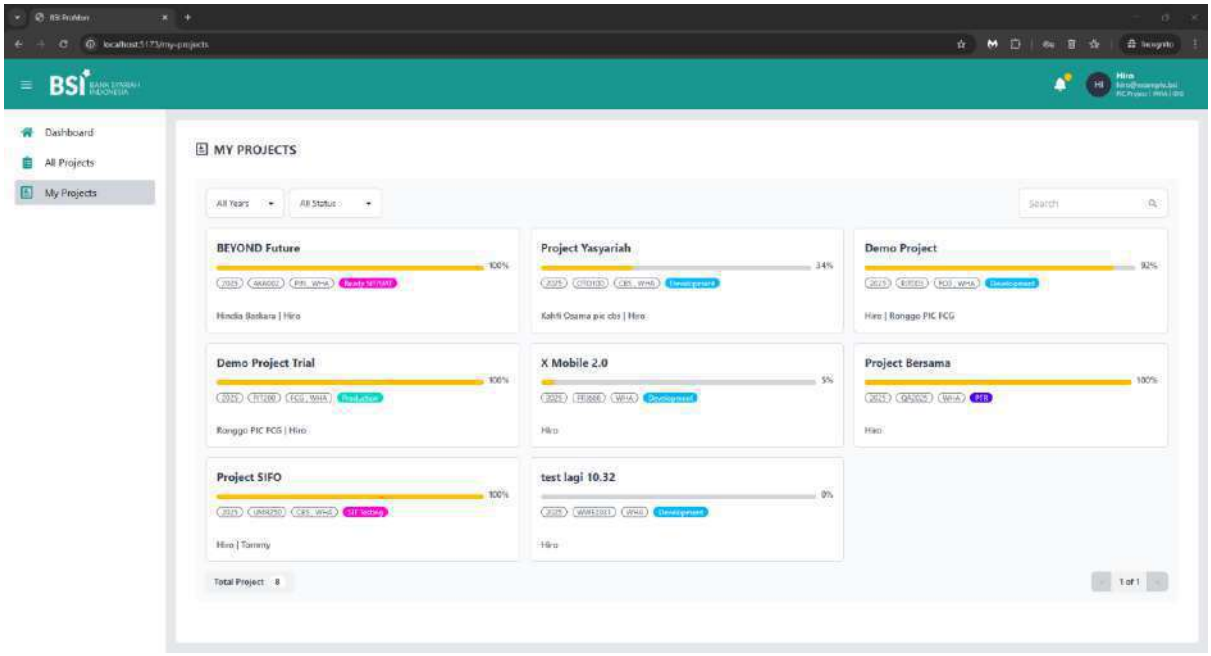
	Dapat melakukan filter proyek berdasarkan tahun dan status pada daftar proyek departemen
	Dapat melakukan pencarian berdasarkan nama proyek

Tabel 3.3 menunjukkan *backlog* pada fitur *all projects* yang menampilkan semua proyek dari departemen sendiri ataupun kolaborasi dengan departemen lain. Untuk fitur *my projects* hanya menampilkan semua proyek yang ditugaskan dari departemen sendiri maupun kolaborasi dengan departemen lain. Pengguna juga dapat melakukan filter proyek berdasarkan tahun dan status, serta dapat melakukan pencarian proyek berdasarkan nama dari proyek.



Gambar 3.8 Tampilan halaman *All Projects*

Gambar 3.8 menampilkan halaman dari fitur *all project* yang menyajikan semua daftar proyek dari departemen sendiri ataupun kolaborasi dengan departemen lain. Pada halaman ini pengguna dapat melakukan filter proyek berdasarkan tahun dan status (seperti *Development, Ready SIT/UAT, SIT, UAT, SIT/UAT, PTR,* dan *Production*), serta dapat melakukan pencarian proyek berdasarkan nama dari proyek. Fitur ini dapat diakses oleh semua peran (*role*).



Gambar 3.9 Tampilan halaman *My Projects*

Gambar 3.9 menampilkan halaman dari fitur *my projects* yang menyajikan semua daftar proyek yang ditugaskan oleh *Project Admin* atau *Department Head* atau PIC dari departemen sendiri ataupun kolaborasi dengan departemen lain. Pada halaman ini pengguna juga dapat melakukan filter proyek berdasarkan tahun dan status (seperti *Development, Ready SIT/UAT, SIT, UAT, SIT/UAT, PTR,* dan *Production*), serta dapat melakukan pencarian proyek berdasarkan nama dari proyek. Fitur ini hanya dapat diakses oleh *role* PIC dan *Team Member*.

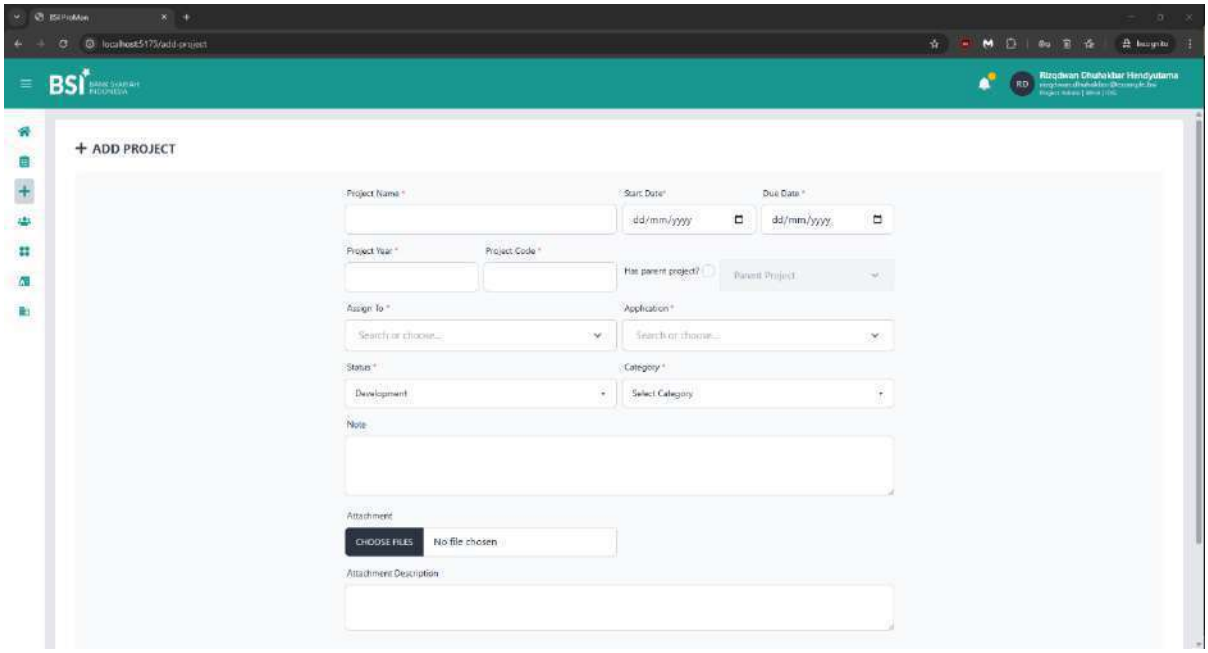
3.3.2.4 Fitur *Add Project*

Tabel 3.4 *Backlog* fitur *Add Project*

Backlog	*Acceptance Criteria*
Sebagai *Project Admin* dapat mengakses tampilan *Add Project* pada sistem monitoring proyek berbasis web	Menampilkan form pendaftaran untuk proyek
Sebagai *Project Admin* dapat menambahkan proyek pada departemen sendiri	Dapat menambahkan proyek baru pada departemen sendiri
	Melakukan input data proyek dengan mengirimkan nama proyek, *start date*, *due date*, tahun proyek, kode proyek, *parent project*, *assign to,* aplikasi, status proyek, kategori, *note* dan lampiran file

Sebagai *Project Admin* dapat menugaskan proyek pada departemen sendiri untuk insan BSI dengan peran *Department Head*	Dapat melakukan *assign* proyek kepada *Departmen Head* dari departemen sendiri maupun lainnya

Berdasarkan *backlog* untuk fitur *Add Project* yang disajikan pada Tabel 3.4 menunjukkan bahwa fitur ini hanya dapat diakses oleh *Project Admin*. *Project Admin* dapat menambahkan proyek baru pada departemen sendiri sesuai dengan *acceptance criteria* dan melakukan penugasan kepada *Department Head* dari departemen sendiri ataupun departemen lain untuk proyek baru tersebut.



Gambar 3.10 Tampilan halaman *Add Project*

Gambar 3.10 menampilkan halaman dari fitur *Add Project* yang hanya dapat diakses oleh *Project Admin*. Fitur ini berfungsi untuk menambahkan dan menungaskan proyek baru untuk *Department Head* pada departemen sendiri ataupun berkolaborasi dengan departemen lain. Hasil dari fitur ini akan muncul pada halaman *all project* yang ditunjukkan pada Gambar 3.8.

3.3.2.5 Fitur *Project Management*

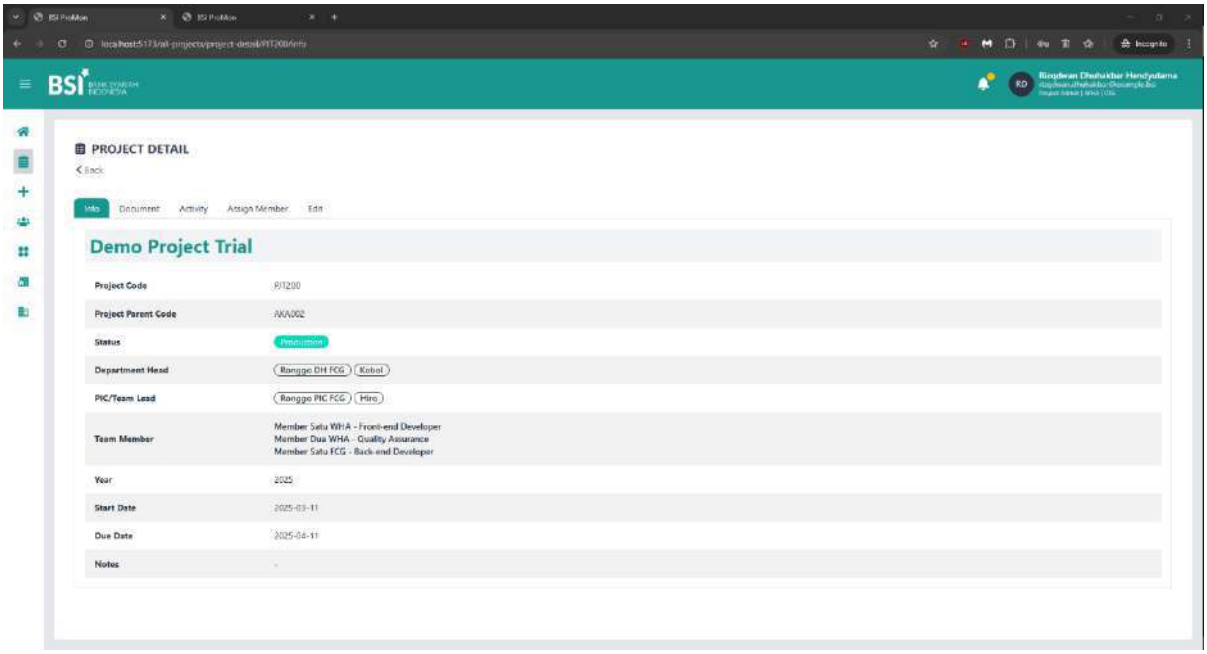
Fitur *project management* adalah kumpulan dari layanan dan fitur tambahan yang ada saat pengguna mengakses tampilan *card* pada halaman *all projects* dan *my projects* yang di tampilkan pada Gambar 3.8 dan Gambar 3.9.

Tabel 3.5 *Backlog* fitur *Project Management*

Backlog	*Acceptance Criteria*
Sebagai insan BSI dapat mengakses tampilan *card project* yang ada dihalaman *all projects* dan *my projects* pada departemen sendiri	Terdapat halaman informasi proyek yang dituju berisikan nama proyek, kode proyek, kode relasi proyek, status, nama *Department Head*, nama PIC/*Team lead*, tahun, *start date, due date*, dan *note*
	Terdapat halaman proyek dokumen
	Terdapat halaman proyek aktivitas yang menampilkan daftar anggota pada proyek
	Terdapat halaman penugasan anggota proyek yang menampilkan informasi daftar anggota pada proyek
	Terdapat halaman *edit* proyek
Sebagai *Project Admin* dapat menambahkan dokumen proyek pada proyek tertentu pada departemen sendiri	Dapat menambahkan deskripsi dan lampiran file
Sebagai *Project Admin* dapat menghapus dokumen proyek pada proyek tertentu pada departemen sendiri	Dokumen dihapus maka dokumen yang berelasi dengan proyek lain juga terhapus
Sebagai *Project Admin* dapat mengubah data detail informasi proyek pada proyek tertentu pada departemen sendiri dari halaman *edit* proyek	Dapat melakukan perubahan pada nama proyek, tahun, *start date, due date*, dan kategori
Sebagai *Project Admin* dapat menghapus proyek proyek tertentu pada departemen sendiri	proyek dihapus maka proyek yang berelasi dengan proyek lain juga terhapus
Sebagai *Department Head* dapat mengubah atau menambahkan penugasan kepada insan BSI dengan peran PIC	Dapat menambahkan *note* setelah melakukan penugasan (*assign to*)
Sebagai PIC dapat dapat menambahkan dokumen proyek pada proyek tertentu yang telah ditugaskan pada departemen sendiri	Dapat menambahkan deskripsi dan lampiran file
Sebagai PIC dapat menghapus dokumen proyek pada proyek tertentu yang telah ditugaskan pada departemen sendiri	Dokumen dihapus maka dokumen yang berelasi dengan proyek lain juga terhapus
Sebagai PIC dapat menambahkan aktivitas proyek yang dikerjakan oleh insan BSI dengan peran *Team Member* pada proyek tertentu yang telah ditugaskan pada departemen sendiri	Dapat menambahkan persen, detail pekerjaan, dan status
	Terdapat pilihan untuk status aktivitas sepeti *Not started, Hold, Progress,* dan *Completed*

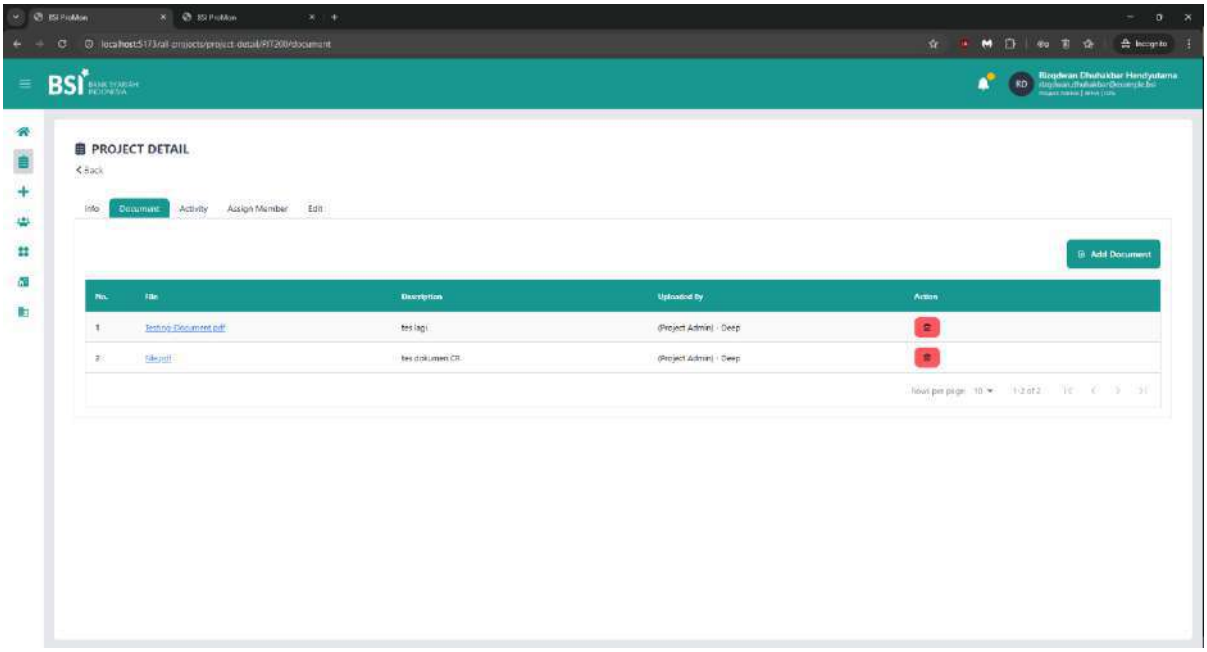
Sebagai PIC dapat menghapus dokumen proyek pada proyek tertentu yang telah ditugaskan pada departemen sendiri	Proyek dihapus maka proyek yang berelasi juga terhapus
Sebagai PIC dapat mengubah aktivitas proyek yang dikerjakan oleh insan BSI dengan peran *Team Member* pada proyek yang telah ditugaskan pada departemen sendiri	Dapat mengubah data aktivitas *Team Member* yang ditugaskan pada proyek
	Dapat mengubah data persen, detail pekerjaan, dan status
	Terdapat pilihan untuk status aktivitas sepeti *Not started, Hold, Progress,* dan *Completed*
Sebagai PIC dapat menghapus aktivitas proyek yang dikerjakan oleh insan BSI dengan peran *Team Member* pada proyek tertentu yang telah ditugaskan pada departemen sendiri	Menghapus aktivitas proyek anggota tim yang ditugaskan pada proyek tersebut
Sebagai *Team Member* hanya dapat mengakses tampilan *card project* yang ada dihalaman *all projects* dan *my projects* dari departemen sendiri yang ditugaskan	Terdapat halaman informasi proyek yang dituju berisikan nama proyek, kode proyek, kode relasi proyek, status, nama *Department Head*, nama PIC/*Team lead*, tahun, *start date, due date*, dan *note*
	Terdapat halaman proyek dokumen
	Terdapat halaman proyek aktivitas yang menampilkan daftar anggota pada proyek
	Terdapat halaman penugasan anggota proyek yang menampilkan informasi daftar anggota pada proyek
	Terdapat halaman *edit* proyek

Tabel 3.5 menunjukkan *backlog* dari fitur *project management* yang memiliki batasan hak akses dari berbagai peran pengguna yang disesuaikan dengan *acceptance criteria* nya. Untuk peran *Project Admin* dapat mengubah detail informasi proyek yang dilakukan dari halaman *edit* proyek yang nanti perubahan dapat terlihat di halaman *project detail*, juga dapat menambahkan dan menghapus dokumen proyek di halaman *document*. Untuk peran *Department Head* juga dapat menambahkan penugasan proyek untuk *Department Head* dari department lain. Untuk peran PIC dapat memperbarui status proyek dari halaman *edit*, juga dapat juga dapat menambahkan dan menghapus dokumen proyek di halaman *document*, serta menambahkan, mengubah, dan menghapus aktivitas proyek anggota tim dengan peran *Team Member* dari departemen sendiri. Dan terakhir untuk peran *Team Member* hanya dapat melihat detail informasi, dokumen dan aktivitas proyek.



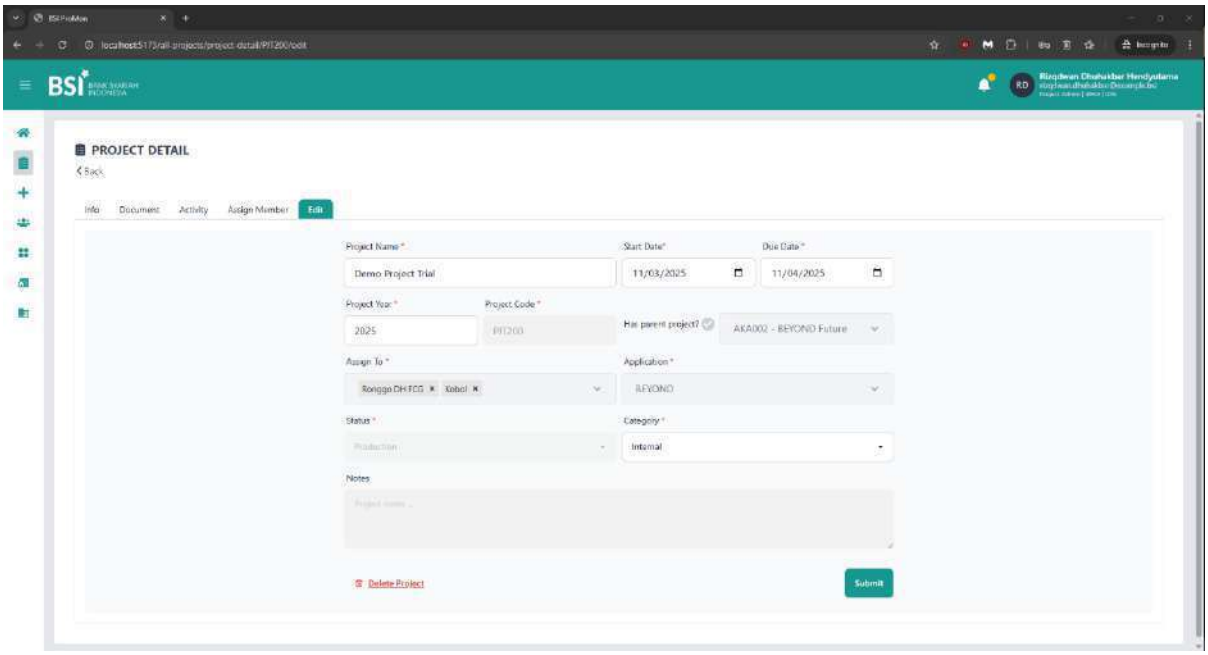
Gambar 3.11 Tampilan halaman *info* dari sudut pandang *Project Admin*

Gambar 3.11 menunjukkan tampilan halaman *info* dari sudut pandang *Project Admin*. Perbedaan mendasar bagi semua peran (*role*) untuk tampilan ini terletak pada bagian isi *note*. *Note* disini berfungsi sebagai catatan yang diberikan oleh *user* yang menugaskan (*assigned*) proyek tersebut kepada *user* lain. Fungsi dari halaman ini sebagai informasi proyek secara detail dari proyek yang dipilih.



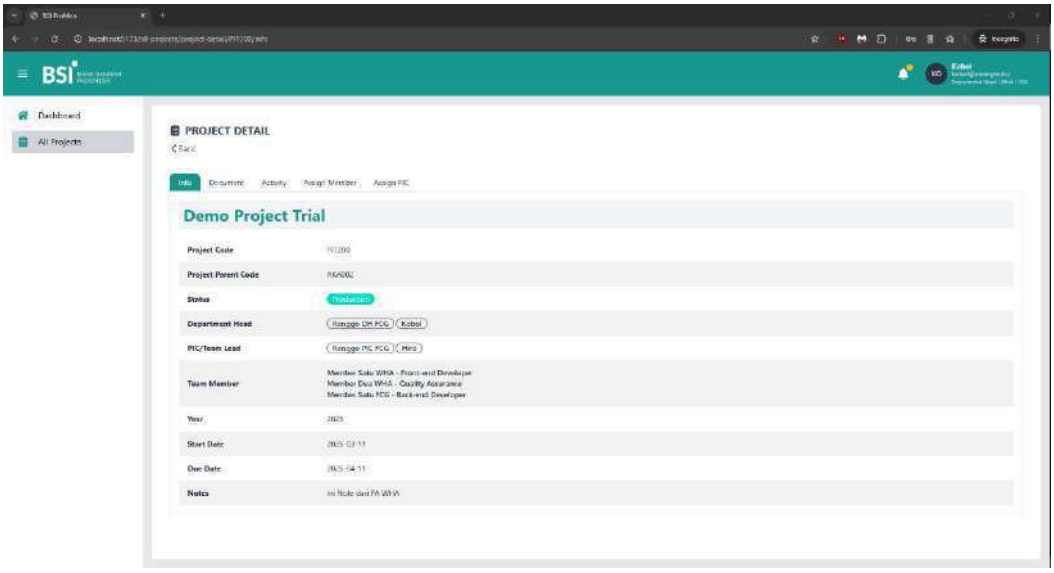
Gambar 3.12 Tampilan halaman *project document* dari sudut pandang *Project Admin*

Gambar 3.12 menunjukkan tampilan halaman *project document* dari sudut pandang *Project Admin*. Disini *Project Admin* dapat menambah dan menghapus dokumen proyek dari departemen sendiri. Peran (*role*) PIC juga dapat menambah dan menghapus dokumen proyek khusus proyek yang ditugaskan kepadanya. Fungsi dari halaman ini untuk menyimpan dokumen terkait proyek seperti proposal atau spesifikasi teknis proyek.



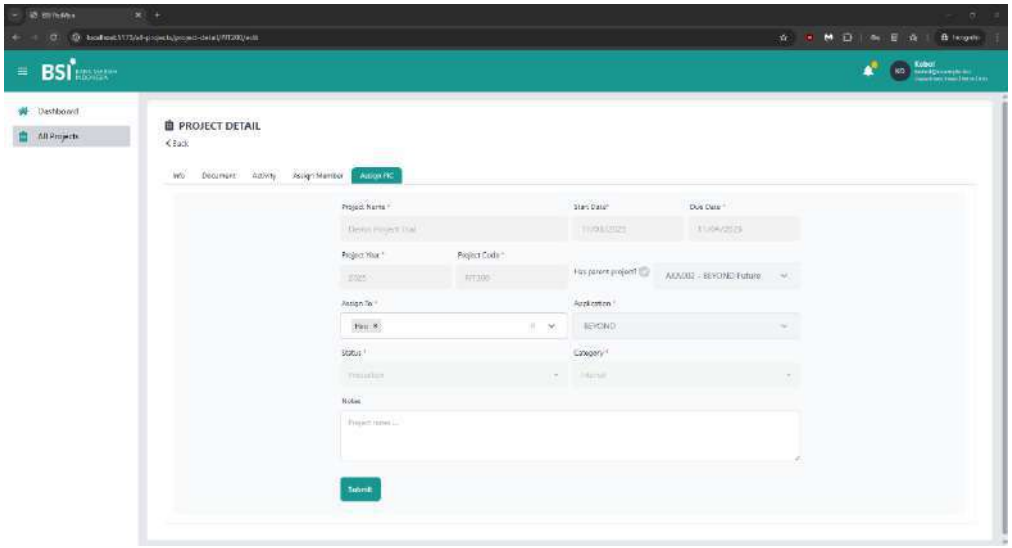
Gambar 3.13 Tampilan halaman *edit* dari sudut pandang *Project Admin*

Gambar 3.13 menunjukkan tampilan halaman *edit* dari sudut pandang *Project Admin*. *Project Admin* dapat melakukan perubahan pada halaman *edit* seperti nama proyek, tahun, *start date, due date*, dan kategori. Dan juga *Project Admin* dapat menghapus proyek tersebut. Fungsi halaman ini untuk melakukan perubahan pada beberapa detail informasi proyek yang ada pada halaman *info*.



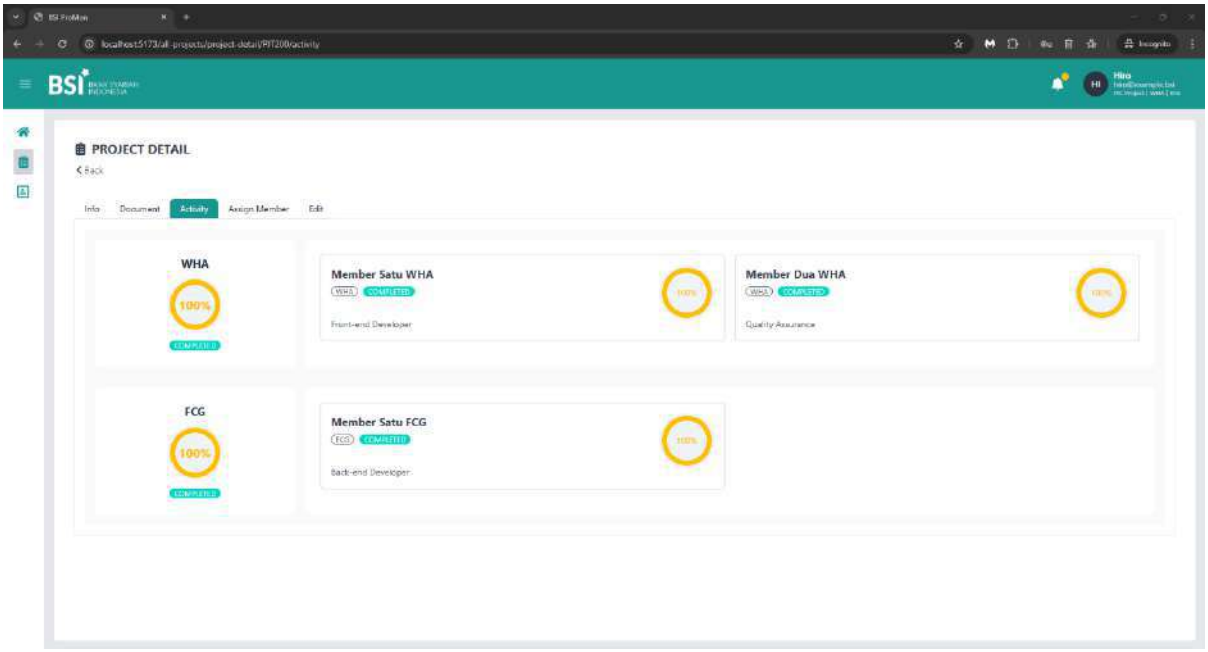
Gambar 3.14 Tampilan halaman *project detail* dari sudut pandang *Department Head*

Gambar 3.14 menunjukkan tampilan halaman *project detail* dari sudut pandang *Department Head* yang dapat terlihat perbedaannya pada bagian isi *note*. *Note* tersebut dikirimkan oleh *Project Admin* dari departemen terkait yang dikasus ini dari departemen WHA.



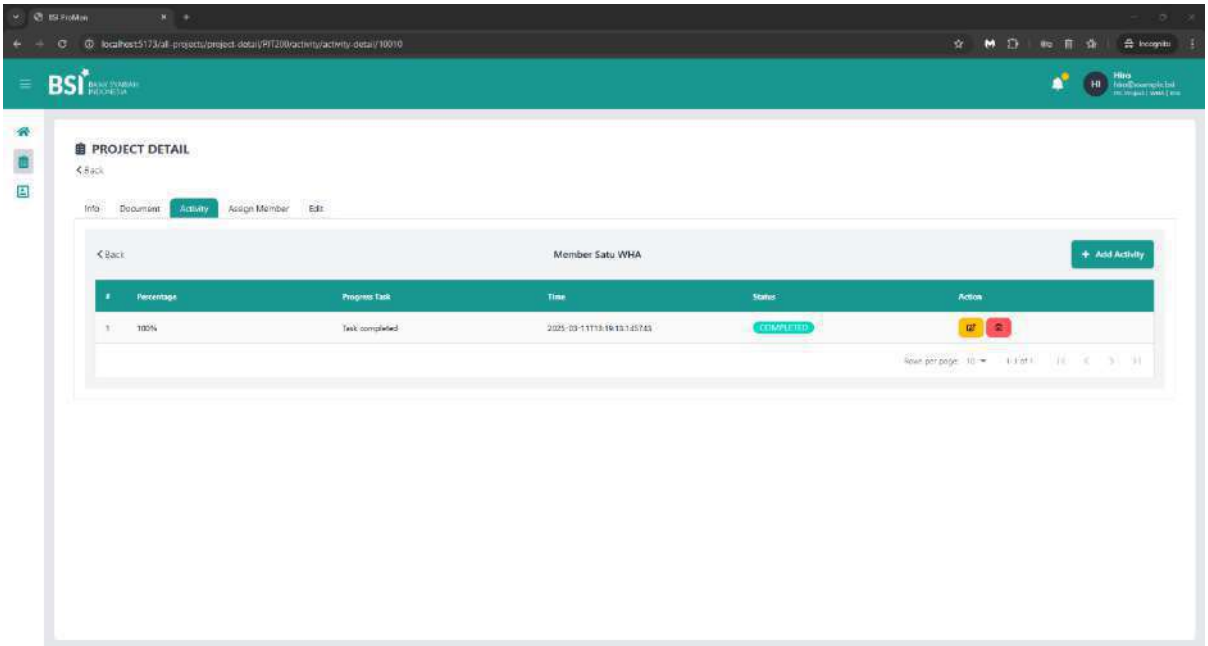
Gambar 3.15 Tampilan halaman *edit* dari sudut pandang *Department Head*

Gambar 3.15 menunjukkan tampilan halaman *edit* dari sudut pandang *Department Head*. Disini *Department Head* hanya dapat menambahkan penugasan (*assigned*) proyek untuk peran (*role*) PIC dan juga dapat memberikan *note*. Fungsi halaman ini untuk melakukan perubahan pada beberapa detail informasi proyek yang ada pada halaman *info*.



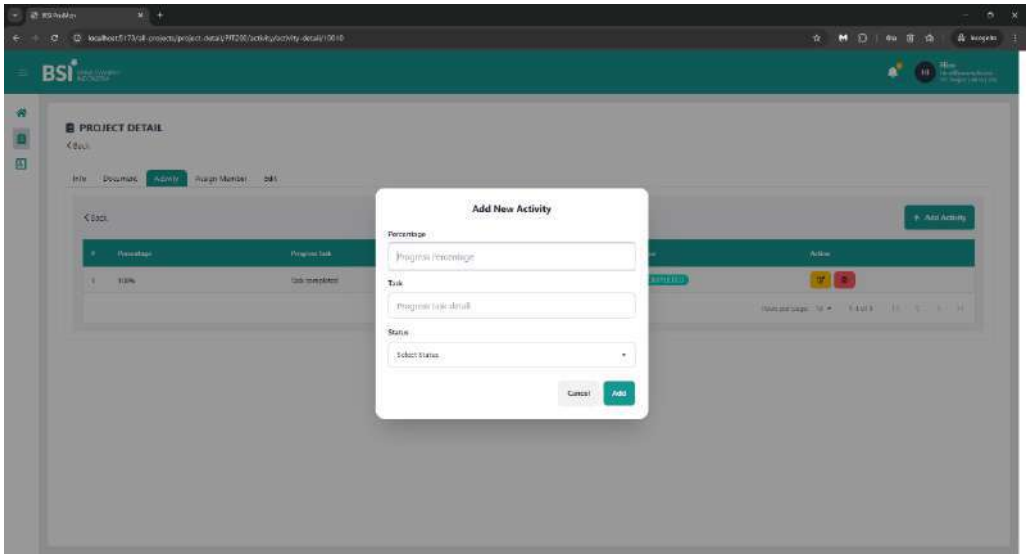
Gambar 3.16 Tampilan halaman *activity project* dari sudut pandang PIC

Gambar 3.16 menunjukkan tampilan halaman *activity project* dari sudut pandang PIC. Fungsi dari halaman *activity project* untuk memberikan informasi terkait perkembangan proyek dari departemen sendiri maupun dari departemen lain yang berkolaborasi.



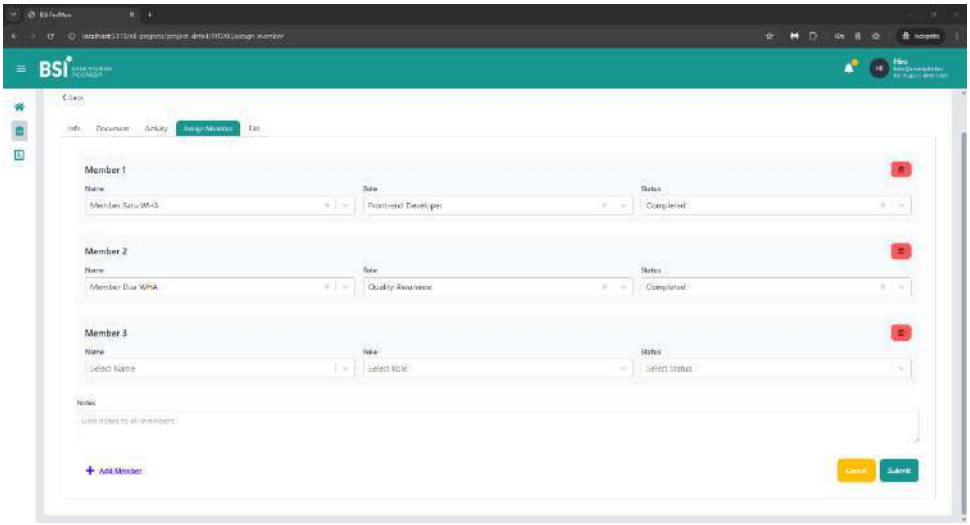
Gambar 3.17 Tampilan isi *card* anggota tim proyek dari halaman *activity project* dari sudut pandang PIC

Gambar 3.17 menunjukkan tampilan isi *card* anggota tim proyek dari halaman *activity project* dari sudut pandang PIC. Halaman ini menampilkan informasi aktivitas dari anggota tim proyek seperti persenan pekerjaan, tugas yang dikerjakan, dan status pekerjaannya.



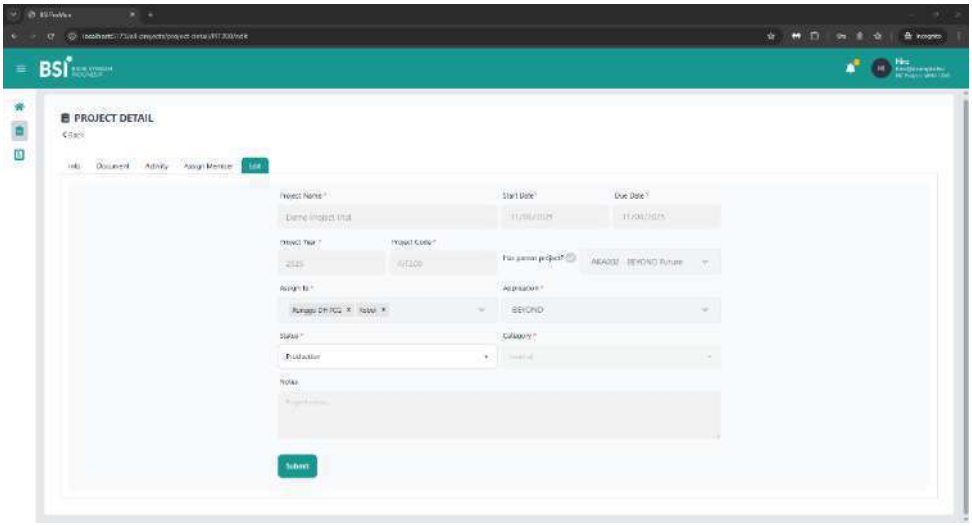
Gambar 3.18 Tampilan dari form data dokumen baru pada halaman *activity project* dari sudut pandang PIC

Gambar 3.18 menunjukkan tampilan form pengisian data untuk menambah aktivitas baru anggota tim dari halaman *activity project*. PIC menambahkan data aktivitas baru anggota tim tertentu dengan mengisikan persenan, detail pekerjaan, dan status pekerjaan. Untuk status pekerjaan dibagi menjadi *Not started, Hold, Progress,* dan *Completed*.



Gambar 3.19 Tampilan halaman *assign member* dari sudut pandang PIC

Gambar 3.19 menunjukkan tampilan halaman *assign member* dari sudut pandang PIC. Halaman ini berfungsi untuk menambahkan anggota tim pada proyek yang hanya dapat dilakukan oleh PIC proyek. PIC dapat menambahkan anggota tim dengan memilih daftar pengguna dengan peran *team member* dan memilih *role* yang diperlukan pada proyek. Pada tampilan terdapat bagian status saat menambahkan anggota tim, hasil dari pemilihan status terhubung dengan halaman *activity project*.



Gambar 3.20 Tampilan halaman *edit* dari sudut pandang PIC

Gambar 3.20 menunjukkan tampilan halaman *edit* dari sudut pandang PIC. PIC hanya dapat mengubah status proyek dihalaman *edit*. Pengubahan status proyek hanya dapat dilakukan jika progres proyek yang dapat dilihat pada halaman *activity project* selesai semua dari departemen yang terkait oleh proyek.

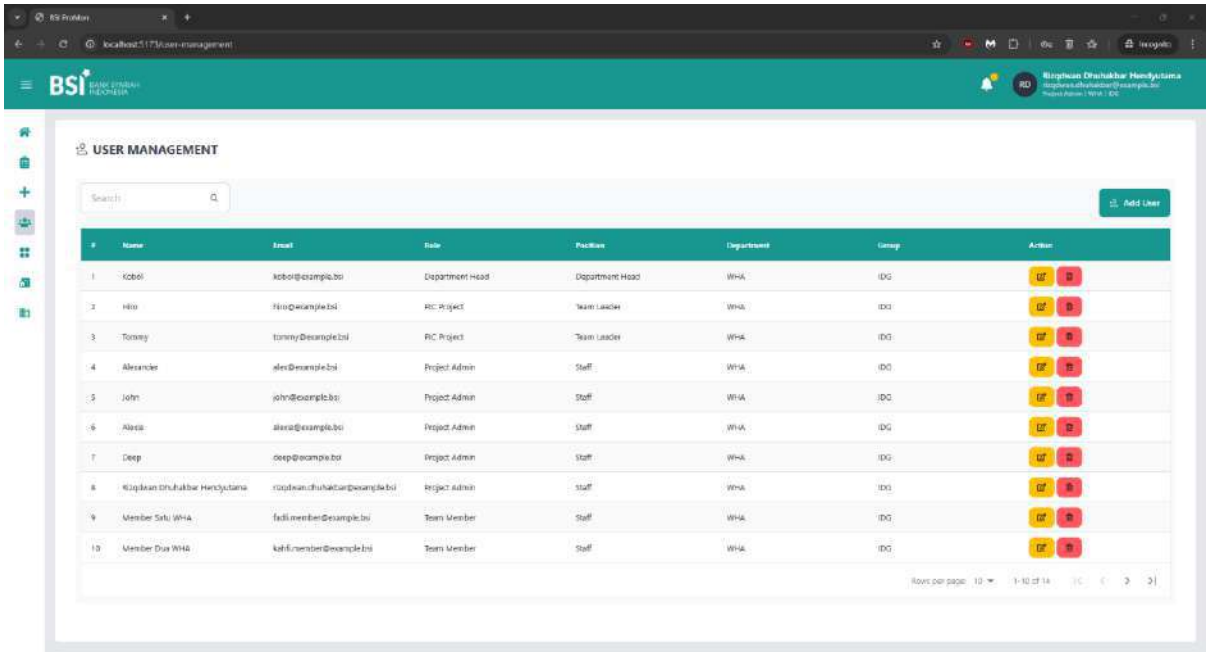
3.3.2.6 Fitur *User management*

Tabel 3.6 *Backlog* fitur *User Management*

Backlog	*Acceptance Criteria*
Sebagai *Project Admin* dapat mengakses tampilan *user management* pada sistem monitoring proyek berbasis web	Dapat melihat daftar pengguna dari departemen sendiri
	Dapat melakukan filter pencarian berdasarkan nama pengguna
Sebagai *Project Admin* dapat menambahkan data pengguna baru pada departemen sendiri	Dapat menambahkan pengguna baru pada departemen sendiri

	Melakukan input data dengan mengirimkan nama lengkap, *email*, *role user*, posisi atau jabatan, departemen, *password*, dan konfirmasi *password*
Sebagai *Project Admin* dapat mengubah data pengguna pada departemen sendiri	Dapat melakukan perubahan untuk beberapa pengguna dengan peran tertentu, sepeti nama
Sebagai *Project Admin* dapat menghapus data pengguna pada departemen sendiri	Data pengguna yang dihapus dan memiliki relasi maka juga terhapus

Tabel 3.6 menunjukkan *backlog* dari fitur *user management* yang hanya dapat diakses oleh *Project Admin*. *Project Admin* dapat menambahkan data pengguna baru atau registrasi yang ditampilkan pada Gambar 3.5, serta dapat mengubah data pengguna untuk beberapa pengguna dengan peran tertentu dan menghapus data pengguna.



Gambar 3.21 Tampilan halaman fitur *User Management*

Gambar 3.21 menampilkan halaman dari fitur *user management* yang hanya dapat diakses oleh *Project Admin*. Fitur ini berguna untuk memberikan informasi internal pada daftar pengguna dari departemen sendiri yang ada pada sistem.

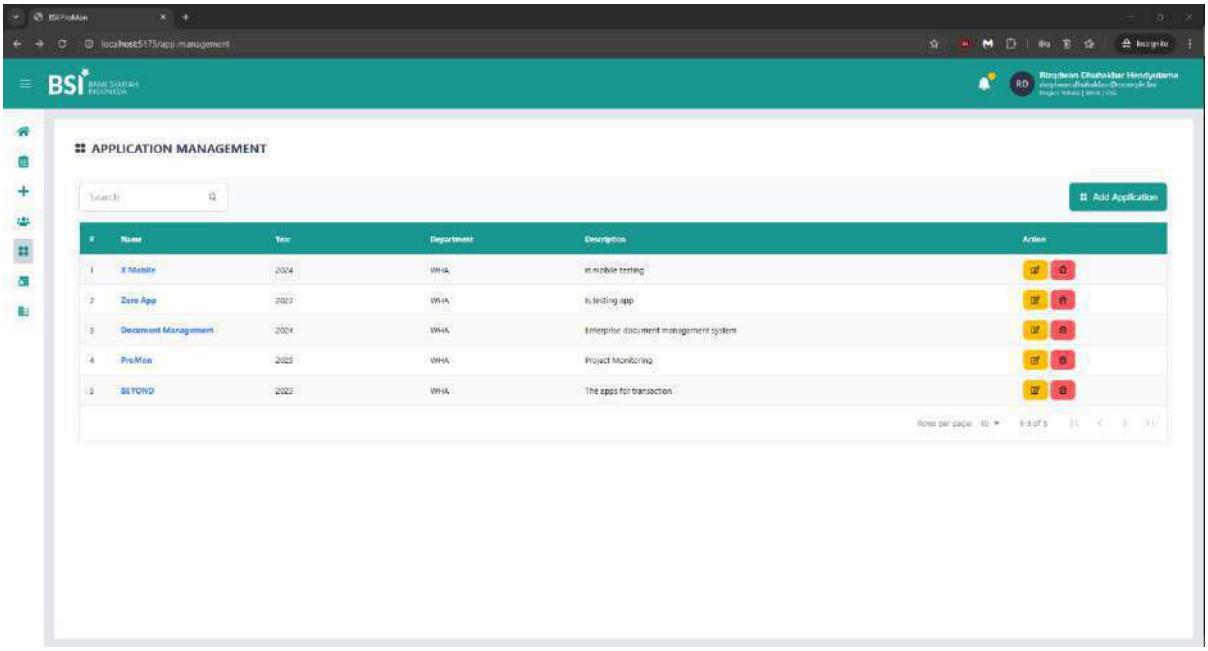
3.3.2.7 Fitur *Application Management*

Tabel 3.7 *Backlog* fitur *Application Management*

Backlog	*Acceptance Criteria*
Sebagai *Project Admin* dapat mengakses tampilan *application management* pada sistem monitoring proyek berbasis web	Terdapat daftar aplikasi dari departemen sendiri yang menampilkan informasi nama, tahun, departemen dan deskripsi
	Dapat melakukan filter pencarian aplikasi berdasarkan nama pada daftar aplikasi
Sebagai *Project Admin* dapat menambahkan aplikasi baru pada departemen sendiri dari *application management*	Dapat menambahkan aplikasi baru pada departemen sendiri
	Melakukan input data aplikasi dengan mengirimkan nama aplikasi, tahun aplikasi, departemen, dan deskripsi
Sebagai *Project Admin* dapat mengubah detail aplikasi pada departemen sendiri dari *application management*	Dapat mengubah data aplikasi pada nama aplikasi, tahun, dan deskripsi
Sebagai *Project Admin* dapat menghapus detail aplikasi pada departemen sendiri dari *application management*	Aplikasi dihapus maka aplikasi yang berelasi dengan aplikasi lain juga terhapus
Sebagai *Project Admin* dapat menambahkan *server* aplikasi pada departemen sendiri dari *application management*	Dapat menambahkan data *server* aplikasi baru pada aplikasi
	Melakukan input data aplikasi dengan mengirimkan *environment* aplikasi, kategori aplikasi, *server/ip* , nama *server*, domain, os, lokasi *server* dan tipe *server*
	Terdapat pilihan untuk *environment* aplikasi sepeti *development, PTR, testing,* dan *production*
	Terdapat pilihan untuk kategori aplikasi sepeti *on premise* dan *cloud*
	Terdapat pilihan untuk lokasi *server* aplikasi sepeti DC, DRC, dan DCI
	Terdapat pilihan untuk tipe *server* sepeti web, *application*, dan *database*
Sebagai *Project Admin* dapat mengubah *server* aplikasi pada departemen sendiri dari *application management*	Dapat mengubah data *server* pada *environment* aplikasi, kategori aplikasi, nama *server*, domain, os, lokasi *server* dan tipe *server*
Sebagai *Project Admin* dapat menghapus detail aplikasi pada departemen sendiri dari *application management*	-
Sebagai *Project Admin* dapat menambahkan *installation package server* aplikasi pada	Dapat menambahkan data *installation package server* aplikasi baru pada aplikasi

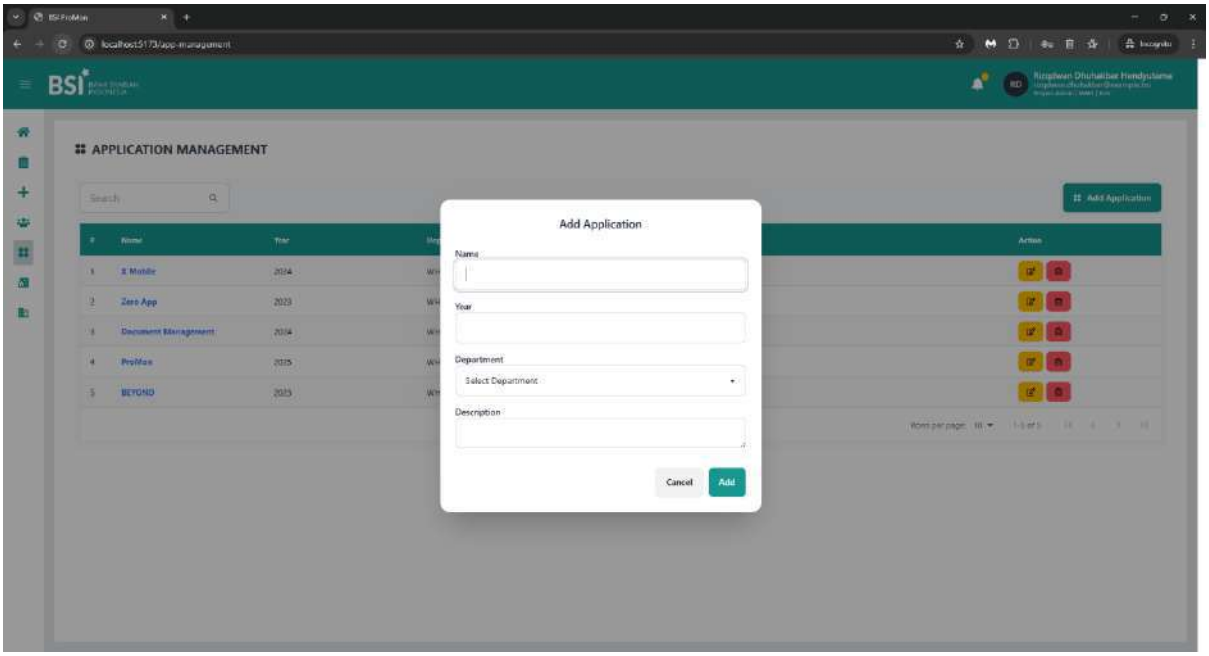
departemen sendiri dari *application management*	Melakukan input data *installation package server* aplikasi dengan mengirimkan nama *package*
Sebagai *Project Admin* dapat mengubah *installation package server* aplikasi pada departemen sendiri dari *application management*	Dapat mengubah data *installation package server* aplikasi pada nama *package*
Sebagai *Project Admin* dapat menghapus *installation package server* aplikasi pada departemen sendiri dari *application management*	-
Sebagai *Project Admin* dapat menambahkan *surrounding* aplikasi pada departemen sendiri dari *application management*	Dapat menambahkan aplikasi baru pada departemen sendiri
Sebagai *Project Admin* dapat mengubah *surrounding* aplikasi pada departemen sendiri dari *application management*	Dapat mengubah data *surrounding* aplikasi dengan memilih aplikasi lain
Sebagai *Project Admin* dapat menghapus *surrounding* aplikasi pada departemen sendiri dari *application management*	-
Sebagai *Project Admin* dapat menambahkan dokumen aplikasi pada aplikasi tertentu pada departemen sendiri	Dapat menambahkan deskripsi dan lampiran file
Sebagai *Project Admin* dapat menghapus dokumen aplikasi pada aplikasi tertentu pada departemen sendiri	Dokumen dihapus maka dokumen yang berelasi dengan aplikasi lain juga terhapus

Tabel 3.7 menyajikan *backlog* dari fitur *application management*. Fitur ini hanya dapat diakses oleh *Project Admin*. *Project Admin* dapat menambahkan aplikasi, mengubah informasi aplikasi dan menghapus aplikasi. Pada fitur ini terdapat fitur tambahan seperti *server application, surrounding application,* dan *document application*. *Project Admin* dapat menambahkan, mengubah dan menghapus data pada fitur tambahan tersebut yang disesuaikan pada *acceptance criteria* nya.



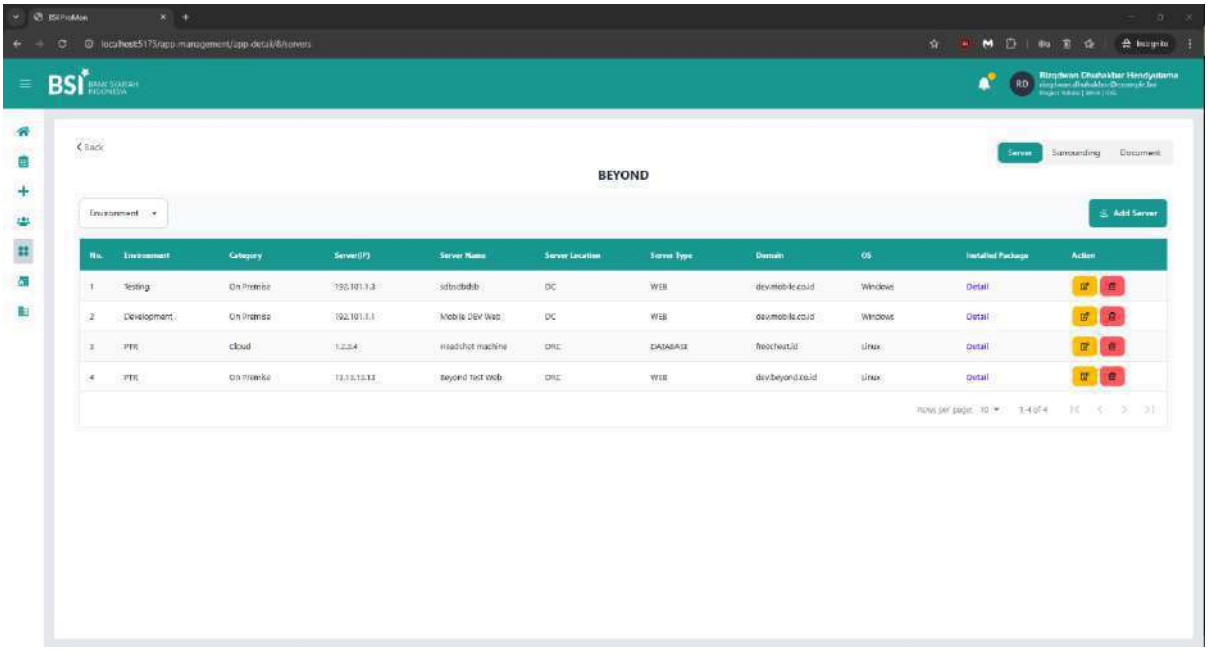
Gambar 3.22 Tampilan halaman fitur *Application Management*

Gambar 3.22 menunjukkan tampilan dari halaman fitur *Application Management*. Halaman ini menampilkan daftar aplikasi pada departemen terkait juga informasi aplikasi seperti nama, tahun, departemen, dan deskripsi. Fitur ini hanya dapat diakses oleh *Project Admin*. Fungsi fitur ini untuk menampilkan daftar aplikasi pada departemen sendiri.



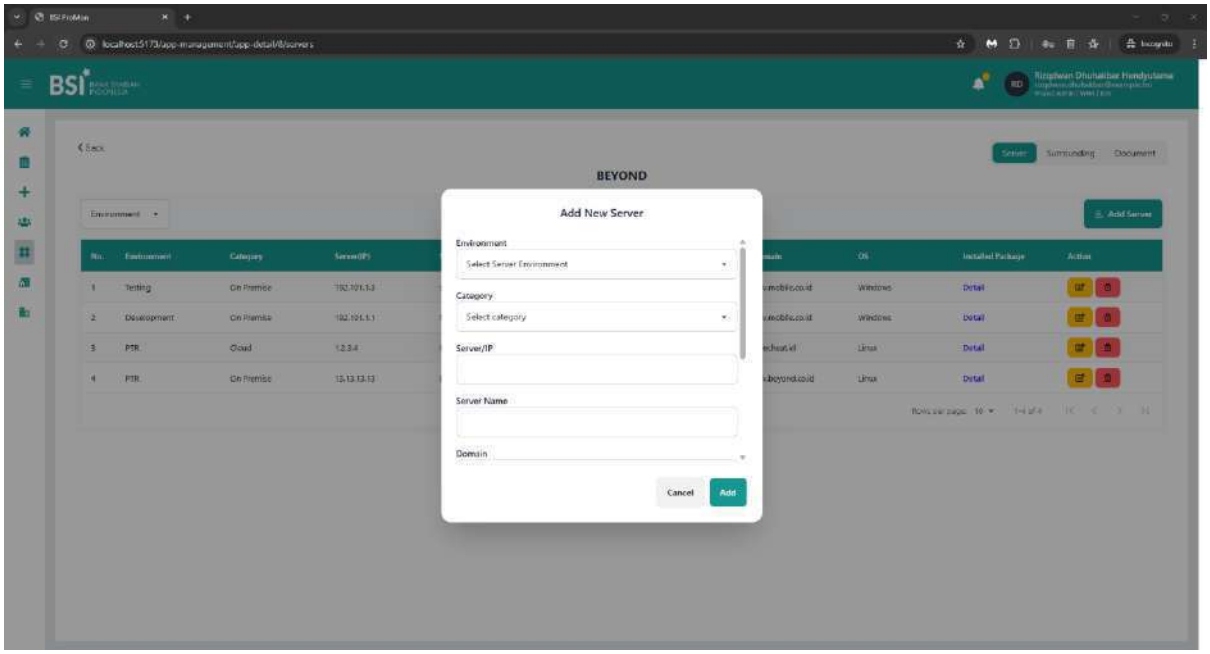
Gambar 3.23 Tampilan halaman form data aplikasi baru dari halaman fitur *Application Management*

Gambar 3.23 menunjukkan tampilan dari halaman form pengisian data aplikasi untuk menambahkan aplikasi baru pada fitur *Application Management*. Fitur ini hanya dapat dilakukan oleh *Project Admin*. *Project Admin* perlu memasukkan data seperti nama aplikasi, tahun aplikasi, departemen dan deskripsi untuk menambahkan aplikasi baru.



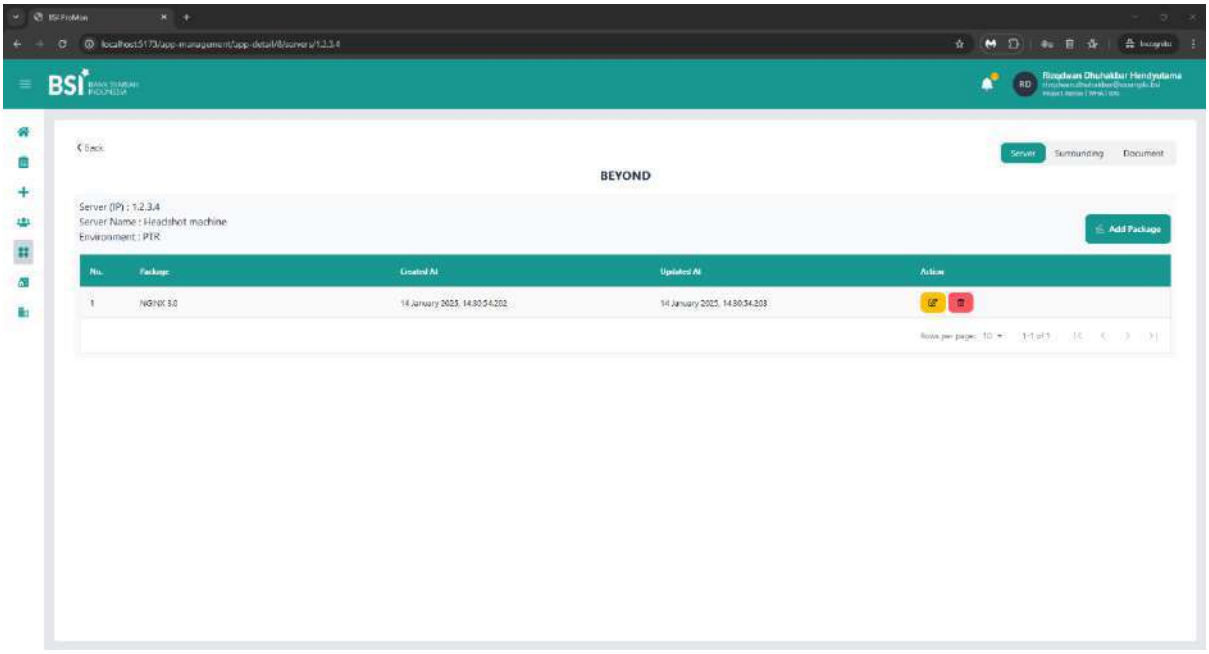
Gambar 3.24 Tampilan halaman *server* dari fitur *Application Management*

Gambar 3.24 menunjukkan tampilan dari halaman *server* dari fitur *Application Management*. Fitur ini hanya dapat diakses oleh *Project Admin*. Halaman ini berfungsi untuk menampilkan daftar *server* pada aplikasi dan memberikan detail informasi seperti *environment*, kategori, *server/ip*, nama *server*, lokasi *server*, tipe *server*, domain, os, dan *installation package*.



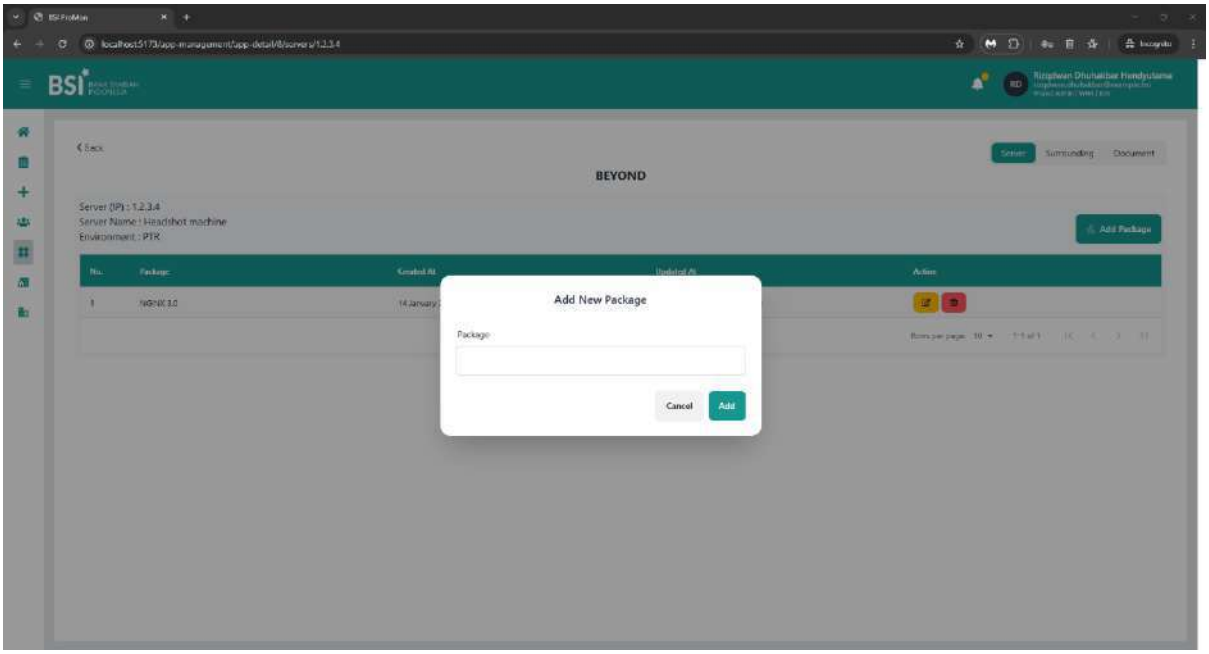
Gambar 3.25 Tampilan halaman form data *server* aplikasi baru pada halaman *server* dari fitur *Application Management*

Gambar 3.25 menunjukkan tampilan dari halaman form pengisian data s*erver* aplikasi untuk menambahkan s*erver* aplikasi baru pada fitur *Application Management*. Fitur ini hanya dapat dilakukan oleh *Project Admin*. *Project Admin* perlu memasukkan data seperti *environment*, kategori, *server/ip*, nama *server*, lokasi *server*, tipe *server*, domain, os, dan *installation package* untuk menambah data s*erver* aplikasi baru. Pilihan e*nvironment* yang terdiri dari *development, PTR, testing,* dan *production*. Pilihan kategori aplikasi yang terdiri dari *on premise* dan *cloud*. Pilihan lokasi *server* yang terdiri dari DC, DRC, dan DCI. Terakhir, pilihan tipe *server* yang terdiri dari web, *application*, dan *database*.



Gambar 3.26 Tampilan halaman *server detail* dari fitur *Application Management*

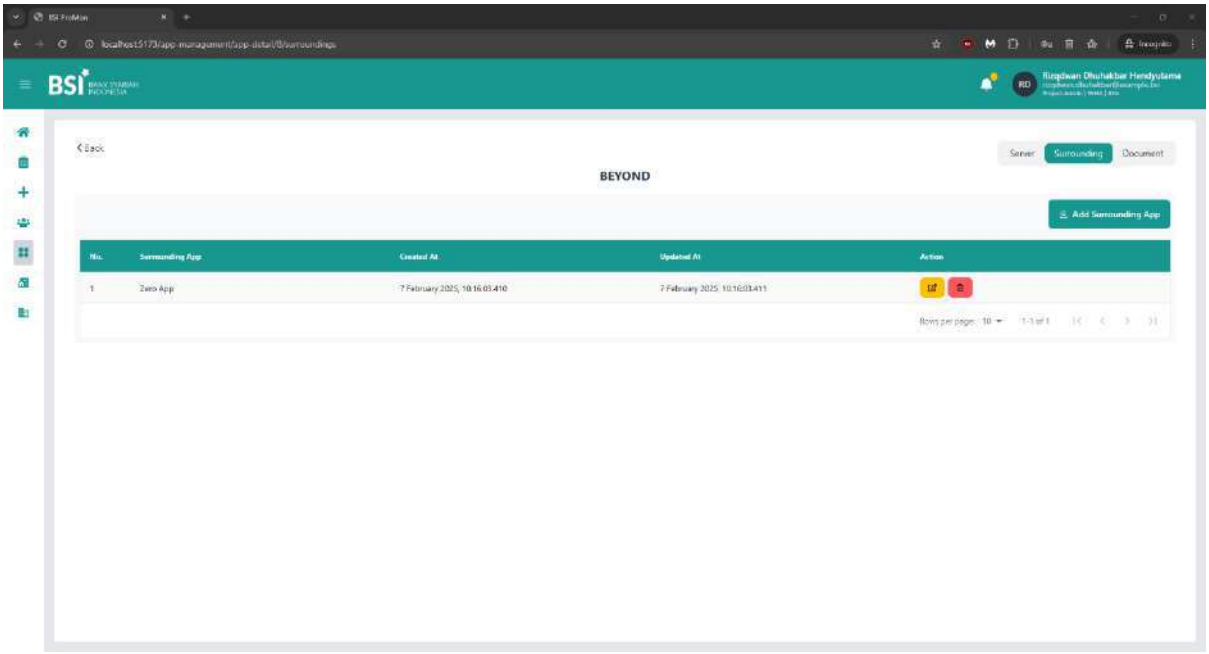
Gambar 3.26 menunjukkan tampilan halaman *server detail* pada halaman *server* dari fitur *Application Management*. Halaman ini menampilkan informasi *installation package* dari *server* aplikasi. *Installation package* berisi teknologi yang diterapkan pada *server* aplikasi.



Gambar 3.27 Tampilan halaman form data *package* baru dari halaman *server detail*

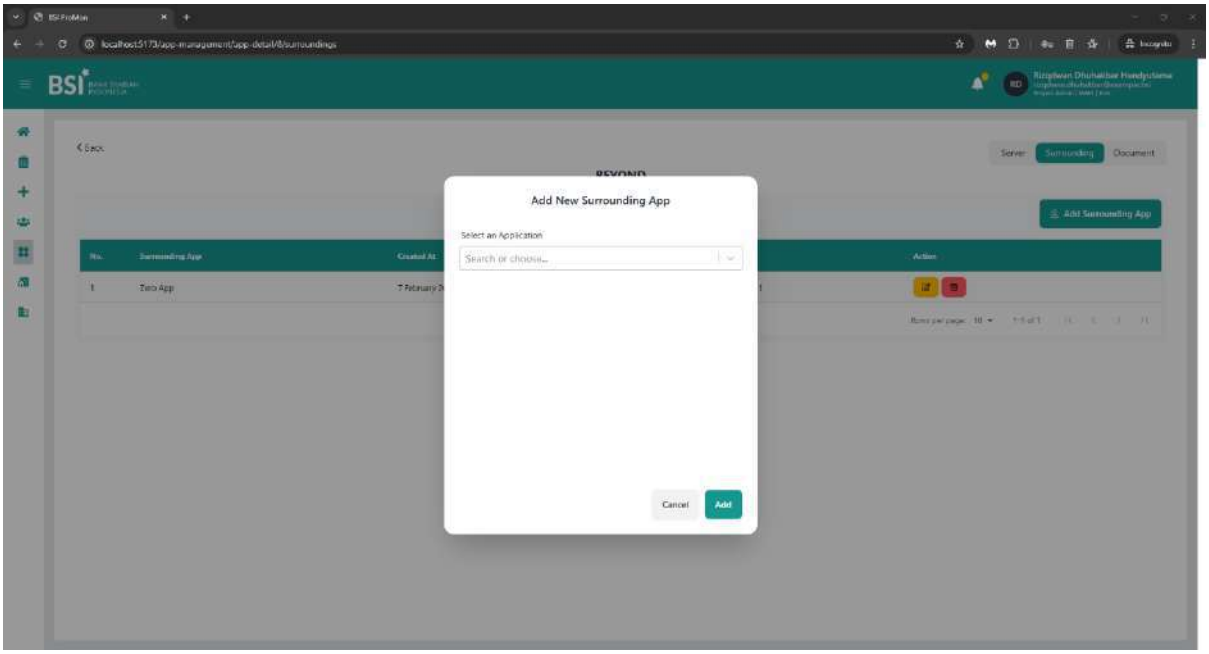
Gambar 3.27 menunjukkan tampilan halaman form pengisian data *package* baru pada halaman *server detail* dari fitur *Application Management*. Fitur ini hanya dapat dilakukan oleh

Project Admin. *Project Admin* perlu memasukkan data nama *package* untuk menambah daftar baru.



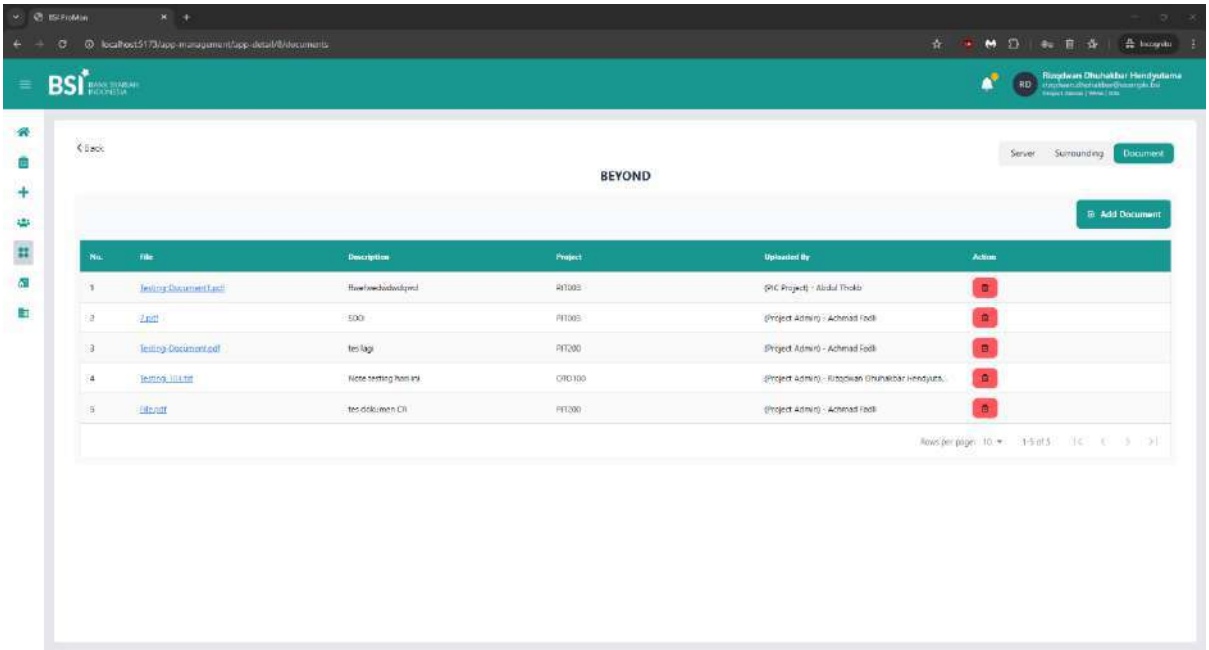
Gambar 3.28 Tampilan halaman *surrounding application* pada fitur *Application Management*

Gambar 3.28 menunjukkan halaman *surrounding application* yang menampilkan daftar relasi aplikasi. Halaman ini berfungsi untuk memberikan informasi aplikasi yang berelasi dengan aplikasi lain. Fitur ini hanya dapat diakses oleh *Project Admin*.



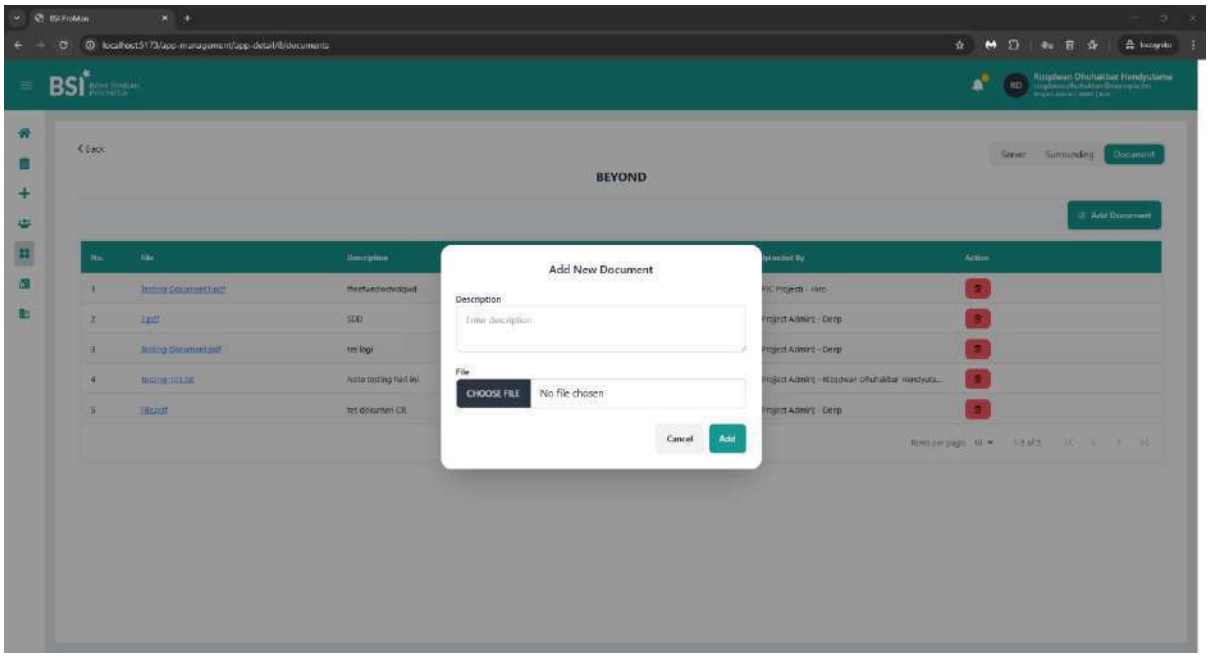
Gambar 3.29 Tampilan halaman form data *surrounding application* baru dari halaman *surrounding application*

Gambar 3.29 menunjukkan halaman form pengisian data *surrounding application* baru pada aplikasi. Fitur ini hanya dapat dilakukan oleh *Project Admin*. *Project Admin* perlu memilih daftar aplikasi untuk dijadikan relasi baru dari aplikasi saat ini.



Gambar 3.30 Tampilan halaman *document application* dari fitur *Application Management*

Gambar 3.30 menunjukkan halaman *document application* dari fitur *Application Management*. Halaman ini berfungsi untuk menampilkan daftar dokumen pada aplikasi dan informasi dokumen seperti nama file, deskripsi, relasi dokumen proyek, dan *upload by*.



Gambar 3.31 Tampilan halaman form data dokumen baru pada halaman *document application*

Gambar 3.31 menunjukkan halaman pengisian form penambahan dokumen baru pada *document application*. Fitur ini hanya dapat dilakukan oleh *Project Admin*. *Project Admin* perlu memasukkan data deskripsi dan lampiran file untuk menambah daftar dokumen aplikasi.

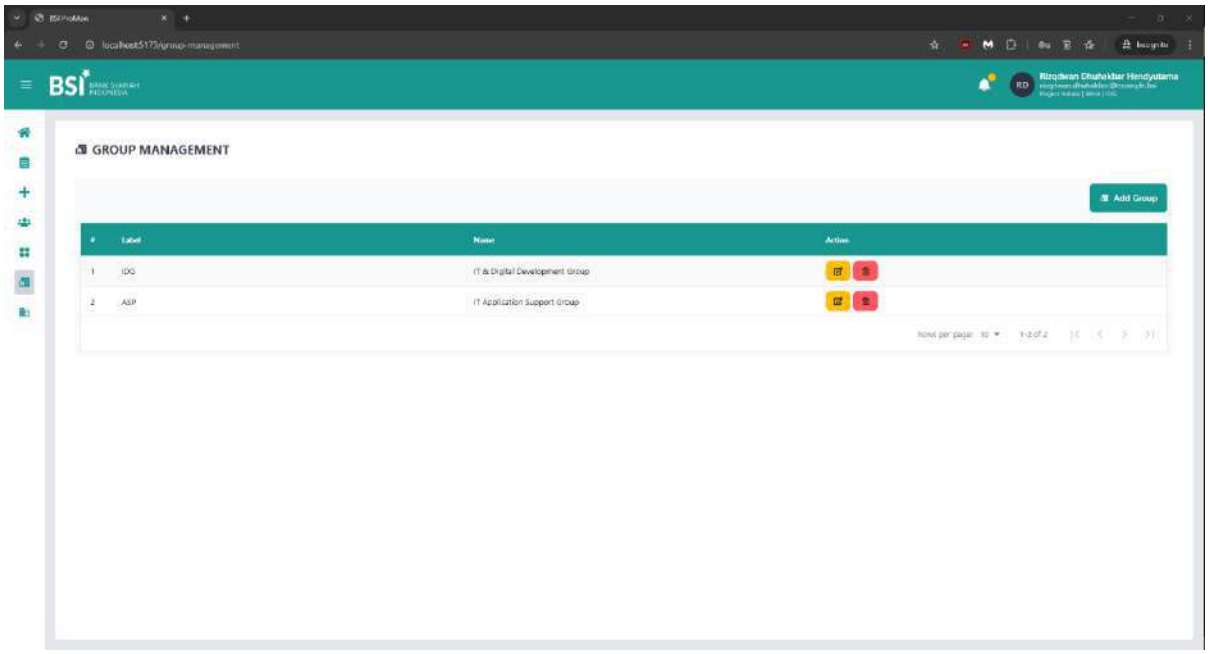
3.3.2.8 Fitur *Group* dan *Department Management*

Tabel 3.8 *Backlog* fitur *Group* dan *Department Management*

Backlog	*Acceptance Criteria*
Sebagai *Project Admin* dapat mengakses tampilan *group management* pada sistem monitoring proyek berbasis web	Menampilkan daftar grup pada sistem
Sebagai *Project Admin* dapat menambahkan data grup baru pada sistem monitoring proyek berbasis web	Melakukan input data dengan mengirimkan label dan nama grup
Sebagai *Project Admin* dapat mengubah data grup pada sistem monitoring proyek berbasis web	Dapat mengubah data grup pada label dan nama

Sebagai *Project Admin* dapat menghapus data grup pada sistem monitoring proyek berbasis web	-
Sebagai *Project Admin* dapat mengakses tampilan *department management* pada sistem monitoring proyek berbasis web	Menampilkan daftar departemen pada sistem
Sebagai *Project Admin* dapat menambahkan data departemen baru pada sistem monitoring proyek berbasis web	Melakukan input data dengan mengirimkan label, nama, dan grup departemen
Sebagai *Project Admin* dapat mengubah data departemen pada sistem monitoring proyek berbasis web	Dapat mengubah data departemen pada label, nama dan grup
Sebagai *Project Admin* dapat menghapus data departemen pada sistem monitoring proyek berbasis web	-

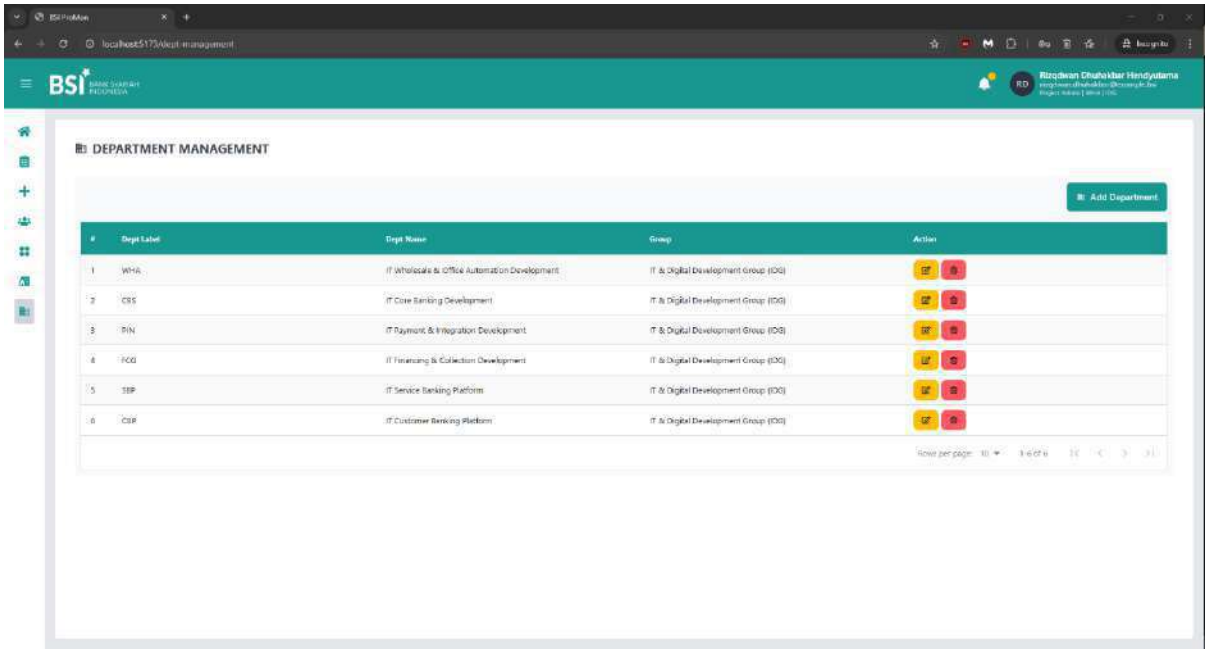
Tabel 3.8 yang menunjukkan *backlog* dari fitur *group* dan *department management* . Fitur ini hanya dapat diakses oleh *Project Admin*. *Project Admin* dapat menambahkan data grup dan departemen baru sesuai dengan *acceptance criteria*, serta menghapus data tersebut.



Gambar 3.32 Tampilan halaman *Group Management*

Gambar 3.32 menampilkan halaman dari fitur *Group Management* yang hanya dapat diakses oleh *Project Admin*. Fitur ini berfungsi sebagai informasi internal terkait daftar grup

pada departemen yang terhubung dengan sistem. *Project Admin* dapat menambahkan data baru untuk daftar grup

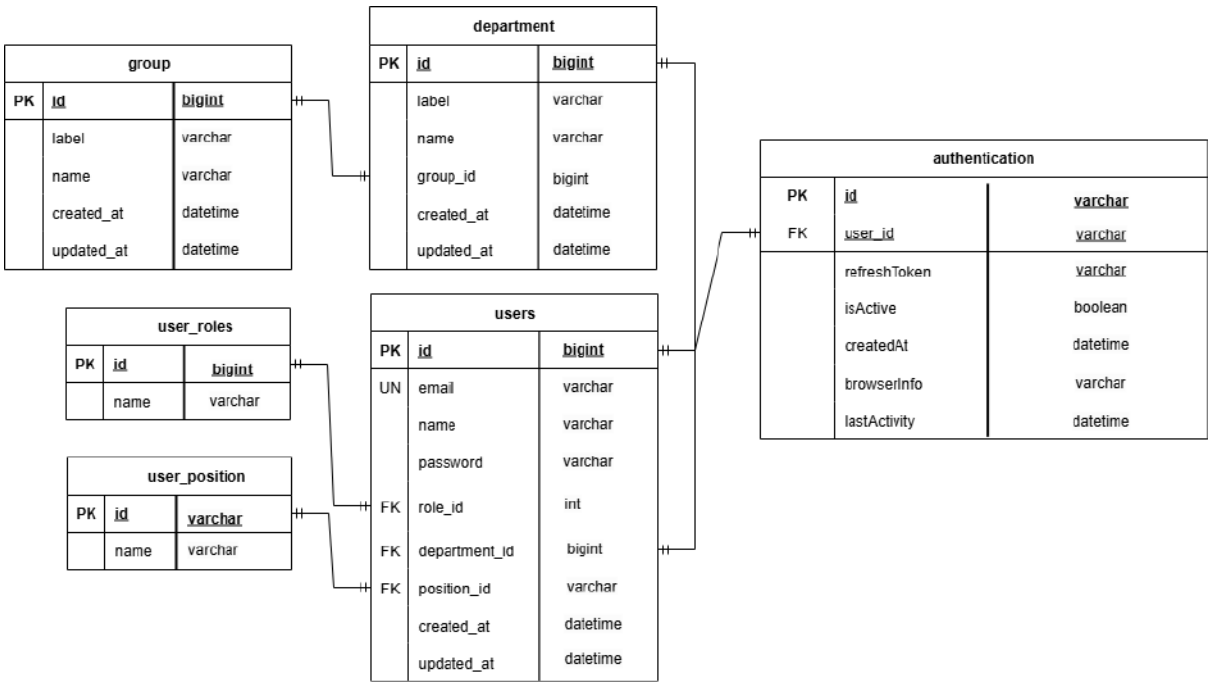


Gambar 3.33 Tampilan halaman *Department Management*

Gambar 3.33 menampilkan halaman dari fitur *Group Management* yang hanya dapat diakses oleh *Project Admin*. Fitur ini berfungsi sebagai informasi internal terkait daftar departemen yang terhubung dengan sistem. *Project Admin* dapat menambahkan data baru untuk daftar departemen. Fitur-fitur yang telah dirancang pada tahap pengembangan ini kemudian diimplementasikan secara teknis menggunakan *framework* dan teknologi yang telah ditentukan. Tahapan implementasi akan dibahas pada subbab berikut, disertai dengan model ERD, serta beberaoa potongan kode untuk masing-masing fitur.

3.3.3 Implementasi Proyek

3.3.3.1 Fitur autentikasi dan registrasi



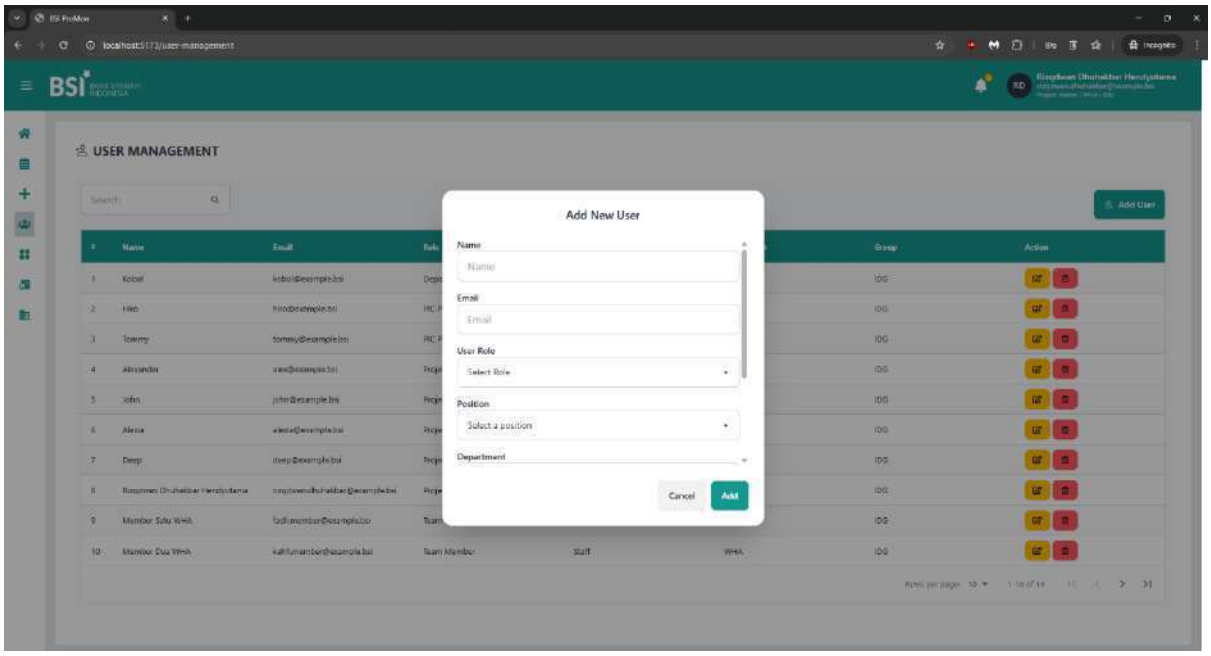
Gambar 3.34 ERD untuk fitur autentikasi dan registrasi

Gambar 3.34 menampilkan *Entity Relation Diagram* (ERD) yang merepresentasikan struktur basis data untuk mendukung fitur autentikasi dan registrasi. Terdapat enam entitas yaitu *users, user_roles, user_positions, group, department,* dan *authentication*. Lima entitas ini digunakan sebagai kebutuhan dalam fungsionalitas pada fitur autentikasi dan registrasi.

- Tabel *users* merupakan entitas utama yang menyimpan informasi pengguna, seperti *email*, nama, *password,* dan atribut relasional ke entitas lain seperti *role_id, department_id,* dan *position_id.*
- Tabel *user_roles* menyimpan daftar peran (*role*) dalam sistem, seperti *Project Admin, Department Head,* PIC, dan *Team Member*. Tabel *user_positions* berfungsi menyimpan daftar posisi atau jabatan pada pengguna, misalnya *Deputy, Department Head, Group Head, Officer, Staff* dan *Team Leader*.

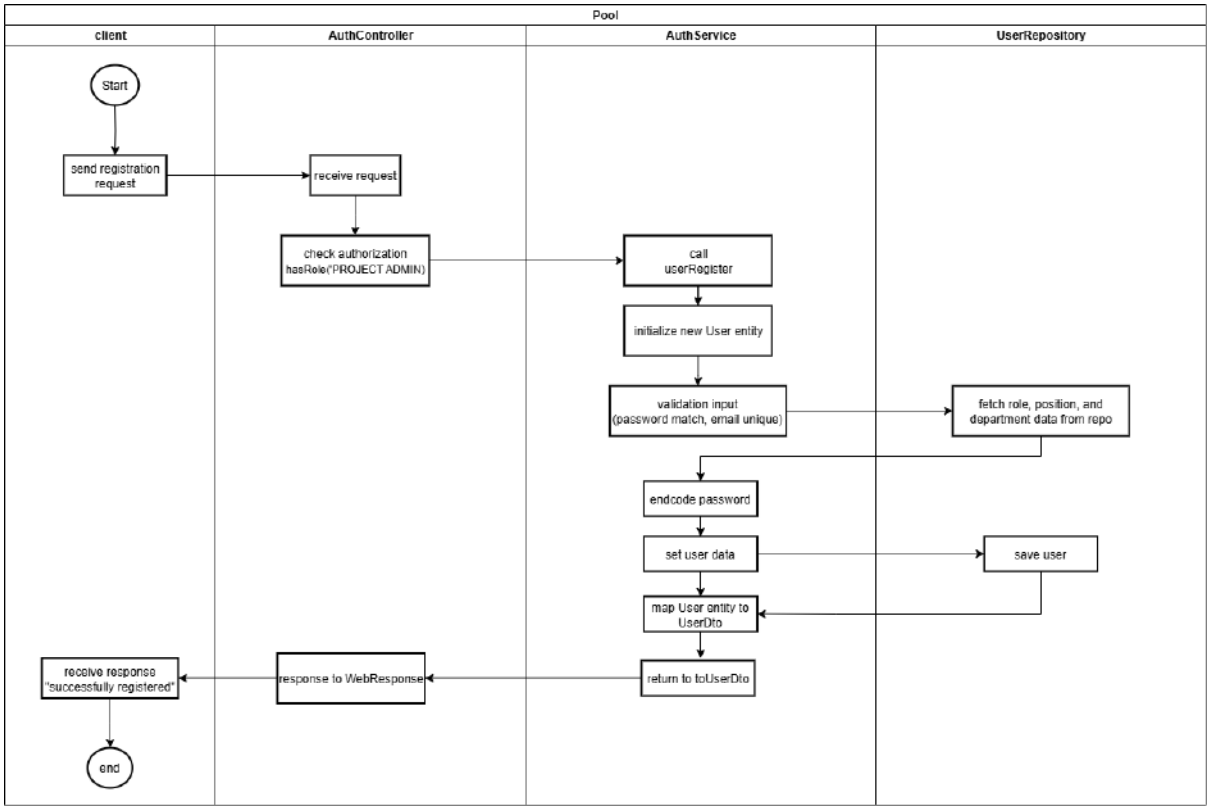
- Tabel *group* dan *department* digunakan untuk mengelompokkan pengguna berdasarkan struktur organisasi di dalam perusahaan. Setiap *department* terhubung dengan satu *group*, dan satu *user* akan memiliki referensi ke *department* tertentu.
- Tabel *authentication* digunakan untuk menyimpan informasi autentikasi tambahan, seperti *refreshToken*, status aktif (*isActive*). *browserInfo*, dan waktu aktivitas terakhir (*lastActivity*). Tabel ini berelasi langsung ke tabel *users* melalui *user_id*, dan digunakan untuk mengelola sesi *login*, validasi token, serta pelacakan aktivitas pengguna.

Registrasi



Gambar 3.35 Tampilan halaman registrasi *user*

Gambar 3.35 menampilkan halaman form registrasi *user* yang hanya dapat dilakukan oleh pengguna dengan peran (*role*) *Project Admin*. *User* melakukan *login* untuk autentikasi, jika *user* memiliki peran *Project Admin* maka dapat melakukan registrasi *user*. Gambar 3.36 menampilkan alur proses sistem dalam registrasi *user*.



Gambar 3.36 Alur aktivitas registrasi *user*

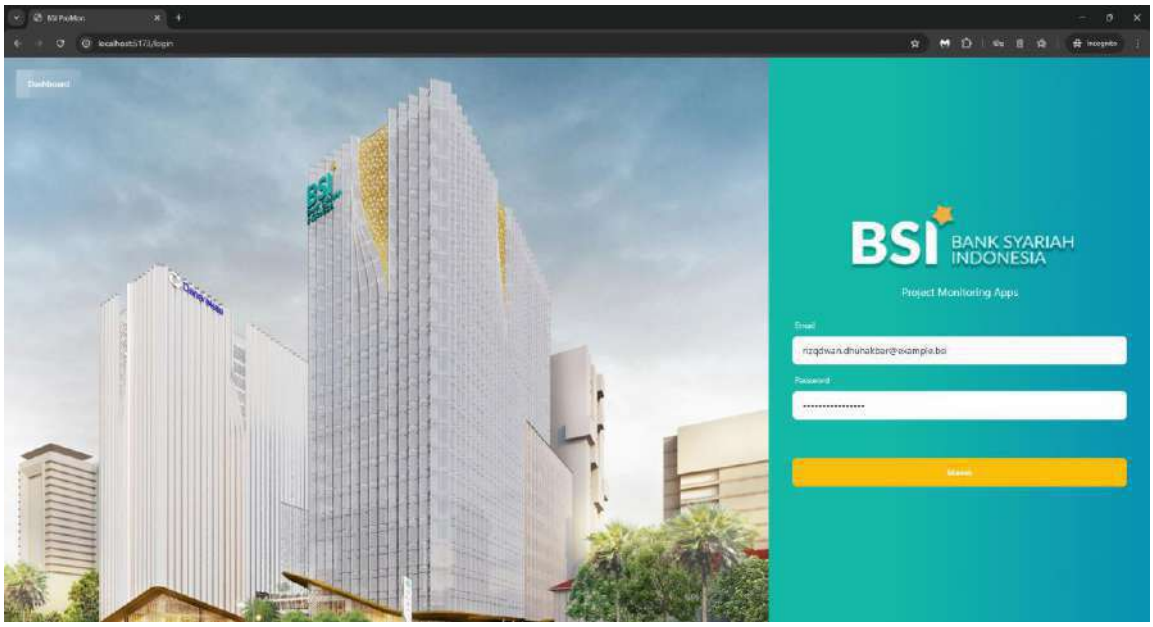
Berdasarkan Gambar 3.36, untuk *user* dapat melakukan registrasi dibutuhkan verifikasi *role* yang dilakukan saat *user* melakukan *login*. Sistem akan melakukan autentikasi untuk pengecekan otorisasi hak *user*. Jika *user* memiliki peran *Project Admin*, maka dapat melakukan registrasi dengan sistem memanggil UserRegister lalu memasukkan isi data *user* dan dilakukan pengecekan validasi data. Jika data *user* sesuai maka registrasi berhasil dan data *user* baru tersimpan. Tampilan *request endpoint* pada fitur registrasi dapat dilihat pada Gambar 3.37.

```
#endpoint
POST /api/v1/auth/signup
#request
{
    "email" : "antonh@example.bsi",
    "password" : "SemangatKita#123",
    "confirmPassword" : "SemangatKita#123",
    "name" : "Antonius",
    "role" : "DEPARTMENT_HEAD",
    "department" : "CBS",
    "position" : "Department Head"
}
```

```
#response
{
    "status": 201,
    "message": "Successfully registered",
    "data": {
        "id": 30057,
        "name": "Antonius",
        "email": "antonh@example.bsi",
        "roleName": "Department Head",
        "departmentName": "IT Core Banking Development",
        "position": "Department Head",
        "active": false
    }
}
```

Gambar 3.37 *Request endpoint register*

Autentikasi

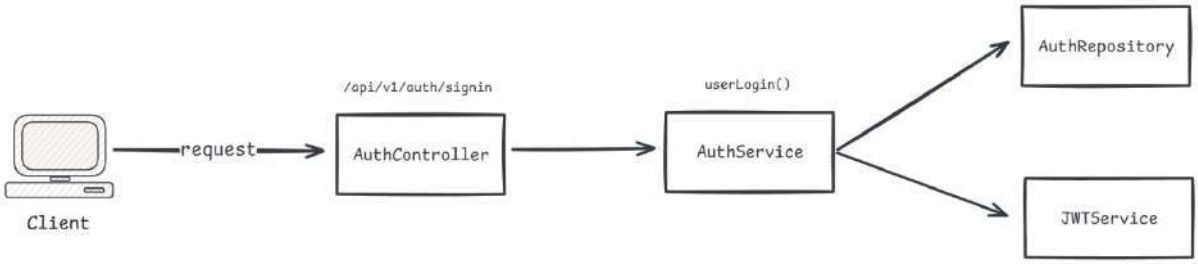


Gambar 3.38 Halaman *login*

User perlu memasukkan *email* dan *password* sebagai kredensial untuk melakukan *login* pada sistem seperti pada Gambar 3.38. Dalam teknisnya *client* akan melakukan *request* seperti pada Gambar 3.39, dengan mengirim kredensial lalu sistem akan melakukan proses autentikasi untuk pengecekan dan jika valid maka sistem akan mengirimkan token JWT yang dapat digunakan *client* dalam mengakses *endpoint* sesuai peran dan tanggung jawabnya.

```
#endpoint
POST /api/v1/auth/signin
#request
{
    "email": "rizqdwan.dhuhakbar@example.bsi",
    "password": "SemangatKita#123"
}
#response
{
    "status": 200,
    "message": "Successfully logged in",
    "data": {
        "accessToken":
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJyaXpxZHdhbi5kaHVoYWtiYXJAZXhhbXBsZS5ic2kiLCJ0eXBlI
joiYWNjZXNzIiwiYXV0aG9yaXRpZXMiOlsicHJvamVjdDpyZWFkIiwiZG9jdW1lbnQ6cmVhZCIsInVzZXI6
ZGVsZXRlIiwiZG9jdW1lbnQ6dXBkYXRlIiwiUk9MRV9QUk9KRUNUX0FETUlOIiwidXNlcjp1cGRhdGUiLCJ
wcm9qZWN0OmRlbGV0ZSIsImRvY3VtZW50OmNyZWF0ZSIsInVzZXI6Y3JlYXRlIiwicHJvamVjdDp1cGRhdG
UiLCJ1c2VyOnJlYWQiLCJwcm9qZWN0OmNyZWF0ZSIsInByb2dyZXNzOnVwZGF0ZSJdLCJpYXQiOjE3NTI2M
DQ0NTYsImV4cCI6MTc1MjYwNDc1NiwianRpIjoiOTU4ZjBjMTgtMGZkNS00NTQyLTg2OTUtYTNkNjFjZGIw
ODU2In0.VDvOSf0ma1zkLycGSWzDSmUiz6J44rPT0yeNxFB9S2QjKRkAxfvujjDo1pVQnisZUqNmtSkZUFD
AmaDXe8VtUw",
        "refreshToken":
"eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJyaXpxZHdhbi5kaHVoYWtiYXJAZXhhbXBsZS5ic2kiLCJpYXQiO
jE3NTI2MDQ0NTYsImV4cCI6MTc1MzIwOTI1NiwianRpIjoiOGM0ZWNjMmMtMmVkZi00MjAwLWEwMjAtMDQ0
MDBkNDIzN2I5IiwidHlwZSI6InJlZnJlc2gifQ.GGcXEUzvDHGnGYaAvg5x12Or87heyVMuYp3GRWnrW5xc
rnOwAX0xB6OYE4YbsMHWc9LdIVDfUVBnqa4K6_S-9g",
        "tokenType": "Bearer",
        "expiresIn": 300
    }
}
```

Gambar 3.39 *Request endpoint login*

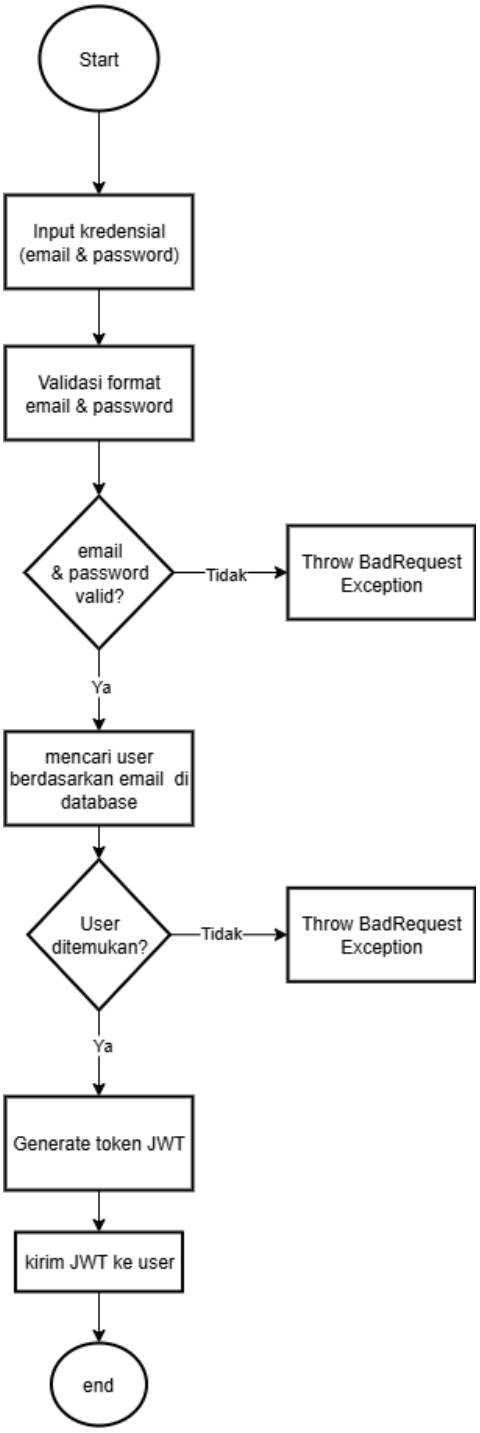


Gambar 3.40 Alur *login* pada sistem

Gambar 3.40 menunjukkan alur pemrosesan *login* pada sistem. Proses diawali ketika *client* mengirimkan pemintaan *login* melalui *endpoint* ke AuthController, yang kemudian meneruskan permintaan tersebut ke AuthService dengan memanggil *method* userLogin(), seperti pada Gambar 3.42. Di service ini proses verifikasi kredensial dengan mengakses data pengguna melalui AuthRepository, dan jika valid, sistem akan menghasilkan token JWT dengan *custom claims* sebagai autentikasi melalui JWTService yang ditampilkan pada Gambar 3.41.


```
{
  "sub": "rizqdwan.dhuhakbar@example.bsi",
  "type": "access",
  "authorities": [
    "project:read",
    "document:read",
    "user:delete",
    "document:update",
    "ROLE_PROJECT_ADMIN",
    "user:update",
    "project:delete",
    "document:create",
    "user:create",
    "project:update",
    "user:read",
    "project:create",
    "progress:update"
  ],
  "iat": 1752601937,
  "exp": 1752602237,
  "jti": "303e1fed-0ef7-4b04-9f6c-ecd67591430f"
}
```


Gambar 3.41 Tampilan *claims* JWT pada sistem

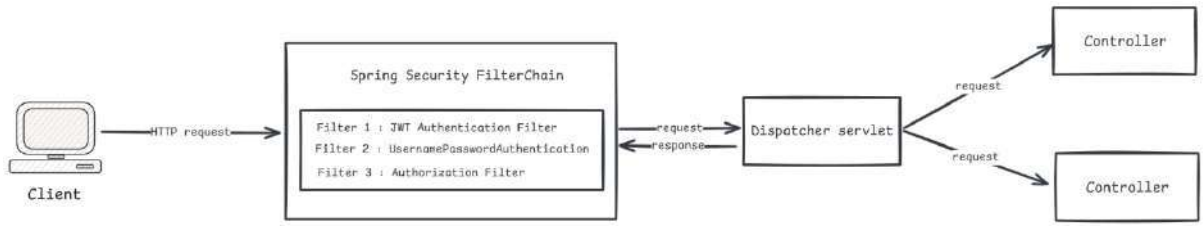


Gambar 3.42 Alur proses mendapatkan token JWT

Authorization

Setelah pengguna melakukan *login* atau autentikasi, maka sistem akan melanjutkan ke proses otorisasi untuk menentukkan batasan hak akses sesuai dengan peran yang diterapkan. Proses ini memanfaatkan fitur dari Spring Security yang memungkinkan konfigurasi serta pengaturan keamanan secara komprehensif, termasuk autentikasi dan otoisasi akses *endpoint*.

Sistem otoriasi yang diterapkan menggunakan pendekatan berlapis (*multi-layered*) yang mencakup *public endpoints, authenticated endpoints, role-based endpoints,* dan *permission-based endpoints*. Gambar 3.43 memperlihatkan alur proses *request* dari *client* yang melewati filter Spring Security sebelum diteruskan ke *controller*. Sementara itu, Gambar 3.44 menunjukkan potongan konfigurasi SecurityFilterChain yang digunakan untuk mengatur otorisasi pada sistem.



Gambar 3.43 Alur Spring Security Filter

```
@Bean
SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception
{
    String[] publicEndpoints = {
            "/api/v1/auth/signin",
            "/api/v1/dashboard/**"
    };

    String[] authenticatedEndpoints = {
            "/api/v1/auth/changePassword",
            "/api/v1/auth/logout",
            "/api/v1/users/current"
    };

    String[] adminEndpoints = {
            "/api/v1/auth/signup",
            "/api/v1/users",
            "/api/v1/departments/**",
    };

)
.sessionManagement(session ->
        session.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
)

.authorizeHttpRequests(auth->auth
```

```
.requestMatchers(publicEndpoints).permitAll()
.requestMatchers(authenticatedEndpoints).authenticated()
.requestMatchers(adminEndpoints).hasRole("PROJECT_ADMIN")

 .requestMatchers("/api/v1/projects/**").hasAnyAuthority(
Permission.PROJECT_READ.getPermission(),
Permission.PROJECT_CREATE.getPermission(),
Permission.PROJECT_UPDATE.getPermission(),
Permission.PROJECT_DELETE.getPermission()
)
.requestMatchers("/api/v1/documents/**").hasAnyAuthority(
Permission.DOCUMENT_READ.getPermission(),
Permission.DOCUMENT_CREATE.getPermission(),
Permission.DOCUMENT_UPDATE.getPermission()
)
 .anyRequest().authenticated()
 )
 .addFilterBefore(authTokenFilter(), UsernamePasswordAuthenticationFilter.class)

return httpSecurity .build();
}
```

Gambar 3.44 Potongan kode konfigurasi pada Spring Security

Dengan menambahkan konfigurasi tersebut, sistem otomatis menambahkan fungsionalitas authorization yang mengatur akses berdasarkan kategori *endpoint* dan hak akses pengguna. *Endpoint* publik seperti *login* dan *dashboard* dapat diakses tanpa autentikasi, sementara *endpoint* seperti *logout* dan pengambilan data pengguna hanya bisa diakses setelah pengguna berhasil *login*. *Endpoint* khusus, seperti pendaftaran akun dan manajemen departemen, dibatasi untuk peran PROJECT_ADMIN menggunakan hasRole(). Selain itu, akses ke fitur proyek dan dokumen dikendalikan lebih detail melalui izin tertentu menggunakan hasAnyAuthority(), mengacu pada *permission* seperti PROJECT_READ, PROJECT_CREATE, dan sebagainya.

Pendekatan ini yang memastikan bahwa setiap *request* hanya diproses jika sesuai dengan peran dan izin yang dimilik pengguna. Filter authTokenFilter() juga ditambahkan sebelum UsernamePasswordAuthenticationFilter untuk menvalidasi token JWT dalam setiap *request*, sehingga proses otorisasi tidak hanya aman tetapi juga sesuai dengan struktur kontrol

akses berbasis peran dan izin yang telah ditentukan. Dan juga penerapan anotasi @PreAuthorize dari Spring Security untuk mengatur kontrol akses pada tingkat *method* pada *layer controller*.

3.3.3.2 Fitur *Dashboard*

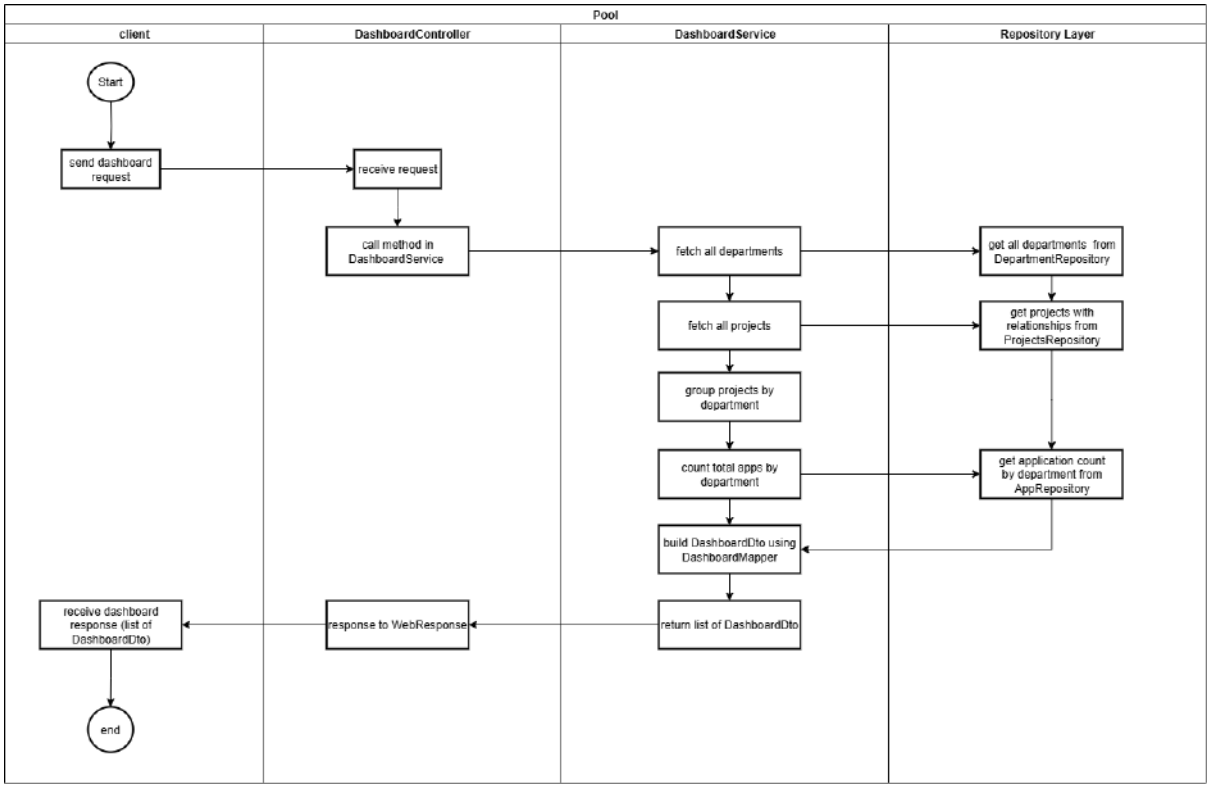
Pengguna dapat secara langsung mengakses fitur ini melalui *button dashboard* yang ada pada halaman *login*, seperti pada Gambar 3.38. Pengguna dapat mengaksesnya tanpa harus melakukan autentikasi terlebih dahulu. Fitur ini menampilkan informasi rekapitulasi data proyek berdasarkan status (seperti *Development*, SIT/UAT, *Production*), jumlah aplikasi, serta distribusi proyek berdasarkan tahun.

```
#endpoints
GET /api/v1/dashboard/departments
#response
{
    "status": 200,
    "message": "Department dashboards retrieved successfully",
    "data": [
        {
            "name": "IT Wholesale & Office Automation Development",
            "label": "WHA",
            "totalApps": 5,
            "totalProjects": 15,
            "readySitUat": 2,
            "sitStatus": 1,
            "uatStatus": 0,
            "sitUatStatus": 0,
            "ptrStatus": 1,
            "prodStatus": 2,
            "devStatus": 9
        },
       ...
       ]
}
```

Gambar 3.45 *Request endpoint* dari halaman *Dashboard*

Untuk melihat struktur *request* dan *response* dari fitur ini, dapat dilihat pada Gambar 3.45 yang memperlihatkan bagaimana sistem merespon *request* pengguna dengan

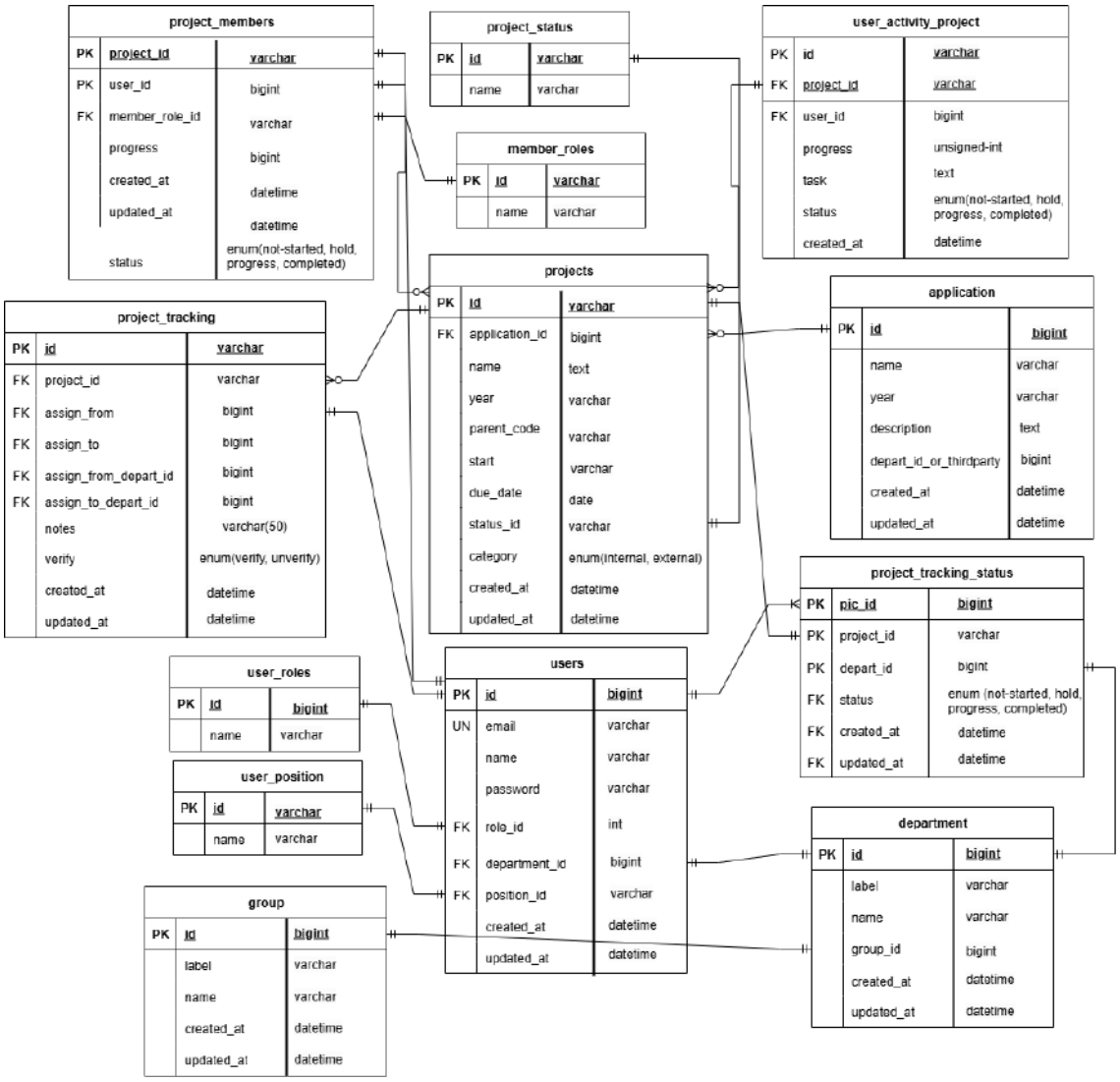
mengembalikan data *dashboard* per departemen. Informasi yang ditampilkan mencakup nama departemen, jumlah proyek, serta rincian jumlah proyek berdasarkan status pengembangan. *Request* ini dapat diakses baik dengan atau tanpa token autentikasi, menandakan bahwa *endpoint* tersebut bersifat publik.



Gambar 3.46 Alur mengambil data dari fitur *dashboard*

Gambar 3.46 menampilkan alur pengambilan data *dashboard* dari sistem yang menggambarkan bagaimana *request* dari *client* diteruskan ke DashboardController, yang kemudian memanggil *method* getAllDepartmentDashboards pada DashboardService. Pada *service* ini, sistem akan mengambil data dari berbagai *repository* seperti DepartmentRepository, ProjectsRepository, dan AppRepository. Setelah itu, proyek dikelompokan berdasarkan departemen dan dihitung jumlah aplikasinya. Data tersebut kemudian diubah menjadi objek DashboardDto menggunakan DashboardMapper, dan akhirnya dikembalikan sebagai *response* ke pengguna dalam bentu daftar informasi *dashboard*.

3.3.3.3 Fitur *Project Management*



Gambar 3.47 ERD untuk fitur *Project Management*

Gambar 3.47 menampilkan ERD dari fitur *Project Management* dengan terdapat 13 entitas yaitu projects, project_tracking, projects_tracking_status, project_users, user_activity_project, project_members, member_roles, application, users, user_roles, user_positions, group, dan department. Seluruh tabel tersebut saling berelasi untuk mendukung pengelolaan proyek secara menyeluruh pada sistem. Tabel projects berfungsi sebagai pusat data utama yang menyimpan informasi proyek, termasuk nama, tahun, dan status proyek. Untuk menyimpan informasi pencatatan penugasan (*assigned*) pengguna pada proyek antar departemen digunakan tabel project_tracking, sedangkan untuk menyimpan informasi perkembangan status proyek dari *not-started, hold, progress* dan *completed* yang ditugaskan

kepada PIC dicatat dalam tabel project_tracking_status. Tabel project_members bersama dengan member_roles digunakan untuk menyimpan informasi dan mengelola anggota proyek beserta peran (*role*) masing-masing. Aktivitas pengguna yang khususnya memiliki peran (*role*) *Team Member* selama proyek berlangsung di catat melalui tabel user_activity_project. Sementara itu, tabel application berfungsi untuk menyimpan informasi relasi proyek pada aplikasi. Informasi pengguna disimpan dalam tabel users, yang kemudian dihubungkan dengan tabel user_roles, user_positions, group, dan department untuk mengatur struktur organisasi serta otoriasi akses dalam sistem. Adapun untuk menyimpan infromasi status proyek dikelola melalui *project_status* yang terhubung langsung dengan tabel projects.

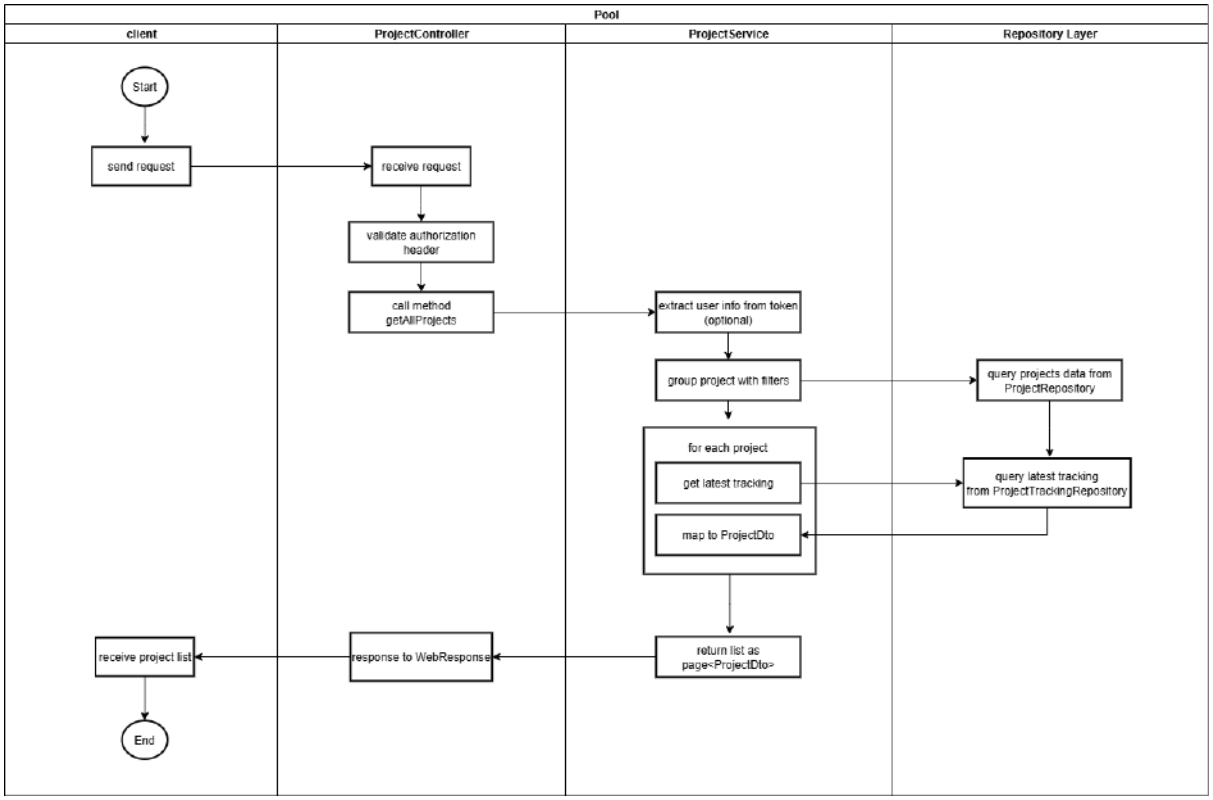
Saat *user* mengakses tampilan dari halaman *All Project*, *user* melakukan *request* pada sistem seperti pada Gambar 3.48 dengan mendapatkan *response* data daftar proyek yang sesuai dengan data *user* dan departemennya. Sistem memproses *request* yang dikirimkan sesuai dengan desain alur yang ditampilkan pada Gambar 3.49.

```
#endpoints
GET /api/v1/projects
#response
{
    "status": 200,
    "message": "Projects retrieved successfully",
    "data": [
        {
            "code": "AKA002",
            "name": "BEYOND Future",
            "year": "2025",
            "parentCode": "PRJ100",
            "applicationId": 6,
            "start": "2025-02-04",
            "dueDate": "2025-02-28",
            "progress": 100,
            "status": "Ready SIT/UAT",
            "category": "INTERNAL",
            "assignTo": [
                {
                    "id": 10048,
                    "name": "Ronggo Member",
                    "email": "ronggo.member@example.bsi",
                    "roleName": "Team Member",
```

```
            "roleLabel": "TM",
           "departmentName": "IT Wholesale & Office Automation Development",
            "departmentLabel": "WHA",
            "position": "Staff",
            "active": true,
            "lastLoginAt": null
        },
        ...
    ],
    "departmentHeads": [
        {
            "id": 10,
            "name": "Dito Pramono",
            "email": "dito@example.bsi",
            "roleName": "Department Head",
            "roleLabel": "DH",
            "departmentName": "IT Payment & Integration Development",
            "departmentLabel": "PIN",
            "position": "Department Head",
            "active": true,
            "lastLoginAt": null
        },
      ...
    ],
    "pics": [
        {
            "id": 10011,
            "name": "Hindia Baskara",
            "email": "hindia@example.bsi",
            "roleName": "PIC Project",
            "roleLabel": "PIC",
            "departmentName": "IT Payment & Integration Development",
            "departmentLabel": "PIN",
            "position": "Officer",
            "active": true,
            "lastLoginAt": null
        },
      ...
    ],
    "teamMembers": [
        {
            "id": 10010,
            "name": "Member Satu WHA",
            "email": "fadli.member@example.bsi",
           "departmentName": "IT Wholesale & Office Automation Development",
```

```
                "roleName": "Front-end Developer",
                "progress": 100,
                "status": "COMPLETED",
                "task": "Task completed",
                "assignedDate": "2025-03-04T12:21:54.770784"
            },
            {
                "id": 10025,
                "name": "Justin Bieber ",
                "email": "justin.bieber@example.bsi",
                "departmentName": "IT Payment & Integration Development",
                "roleName": "Software Developer",
                "progress": 100,
                "status": "COMPLETED",
                "task": "Task completed",
                "assignedDate": "2025-02-27T15:52:32.702076"
            },
          ...
        ],
        "notes": null
    },
```


Gambar 3.48 *Request endpoint* mendapatkan semua daftar proyek



Gambar 3.49 Alur mendapatkan data semua daftar proyek

Proses mendapatkan data daftar proyek yang ditampilkan pada halaman *All Project* diawali dengan *client* mengirimkan *request* pada *endpoint*. *Request* tersebut akan masuk ke dalam ProjectController dan menvalidasi otorisasi guna memastikan bahwa pengguna memiliki hak akses yang sesuai. Setelah proses validasi berhasil, *controller* akan memanggil *method* getAllProjects pada ProjectService. Di dalam *service* ini , sistem mengektrak informasi pengguna dari token (jika dibutuhkan) dan mengelompokkan data proyek berdasarkan filter tertentu, seperti peran pengguna, departemen terkait, dan proyek yang ditugaskan.

Setelah filter diterapkan, ProjectService melakukan pengambilan data proyek dari ProjectRepository. Selanjutnya, untuk setiap proyek yang berhasil diperoleh, sistem akan mengambil informasi pelacakan proyek terbaru dari ProjectTrackingRepository. Hasil dari masing-masing proyek kemudian dipetakan ke dalam objek ProjectDto. Setelah seluruh proses selesai, sistem mengembalikan hasilnya ke dalam bentuk *paged list* Page<ProjectDto> kepada *client*. *Client* kemudian menerima data tersebut dalam tampilan daftar proyek pada halaman *All Project*, seperti pada Gambar 3.8.

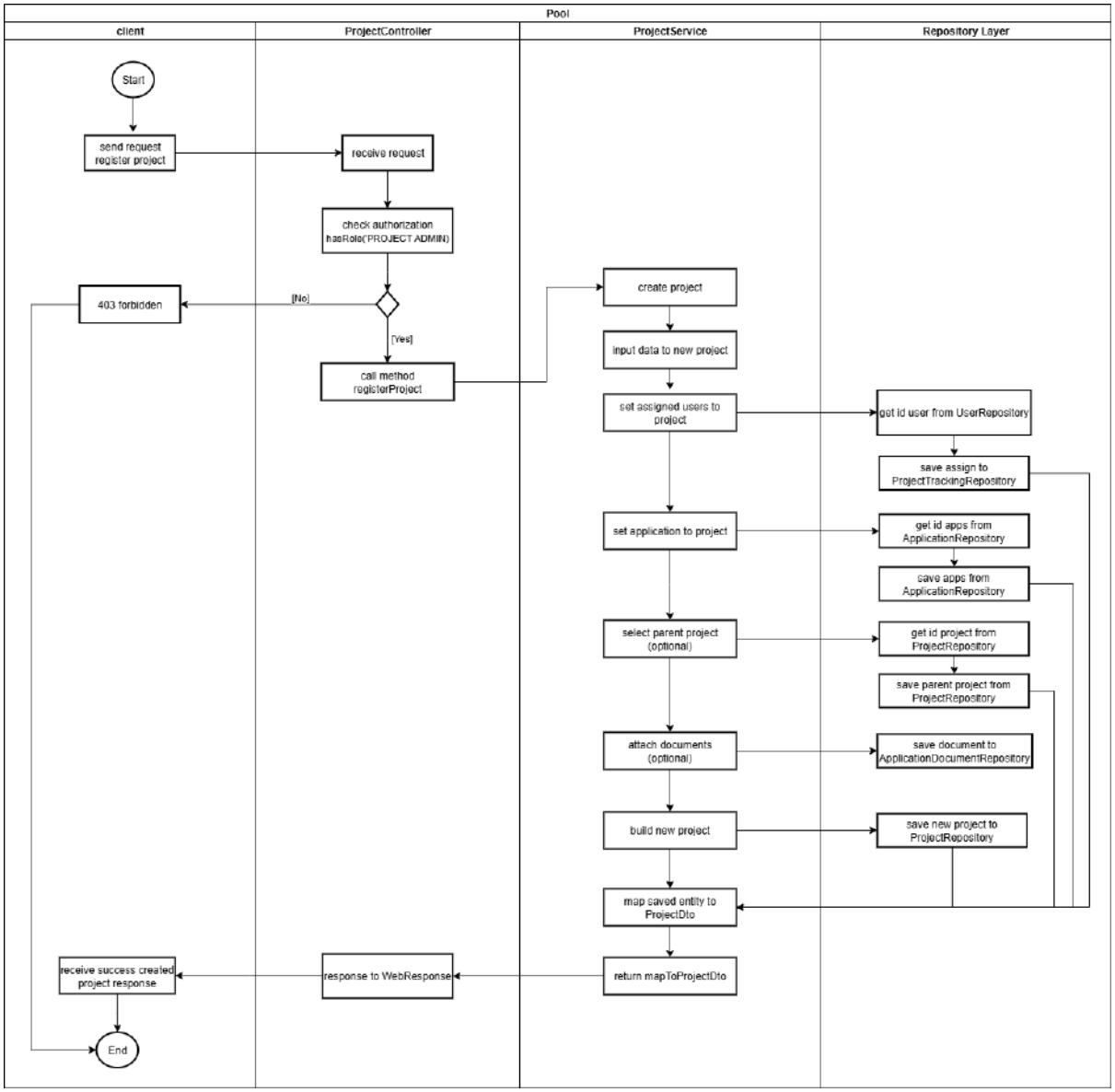
Setelah proses pengambilan data daftar proyek berhasil ditampilkan pada halaman *All Project,* sistem ini juga menyediakan fitur untuk melakukan registrasi proyek baru melalui

halaman *Add Project* yang ditampilkan pada Gambar 3.10. Akses terhadap fitur ini dibatasi hanya untuk pengguna dengan peran (*role*) *Project Admin*. *Client* melakukan *request* ke *endpoint* tersebut dengan mengirimkan beberapa ketentuan untuk registrasi proyek, lalu sistem akan mengembalikan *response* yang memuat detail proyek yang baru diregistrasikan, seperti pada Gambar 3.50. Alur proses *back-end* dalam melakukan registrasi proyek ditunjukkan pada Gambar 3.51.

```
#endpoints
POST /api/v1/projects/register
#request
{
  "code": "PRJ1026",
  "name": "Software 2000",
  "year": "2030",
  "start": "2025-05-10",
  "dueDate": "2030-02-01",
  "assignTo": [1],
  "applicationId": 6,
  "status": "DEV",
  "category": "INTERNAL",
  "parentCode": null,
  "notes": "penugasan untuk DH WHA",
  "description": [],
  "attachment": []
}
#response
{
    "status": 201,
    "message": "Project successfully created",
    "data": {
        "code": "PRJ1026",
        "name": "Software 2000",
        "year": "2030",
        "parentCode": null,
        "applicationId": 6,
        "start": "2025-05-10",
        "dueDate": "2030-02-01",
        "progress": 0,
        "status": "Development",
        "category": "INTERNAL",
        "assignTo": [
            {
```

```
                "id": 1,
                "name": "Kobol",
                "email": "kobol@example.bsi",
                "roleName": "Department Head",
                "roleLabel": "DH",
                "departmentName": "IT Wholesale & Office Automation Development",
                "departmentLabel": "WHA",
                "position": "Department Head",
                "active": true,
                "lastLoginAt": null
            }
        ],
        "departmentHeads": [
            {
                "id": 1,
                "name": "Kobol",
                "email": "kobol@example.bsi",
                "roleName": "Department Head",
                "roleLabel": "DH",
                "departmentName": "IT Wholesale & Office Automation Development",
                "departmentLabel": "WHA",
                "position": "Department Head",
                "active": true,
                "lastLoginAt": null
            }
        ],
        "pics": [],
        "teamMembers": [],
        "notes": null
    }
}
```


Gambar 3.50 *Request endpoint* registrasi proyek



Gambar 3.51 Alur proses registrasi proyek

Untuk proses registrasi proyek diawali *client* mengirimkan *request* ke *endpoit*. *Request* ini diterima oleh ProjectController, yang kemudian memverifikasi otorisasi pengguna dengan peran *Project Admin*. Jika tidak valid, sistem mengembalikan respon 403 *Forbidden*. Jika valid, *method* registerProject dipanggil dan proses dilanjutkan ke ProjectService. Di dalam *service*, sistem memberikan form ketentuan pengisian data registrasi proyek, seperti nama proyek, *start date*, *due date*, tahun proyek, kode proyek, *parent project*, *assign to,* aplikasi, status proyek, kategori, *note* dan lampiran file. Untuk *assign user* pada proyek ditetapkan dengan data ID *user* yang diambil dari UserRepository, lalu menyimpan historisnya ke dalam

ProjectTrackingRepository. Untuk pemilihan *application* sebagai relasi pada proyek, data diambil dari ApplicationRepository.

Jika proyek yang didaftarkan memiliki *parent project*, maka data diambil dari ID proyek melalui ProjectRepository. Selain itu, apabila terdapat dokumen yang dilampirkan pengguna, maka dokumen tersebut akan disimpan melalui ApplicationDocumentRepository. Setelah semua entitas dan relasi berhasil diproses, maka data hasil registrasi proyek tersebut akan disimpan ke dalam ProjectRepository. Data proyek tersebut kemudian akan diubah ke bentuk DTO melalui proses mapping dan dikembalikan ke *client* melalui WebResponse. Data kemudian ditampilkan melalui halaman *All Project* atau *My Project,* seperti pada Gambar 3.8 dan Gambar 3.9.

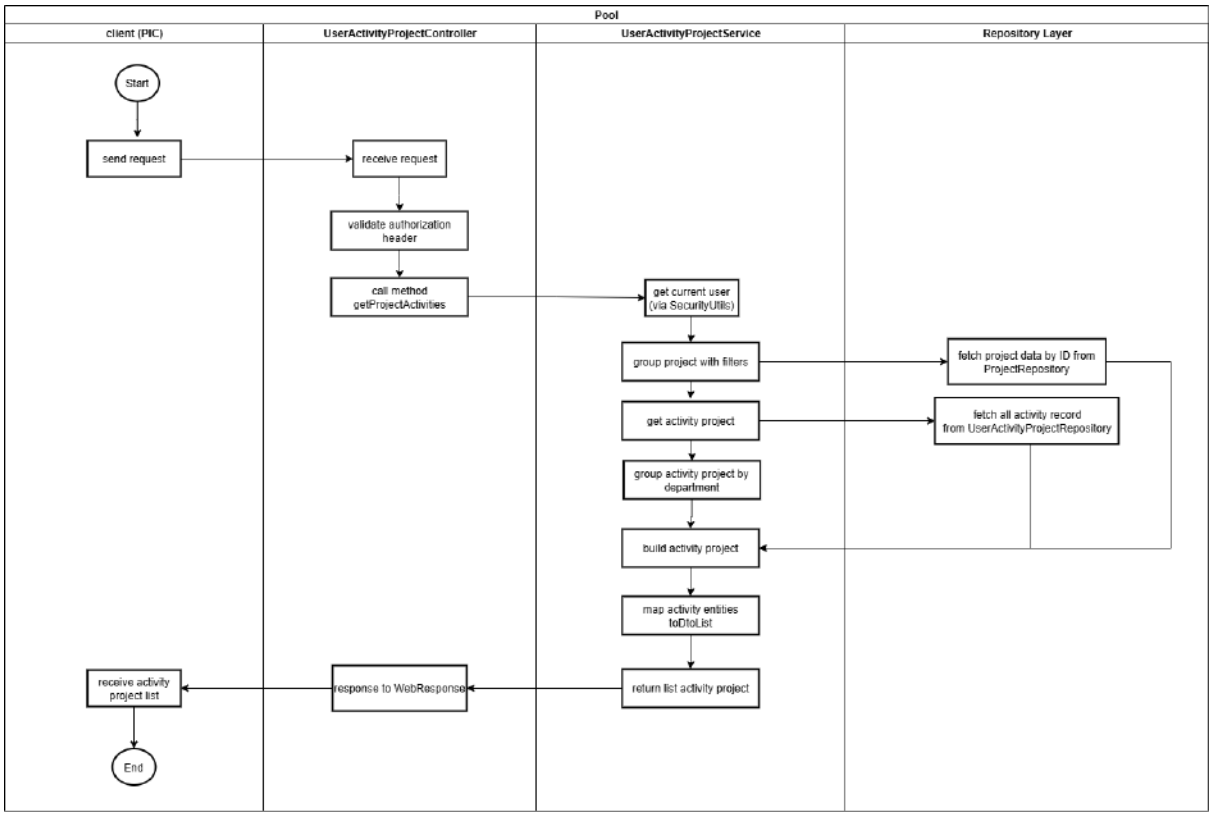
Namun, untuk mendukung proses monitoring proyek terdapat halaman *Activity project* yang menampilkan perkembangan proyek secara lebih rinci. Pada halaman ini terdapat informasi progres setiap anggota tim dari berbagai departemen yang berkolaborasi. Saat pengguna mengakses halaman ini, *client* melakukan *request* ke *endpoint activity project* tersebut, lalu sistem akan mengembalikan *response* yang memuat detail aktivitas proyek, seperti pada Gambar 3.52 dan jika pengguna mengakses tampilan *card* pada halaman *activity project* untuk melihat lebih detail aktivitas proyek dari *user* tertentu maka, *client* melakukan *request* pada *activity user* tersebut, lalu sistem akan mengembalikan *response* yang memuat detail aktivitas proyek dari *user* tertentu, seperti pada Gambar 3.54 dan hasilnya di tampilkan pada Gambar 3.17.


```
#endpoints
GET /api/v1/projects/AKA002/activities
#response
{
    "status": 200,
    "message": "Project activities retrieved successfully",
    "data": [
        {
            "department": "IT Wholesale & Office Automation Development",
            "label": "WHA",
            "information": [
                {
                    "id": "1741333905008",
                    "projectId": "AKA002",
                    "projectName": "BEYOND Future",
                    "userId": 10010,
```

```
                "userName": "Member Satu WHA",
                "email": "fadli.member@example.bsi",
                "departmentName": "IT Wholesale & Office Automation Development",
                "roleName": "Team Member",
                "roleMember": "Front-end Developer",
                "progress": 100,
                "task": "done",
                "status": "COMPLETED",
                "createdAt": "2025-03-07T14:51:45.911074"
            },
            ...
    {
        "department": "IT Payment & Integration Development",
        "label": "PIN",
        "information": [
            {
                "id": "1740973332000",
                "projectId": "AKA002",
                "projectName": "BEYOND Future",
                "userId": 10025,
                "userName": "Justin Bieber ",
                "email": "justin.bieber@example.bsi",
                "departmentName": "IT Payment & Integration Development",
                "roleName": "Team Member",
                "roleMember": "Software Developer",
                "progress": 100,
                "task": "selesai konser",
                "status": "COMPLETED",
                "createdAt": "2025-03-03T10:42:12.699015"
            },
            {
                "id": "1740971554001",
                "projectId": "AKA002",
                "projectName": "BEYOND Future",
                "userId": 10025,
                "userName": "Justin Bieber ",
                "email": "justin.bieber@example.bsi",
                "departmentName": "IT Payment & Integration Development",
                "roleName": "Team Member",
                "roleMember": "Software Developer",
                "progress": 60,
                "task": "mulai konser",
                "status": "PROGRESS",
                "createdAt": "2025-03-03T10:12:34.117484"
            },
```

```
                ...
            ]
        }
    ]
}
```

Gambar 3.52 *Request endpoint Activity Project*



Gambar 3.53 Alur mendapatkan data *activiy project*

Gambar 3.53 menampilkan proses mendapatkan data *actvitiy projet* diawali dengan *client* mengirimkan *request* ke *endpoit*. *Request* ini diterima oleh UserActivityProjectController, kemudian dilakukan proses validasi otorisai untuk memastikan hak akses terhadap proyek. Setelah validasi berhasil, *controller* akan memanggil *method* getProjectActivity pada *service layer*. Di dalam *service*, sistem akan mengambil data *client* melalui SecurityUtil untuk mengecek departemen terkait. Sistem lalu mengambil data proyek dari ProjectRepository dan seluruh aktivitas proyek dari UserActivityProjectRepository. Aktivitas-aktivitas ini dikelompokan berdasarkan departemennya dan dipetakan ke dalam

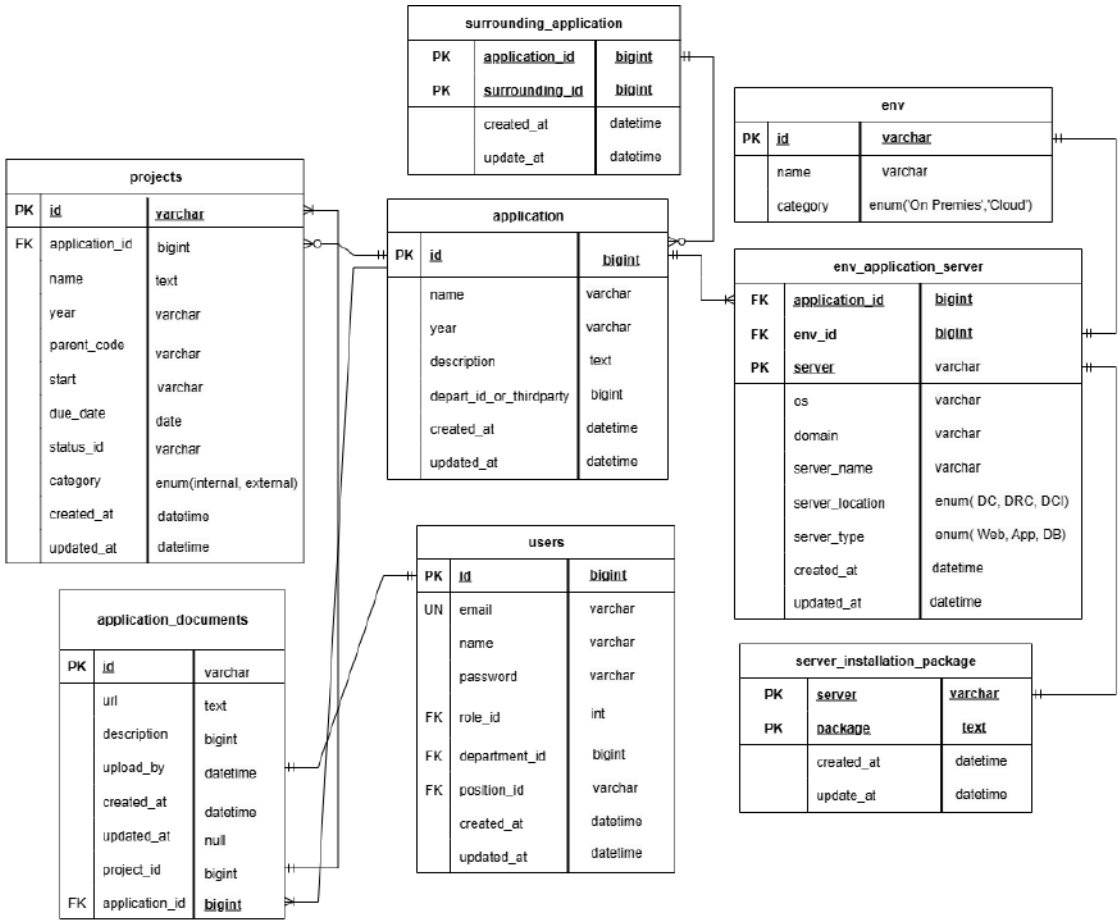
bentuk Dto. Hasil data ini dikembalikan ke *client* melalui WebResponse yang nantinya ditampilkan pada halaman *Activity Project*, seperti Gambar 3.16.


```
#endpoints
GET /api/v1/projects/AKA002/activities/users/10010
#response
{
    "status": 200,
    "message": "User activities retrieved successfully",
    "data": [
        {
            "department": "IT Wholesale & Office Automation Development",
            "label": "WHA",
            "information": [
                {
                    "id": "1741333905008",
                    "projectId": "AKA002",
                    "projectName": "BEYOND Future",
                    "userId": 10010,
                    "userName": "Member Satu WHA",
                    "email": "fadli.member@example.bsi",
                    "departmentName": "IT Wholesale & Office Automation Development",
                    "roleName": "Team Member",
                    "roleMember": "Front-end Developer",
                    "progress": 100,
                    "task": "done",
                    "status": "COMPLETED",
                    "createdAt": "2025-03-07T14:51:45.911074"
                },
                {
                    "id": "1741333747005",
                    "projectId": "AKA002",
                    "projectName": "BEYOND Future",
                    "userId": 10010,
                    "userName": "Member Satu WHA",
                    "email": "fadli.member@example.bsi",
                    "departmentName": "IT Wholesale & Office Automation Development",
                    "roleName": "Team Member",
                    "roleMember": "Front-end Developer",
                    "progress": 47,
                    "task": "test nambah",
                    "status": "PROGRESS",
                    "createdAt": "2025-03-07T14:49:07.520774"
                },
```

```
                {
                    "id": "1741065714004",
                    "projectId": "AKA002",
                    "projectName": "BEYOND Future",
                    "userId": 10010,
                    "userName": "Member Satu WHA",
                    "email": "fadli.member@example.bsi",
                    "departmentName": "IT Wholesale & Office Automation Development",
                    "roleName": "Team Member",
                    "roleMember": "Front-end Developer",
                    "progress": 0,
                    "task": "On development",
                    "status": "PROGRESS",
                    "createdAt": "2025-03-04T12:21:54.787438"
                }
            ]
        }
    ]
}
```


Gambar 3.54 *Request endpoint Activity Project* dari anggota tim

3.3.3.4 Fitur *Application Management*



Gambar 3.55 ERD untuk fitur *Application Management*

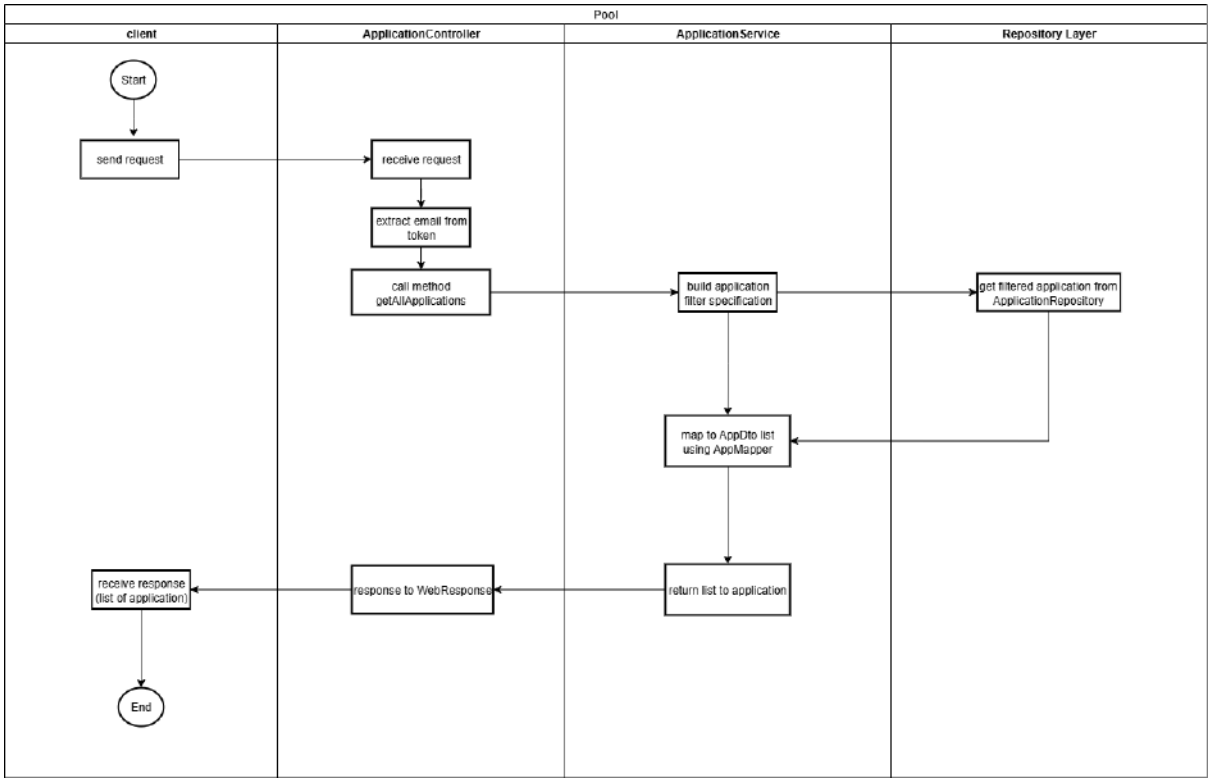
Gambar 3.55 menampilkan ERD dari fitur *Application Management* dengan terdapat delapan entitas yaitu projects, users, application, application_documents, surrounding_application, env, env_application_server, dan server_installation_package. Tabel projects digunakan untuk menyimpan informasi proyek yang berelasi dengan aplikasi. Tabel users berfungsi untuk mencatat data pengguna sistem beserta peran dan keterkaitannya dengan departemen. Tabel application merupakan inti dari manajemen aplikasi yang menyimpan informasi utama terkait aplikasi yang terdaftar. Tabel application_documents digunakan untuk mengelola dokumen yang berkaitan dengan aplikasi. Tabel surrounding_application digunakan untuk menyimpan keterkaitan antar aplikasi yang saling berhubungan satu dengan yang lain. Tabel env digunakan untuk menyimpan data lingkungan sistem seperti *On-Premises* atau *Cloud*. Tabel env_application_server digunakan untuk menyimpan informasi konfigurasi *server* berdasarkan aplikasi dan lingkungan. Terakhir, tabel server_installation_package

digunakan untuk menyimpan informasi paket instalasi yang terpasang pada masing-masing *server*.

Saat *user* mengakses tampilan dari halaman *Application Management*, *user* melakukan *request* pada sistem seperti pada Gambar 3.56 dengan mendapatkan *response* data daftar aplikasi yang sesuai departemennya. Sistem memproses *request* yang dikirimkan sesuai dengan desain alur yang ditampilkan pada Gambar 3.57.

```
#endpoints
GET /api/v1/applications
#request
Token
#response
{
    "status": 200,
    "message": "Applications retrieved successfully",
    "data": [
        {
            "id": 1,
            "name": "X Mobile",
            "year": "2024",
            "description": "is mobile testing",
            "departIdOrThirdparty": {
                "id": 1,
                "label": "WHA",
                "name": "IT Wholesale & Office Automation Development",
                "group": {
                    "id": 1,
                    "label": "IDG",
                    "name": "IT & Digital Development Group"
                }
            }
        },
       ...
       ]
}
```

Gambar 3.56 *Request endpoint* mendapatkan daftar aplikasi



Gambar 3.57 Alur mendapatkan daftar aplikasi

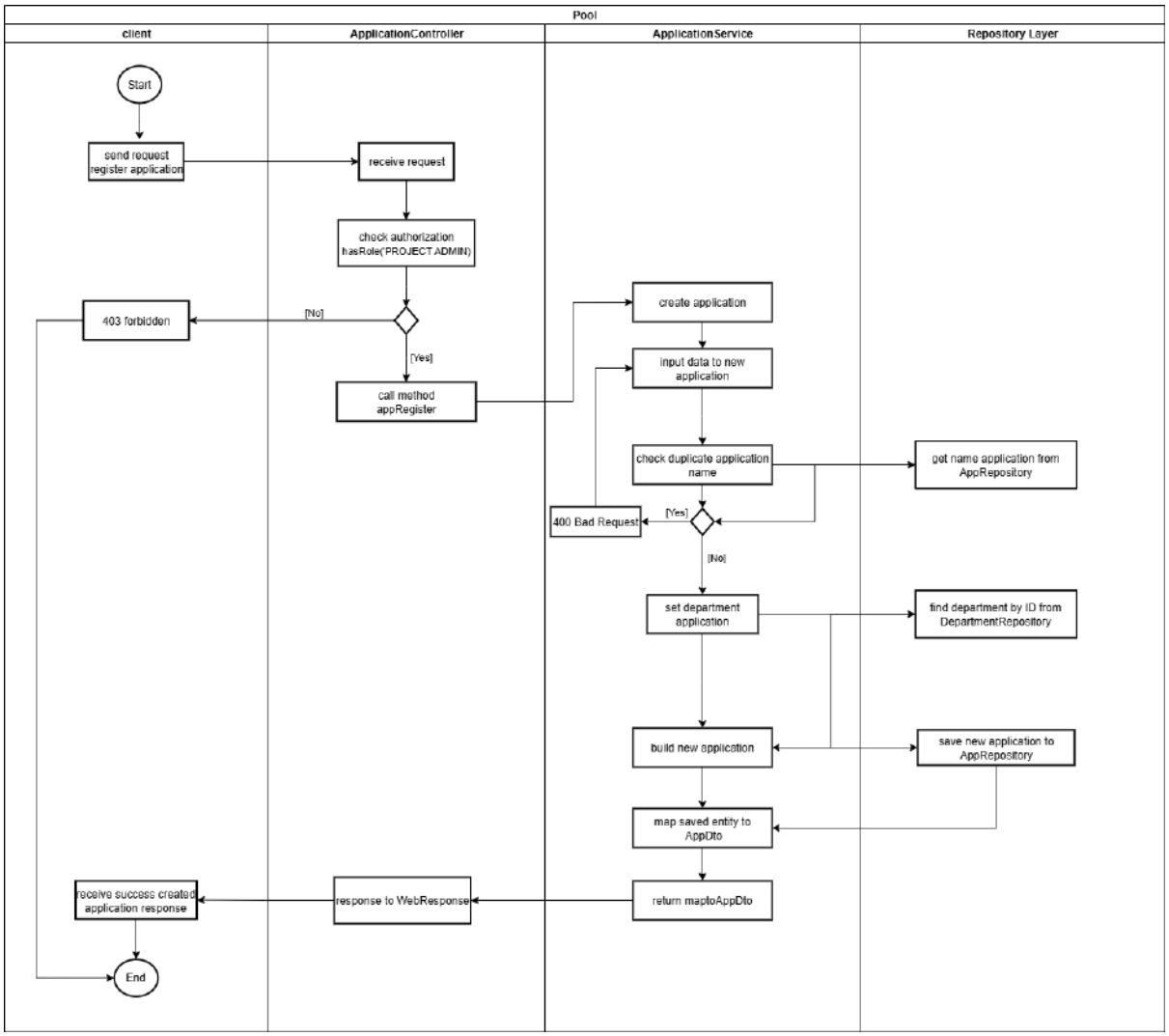
Proses mendapatkan semua daftar aplikasi dimulai ketika *client* mengirimkan *request* ke *endpoint* tersebut. *Request* ini diterima oleh AppliactionController, yang kemudian melakukan validasi terhadap otorisasi dan mengeksrak token untuk mengkonfirmasi peran (*role*) sebagai *Project Admin* dan informasi departemen terkait. Selanjutnya *controller* memanggil *method* getAllApplication pada ApplicationService. Di dalam *service*, data aplikasi difilter dengan data pengguna melalui UserRepository. Setelah entitas pengguna ditemukanm sistem mengambil data departemen terkait yang kemudian digunakan untuk memfilter data aplikasi sesuai departemen pengguna melalui AppRepository. Hasil data daftar aplikasi tersebut kemudian dipetakan ke bentuk AppDto menggunakan AppMapper dan selanjutnya dibungkus ke dalam WebResponse, serta dikirimkan kembali ke *client* sebagai respon. Data response tersebut ditampilkan ke *client* dalam bentuk tampilan, seperti pada Gambar 3.22.

Setelah proses pengambilan data daftar aplikasi berhasil ditampilkan pada halaman *Application Management,* sistem ini juga menyediakan fitur untuk melakukan registrasi aplikasi baru melalui halaman *Application Management* yang ditampilkan pada Gambar 3.23. Akses terhadap fitur ini dibatasi hanya untuk pengguna dengan peran (*role*) *Project Admin*. *Client* melakukan *request* ke *endpoint* tersebut dengan mengirimkan beberapa ketentuan untuk

registrasi aplikasi, lalu sistem akan mengembalikan *response* yang memuat detail aplikasi yang baru diregistrasikan, seperti pada Gambar 3.58. Alur proses *back-end* dalam melakukan registrasi proyek ditunjukkan pada Gambar 3.59.

```
#endpoints
GET /api/v1/applications/register
#request
{
    "name": "Inventory System",
    "year": "15/05/2025",
    "description": "Warehouse management system",
    "departmentId": 1
}
#response
{
    "status": 201,
    "message": "Application Successfully created",
    "data": {
        "id": 20046,
        "name": "Inventory System",
        "year": "15/05/2025",
        "description": "Warehouse management system",
        "departIdOrThirdparty": {
            "id": 1,
            "label": "WHA",
            "name": "IT Wholesale & Office Automation Development",
            "group": {
                "id": 1,
                "label": "IDG",
                "name": "IT & Digital Development Group"
            }
        }
    }
}
```

Gambar 3.58 *Request endpoint* registrasi aplikasi



Gambar 3.59 Alur proses registrasi aplikasi

Untuk proses registrasi proyek diawali *client* mengirimkan *request* ke *endpoit*. *Request* ini diterima oleh ApplicationController, yang kemudian memverifikasi otorisasi pengguna dengan peran *Project Admin*. Jika tidak valid, sistem mengembalikan respon 403 *Forbidden*. Jika valid, *method* appRegister dipanggil dan proses dilanjutkan ke ApplicationService. Di dalam *service*, sistem memberikan form ketentuan pengisian data registrasi aplikasi, seperti nama aplikasi, tahun aplikasi, departemen, dan deskripsi. Untuk nama aplikasi akan dilakukan pengecekan terhadap duplikasi dengan memanggil *method* findByName() pada

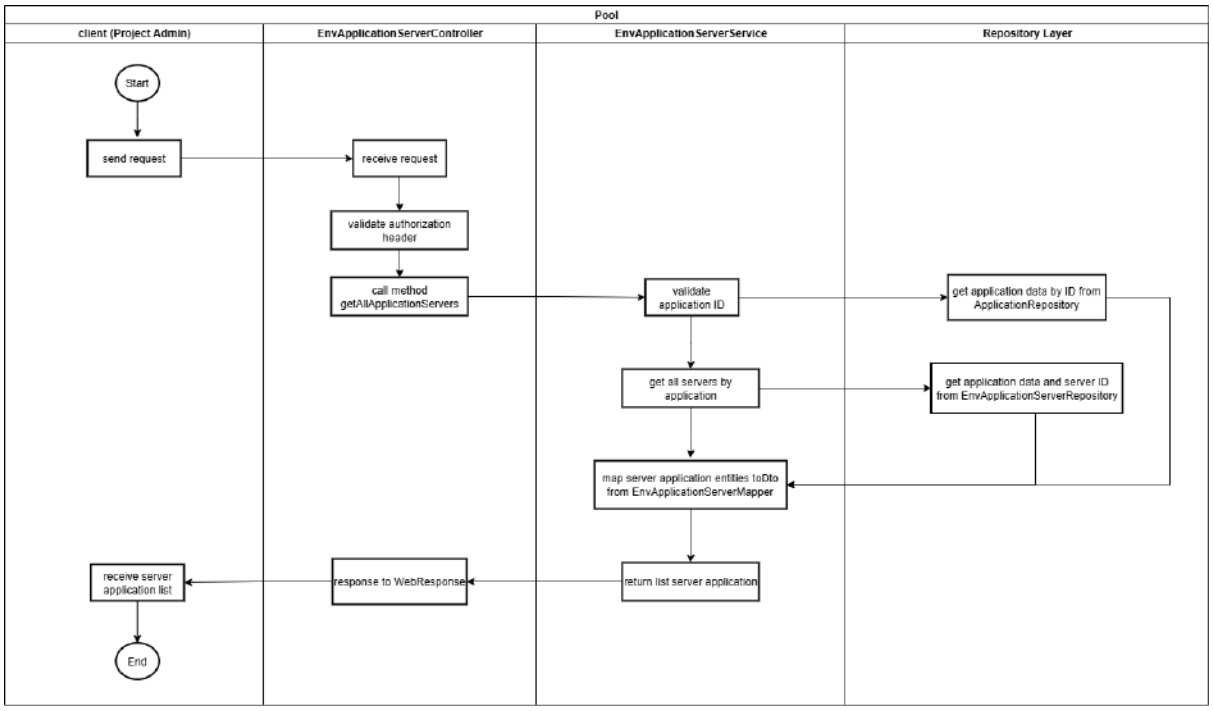
AppRepository. Jika nama aplikasi sudah terdaftar, maka sistem akan memberikan *respons* 404 Bad Request. Jika tidak, maka pengisian data registrasi aplikasi berlanjut ke pilihan departemen yang diambil datanya berdasarkan ID melalui DepartmentRepository. Jika pengisian pada departemen sudah sesuai maka data akan disisipkan ke dalam objek aplikasi, lalu disimpan ke dalam basis data menggunakan *method* save() dari AppRepository. Hasil data kemudian dikonveris menjadi objek AppDto menggunakan AppMapper, lalu dikirimkan kembali ke *client* melalui WebResponse dan data hasil registrasi aplikasi kemudian ditampilkan melalui halaman *Application Management,* seperti pada Gambar 3.21.

Pada tampilan dari halaman *Application Management*, terdapat halaman lanjutan yang menampilkan informasi *server* pada aplikasi. Pengguna dapat mengakses melalui nama aplikasi yang diberikan warna berbeda yang akan diarahkan ke halaman *server*. Saat pengguna melakukan akses pada halaman tersebut, *client* melakukan *request* pada *endpoint server* tersebut, lalu sistem akan mengembalikan *response* yang memuat detail *server* aplikasi tertentu, seperti pada Gambar 3.60 dan alur prosesnya ditampilkan pada Gambar 3.61.


```
#endpoints
GET /api/v1/applications/8/servers
#response
{
    "status": 200,
    "message": "Successfully retrieved all servers",
    "data": [
        {
            "server": "192.101.1.3",
            "appId": 8,
            "appName": "BEYOND",
            "envId": "QA",
            "envName": "Testing",
            "categoryDisplayName": "On Premise",
            "os": "Windows",
            "domain": "dev.mobile.co.id",
            "serverName": "sdbsdbdsb",
            "serverLocation": "DC",
            "serverType": "WEB",
            "createdAt": "2025-01-13T09:28:30.801302",
            "updatedAt": "2025-05-25T19:26:57.246361"
        },
       ...
    ]
```

```
}
```

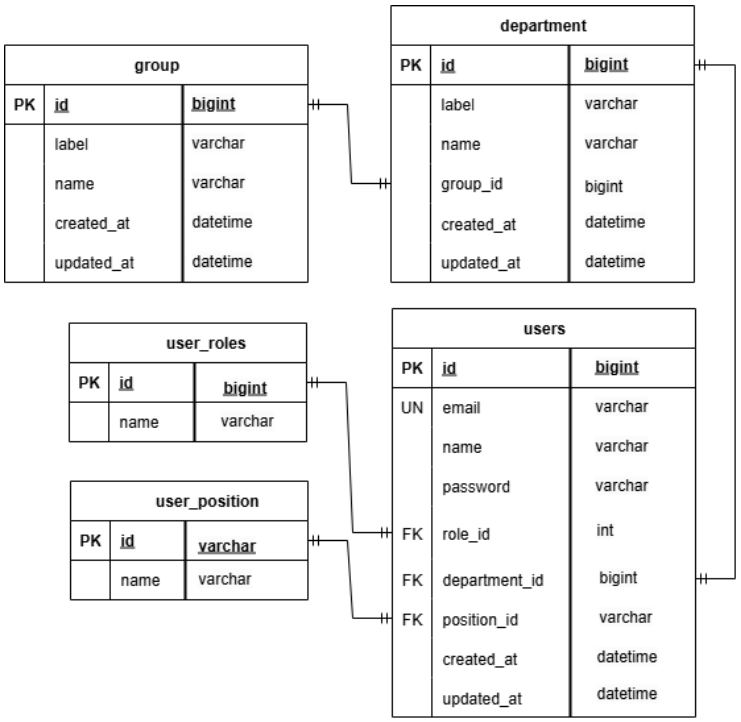
Gambar 3.60 *Request endpoint server* untuk mendapatkan daftar *server* pada aplikasi tertentu



Gambar 3.61 Alur mendapatkan semua daftar *server application*

Proses pengambilan seluruh daftar *server application* dimulai ketika *client* dengan peran *Project Admin* mengirimkan *request* ke *endpoint* tersebut. *Request* diterima oleh *controller* melalui EnvApplicationServerController dan juga melakukan pengecekan otorisasi hak akses. Selanjutnya *controller* meneruskan ke *service layer* dengan memanggil *method* getAllApplicationServers. Di dalam *service,* sistem terlebih dahulu memvalidasi aplikasi berdasarkan ID yang dikirimkan dengan menggunakan findById melalui AppRepository. Jika aplikasi valid, sistem akan mengambil seluruh daftar *server* yang terkait aplikasi menggunakan *method* findByApplicationOrderByCreatedAtAsc dari EnvApplicationServerRepository. Hasil pengelompokan data tersebut kemudian dipetakan ke dalam objek Dto dengan memanfaatkan EnvApplicationServerMapper. Data dto yang dihasikkan kemudian dikembalikan ke *client* melalui WebResponse dan ditampilkan pada halaman *server*, seperti pada Gambar 3.24.

3.3.3.5 Fitur *User Management*



Gambar 3.62 ERD untuk fitur *User Management*

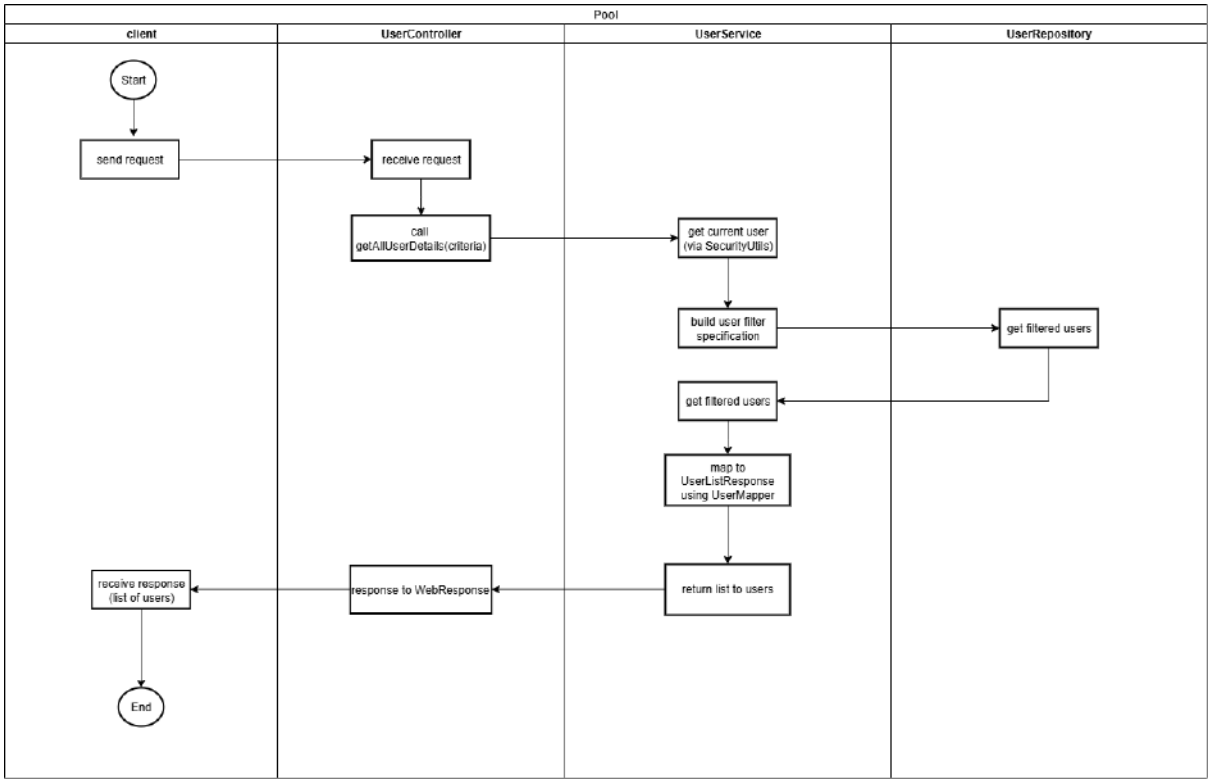
Gambar 3.62 menampilkan ERD dari fitur *User Management* dengan terdapat lima entitas yaitu *users, user_roles, user_positions, group,* dan *department*. Lima entitas ini digunakan sebagai kebutuhan dalam fungsionalitas pada fitur autentikasi dan registrasi. Tabel *user_roles* berisikan daftar peran pada pengguna seperti *Project Admin, Department Head,* PIC*,* dan *Team Member*. Tabel *user_positions* berisikan daftar posisi atau jabatan pada pengguna seperti *Deputy, Department Head, Group Head, Officer, Staff* dan *Team Leader*.

Saat *user* mengakses tampilan dari halaman *User Management*, *user* melakukan *request* pada sistem seperti pada Gambar 3.63 dengan mendapatkan *response* data daftar users yang sesuai dengan data *user* dan departemennya. Sistem memproses *request* yang dikirimkan sesuai dengan desain alur yang ditampilkan pada Gambar 3.64.

```
#endpoints
GET Api/v1/users
#request
Token
#response
{
```

```
    "status": 200,
    "message": "Successfully retrieved all users",
    "data": [
        {
            "id": "1",
            "name": "Kobol",
            "email": "kobol@example.bsi",
            "roleName": "Department Head",
            "department": {
                "id": 1,
                "label": "WHA",
                "name": "IT Wholesale & Office Automation Development",
                "group": {
                    "id": 1,
                    "label": "IDG",
                    "name": "IT & Digital Development Group"
                }
            },
            "position": {
                "id": "DH",
                "name": "Department Head"
            },
            "lastLoginAt": null,
            "active": true
        },
       ...
       ]
}
```

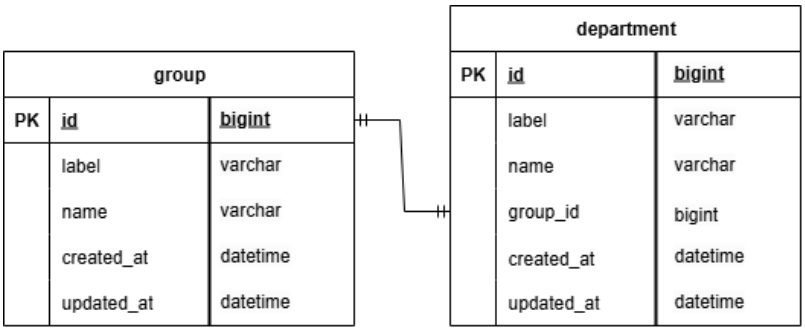

Gambar 3.63 *Request endpoint* dari halaman *user management*



Gambar 3.64 Alur mendapatkan semua daftar *user*

Proses mendapatkan data daftar *user* yang ditampilkan pada halaman *User Management* diawali dengan *client* mengirimkan *request* pada *endpoint*. *Request* tersebut diterima oleh UserController dan diteruskan dengan memanggil *method* getAllUserDetails. Selanjutnya, UserService akan mengambil informasi departemen pengguna melalui SecurityUtils untuk menampilkan daftar *user* dari departemen tersebut. Data itu didapat dari filter *query* melalui UserRepository. Hasil tersebut kemudian di-*map* ke dalam bentu UserListResponse menggunakan UserMapper. Setelah prosess *mapping* selesai, data daftar *user* dikembalikan ke *service* dan dibungkus dalam format WebResponse. *Response* tersebut dikirimkan kembali ke *client* dalam bentuk daftar *user* yang ditampilkan pada Gambar 3.21.

3.3.3.6 Fitur *Group* dan *Department Management*



Gambar 3.65 ERD untuk fitur *Group* dan *Department Management*

Gambar 3.65 menampilkan ERD dari fitur *Group* dan *Department Management* dengan dua entitas yaitu group, dan department. Terdapat dua entitas, yaitu group dan department yang saling memiliki relasi *one-to-many*, di mana satu *group* dapat memiliki banyak *department*. Struktur ini digunakan untuk mempermudah pengelompokan pengguna berdasarkan struktur organisasi di dalam perusahaan.

```
#endpoints
GET /api/v1/groups
#request
Token
#response
{
    "status": 200,
    "message": "Successfully retrieved all group",
    "data": [
        {
            "id": 1,
            "label": "IDG",
            "name": "IT & Digital Development Group"
        },
        {
            "id": 17,
            "label": "ASP",
            "name": "IT Application Support Group"
        }
    ]
}
```

Gambar 3.66 *Request endpoint* dari halaman *group management*

Gambar 3.66 menampilkan *request* dari *client* untuk mendapatkan semua daftar grup pada sistem. *Request* ini hanya dapat dilakukan oleh pengguna yang memiliki peran (*role*) *Project Admin*. Sistem akan meverifikasi otorisasi pengguna yang mencoba *request* tersebut, jika pengguna tidak memiliki hak akses maka sistem akan mengirimkan respon 403 Forbidden. Sama halnya dengan Gambar 3.67 yang menampilkan *request* untuk mendapatkan daftar departmen pada sistem yang hanya dapat dilakukan oleh *Project Admin*.

```
#endpoints
GET /api/v1/departments
#request
Token
#response
{
    "status": 200,
    "message": "Successfully retrieved all departments",
    "data": [
        {
            "id": 1,
            "label": "WHA",
            "name": "IT Wholesale & Office Automation Development",
            "group": {
                "id": 1,
                "label": "IDG",
                "name": "IT & Digital Development Group"
            }
        },
       ...
       ]
 }
```

Gambar 3.67 *Request endpoint* dari halaman *department management*

3.4 Hasil Pengembangan Proyek

Pengembangan proyek ini menghasilkan berbagai fitur dan layanan yang mendukung kebutuhan sistem. Bab ini akan menyajikan hasil implementasi dari setiap fitur dan layanan yang telah dikerjakan oleh penulis. Setiap fitur dan layanan yang terintegrasi melalui API diimplementasikan dengan menerapkan arsitektur RESTful API.

Tabel 3.9 *Endpoint* autentikasi dan registrasi

Metode	URL Endpoint	Deskripsi
POST	/api/v1/auth/signup	Melakukan pendaftaran akun pengguna.
POST	/api/v1/auth/signin	Melakukan login atau autentikasi untuk memulai sesi.
DELETE	/api/v1/auth/logout	Mengakhiri sesi penggunaan akun.
POST	/api/v1/auth/renew	Melakukan pembaharuan sesi token.

Tabel 3.9 menyajikan daftar *endpoint* yang digunakan untuk proses autentikasi dan registrasi pada sistem. Setiap *endpoint* dikembangkan menggunakan metode HTTP. Layanan signup digunakan untuk melakukan pendaftaran akun pengguna baru kedalam sistem, namun akses layanan ini hanya dapat dilakukan oleh *Project Admin* melaui fitur *user management*. Pada layanan *sign-in* berfungsi untuk melakukan proses autentikasi pengguna, sehingga sistem dapat memverifikasi kredensial dan memulai sesi. Sistem ini membatasi sesi autentikasi hanya pada satu akun dalam satu *browser* dan perangkat yang sama. Melakukan login dengan akun yang sama di perangkat berbeda atau *browser* berbeda dalam satu perangkat tidak diizinkan. layanan *renew* digunakan untuk melakukan pembaruan token autentikasi secara otomatis guna menjaga sesi autentikasi tetap aktif tanpa melakukan login ulang, selama refresh token masih berlaku.

Tabel 3.10 *Endpoint* fitur *Dashboard*

Metode	URL Endpoint	Deskripsi
GET	/api/v1/dashboard/departments	Mendapatkan daftar data kalkulasi proyek dari semua departemen.
GET	/api/v1/dashboard/departments/{departmentId}	Mendapatkan data informasi proyek dari departemen tertentu.

Tabel 3.10 menyajikan daftar *endpoint* pada fitur *dashboard* yang dikembangkan. Fitur ini dapat diakses oleh pengguna yang melakukan login ataupun tidak. Kalkulasi data proyek berupa informasi total aplikasi, total proyek, proyek yang selesai, proyek yang sudah masuk

dalam produksi dan juga beberapa tahapan status pada proyek. Fitur ini membantu pengguna dalam pengelolaan dan pemantauan proyek dari departemen sendiri maupun hasil kolaborasi.

Tabel 3.11 *Endpoint* fitur *Project Management*

Metode	URL Endpoint	Deskripsi
GET	/api/v1/projects/department/all	Mendapatkan daftar data informasi proyek dari semua departemen.
GET	/api/v1/projects/my-project	Mendapatkan semua data informasi proyek yang telah ditugaskan kepada pengguna.
POST	/api/v1/projects/register	Membuat proyek baru.
DELETE	/api/v1/projects/{projectId}	Menghapus proyek tertentu.
GET	/api/v1/projects/{projectId}	Mendapatkan detail informasi proyek tertentu.
PUT	/api/v1/projects/{projectId}	Melakukan pembaharuan pada detail informasi proyek tertentu.
PUT	/api/v1/projects/{projectId}/status	Melakukan pembaharuan status pada proyek tertentu.
GET	/api/v1/projects/{projectId}/activities	Mendapatkan semua data informasi aktivitas proyek dari proyek tertentu.
GET	/api/v1/projects/{projectId}/documents	Mendapatkan semua data dokumen dari proyek tertentu.
POST	/api/v1/projects/{projectId}/documents	Membuat beberapa dokumen baru pada proyek tertentu.
POST	/api/v1/projects/{projectId}/document	Membuat dokumen baru pada proyek tertentu.
DELETE	/api/v1/projects/{projectId}/documents/{documentId}	Menghapus dokumen dari proyek tertentu.
GET	/api/v1/projects/{projectId}/members/assignment/all	Mendapatkan data informasi anggota tim proyek dari proyek tertentu.
PUT	/api/v1/projects/{projectId}/members/assignment	Melakukan pembaharuan penugasan dan peran

		anggota tim proyek dari proyek tertentu.

Pada Tabel 3.11 menyajikan daftar beberapa *endpoint* pada fitur *projects management* yang dikembangkan. Fitur ini dapat diakses oleh semua *role* yakni *Project Admin, Department Head,* PIC*,* dan *team member* dengan penyesuaian khusus sesuai dengan tanggung jawabnya. Contohnya, *role Department Head* dapat mengakses fitur proyek dokumen, namun tidak dapat melakukan perubahan ataupun upload dokumen karena itu tanggung jawab dari *Project Admin* dan PIC. Untuk fitur registrasi dan penghapusan proyek hanya dapat dilakukan oleh *Project Admin.* Layanan API ini dirancang modular dan terstruktur melalui arsitektur REST API dengan adanya penerapan autentikasi JWT dan kontrol akses berbasis RBAC memungkinkan setiap layanan dan pengguna terintergrasi dengan baik dalam mengaksees informasi secara selektif sesuai peran yang mendukung efektivitas pengelolaan antar departemen.

Tabel 3.12 *Endpoint* fitur *Pser Managemet*

Metode	URL Endpoint	Deskripsi
PUT	/api/v1/users/details/{userId}	Melakukan pembaharuan pada detail informasi *user* tertentu.
GET	/api/v1/users	Mendapatkan semua daftar data user dari departemen tertentu.
DELETE	/api/v1/users/{userId}	Menghapus *user* tertentu.

Tabel 3.12 menyajikan daftar *endpoint* pada fitur *user management* yang dikembangkan. Fitur ini hanya dapat diakses oleh *Project Admin*. Karena dalam sistem ini peran *Project Admin* sebagai perwakilan dari departemennya sendiri yang akan bertanggung jawab dalam mengelola dan manajemen pengguna dari departemen tersebut. Fitur ini berfungsi untuk mempermudah *Project Admin* dalam mengelola pengguna dari departemen terkait.

Tabel 3.13 *Endpoint* fitur *Application Management*

Metode	URL Endpoint	Deskripsi
GET	/api/v1/applications	Mendapatkan semua daftar data user dari departemen tertentu.
GET	/api/v1/applications/{applicationId}	Mendapatkan detail informasi aplikasi tertentu.
POST	/api/v1/applications/register	Membuat aplikasi baru.
PUT	/api/v1/applications/{applicationId}	Melakukan pembaharuan pada detail informasi aplikasi tertentu.

DELETE	/api/v1/applications/{applicationId}	Menghapus aplikasi tertentu.
GET	/api/v1/applications/{applicationId}/documents	Mendapatkan data dokumen dari aplikasi tertentu.
POST	/api/v1/applications/{applicationId}/documents/register	Membuat dokumen baru dari aplikasi tertentu.
PUT	/api/v1/applications/{applicationId}/documents/{documentId}	Melakukan pembaharuan data dokumen dari aplikasi tertentu.
DELETE	/api/v1/applications/{applicationId}/documents/{documentId}	Menghapus dokumen dari aplikasi tertentu.
GET	/api/v1/applications/{applicationId}/surrounding-application/{surroundingId}	Mendapatkan data aplikasi yang berkaitan pada aplikasi tertentu.
GET	/api/v1/applications/{applicationId}/surrounding-application	Mendapatkan semua daftar data aplikasi yang berkaitan pada aplikasi tertentu.
POST	/api/v1/applications/{applicationId}/surrounding-application/register	Membuat data baru aplikasi yang berkaitan dari aplikasi tertentu.
PUT	/api/v1/applications/{applicationId}/surrounding-application/{surroundingId}	Melakukan pembaharuan data aplikasi yang berkaitan pada aplikasi tertentu.
DELETE	/api/v1/applications/{applicationId}/surrounding-application/{surroundingId}	Menghapus data aplikasi yang berkaitan pada aplikasi tertentu.
GET	/api/v1/applications/{applicationId}/servers/environment/ {environmentId}	Mendapatkan data detail informasi *server* dari *environment* tertentu.
GET	/api/v1/applications/{applicationId}/servers	Mendapatkan semua daftar data informasi *server* dari aplikasi tertentu.
GET	/api/v1/applications/{applicationId}/servers/{serverId}	Mendapatkan data informasi aplikasi *server* dari *server* tertentu.
POST	/api/v1/applications/{applicationId}/servers/register	Membuat data *server* baru pada aplikasi tertentu.
PUT	/api/v1/applications/{applicationId}/servers/{serverId}	Melakukan pembaharuan data informasi *server*

		tertentu pada aplikasi tertentu.
DELETE	/api/v1/applications/{applicationId}/servers/{serverId}	Menghapus data *server* tertentu dari aplikasi tertentu.
GET	/api/v1/applications/{applicationId}/servers/{serverId}/package-installation	Mendapatkan data detail informasi paket instalansi *server* dari apliksasi dan *server* tertentu.
POST	/api/v1/applications/{applicationId}/servers/{serverId}/package-installation/register	Membuat data paket instalasi *server* baru pada aplikasi dan *server* tertentu.
PUT	/api/v1/applications/{applicationId}/servers/{serverId}/package-installation	Melakukan pembaharuan data detail informasi paket instalasi *server* pada aplikasi dan *server* tertentu.
DELETE	/api/v1/applications/{applicationId}/servers/{serverId}/package-installation	Menghapus data paket instalasi *server* pada aplikasi dan *server* tertentu.

Tabel 3.13 menyajikan daftar *endpoint* pada fitur *application management* yang dikembangkan. Fitur ini hanya dapat diakses oleh *Project Admin*. Pada fitur *application management* terdapat beberapa fitur lainnya seperti *surrounding application, environment application,* dan *server installation package.* Dalam fitur ini *Project Admin* dapat manajemen daftar aplikasi dari berbagai proyek. Setiap aplikasi memiliki detail informasi yang terstruktur seperti aplikasi yang berkaitan satu sama lain, *environment server* pada aplikasi, *server installation package* pada aplikasi, dan dokumen aplikasi. Informasi yang terstruktur dan detail mempermudah departemen dalam mengorganisir daftar aplikasi.

3.5 Status Proyek

Pada proyek pengembangan sistem monitoring proyek berbasis web ini , seluruh proses pengembangan telah mencapai tahap selesai sesuai dengan ketentuan yang ditetapkan oleh departemen WHA. Proses pengembangan dimulai dari perencanaan sampai tahap implementasi berlangsung selama lima bulan. Penulis telah mengembangankan beberapa fitur utama seperti *dashboard, project management, application management*, *user management*,

department management, *group management*, dan *server installation management*. Selain itu, sistem ini telah dilengkapi dengan mekanisme autentikasi dan otorisasi berbasis peran (*Role-Based Access Control*) guna memastikan keamanan serta kemudahan dalam mengelola hak akses pengguna terhadap fitur-fitur yang tersedia. Sistem juga telah melalui tahapan pengujian yang dilakukan oleh tim *Quality Assurance* (QA) untuk memastikan bahwa setiap fungsionalitas berjalan sesuai dengan perancangan dan memenuhi standar kualitas yang ditetapkan.

Meskipun sistem secara fungsional telah selesai dikembangkan dan telah melewati tahap pengujian, saat ini sistem masih berada dalam tahap evaluasi lanjutan dan penyesuian untuk kemungkinan diimplementasikan sebagai sistem internal di lingkungan departemen WHA. Ke depannya, tidak menutup kemungkinan sistem ini akan diekspansi dan diintegrasikan ke dalam grup *IT & Digital Development Group* (IDG), apabila hasil uji coba awal menunjukkan performa yang baik serta sistem dinilai mampu memenuhi kebutuhan lintas departemen.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Berdasarkan hasil teori-teori yang telah dijelaskan pada Bab 2 mengenai metodologi *Semi-Agile*, penerapan RBAC, *framework* Spring Boot dan SQL Server sebagai infrastruktur basis data, ditemukan sejumlah kesenjangan antara kajian akademik dengan praktik nyata selama pelaksanaan magang. Lebih diutamakan pada pengembangan menggunakan metode *Semi-Agile*. Kesenjangan ini menunjukkan bahwa dalam dunia industri, penerapan teori tidak selalu berjalan secara ideal dan sering kali mengalami penyesuaian terhadap kondisi organisasi, ketersediaan sumber daya, serta kebutuhan bisnis yang dinamis.

Pertama, dalam aspek pendekatan *Semi-Agile* dijelaskan sebagai penggabungan antara prinsip *Watefall* yang menekankan dokumentasi dan perencanaan awal, serta *Agile* yang menekankan iterasi dan fleksibilita. Namun, dalam implementasinya proyek ini tidak secara eksplisit diterapkan secara adaptif. Meskipun tidak secara ketat mengikuti *Scrum framework*, pengembangan sistem tetap menggunakan *backlog* per fitur, sesi diskusi berkala, dan tahapan iteratif sebagaimana prinsip *Agile*. Kesenjangan yang ditemukkan adalah tidak digunakannya *tools* manajemen proyek secara umum seperti Jira atau Trello secara tim, melainkan hanya Excel sheet untuk mencatat progres dan pembagian tugas, sehingga membuat praktik iterasi lebih bersifat dokumentasi daripada sistematis. Kesenjangan ini menunjukkan bahwa prinsip *Agile* dapat diterapkan dengan pendekatan sederhana tanpa perangkat pendukung formal, namun berpotensi menurunkan efektivitas *tracking* dan distribusi kerja, terutama dalam tim berskala lebih besar. Hal ini menjadi pembelajaran bahwa implementasi metode pengembangan dalam organisasi sering kali disesuaikan dengan tingkat adaptasi tim dan budaya kerja yang berlaku.

Kedua, dari sisi desain arsitektur dan dokumentasi sistem dalam metode *Semi-Agile*, teori menyebutkan bahwa tahapan awal pengembangan sistem informasi seharusnya mencakup penyusunan dokumen teknis seperti *use case diagram* (UCD), *activity diagram*, *entity relationship diagram* (ERD), dan *prototype* antarmuka pengguna. Namun, pada praktiknya penyusunan sebagian besar dokumentasi tersebut tidak dilakukan secara formal dan sistematis oleh tim pengembang. Penyusunan ERD, misalnya memang dilakukan sejak awal oleh mentor yang juga merupakan perwakilan dari Departemen WHA sebagai *stakeholder*. Namun, skema

tersebut bersifat dinamis dan mengalami perubahan seiring dengan penyesuaian kebutuhan selama proses iterasi. Hal ini membuat pengembangan proyek menjadi tertunda beberapa waktu karena adanya perubahan skema pada basis data. Sementara itu, diagram seperti UCD dan *activity diagram* tidak disusun secara eksplisit dalam bentuk visual, tetapi alur interaksi antar fitur pengguna tetap dianalisis oleh tim pengembang secara kolaboratif dan dituangkan dalam struktur logika sistem. Dan untuk desain UI diberikan dalam bentuk *prototype* kasar yang kemudian dikembangkan lebih lanjut oleh tim UI/UX melalui improvisasi. Kurangnya dokumen dan penjelasan untuk identifikasi terkait kebutuhan sistem menjadikan penulis dan tim menjadikan pengembangan proyek tidak berjalan sesuai dengan perencanaan yang ditetapkan.

Ketiga, dalam aspek pengelolaan hak akses pengguna, sistem telah menerapkan konsep *Role-Based Access Control* (RBAC) sebagaimana dijelaskan pada Bab 2. Empat peran utama telah didefinisikan secara jelas, yakni *Project Admin, Department Head,* PIC, dan *Team Member*, dengan masing-masing hak akses yang spesifik terhadap fitur sistem. Akan tetapi, konfigurasi otorisasi tersebut masih diatur secara statis melalui konfigurasi kode, belum melalui *dashboard* admin yang dinamis. Secara ideal, RBAC pada sistem skala perusahaan seharusnya dapat terintegrasi dengan layanan direktori seperti *Lightweight Directory Access Protocol* (LDAP) untuk pengelolaan akses yang lebih terpusat dan aman. Dalam rencana pengembangan lebih lanjut, sistem ini akan dihubungkan dengan sistem LDAP perusahaan, di mana unit keamanan internal akan bertindak sebagai pengelola manajemen pengguna pada sistem. Namun, selama program magang berlangsung, integrasi ini belum dapat direalisasikan karena prosedur otorisasi internal yang ketat serta kebutuhan koordinasi dengan unit pusat.

Keempat, penerapan *framework* Spring Boot pada pengembangan sistem menjadi pengalaman baru bagi penulis dalam menggunakan dan mempelajari teknologi yang sebelumnya belum diajarkan secara mendalam di lingkungan akademik. Pada akademik, pembelajaran bahasa pemrograman Java umumnya terbatas pada konsep dasar tanpa penggunaan *framework* pendukungnya. Melalui proyek ini, penulis memperoleh pemahaman langsung mengenai cara kerja Spring Boot, termasuk konfigurasi otomatis, penerapan arsitektur MVC, integrasi dengan Spring Data JPA, penggunaan Spring Security sebagai struktur keamanan sistem dan pengembangan layanan RESTful yang terstruktur. Pengalaman ini menjadi penghubung penting antara teori dasar pemrograman Java yang dipelajari pada akademik dengan praktik pengembangan aplikasi berskala industri.

Kelima, penggunaan Microsoft SQL Server sebagai basis data dalam proyek ini juga memberikan relevansi akademik yang siginifikan. Selama di akademik, penulis lebih sering menggunakan MySQL sebagai *database engine*, sehingga implementasi SQL Server menjadi kesempatan untuk memperlajari perbedaan, fitur, dan keunggulan yang ditawarkan, seperti manajemen transakasi yang memenuhi prinsip ACID, penggunaan *indexing* untuk optimasi *query*, penerapan *stored procedur*, serta adanya pengaturan keamanan berbasis peran. Integrasi SQL Server dengan Spring Boot melalui JDBC (*Java Database Connectivity*) dan JPA (*Java Persistence API*) juga memperkuat keterampilan penulis dalam menghubungkan konsep teori basis data dengan penerapan langsung pada platform yang umum digunakan di industri perbankan.

4.2 Pembelajaran Magang

Kegiatan magang merupakan salah satu tahapan penting yang dalam perjalanan akademik mahasiswa, meskipun sering kali kurang mendapatkan perhatian. Tantangan seperti kurangnya rasa percaya diri atau kekhawatiran terhadap kemampuan yang dimiliki kerap menjadi alasan minimnya minat terhadap program ini. Padahal magang memberikan kesempatan bagi mahasiswa untuk mengembangkan keterampilan praktis dan menerapkan pengetahuan yang telah diperoleh selama perkuliahan ke dalam konteks dunia industri. Selain memberikan pengalaman kerja nyata, mahasiswa juga dapat berkontribusi secara langsung terhadap kegiatan perusahaan tempat magang. Pengalaman ini menjadi bekal penting dalam mempersiapkan diri menghadapi dunia kerja.

Selama menjalani program magang, penulis memperoleh berbagai pengalaman berharga, baik dalam aspek teknis maupun non-teknis. Penulis mendapat kesempatan untuk memahami secara mendalam operasional sebuah lembaga perbankan, mempelajari teknologi terkini dalam industri keuangan, serta mengembangkan kemampuan interpersonal melalui interaksi dengan rekan kerja. Selain itu, program magang ini turut berkontribusi meningkatkan kemampuan manajemen diri, kerja sama tim, dan komunikasi penulis. Salah satu manfaat dari program ini adalah pelaksanaan OJT (*On the Job Training*) yang memberikan pemahaman secara langsung mengenai operasional dan cara kerja perbankan. Melalui kegiatan ini, penulis melakukan observasi dan identifikasi terhadap kendala IT yang terjadi di lingkungan kerja. Kegiatan ini tidak hanya meningkatkan pemahaman terhadap sistem yang digunakan di perbankan, tetapi juga melatih cara kemampuan analisis dan komunikasi dalam menyampaikan hasil observasi.

Selain itu, kegiatan *Specialis IT* memberikan manfaat signifikan dalam memahami teknologi yang digunakan dalam industri secara langsung lebih tepatnya industri perbankan. Materi yang disampaikan oleh para ahli di bidangnya memungkinkan penulis untuk memperoleh wawasan yang lebih mendalam serta kesempatan untuk berdiskusi terkait teknologi yang diterapkan di industri. Kegiatan ini juga memberikan gambaran mengenai relevansi materi perkuliahaan dengan kebutuhan di dunia kerja, sehingga dapat menjadi acuan dalam mengembangkan keterampilan yang lebih sesuai dengan industri.

Lebih lanjut, penulis turut terlibat dalam kegiatan *On the Job Assignment Project* sebagai *back-end developer* yang di mana penulis diberikan kesempatan untuk mengembangkan sebuah proyek secara tim berdasarkan kebutuhan departemen terkait. Pengalaman ini melatih keterampilan kerja sama dalam tim, komunikasi yang efektif, serta kemampuan *problem-solving* dalam menyelesaikan tantangan proyek. Penggunaan teknologi yang baru bagi penulis pada pengembangan sistem monitoring proyek berbasis web ini, memberikan manfaat dan pengalaman dalam mengembangkan keterampilan teknis dalam penerapan *framework* Spring Boot yang menggunakan konsep REST API sebagai sarana komunikasi sistem dan implementasi mekanisme RBAC sebagai metode keamanan pada sistem. Penggunaan teknologi Spring Boot membuat penulis mengetahui dan mempelajari keunggulan dari fitur-fitur yang diberikan pada teknologi tersebut seperti *auto-configuration, embedded servers,* dan *starter dependencies,* serta mengetahui teknologi lainnya seperti Spring Security dan Spring Data JPA dalam mengintegrasikan *database* yang digunakan pada hal ini adalah Microsoft SQL Server.

Peran mentor dalam proyek ini memberikan kontribusi besar dalam proses pembelajaran, khususnya dalam memahami konsep-konsep teknis yang belum dikuasai. Bimbingan yang diberikan juga memperkuat kesiapan penulis dalam menghadapi tantangan nyata di lingkungan kerja. Secara keseluruhan, keterlibatan dalam pengembangan proyek ini memberikan pengalaman menyeluruh dalam pengelolaan sistem informasi, mulai dari proses analisis kebutuhan hingga implementasi teknis yang sesuai dengan kebutuhan departemen sebagai *client*. Dengan demikian, pengalaman magang ini menjadi landasan kuat bagi penulis untuk mengintegrasikan teori dan praktik secara nyata, serta memperkuat kesiapan dalam menghadapi tantangan profesional di bidang teknologi informasi.

BAB V
PENUTUP

5.1 Kesimpulan

Pengembangan sistem monitoring proyek berbasis web menggunakan *framework* Spring Boot dan mekanisme *Role-Based Access Control* (RBAC) telah berhasil diimplementasikan. Hasil pengembangan telah sesuai dengan kebutuhan dan tujuan utama, yaitu mengembangkan solusi digital terintegrasi untuk meningkatkan efisiensi, transparansi dan koordinasi antar tim dalam lingkup departemen WHA (*IT Wholesale & Office Automation Development*) dan rencana ke depannya dapat diterapkan juga pada beberapa departemen di bawah naungan grup IDG (*IT Development Group*) di Bank Syariah Indonesia. Sistem yang dikembangkan menggunakan pendekatan metode *Semi-Agile* yang memadukkan struktur dokumentasi *Waterfall* dan fleksibilitas *Agile*, serta mengintegrasikan teknologi *back-end* Spring Boot, Vite React sebagai *interface build* dan Microsoft SQL Server sebagai basis data. Seluruh fitur utama seperti autentikasi, *management, project, user, application, department, group*, serta *dashboard* telah dikembangkan dan diuji dengan mengacu pada *backlog* dan *acceptance criteria* yang disusun berdasarkan kebutuhan pengguna.

Penerapan RBAC dalam sistem terbukti efektif dalam mengatur hak akses pengguna sesuai peran, seperti *Project Admin, Department Head,* PIC, dan *Team Member*, sehingga mendukung keamanan dan struktur tata kelola akses informasi. Selain itu, sistem ini juga mampu menyediakan data proyek secara *real-time* yang mendukung proses pengambilan keputusan, mempermudah kolaborasi lintas tim, serta menyajikan informasi historis yang dapat dimanfaatkan sebagai dasar evaluasi kinerja. Berdasarkan hasil pengembangan dan pembahasan sebelumnya, dapat disimpulkan bahwa sistem ini telah mencapai seluruh target pengembangan serta memberikan manfaat nyata dalam mendukung proses kerja di lingkungan organisasi. Dengan dikembangkannya sistem ini, diharapkan dapat menjadi kontribusi nyata dalam mendukung transformasi digital di lingkungan kerja, sekaligus menjadi referensi bagi pengembangan sistem serupa di masa mendatang.

5.2 Saran

Terdapat saran bagi pengembangan sistem monitoring proyek berbasis web untuk pengembangan selanjutnya. Adanya integrasi fitur notifikasi otomatis terhadap setiap

penugasan proyek yang dilakukan oleh *Project Admin*, *Department Head*, dan PIC. Fitur ini bertujuan agar membantu pengguna dalam menerima informasi terkait penugasan proyek secara langsung dan tepat waktu, baik dari departemen sendiri maupun departemen lain. Selain itu, untuk meningkatkan kualitas dan keberhasilan proyek ke depannya dalam program magang, disarankan penggunaan platform manajemen proyek yang umum digunakan dalam pengembangan proyek yang kompleks dan memerlukan kolaborasi antar tim, seperti Jira ataupun Trello. Penggunaan tersebut dapat membantu pengelolaan dan manajemen pada pengembangan proyek menjadi lebih terstruktur dan efektif.

DAFTAR PUSTAKA

Aggarwal, S., & Pandit, S. (2023). Spring Boot Application using Three Layered Architecture in Java. *Jaypee University of Information Technology (JUIT)*. Retrieved from http://ir.juit.ac.in:8080/jspui/jspui/handle/123456789/10202

Dawis, A. M., Putra, Y. W. S., Fitria, F., Hamidin, D., Yutia, S. N., Maniah, M., … Natsir, F. (2023). *Rekayasa perangkat lunak panduan praktis untuk pengembangan aplikasi berkualitas*. Penerbit Widina. Retrieved from https://books.google.co.id/books?hl=id&lr=&id=ttnVEAAAQBAJ&oi=fnd&pg=PA1&dq=Pengembangan+Berulang+pada+Pengembangan+perangkat+lunak+&ots=UWkXw2bo9E&sig=S5XZXmfLhOmyXQJYhs--YhrzxPQ&redir_esc=y#v=onepage&q=Pengembangan%20Berulang%20pada%20Pengembangan%20perangkat%20lunak&f=false

Hyytiäinen, J. (2024). Access management solution to corporate service. *Aaltodoc - Aalto University*, 81–89. Retrieved from https://urn.fi/URN:NBN:fi:aalto-202405263820

Java Persistence with Spring Boot 3, Spring Data JPA with Hibernate, and the Oracle Database 23c Free — Developer Release | by Juarez Junior | Medium. (2023). Retrieved July 14, 2025, from https://juarezjunior.medium.com/java-persistence-with-spring-boot-3-spring-data-jpa-with-hibernate-and-the-oracle-database-23c-9c0946d4838f

Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A. M., & Luo, B. (2023). SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review. *Big Data and Cognitive Computing*, *7*(2), 97. Retrieved from https://doi.org/10.3390/bdcc7020097

Lai, Y. (2025). Implementation of online network file management system based on Spring Boot and Vue. *Theseus*, 46. Retrieved from https://urn.fi/URN:NBN:fi:amk-2025052817468

Luan, X. (2021). Implementation and analysis of software development in spring boot. *Doctoral Dissertation*. Retrieved from http://hdl.handle.net/20.500.12680/2j62sb23d

MVC Framework Introduction - GeeksforGeeks. (2024). Retrieved July 14, 2025, from https://www.geeksforgeeks.org/software-engineering/mvc-framework-introduction/

Nevarez, B. (2022). *SQL Server Query Tuning and Optimization: Optimize Microsoft SQL Server 2022 Queries and Applications*. Packt Publishing. Retrieved from http://ieeexplore.ieee.org/document/10163371

Nguyen, U. (2024). Streamlining project management: Enhancing efficiency and progress tracking through a CRM website. *Theseus*. Retrieved from https://urn.fi/URN:NBN:fi:amk-2024052917438

Penggalih, V. G., & Silmina, E. P. (2025). IMPLEMENTASI DASHBOARD PROYEK DENGAN SISTEM RBAC PADA PT. XYZ. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *9*(2), 2135–2142. Retrieved from https://doi.org/10.36040/jati.v9i2.12978

Prayogi, A. A., Niswar, M., Indrabayu, & Rijal, M. (2020). Design and implementation of REST API for academic information system. *IOP Conference Series: Materials Science and Engineering*, *875*(1), 12047. https://doi.org/10.1088/1757-899X/875/1/012047

Rachmadi, T. S. Kom. (2020). *Sistem Basis Data* (Vol. 1). Tiga Ebook. Retrieved from https://books.google.co.id/books?id=b7_dDwAAQBAJ&printsec=frontcover#v=onepage&q&f=false

Romindo, R., Yusnanto, T., Heryana, N., Jamaludin, A. P. A., Permana, A. A., Aisa, S., … Sihombing, F. A. H. (2023). Rekayasa Perangkat Lunak. *Padang: PT Global Eksekutif Teknologi*. Retrieved from https://www.researchgate.net/publication/370471310_BAB_4_REKAYASA_PERSYARATAN_PERANGKAT_LUNAK

Saleh, K. R., & Paputungan, I. V. (2024). Implementasi Metode Agile Serta Proses Bisnis Dalam Pengembangan Dan Perancangan Aplikasi Bergerak Mecha Sebagai Penyedia Layanan Perbaikan Kendaraan. *EDUSAINTEK: Jurnal Pendidikan, Sains Dan Teknologi*, *11*(1), 87–103. Retrieved from https://doi.org/10.47668/edusaintek.v11i1.959

SQL Server | Microsoft Learn. (2025). Retrieved July 11, 2025, from https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver17

Suzanti, I. O., Fitriani, N., Jauhari, A., & Khozaimi, A. (2020). REST API implementation on android based monitoring application. *Journal of Physics: Conference Series*, *1569*(2), 22088. https://doi.org/10.1088/1742-6596/1569/2/022088

Torredimare, A. (2024). *Extension of an enterprise web application for top-management reporting: a modular approach to Web Application development* (Politecnico di Torino). Politecnico di Torino. Retrieved from http://webthesis.biblio.polito.it/id/eprint/33954

Trista, R. T., & Wisdariah, W. (2021). PENGEMBANGAN SISTEM MONITORING MY SCHOOL DENGAN MENGGUNAKAN NETBEANS. *Jurnal Manajamen Informatika*

Jayakarta, *1*(4), 427–435. Retrieved from https://doi.org/10.52362/jmijayakarta.v1i4.1578

Walls, C. (2022). *Spring in Action, Sixth Edition*. Simon and Schuster. Retrieved from https://books.google.co.id/books?hl=id&lr=&id=2zVbEAAAQBAJ&oi=fnd&pg=PA1&dq=Spring+in+Action,+Sixth+Edition&ots=PhUCvx1BMf&sig=IKTfFr3_Fy1eFDvvKc6KmheLpf0&redir_esc=y#v=onepage&q=Spring%20in%20Action%2C%20Sixth%20Edition&f=false

Wibowo, N. A., & Idris, M. (2025). Penerapan waterfall dalam pengembangan web inventori EDC dengan Spring Boot dan Thymeleaf. *EDUSAINTEK: Jurnal Pendidikan, Sains Dan Teknologi*, *12*(1), 173–188. Retrieved from https://doi.org/10.47668/edusaintek.v12i1.1349

LAMPIRAN