PERANCANGAN DAN IMPLEMENTASI SISTEM INFORMASI PELAPORAN ABNORMALITAS MESIN BERBASIS WEB UNTUK EFEKTIVITAS MANAJEMEN PERAWATAN DI PT. XYZ

PROYEK AKHIR

Disusun Oleh:

Nayla Nabillah Arishima

3312301025

Disusun untuk memenuhi salah satu syarat memperoleh gelar Ahli Madya Teknik Informatika

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

POLITEKNIK NEGERI BATAM

2026

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan proposal Tugas Akhir yang berjudul "Perancangan dan Implementasi Sistem Informasi Pelaporan Abnormalitas Mesin Berbasis *Web* untuk Efektivitas Manajemen Perawatan".

Penyusunan proposal ini merupakan bagian dari proses akademik yang harus ditempuh oleh penulis dalam menyelesaikan pendidikan pada Program Studi Teknik Informatika, Jurusan Teknik Informatika, Politeknik Negeri Batam. Proposal ini sekaligus menjadi landasan awal dalam perancangan dan pengembangan sistem informasi berbasis *web Machine Abnormality*, yang diharapkan mampu mendukung proses pelaporan kondisi mesin secara terstruktur serta membantu peningkatan efektivitas manajemen perawatan.

Selama proses penyusunan proposal ini, penulis memperoleh banyak dukungan, arahan, dan motivasi dari berbagai pihak. Oleh karena itu, dengan penuh rasa hormat dan terima kasih, penulis menyampaikan apresiasi kepada:

1. Ibu Miratul Khusna Mufida, selaku Dosen Pembimbing, atas bimbingan, arahan, serta masukan yang diberikan secara berkelanjutan sehingga proposal ini dapat disusun dengan lebih terarah dan sistematis.
2. Seluruh Dosen dan Tenaga Kependidikan Program Studi Teknik Informatika Politeknik Negeri Batam, yang telah berperan dalam membekali penulis dengan ilmu pengetahuan dan pengalaman selama masa perkuliahan.
3. Keluarga tercinta, khususnya Mama, Papa, dan adik-adik, yang senantiasa memberikan doa, dukungan, dan semangat sehingga penulis mampu menyelesaikan penyusunan proposal ini.
4. Rekan satu tim dan rekan mahasiswa, yang telah menjadi tempat berdiskusi, bertukar pikiran, serta memberikan dukungan moral selama proses pengerjaan Tugas Akhir.
5. BTS (Bangtan Sonyeondan), yang melalui karya musik dan pesan-pesan

positifnya turut memberikan dorongan semangat serta ketenangan bagi penulis dalam menjaga konsistensi dan motivasi selama menyusun laporan Tugas Akhir ini.

Penulis menyadari bahwa proposal ini masih jauh dari sempurna, baik dari segi isi maupun penyajian. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang bersifat membangun demi penyempurnaan proposal dan proyek ini di masa mendatang.

Akhir kata, penulis berharap semoga proposal ini dapat memberikan manfaat, baik bagi pihak industri dalam meningkatkan efektivitas manajemen perawatan mesin, maupun bagi pembaca sebagai referensi dalam pengembangan sistem informasi berbasis web di bidang industri.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

ABSTRAK

Perawatan mesin yang efektif merupakan faktor penting dalam menjaga kelancaran proses produksi di lingkungan industri. Namun, sistem pelaporan abnormalitas mesin yang masih dilakukan secara manual sering kali menyebabkan keterlambatan dalam penanganan masalah dan pengambilan keputusan. Oleh karena itu, proyek ini bertujuan untuk merancang dan mengimplementasikan Sistem Informasi Pelaporan Abnormalitas Mesin berbasis *web* yang diberi nama *Machine Abnormality* di PT. XYZ. Sistem ini dirancang untuk memudahkan pihak *maintenance* dalam melaporkan kondisi mesin yang tidak normal secara cepat, akurat, dan terintegrasi. Metode pengembangan yang digunakan adalah model *Waterfall,* dengan fokus pada pembuatan antarmuka yang responsif serta sistem basis data yang dapat menampung laporan abnormalitas mesin. Hasil dari proyek ini berupa sebuah *website* yang memungkinkan pengguna untuk melakukan *input* data abnormalitas mesin, melihat riwayat laporan dan melihat *Dashboard* analisis. Dengan adanya sistem ini, diharapkan proses pelaporan dan pengelolaan data abnormalitas mesin di PT. XYZ dapat menjadi lebih efisien, transparan, dan terdigitalisasi, sehingga meningkatkan efektivitas manajemen perawatan mesin secara keseluruhan.

Kata Kunci: Sistem Informasi, Abnormalitas Mesin, Manajemen Perawatan, *Web-Based Application*.

BAB I PENDAHULUAN

1.1 Latar Belakang

Dalam industri manufaktur modern, keandalan mesin produksi merupakan faktor krusial yang mempengaruhi kelancaran proses operasional perusahaan. Setiap gangguan atau abnormalitas pada mesin dapat menyebabkan *downtime*, penurunan produktivitas, terganggunya jadwal produksi, serta menurunnya nilai *Overall Equipment Effectiveness* (OEE). Oleh karena itu, diperlukan suatu sistem yang mampu mendukung proses pelaporan dan penanganan abnormalitas secara cepat, akurat, dan terstruktur.

Berdasarkan hasil observasi yang dilakukan di PT. XYZ, proses pelaporan abnormalitas mesin saat ini masih dilakukan secara manual, seperti menggunakan kertas, pesan singkat, maupun *spreadsheet*. Kondisi ini menimbulkan berbagai permasalahan, di antaranya keterlambatan penyampaian informasi, risiko kesalahan pencatatan data, tidak adanya dokumentasi riwayat yang terstruktur, serta kesulitan dalam melakukan *monitoring* terhadap progres penyelesaian masalah. Selain itu, tidak adanya alur kerja yang baku dalam distribusi penanganan menyebabkan beberapa laporan mengalami keterlambatan penanganan, meskipun tingkat permasalahan relatif ringan.

Selain itu, sistem pelaporan abnormalitas yang berjalan saat ini di PT. XYZ belum memiliki standar digitalisasi yang terintegrasi. Proses pelaporan masih bergantung pada komunikasi manual seperti *chat* atau pencatatan sederhana, sehingga tidak hanya memperlambat penanganan tetapi juga berpotensi menimbulkan biaya tidak langsung seperti *downtime* mesin, kehilangan produktivitas, serta risiko kesalahan komunikasi antar tim. Dari sisi implementasi sistem, perusahaan juga belum memiliki platform khusus yang dapat mengakomodasi *workflow* pelaporan abnormalitas secara terstruktur dan terdokumentasi. Oleh karena itu, diperlukan sistem berbasis web yang tidak hanya

mampu mencatat laporan, tetapi juga mengelola alur kerja, distribusi tugas, serta validasi secara terintegrasi.

Permasalahan tersebut menunjukkan adanya kebutuhan mendesak akan sistem pelaporan abnormalitas mesin yang terintegrasi dan berbasis digital. Tanpa adanya sistem yang terstruktur, perusahaan berpotensi mengalami kerugian yang lebih besar akibat *downtime* yang tidak tertangani secara optimal serta pengambilan keputusan yang kurang didukung oleh data yang akurat dan terdokumentasi.

Untuk mengatasi permasalahan tersebut, penulis merancang dan membangun Sistem Informasi Pelaporan Abnormalitas Mesin Berbasis Web menggunakan teknologi ASP.NET (C#) dan database SQL Server. Sistem ini dirancang agar dapat diimplementasikan secara langsung di lingkungan PT. XYZ dan diakses secara *real-time* oleh pengguna sesuai dengan perannya.

1. Alur Utama (melalui *Action Owner*)
   Requestor → *Action Owner* → *Assigned Action* → *Validator*
   (Jika *Validator* menolak, laporan kembali ke *Action Owner*)
2. Alur Fixed by Myself *Requestor* → *Validator*
   (Jika *Validator* menolak, laporan masuk ke *Action Owner*)

Selain alur kerja tersebut, sistem ini juga mendukung penerapan kategori abnormalitas menggunakan kode warna (Merah, Kuning, Hijau, dan Putih) yang membantu perusahaan mengklasifikasikan tingkat urgensi serta jenis penanganan yang diperlukan. Dengan adanya dua *workflow* yang berbeda, perusahaan dapat menyesuaikan proses penyelesaian berdasarkan tingkat kompleksitas masalah, sehingga penanganan abnormalitas ringan dapat berlangsung lebih cepat tanpa mengganggu alur penanganan masalah yang lebih berat.

Dengan adanya sistem ini, diharapkan proses pelaporan, pemantauan, dan validasi abnormalitas mesin menjadi lebih cepat, terstruktur, dan terdokumentasi dengan baik. Selain itu, sistem ini juga diharapkan mampu meningkatkan efektivitas pengambilan keputusan dalam manajemen perawatan mesin serta mengurangi risiko *downtime* yang berdampak pada produktivitas perusahaan.

1.2 Rumusan Masalah

Berdasarkan latar belakang dan ruang lingkup fitur yang akan dibangun, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana sistem dapat mendukung pencatatan riwayat abnormalitas mesin serta monitoring progres tindak lanjut perawatan secara *real-time*?
2. Bagaimana penerapan mekanisme kategorisasi abnormalitas berbasis kode warna (Merah, Kuning, Hijau, Putih) dalam membantu penentuan prioritas dan distribusi penanganan masalah?
3. Bagaimana sistem dapat secara otomatis menentukan *Action Owner* dan *Assigned SESA* berdasarkan hasil AM Checklist (*Yes/No*), *facility*, dan *tag ID*, serta menentukan *Validator* berdasarkan *facility* pada setiap laporan abnormalitas mesin?
4. Bagaimana menguji fungsionalitas sistem informasi pelaporan abnormalitas mesin untuk memastikan sistem berjalan sesuai kebutuhan pengguna dan mendukung efektivitas manajemen perawatan mesin?

1.3 Tujuan

Sesuai dengan rumusan masalah diatas, maka tujuan penelitian ini adalah:

1. Mengembangkan fitur pencatatan riwayat abnormalitas mesin serta monitoring progres tindak lanjut perawatan guna meningkatkan transparansi dan kemudahan pengawasan.
2. Menerapkan mekanisme kategorisasi abnormalitas berbasis kode warna (Merah, Kuning, Hijau, Putih) sebagai indikator prioritas dalam proses penanganan masalah.
3. Merancang dan mengimplementasikan mekanisme penetapan otomatis *Action Owner* dan *Assigned SESA* berdasarkan hasil *AM Checklist* (*Yes/No*), facility, dan *tag ID*, serta *Validator* berdasarkan *facility* untuk meningkatkan efisiensi distribusi tanggung jawab.
4. Melakukan pengujian sistem menggunakan metode *Blackbox* Testing untuk memastikan fungsionalitas sistem berjalan dengan baik dan sesuai dengan

kebutuhan pengguna.

1.4 Batasan Masalah

Untuk menjaga fokus penelitian dan menyesuaikan dengan fitur yang akan dikembangkan, batasan masalah ditetapkan sebagai berikut:

1. Lingkup hanya aplikasi web tanpa integrasi IoT, AI, atau aplikasi *mobile*.
2. Pengujian dilakukan pada satu area produksi.
3. Sistem hanya menggunakan dua alur kerja yaitu Utama (*Action Owner*) dan *Fixed by Myself*.
4. Pengembangan menggunakan bahasa pemrograman C# dan MySQL server untuk *database*.

1.5 Manfaat

Proyek ini diharapkan dapat memberikan manfaat secara teoritis dan praktis sebagai berikut:

1. Manfaat Teoritis

- Memberikan kontribusi dalam pengembangan ilmu pengetahuan di bidang sistem informasi, khususnya dalam implementasi sistem pelaporan abnormalitas mesin berbasis web.
- Menjadi referensi bagi penelitian atau pengembangan sistem serupa, terutama dalam penerapan *dual workflow* (*Action Owner* dan *Fixed by Myself*) pada sistem maintenance industri.

2. Manfaat Praktis

- Memberikan solusi terstruktur dan terdigitalisasi dalam proses pelaporan serta penanganan abnormalitas mesin di lingkungan industri.
- Mempercepat penyelesaian abnormalitas ringan melalui alur *Fixed by Myself*, sehingga dapat mengurangi downtime mesin.
- Meminimalkan kesalahan pencatatan serta meningkatkan keterlacakan proses melalui sistem yang terintegrasi.

- Meningkatkan efektivitas *monitoring* serta mendukung pengambilan keputusan dalam manajemen perawatan mesin berbasis data.

3. Kontribusi terhadap Sustainable Development Goals (SDGs)

Proyek ini berkontribusi terhadap pencapaian tujuan *Sustainable Development Goals* (SDGs), yaitu:

- SDG 9: *Industry, Innovation, and Infrastructure*
  Sistem yang dikembangkan mendukung inovasi teknologi di sektor industri melalui digitalisasi proses pelaporan dan manajemen perawatan mesin yang lebih efisien dan terintegrasi.
- SDG 12: *Responsible Consumption and Production*
  Dengan meningkatkan efisiensi proses perawatan dan mengurangi downtime mesin, sistem ini membantu optimalisasi penggunaan sumber daya serta mengurangi pemborosan operasional.
- SDG 8: *Decent Work and Economic Growth*
  Sistem membantu menciptakan lingkungan kerja yang lebih produktif, efisien, dan terstruktur, sehingga mendukung peningkatan kinerja operasional perusahaan.

1.6 Kolaborasi dan Sinergi dalam Pelaksanaan Proyek

Proyek akhir ini dilaksanakan secara individu di lingkungan PT. XYZ, namun tetap melibatkan kolaborasi dan sinergi dengan berbagai pihak guna memastikan kualitas, kesesuaian kebutuhan, serta keberhasilan implementasi sistem yang dikembangkan. Bentuk kolaborasi yang dilakukan antara lain:

- Diskusi dengan pembimbing lapangan
  Dilakukan secara berkala untuk memperoleh arahan terkait kebutuhan sistem, pemahaman proses bisnis, serta validasi terhadap solusi yang dirancang.
- Koordinasi dengan tim teknis dan pengguna di perusahaan

Bertujuan untuk memahami kondisi operasional di lapangan, mengidentifikasi kebutuhan pengguna, serta memastikan sistem yang dibangun sesuai dengan kebutuhan nyata.

- Konsultasi dengan dosen pembimbing

  Dilakukan secara rutin untuk memastikan kesesuaian metode pengembangan, kualitas akademik, serta struktur penulisan laporan sesuai standar Proyek Akhir.

- Pengumpulan umpan balik pengguna (*user feedback*)

  Dilakukan melalui uji coba sistem secara langsung oleh calon pengguna, guna mengevaluasi kemudahan penggunaan (*usability*) serta kesesuaian fitur dengan kebutuhan operasional.

Melalui kolaborasi tersebut, proyek ini tidak hanya bersifat individual, tetapi juga melibatkan sinergi berbagai pihak yang berkontribusi dalam menghasilkan solusi yang relevan, aplikatif, dan sesuai dengan kebutuhan industri.

BAB II TINJAUAN PUSTAKA

2.1 Tinjauan Solusi dan Sistem Terkait

Pada bagian ini dibahas beberapa aplikasi atau sistem yang memiliki keterkaitan dengan pengembangan Sistem Informasi Pelaporan Abnormalitas Mesin berbasis web. Tinjauan ini bertujuan untuk memberikan gambaran mengenai fitur umum, teknologi yang digunakan, serta kelebihan dan keterbatasan dari sistem yang sudah ada sebagai bahan pertimbangan dalam pengembangan proyek.

Beberapa sistem yang relevan umumnya termasuk dalam kategori Computerized Maintenance Management System (CMMS), yaitu sistem yang digunakan untuk membantu proses pengelolaan pemeliharaan mesin di industri manufaktur (Campos et al., 2020).

1. CMMS Berbasis Web (Mohd Rofi & Kasim, 2022)

   Sistem ini merupakan aplikasi CMMS berbasis web yang digunakan untuk mengelola data maintenance seperti pencatatan aset, penjadwalan perawatan, serta *work order*. Teknologi yang digunakan adalah sistem berbasis web dengan arsitektur terpusat. Kelebihan dari sistem ini adalah pengelolaan data yang terstruktur dan terintegrasi. Namun, sistem ini masih memiliki keterbatasan karena hanya menggunakan satu alur kerja (*single workflow*) dan belum mendukung pelaporan abnormalitas dengan skenario alternatif seperti *Fixed by Myself*.

2. Sistem Evaluasi CMMS dengan K-Means (Sukmana & Rozi, 2024)

   Sistem ini berfokus pada evaluasi efisiensi tahapan pengembangan CMMS menggunakan algoritma K-Means. Pendekatan ini memberikan insight dalam mengoptimalkan proses pengembangan sistem. Kelebihannya adalah mampu mengidentifikasi tahapan yang kurang efisien dalam proyek. Namun, sistem ini tidak berfokus pada implementasi fitur operasional seperti pelaporan abnormalitas mesin maupun *workflow* pengguna.

3. Sistem Evaluasi Kualitas CMMS (Hidayat et al., 2024)

   Sistem ini menggunakan *framework* COBIT 5 untuk mengevaluasi kualitas CMMS dari segi kematangan sistem, kontrol, dan kualitas data. Kelebihannya adalah memberikan evaluasi yang komprehensif terhadap kualitas sistem. Namun, sistem ini tidak membahas implementasi fitur pelaporan abnormalitas, seperti penggunaan kategori warna maupun mekanisme validasi oleh *validator*.

4. CMMS pada Industri Manufaktur Kendaraan (Shankar et al., 2024)

   Sistem ini diterapkan pada industri manufaktur kendaraan roda dua untuk mengelola maintenance dan aset produksi. Kelebihannya adalah terbukti meningkatkan efektivitas manajemen maintenance di lingkungan industri. Namun, sistem ini lebih berfokus pada pencatatan dan penjadwalan maintenance, serta belum menerapkan *workflow* pelaporan abnormalitas secara spesifik.

Berdasarkan tinjauan tersebut, dapat disimpulkan bahwa sebagian besar sistem yang ada masih berfokus pada pengelolaan maintenance secara umum dan belum mengakomodasi kebutuhan pelaporan abnormalitas mesin dengan alur kerja yang fleksibel dan *multi-role*. Oleh karena itu, pada proyek ini dikembangkan sistem pelaporan abnormalitas mesin berbasis web yang memiliki dua alur kerja, yaitu *Normal Workflow* dan *Fixed by Myself Workflow*, serta mendukung *multi-role user* dengan proses validasi yang terkontrol.

Tabel 2.1 Tinjauan Aplikasi Sejenis

Pengembang / Tahun	**Nama Aplikasi**	**Fitur Utama**	**Teknologi**	**Kelebihan**	**Keterbatasan**
Mohd Rofi & Kasim (2022)	CMMS Web	Manajemen aset, *work order, scheduling*	Web-based CMMS	Data terstruktur dan terintegrasi	Tidak mendukung *dual workflow*

Sukmana & Rozi (2024)	Evaluasi CMMS	Analisis efisiensi proyek	K-Means	*Insight* efisiensi proses	Tidak membahas fitur operasional
Hidayat dkk. (2024)	Evaluasi CMMS	Evaluasi kualitas sistem	COBIT 5	Analisis kualitas komprehensif	Tidak fokus ke pelaporan abnormalitas
Shankar dkk. (2024)	CMMS Industri	Maintenance & manajemen aset	CMMS	Efektif di industri manufaktur	Tidak ada *workflow* pelaporan
Nayla Nabillah Arishima (2026)	Sistem Pelaporan Abnormalitas Mesin	Pelaporan abnormalitas, *dual workflow*, validasi	C#, ASP.NET, SQL Server	Sesuai kebutuhan, fleksibel, *multi-role*	Bergantung kualitas input user

2.2 Landasan Konsep dan Teknologi

Bagian ini memuat konsep-konsep dasar yang mendukung proyek, teknologi yang digunakan, serta profil tempat penerapan solusi. Teori yang diuraikan berkaitan langsung dengan topik dan studi kasus yang dikerjakan, yaitu pengembangan Sistem Informasi Pelaporan Abnormalitas Mesin berbasis web di lingkungan industri manufaktur.

a) Konsep Utama

Berikut adalah konsep-konsep utama yang berkaitan langsung dengan domain sistem yang dikembangkan:

1. **Sistem Informasi**

   Sistem informasi adalah sekumpulan komponen yang saling terintegrasi, meliputi manusia, perangkat keras, perangkat lunak, data,

dan prosedur yang digunakan untuk mengumpulkan, mengolah, dan menyajikan informasi guna mendukung pengambilan keputusan (Laudon & Laudon, 2020). Dalam konteks industri manufaktur, sistem informasi berperan penting dalam meningkatkan efisiensi operasional serta memastikan informasi dapat diakses secara cepat dan akurat.

2. **Manajemen Perawatan (*Maintenance Management*)**

   Menurut Mobley (2021), manajemen perawatan adalah suatu pendekatan sistematis dalam mengelola aktivitas pemeliharaan mesin untuk menjaga keandalan dan kinerja aset produksi. Tujuan utama dari manajemen perawatan adalah mengurangi *downtime*, meningkatkan efisiensi operasional, serta memperpanjang umur pakai mesin. Dalam proyek ini, konsep manajemen perawatan menjadi landasan utama perancangan alur kerja dan fitur sistem.

3. **Computerized Maintenance Management System (CMMS)**

   Menurut Campos et al. (2020), CMMS adalah sistem terkomputerisasi yang digunakan untuk mengelola kegiatan pemeliharaan mesin dan aset secara terstruktur, termasuk pencatatan gangguan, penjadwalan, dan dokumentasi perbaikan. Sistem yang dikembangkan dalam proyek ini merupakan implementasi CMMS berbasis web yang disesuaikan dengan kebutuhan PT. XYZ, sehingga perusahaan dapat meningkatkan reliabilitas mesin dan menurunkan downtime tidak terencana.

4. **Workflow Management**

   Menurut Hidayat et al. (2024), *workflow* merupakan rangkaian aktivitas terotomasi yang menggambarkan bagaimana suatu proses bisnis berjalan dari awal hingga selesai. Dalam sistem ini, terdapat dua alur kerja yang dirancang: alur utama melalui *Action Owner* dan alur *Fixed by Myself*. Penerapan *workflow* yang terstruktur memastikan setiap laporan abnormalitas diproses melalui langkah yang jelas dan terdokumentasi.

5. **Sistem Pelaporan Abnormalitas Mesin**

   Menurut Shankar et al. (2024), sistem pelaporan abnormalitas adalah

mekanisme untuk mencatat, mengklasifikasikan, dan memantau gangguan atau kondisi tidak normal pada mesin produksi. Kondisi abnormalitas yang tidak ditangani dengan baik dapat menyebabkan penurunan performa mesin hingga *downtime* produksi. Dalam sistem ini, abnormalitas diklasifikasikan menggunakan kode warna (Merah, Kuning, Hijau, Putih) untuk menentukan tingkat urgensi penanganan.

b) Teknologi, Framework, dan Tools

Berikut adalah teknologi, *framework*, dan *tools* yang digunakan dalam pengembangan sistem, disertai penjelasan fungsi dan perannya dalam solusi yang dikembangkan:

1. Sistem Berbasis Web

   Menurut Pressman (2020), sistem berbasis web adalah aplikasi yang dapat diakses melalui browser dan berjalan pada jaringan internet atau intranet. Keunggulan sistem berbasis web meliputi kemudahan akses, kemampuan *multi-user*, serta mendukung pengolahan data secara *real-time.* Dalam proyek ini, arsitektur berbasis web dipilih agar sistem dapat diakses oleh seluruh pengguna di PT. XYZ tanpa perlu instalasi perangkat lunak tambahan.

2. Framework ASP.NET MVC

   Framework merupakan kerangka kerja yang mempermudah pengembangan perangkat lunak dengan menyediakan struktur dan komponen yang dapat digunakan kembali (Pressman, 2020). Dalam proyek ini digunakan ASP.NET MVC yang menerapkan arsitektur *Model-View-Controller* untuk memisahkan logika bisnis, tampilan, dan pengolahan data. Pemisahan ini menjadikan sistem lebih mudah dikembangkan, diuji, dan dipelihara.

3. Bahasa Pemrograman C#

   C# adalah bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft dan digunakan secara luas dalam pengembangan aplikasi berbasis web maupun *desktop* (Microsoft, 2024). Dalam sistem ini, C# digunakan pada sisi backend untuk mengelola logika bisnis,

pengolahan data laporan abnormalitas, penetapan otomatis *Action Owner* dan *Validator*, serta integrasi dengan *database*.

4. SQL Server

   SQL Server merupakan Database Management System (DBMS) yang dikembangkan oleh Microsoft untuk menyimpan dan mengelola data dalam jumlah besar secara efisien dan terstruktur (Silberschatz et al., 2020). Dalam sistem ini, SQL Server digunakan untuk menyimpan data laporan abnormalitas, data pengguna, serta histori tindakan perbaikan. Penggunaan *primary key* dan *foreign key* menjaga keterkaitan antar tabel serta integritas data sistem.

c) Profil Tempat Penerapan

PT. XYZ adalah perusahaan yang bergerak di bidang industri manufaktur. Perusahaan ini mengoperasikan sejumlah mesin produksi yang memerlukan pemantauan dan perawatan rutin guna menjaga kelancaran proses operasional. Departemen maintenance di PT. XYZ bertanggung jawab terhadap penanganan setiap kondisi abnormal yang ditemukan selama proses produksi berlangsung.

Kondisi yang relevan dengan proyek ini adalah bahwa proses pelaporan abnormalitas mesin di PT. XYZ saat ini masih bergantung pada *spreadsheet (Microsoft Excel)* dan komunikasi manual seperti pesan singkat. Kondisi ini menimbulkan sejumlah permasalahan operasional, di antaranya: tidak adanya alur kerja yang baku, risiko keterlambatan penanganan laporan, kesulitan dalam melakukan *monitoring* progres secara *real-time*, serta tidak tersedianya dokumentasi riwayat yang terstruktur dan dapat diakses bersama oleh seluruh pihak terkait.

Berdasarkan kondisi tersebut, kebutuhan utama yang ingin diselesaikan melalui proyek ini adalah tersedianya sistem pelaporan berbasis web yang mampu mengotomasi alur distribusi laporan, mengklasifikasikan tingkat urgensi abnormalitas, serta menyajikan data *monitoring* secara terpusat dan *real-time*. Dengan demikian, sistem ini diharapkan dapat meningkatkan efektivitas manajemen perawatan mesin di

PT. XYZ secara keseluruhan.

2.3 Metode Pengembangan Produk

Metode pengembangan sistem yang digunakan dalam tugas akhir ini adalah *Waterfall* Model. Menurut Bassil (2012), model *Waterfall* merupakan metode pengembangan perangkat lunak yang dilakukan secara berurutan dan sistematis, dimulai dari tahap analisis hingga pemeliharaan.

Model *Waterfall* dipilih karena sistem yang dikembangkan memiliki kebutuhan yang relatif jelas dan stabil berdasarkan hasil observasi di lapangan. Selain itu, pendekatan yang terstruktur dan berurutan memudahkan proses dokumentasi serta pengendalian pengembangan sistem. Adapun tahapan dalam metode *Waterfall* yang diterapkan pada penelitian ini adalah sebagai berikut:

1. *Requirement Analysis*: Tahap ini dilakukan untuk mengidentifikasi kebutuhan sistem berdasarkan kondisi yang ada di PT. XYZ. Pengumpulan kebutuhan dilakukan melalui observasi terhadap proses pelaporan abnormalitas yang sedang berjalan serta diskusi berkala dengan mentor teknis di perusahaan.

   Hasil dari tahap ini berupa:

   - kebutuhan fungsional sistem (pelaporan abnormalitas, monitoring, riwayat)
   - penentuan alur sistem (*Action Owner* dan *Fixed by Myself*)
   - penentuan kategori abnormalitas (Merah, Kuning, Hijau, Putih)

2. *System Design*: Pada tahap ini dilakukan perancangan sistem berdasarkan kebutuhan yang telah dianalisis. Perancangan meliputi:

   - desain arsitektur sistem berbasis web
   - perancangan database (tabel dan relasi data)
   - pembuatan *use case* diagram dan *flow* proses sistem
   - perancangan antarmuka pengguna (UI)
   - perancangan logika *workflow* berdasarkan *role* pengguna

3. *Implementation*: Tahap implementasi dilakukan dengan mengembangkan sistem berdasarkan desain yang telah dibuat. Sistem dibangun

menggunakan teknologi ASP.NET berbasis C# dan database SQL Server.

Pada tahap ini dilakukan Implementasi fitur:

- fitur pelaporan abnormalitas
- fitur *monitoring* progres
- fitur riwayat perawatan mesin
- penerapan *workflow Action Owner* dan *Fixed by Myself*

4. *Testing*: Tahap pengujian dilakukan untuk memastikan sistem berjalan sesuai dengan kebutuhan yang telah ditentukan. Metode pengujian yang digunakan adalah *Blackbox* Testing, yang berfokus pada pengujian fungsi sistem tanpa melihat kode program.

   Pengujian dilakukan pada:

   - fungsi *login* dan hak akses pengguna
   - proses pelaporan abnormalitas
   - alur *workflow (Action Owner & Fixed by Myself)*
   - validasi data *input*

5. *Deployment*: Tahap ini merupakan proses penerapan sistem ke lingkungan operasional agar dapat digunakan oleh pengguna. Sistem berbasis web memungkinkan akses secara *real-time* oleh *user* sesuai dengan perannya.
6. *Maintenance*: Melakukan pemeliharaan dan pembaruan sistem secara berkala untuk memperbaiki *bug* serta menyesuaikan kebutuhan baru pengguna. Tahap pemeliharaan dilakukan setelah sistem diimplementasikan, meliputi:

   - perbaikan *bug*
   - peningkatan fitur
   - penyesuaian sistem terhadap kebutuhan baru pengguna

2.4 Pengujian Fungsional

Pengujian fungsional dilakukan untuk memastikan bahwa setiap fitur pada sistem informasi pelaporan abnormalitas mesin berbasis web bekerja sesuai dengan kebutuhan pengguna. Pengujian ini berfokus pada keluaran sistem berdasarkan masukan yang diberikan tanpa memperhatikan proses internal program.

Metode yang digunakan dalam pengujian ini adalah *Black Box* Testing, yaitu metode pengujian perangkat lunak yang berfokus pada fungsi eksternal sistem. Tujuan pengujian ini adalah untuk memverifikasi apakah setiap fungsi pada aplikasi telah berjalan sesuai dengan kebutuhan yang ditentukan pada tahap analisis dan perancangan.

Pada metode *Black Box* Testing, pengujian dilakukan dengan memberikan data uji (*input*) ke dalam sistem dan membandingkan hasil yang dihasilkan (*output*) dengan hasil yang diharapkan. Apabila hasil keluaran sesuai dengan yang diharapkan, maka fungsi tersebut dinyatakan berhasil (*valid*), sedangkan jika tidak sesuai maka fungsi tersebut dinyatakan gagal (*invalid*). Aspek-aspek yang diuji meliputi seluruh fungsi dasar serta dua alur kerja (*workflow*) utama, yaitu Alur Normal (*Action Owner*) dan Alur *Fixed by Myself*, yang masing-masing memiliki karakteristik pengisian data dan perpindahan laporan berbeda.

Adapun aspek fungsional yang diuji pada sistem ini meliputi:

1. Pemilihan kategori warna (Merah, Kuning, Hijau, Putih) digunakan untuk menentukan klasifikasi abnormalitas sekaligus mengindikasikan departemen yang bertanggung jawab atas penanganan dan pelaporan abnormalitas tersebut, yaitu:
   - Warna Merah: *Maintenance*
   - Warna Kuning: *Safety*
   - Warna Hijau: *Environment*
   - Warna Putih: Operator

   Dengan demikian, warna kategori yang dipilih oleh *Requestor* akan secara otomatis menentukan pihak yang menjadi *Action Owner* utama dalam *workflow* penanganan abnormalitas di sistem.
2. Kemunculan kolom *Action Owner* dan *Assigned Action*, yang berperilaku berbeda pada dua *workflow*:
   - Alur Utama: kolom diisi oleh *Action Owner* dan *Assigned Action* pada tahap masing-masing *role*.

- Alur *Fixed by Myself*: kolom *Action Owner* dan *Assigned Action* berpindah ke *Requestor*, sehingga *Requestor* dapat mengisinya langsung sebelum laporan menuju *Validator*.

3. Fungsi upload bukti perbaikan (foto dan keterangan) oleh pihak yang menangani abnormalitas.
4. Pengiriman laporan ke *Validator* untuk kedua alur:
    - Pada Alur Normal, laporan dikirim ke *Validator* setelah *Assigned Action* menyelesaikan tindakan.
    - Pada Alur *Fixed by Myself*, laporan langsung diterima *Validator* setelah *Requestor* menekan "*Submit*".
5. Fungsi *Validator*, yaitu melakukan *Approve* untuk menutup laporan (*Closed*) atau *Reject* untuk mengembalikan tiket:
    - Jika *Reject* pada Alur Normal, laporan dikembalikan ke *Action Owner* untuk diverifikasi dan di (*Closed*).
    - Jika *Reject* pada Alur *Fixed by Myself*, laporan dikembalikan ke *Action Owner* (teknisi yang melakukan *Request*) untuk diverifikasi dan di (*Closed*).
6. Perubahan status laporan pada setiap tahap, termasuk pengujian apakah status berubah sesuai dengan alur transisi pada kedua *workflow*.
7. Perpindahan laporan antar tahap, memastikan sistem memindahkan laporan ke daftar yang tepat untuk *Action Owner, Assigned Action*, dan *Validator* sesuai *workflow* yang dipilih.

Pengujian pada seluruh aspek tersebut memastikan bahwa kedua alur kerja Normal dan *Fixed by Myself* berjalan sesuai dengan desain, dan sistem mampu menampilkan status serta perkembangan penanganan abnormalitas secara akurat, digital, dan terdokumentasi.

BAB III ANALISIS DAN PERANCANGAN

3.1 Analisis Kebutuhan

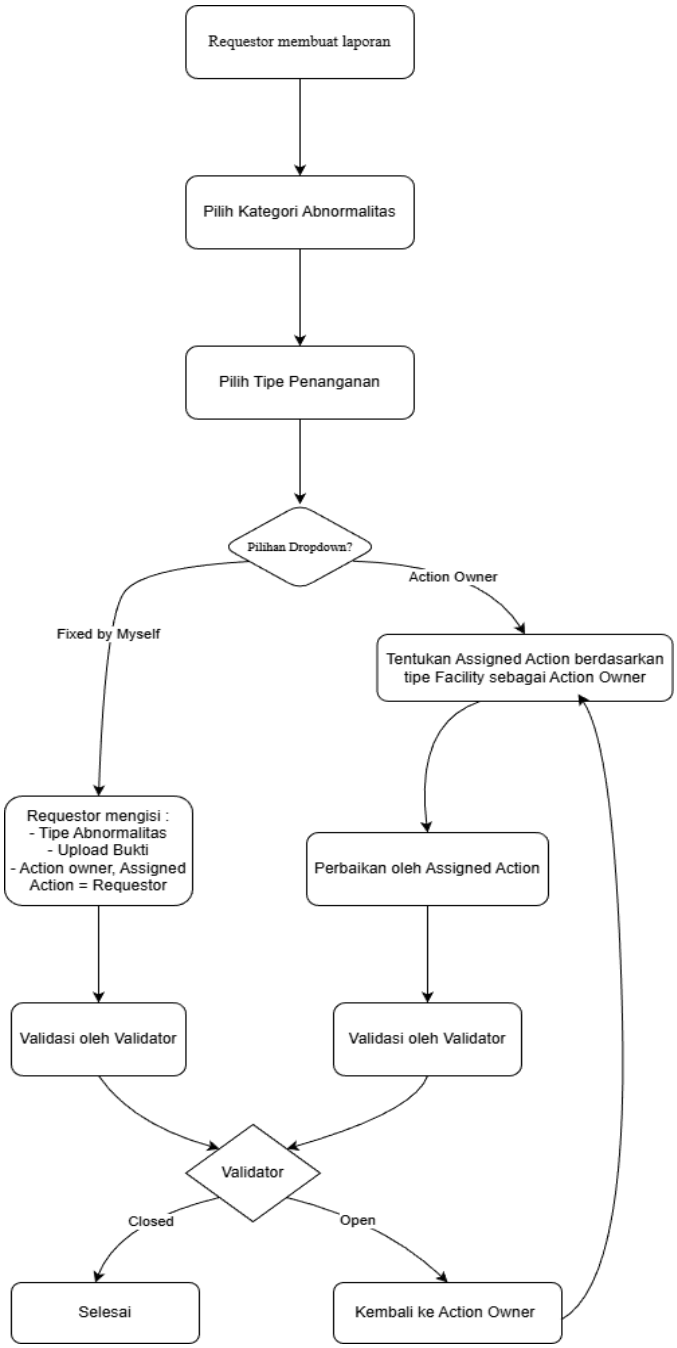
Proses pelaporan abnormalitas mesin di PT. XYZ saat ini masih dilakukan secara manual menggunakan kertas, pesan singkat, atau *spreadsheet* sederhana. Setiap kali terjadi gangguan atau kondisi tidak normal pada mesin, pihak maintenance harus mencatat detail secara manual, termasuk deskripsi masalah, lokasi mesin, dan tingkat urgensi, sebelum melaporkannya ke pihak terkait seperti *Action Owner* atau *Validator.* Namun, pencatatan ini sering kali tidak terstruktur, tanpa klasifikasi jelas berdasarkan tingkat keparahan (seperti kategori warna Merah, Kuning, Hijau, atau Putih), dan tidak mendukung alur kerja yang berbeda untuk masalah ringan maupun kompleks.

Kondisi tersebut menimbulkan berbagai kendala dalam manajemen perawatan mesin. Pihak maintenance kesulitan memantau progres penyelesaian secara *real-time*, sehingga rawan terjadi keterlambatan penanganan, duplikasi tugas, atau hilangnya jejak proses. Selain itu, proses validasi dan distribusi tugas menjadi tidak efisien karena tidak ada sistem terintegrasi yang memungkinkan pelacakan otomatis, termasuk pengembalian laporan jika ditolak oleh *Validator*. Akibatnya, laporan abnormalitas seringkali kurang akurat, memperlambat pengambilan keputusan, dan menurunkan efektivitas keseluruhan manajemen perawatan, yang berdampak pada *downtime* produksi dan nilai OEE (*Overall Equipment Effectiveness*).

Untuk mengatasi permasalahan tersebut, PT. XYZ memerlukan sistem informasi pelaporan abnormalitas mesin berbasis web yang terintegrasi dengan dua alur kerja utama (Alur Utama melalui *Action Owner* dan Alur *Fixed by Myself*) serta mekanisme kategorisasi warna untuk menentukan urgensi. Dengan sistem ini, setiap laporan dapat tercatat secara digital dan *real-time*, proses validasi menjadi lebih transparan dengan jejak audit lengkap, serta penanganan abnormalitas ringan

dapat dipercepat tanpa mengganggu alur masalah berat, sehingga meningkatkan efisiensi, akurasi, dan efektivitas manajemen perawatan mesin secara keseluruhan.

3.1.1 Proses Bisnis Berjalan



Gambar 3.1 Proses Bisnis Berjalan

Proses penanganan abnormalitas pada sistem yang dirancang berjalan secara digital dan terstruktur. *Requestor* memulai dengan membuat laporan baru, kemudian memilih kategori abnormalitas dan tipe penanganan, yaitu *"Fixed by Myself"* atau *"Action Owner"* (alur utama). Pada alur utama *"Action Owner",* sistem secara otomatis menentukan *Assigned Action* dan *Action Owner* berdasarkan *mapping area/facility* yang telah dikonfigurasi. *Action Owner* dan *Assigned Action* (jika berbeda) wajib mengisi data perbaikan serta mengunggah bukti pendukung. Setelah lengkap, laporan otomatis diteruskan ke *Validator*.

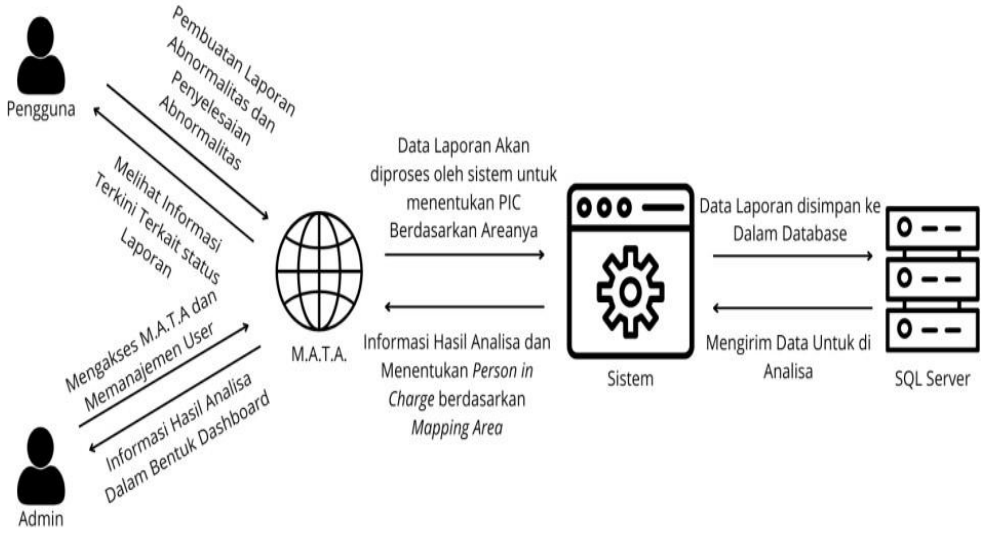
Pada alur *"Fixed by Myself", Requestor* langsung berperan sebagai *Action Owner* sekaligus *Assigned Action,* mengisi semua data perbaikan dan bukti yang diperlukan, kemudian mengirimkan laporan langsung ke *Validator* setelah submit. *Validator* memeriksa kelengkapan data dan bukti serta memastikan perbaikan telah dilakukan dengan benar dan efektif, lalu menetapkan status laporan: *"Closed"* berarti laporan selesai dan ditutup, sedangkan jika *Validator* menetapkan status *"Open"* laporan dikembalikan ke *Action Owner* untuk klarifikasi dan perbaikan ulang. Proses validasi dan klarifikasi dapat berulang hingga *Validator* menetapkan status *"Closed"*. Dengan alur ini, seluruh proses penanganan abnormalitas menjadi lebih terdokumentasi, penugasan otomatis, jejak audit jelas, serta meningkatkan akurasi dan kecepatan penyelesaian.

3.1.2 Gambaran Umum Sistem

Sistem Informasi Pelaporan Abnormalitas Mesin *(Machine Abnormality)* yang dirancang merupakan aplikasi berbasis *web* yang memungkinkan seluruh proses pelaporan, penugasan, perbaikan, hingga validasi abnormalitas mesin berlangsung secara digital, terintegrasi, dan terdokumentasi dengan baik. Sistem ini dapat diakses oleh empat aktor utama, yaitu *Requestor* (biasanya teknisi/*operator* yang menemukan masalah), *Action Owner, Assigned Action,* dan *Validator*, Sistem dapat diakses langsung melalui *web* (*browser*) pada komputer atau laptop di area produksi maupun kantor.

Sistem menyediakan fitur utama berupa pembuatan laporan abnormalitas baru, pemilihan kategori dan tipe penanganan (*Fixed by Myself* atau *Action Owner*),

penugasan otomatis *Action Owner* dan *Assigned Action* berdasarkan lokasi/area mesin, pengisian data perbaikan beserta upload bukti foto/dokumen, serta proses validasi oleh *Validator* dengan dua opsi status (*Closed/Open*). Semua aktivitas tercatat secara *real-time* dalam basis data terpusat, sehingga memudahkan pelacakan status laporan, riwayat perubahan, dan audit proses di kemudian hari. Sistem tidak memerlukan instalasi khusus di perangkat pengguna karena berjalan sepenuhnya pada server perusahaan dan dapat diakses hanya melalui jaringan internet yang aman.



Gambar 3.2 Gambaran Umum Sistem

3.2 Perancangan

Perancangan sistem dilakukan berdasarkan hasil analisis kebutuhan yang telah dijelaskan sebelumnya. Bagian ini menjabarkan rancangan fitur, diagram, dan struktur basis data yang menjadi dasar pengembangan sistem.

3.2.1 Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan fungsi dan fitur utama yang harus tersedia pada sistem agar dapat beroperasi sesuai dengan tujuan yang telah ditetapkan. Adapun kebutuhan fungsional pada *website* disajikan pada tabel berikut:

Tabel 3.1 Kebutuhan Fungsional

No.	Kebutuhan Fungsional
FR-01	Pengguna (*Requestor, Action Owner, Assigned Action, Validator*) dapat melakukan *Login.*
FR-02	Semua Pengguna dapat melihat riwayat laporan, filter berdasarkan status, kategori, atau tanggal.
FR-03	Admin dapat mengelola data pengguna.
FR-04	*Requestor* dapat membuat laporan abnormalitas baru, termasuk *input* deskripsi, lokasi mesin, kategori warna berdasarkan tipe abnormalitas (Merah, Kuning, Hijau, Putih) dan pemilihan alur (Utama atau *Fixed by Myself*).
FR-05	*Action Owner/Assigned Action* dapat mengisi data perbaikan, *upload* Bukti foto/dokumen, dan submit ke tahap validasi laporan abnormalitas.
FR-06	*Validator* dapat meninjau laporan, selesai (ubah status ke *Closed*)atau *open* (kembalikan ke *Action Owner* dengan catatan).
FR-07	Admin dan Pengguna dapat mengakses *dashboard* analisis abnormalitas.
FR-08	Admin dan Pengguna dapat mengubah *password*.
FR-09	Admin dan Pengguna (*Requestor, Action Owner, Assigned Action, Validator*) dapat melakukan *logout* dari sistem abnormalitas.

3.2.2 Kebutuhan Non-Fungsional

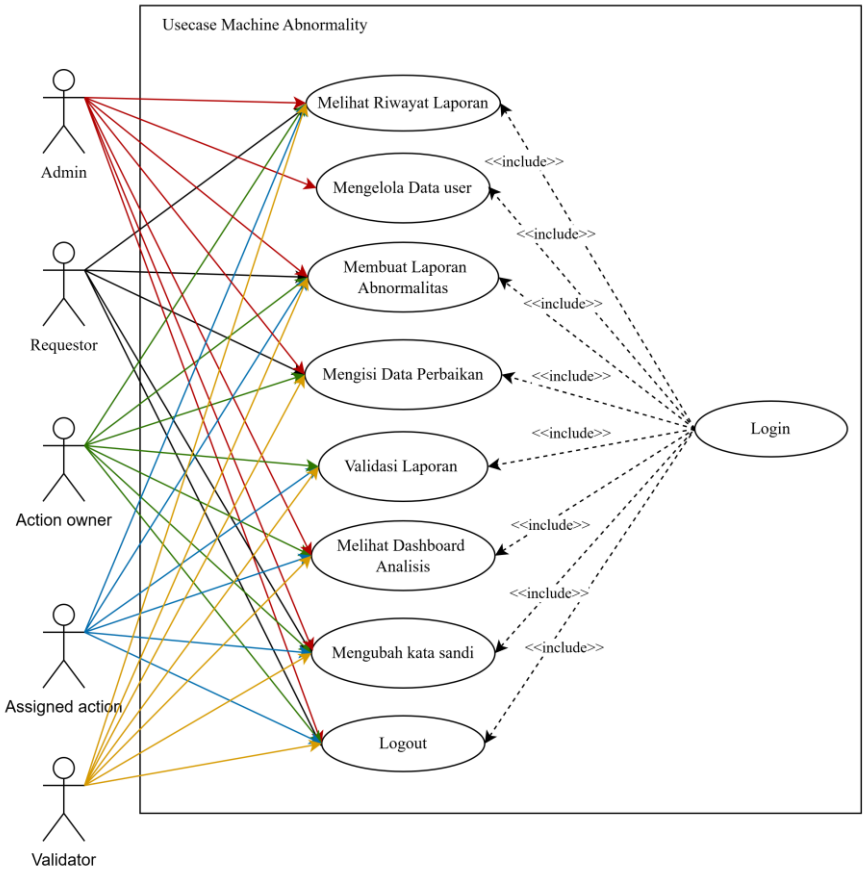
Kebutuhan *Non-Fungsional* pada *website* dapat dilihat pada tabel berikut:

Tabel 3.2 Kebutuhan *Non-Fungsional*

No.	Kebutuhan *Non-Fungsional*
NFR-01	Sistem harus mampu menampilkan halaman atau memproses permintaan pengguna dengan waktu respon maksimal 3 detik untuk setiap permintaan pada kondisi penggunaan normal.
NFR-02	Sistem harus memiliki mekanisme keamanan untuk melindungi data

	sensitif dengan menerapkan validasi input untuk mencegah SQL Injection dan XSS.

3.2.3 Diagram *Use Case*



Gambar 3.3 *Use Case Diagram*

Use Case Diagram pada sistem *Machine Abnormality* menggambarkan dua aktor utama, yaitu *Admin* dan *User*, yang berinteraksi dengan sistem untuk mendukung manajemen perawatan mesin. *Admin* memiliki hak akses penuh yang mencakup kemampuan eksklusif untuk mengelola data pengguna serta akses beberapa fitur operasional seperti membuat laporan abnormalitas.

Sementara itu, aktor *User* merepresentasikan peran operasional di lapangan yang dapat membuat laporan, mengisi data perbaikan, memvalidasi, serta mengakses informasi melalui fitur melihat riwayat laporan dan *dashboard* analisis

tanpa hak akses manajemen pengguna. Setiap proses pengelolaan dan pelaporan data tersebut terhubung dengan *use case Login* melalui relasi *include*, yang mewajibkan seluruh aktor untuk melakukan otentikasi terlebih dahulu sebelum dapat mengakses fitur-fitur utama sistem. Diagram ini secara keseluruhan menunjukkan alur interaksi sistem yang terstruktur untuk memastikan setiap pelaporan, penanganan, dan validasi abnormalitas mesin terdokumentasi secara digital dan terintegrasi.

3.2.4 Skenario *Use Case*

Bagian ini menjelaskan detail langkah-langkah interaksi antara aktor dengan sistem untuk setiap *use case* yang telah dirancang.

a) Use Case Scenario login

Tabel 3.3 *Use Case Scenario Login*

Use Case Scenario* 1: *"Login"	
Nomor	UC-01
Nama	*Login*
Deskripsi	Admin memasukan *Username* dan *Password* untuk masuk ke *Web*.
Primary Actor	Semua aktor
Preconditions	Semua aktor telah memiliki akun terdaftar di aplikasi tetapi belum *login.*
Postconditions	Semua aktor berhasil masuk dan diarahkan ke halaman *Observation.*
Main Succes Scenario	1. Semua aktor membuka *web*. 2. Semua aktor memasukan *Username* dan *Password.* 3. Semua aktor menekan tombol "*Log in".* 4. Sistem memvalidasi data yang dimasukkan aktor. 5. Jika data valid, sistem mengarahkan Admin dan Pengguna ke halaman *Observation.*

Alternative Flow (Extension)	1. Semua aktor memasukan *Username* dan *Password* yang salah. Sistem menampilkan pesan kesalahan "*User and Password not Registered"* dan meminta pengguna untuk mengisi ulang data yang benar.

b) *Use case Scenario* Melihat Riwayat Laporan

Tabel 3.4 *Use Case Scenario* Melihat Riwayat Laporan

***Use Case Scenario* 2: "Melihat Riwayat Laporan"**	
Nomor	UC-02
Nama	Melihat Riwayat Laporan
Deskripsi	Semua aktor dapat melihat riwayat/status laporan.
Primary Actor	Semua aktor
Preconditions	Semua aktor sudah *login* dan berada di halaman *Observation.*
Postconditions	Semua aktor dapat melihat riwayat/status laporan.
Main Succes Scenario	1. Semua aktor milih satu laporan yang ingin dilihat riwayat dan statusnya. 2. Semua aktor menekan tombol mata pada laporan. 3. Sistem menampilkan detail riwayat/status laporan.
Alternative Flow (Extension)	1. Internet terputus saat mengambil data riwayat/status. Sistem menampilkan pesan kesalahan "*An error occurred while loading the report history. Please try again later.*"

c) *Use Case Scenario* Mengelola Data *User*

Tabel 3.5 *Use Case Scenario* Mengelola Data User

***Use Case Scenario* 3: "Mengelola Data *User*"**	
Nomor	UC-03
Nama	Mengelola Data *User*

Deskripsi	Admin dapat melakukan pengelolaan data pengguna melalui operasi CRUD *(Create, Read, Update, Delete)* untuk memastikan data pengguna tersimpan secara akurat dan selalu diperbarui.
Primary Actor	Admin
Preconditions	Admin telah berhasil *login* ke dalam sistem dan diarahkan ke *Dashboard* admin.
Postconditions	Data pengguna berhasil ditambahkan, diperbarui, atau dihapus, dan perubahan tersebut langsung ditampilkan pada halaman *User Management* dalam bentuk tabel yang terbarui.
Main Succes Scenario	1. Admin memilih menu "*User Management*". 2. Sistem menampilkan daftar data pengguna dalam bentuk tabel. 3. Admin memilih salah satu aksi berikut: • *Add User* untuk menambahkan data pengguna baru. • *Edit* untuk mengubah data pengguna. • *Delete* untuk menghapus data pengguna. 4. Admin mengisi atau memperbarui informasi data *User*. 5. Admin menyimpan perubahan data. 6. Sistem memvalidasi data yang di *input*. 7. Sistem menyimpan data ke dalam *database* dan memperbarui tampilan halaman *User Management*.
Alternative Flow (Extension)	1. Jika data yang di *input* tidak lengkap atau tidak valid. Sistem menampilkan pesan validasi dan admin diminta untuk memperbaiki data.

d) *Use case Scenario* Membuat Laporan abnormalitas

Tabel 3.6 *Use case Scenario* Membuat Laporan abnormalitas

***Use case Scenario* 4: "Membuat Laporan abnormalitas"**	
Nomor	UC-04

Nama	Membuat Laporan abnormalitas
Deskripsi	Semua aktor dapat membuat laporan abnormalitas dengan mengisi data abnormalitas mesin secara lengkap, termasuk informasi waktu kejadian, lokasi, kategori abnormalitas, alur penanganan, serta bukti pendukung untuk ditindaklanjuti oleh pihak terkait.
Primary Actor	Semua aktor
Preconditions	Semua aktor telah berhasil *login* ke dalam sistem dan memiliki hak akses untuk membuat laporan abnormalitas.
Postconditions	Laporan abnormalitas berhasil tersimpan di dalam sistem dan status laporan berada pada tahap awal sesuai dengan alur yang dipilih (Utama atau *Fixed by Myself*).
Main Succes Scenario	1. Semua aktor memilih menu *"Add New Abnormality"*. 2. Sistem menampilkan *form* pembuatan laporan abnormalitas. 3. Semua aktor mengisi data laporan abnormalitas. 4. Semua aktor mengunggah foto temuan (*finding picture*) sebagai bukti abnormalitas. 5. Semua aktor menekan tombol *Submit.* 6. Sistem melakukan validasi terhadap data yang di*input*. 7. Sistem menyimpan laporan abnormalitas ke dalam *database*. 8. Sistem menampilkan notifikasi bahwa laporan abnormalitas berhasil dibuat.
Alternative Flow (Extension)	1. Jika terdapat data yang belum diisi atau tidak valid. Sistem menampilkan pesan validasi dan Admin atau Pengguna diminta untuk melengkapi atau memperbaiki data.

e) *Use case Scenario* Mengisi Data Perbaikan

Tabel 3.7 *Use Case Scenario* Mengisi Data Perbaikan

***Use case Scenario* 5: "Mengisi Data Perbaikan"**	
Nomor	UC-05
Nama	Mengisi Data Perbaikan
Deskripsi	*Use case* ini memungkinkan aktor (*Action Owner* atau *Assigned Action*) untuk mengisi data perbaikan pada laporan abnormalitas yang telah ditugaskan. *Form* yang ditampilkan berbeda tergantung peran: *Action Owner* mengisi analisis akar masalah dan penugasan, sedangkan *Assigned Action* mengisi tindakan perbaikan, bukti (foto/*file*), dan target penyelesaian. Setelah pengisian selesai dan disimpan, laporan akan berpindah ke tahap berikutnya sesuai *workflow.*
Primary Actor	*User* (sebagai *Action Owner, Assigneed Action,* atau *Requestor* pada alur *Fixed by Myself*). Admin juga dapat mengakses jika diperlukan untuk pengelolaan.
Preconditions	• *User* telah *login* ke sistem. • Terdapat laporan abnormalitas dengan status "*Open*" yang telah ditugaskan ke *User* (baik sebagai *Action Owner* atau *Assigned Action*). • Data dasar laporan telah diisi pada tahap sebelumnya (oleh *Requestor*). • Untuk *Assigned Action*: *Action Owner* telah menyelesaikan bagiannya.
Postconditions	• Data perbaikan yang telah di isi tersimpan di *database*. • Foto/bukti perbaikan dan *finding picture* ter-*upload* dan terhubung dengan laporan abnormalitas. • Status laporan tetap "*Open*" tetapi tahap berpindah (misalnya, dari *Action Owner* ke *Assigned Action,* atau dari *Assigned Action* ke *Validator*).

	• Riwayat perubahan (*History*) diperbarui dengan aktivitas pengisian terbaru.
Main Succes Scenario	1. *User* membuka laporan yang ditugaskan kepadanya dari *list* tabel laporan abnormalitas tersebut. 2. Sistem menampilkan form pengisian sesuai peran *User*: ● Jika *User* adalah *Action Owner*: *form* menampilkan kolom yang perlu diisi oleh *Action owner.* ● Jika *User* adalah *Assigned Action*: form menampilkan kolom yang harus diisi oleh *assigned action.* 3. User mengisi semua *field* wajib (ditandai *) dan mengupload *file*/foto yang diperlukan. 4. User menekan tombol "*Submit*" (*Action Owner*) atau "*Assigned*" (*Assigned Action*). 5. Sistem memvalidasi kelengkapan data. 6. Jika *valid*, sistem menyimpan data, memperbarui riwayat dan memindahkan laporan ke tahap berikutnya. 7. Sistem menampilkan notifikasi sukses dan mengarahkan *User* kembali ke *list* tabel laporan abnormalitas.
Alternative Flow (Extension)	1. Jika pada tahap validasi terdapat *field* wajib yang belum diisi atau data tidak sesuai format. Sistem menampilkan pesan validasi dan *User* diminta untuk melengkapi atau memperbaiki data sebelum dapat melanjutkan proses *submit*. 2. Jika *file* atau foto yang diunggah tidak sesuai ketentuan (format tidak didukung atau ukuran melebihi batas). Sistem menampilkan pesan kesalahan unggah dan *User* diminta untuk mengunggah ulang file yang sesuai. 3. Jika proses penyimpanan data gagal akibat gangguan sistem atau koneksi. Sistem menampilkan notifikasi

	kegagalan dan laporan tetap berada pada tahap sebelumnya. 4. Jika *User* membatalkan proses sebelum menekan tombol *Submit/Assigned.* Sistem tidak menyimpan perubahan dan mengarahkan *User* kembali ke *form* atau daftar laporan abnormalitas.

f) *Use case Scenario* Validasi Laporan

Tabel 3.8 *Use case Scenario* Validasi Laporan

***Use case Scenario* 6: "Validasi Laporan"**	
Nomor	UC-06
Nama	Validasi Laporan
Deskripsi	*Use case* ini memungkinkan *Validator* untuk melakukan validasi terhadap laporan abnormalitas mesin yang telah diisi oleh *Action Owner* dan *Assigned Action*. *Validator* meninjau kelengkapan dan kebenaran data serta bukti perbaikan, memberikan *remark* apabila diperlukan, dan menentukan status akhir laporan (*Closed* atau *Open*). Apabila laporan disetujui (*Closed*), laporan dinyatakan selesai. Sebaliknya, apabila ditolak (*Open*), laporan dikembalikan ke aktor terkait untuk dilakukan perbaikan lanjutan.
Primary Actor	*Validator*
Preconditions	• *User* telah berhasil *login* ke dalam sistem • Terdapat laporan abnormalitas dengan status *"Open"* yang telah selesai diisi data perbaikan oleh *(Assigned Action* atau *Requestor)* pada alur *(Fixed by Myself).* • *laporan berada pada tahap siap divalidasi.*
Postconditions	• Status laporan diperbarui menjadi "*Closed*" (jika disetujui) atau tetap "*Open"* apabila ditolak dan akan proses ke tahap *clarify*. • *Remark* hasil validasi tersimpan di *database* dan terhubung

	dengan laporan terkait. • Riwayat perubahan (*History*) diperbarui dengan aktivitas validasi. • Apabila status *"Closed"*, laporan masuk ke riwayat laporan selesai dan tidak dapat diedit kembali.
Main Succes Scenario	1. *Validator* membuka laporan yang berada pada status siap di validasi dari daftar laporan abnormalitas. 2. Sistem menampilkan *form* validasi yang berisi detail laporan dalam kondisi *read-only*, meliputi kolom yang sebelumnya telah diisi oleh *Assigned Action* serta *field* yang dapat diedit yaitu Status* dan *Remark**. 3. *Validator* meninjau seluruh data dan bukti perbaikan yang telah diunggah. 4. *Validator* mengisi *Remark* (wajib apabila melakukan penolakan, opsional apabila menyetujui). 5. *Validator* memilih status validasi, yaitu *"Closed"* atau *"Open"*. 6. Validator menekan tombol *Validate*. 7. Sistem melakukan validasi terhadap *input*, termasuk memastikan *Remark* terisi apabila status *"Open"* dipilih. 8. Jika *valid*, sistem menyimpan hasil validasi, memperbarui status laporan, memperbarui riwayat. 9. Sistem menampilkan notifikasi keberhasilan dan mengarahkan *Validator* kembali ke list daftar laporan abnormalitas.
Alternative Flow (Extension)	1. *Validator* meninjau laporan yang diisi langsung oleh *Requestor*. Proses persetujuan atau penolakan dilakukan dengan alur yang sama seperti proses utama (alur *Fixed by Myself*). 2. *Validator* memilih status *"Closed"*, mengisi *Remark*

	(opsional), lalu menekan *Validate*. Sistem menandai laporan sebagai selesai. 3. *Validator* memilih status "*Open*", mengisi *Remark* alasan penolakan (wajib), lalu menekan *Validate*. Sistem mengembalikan laporan abnormalitas ke aktor sebelumnya untuk dilakukan perbaikan(*clarify*)

g) *Use case Scenario* Melihat *Dashboard* Analisis

Tabel 3.9 *Use case Scenario* Melihat *Dashboard* Analisis

***Use case Scenario* 7: "Melihat *Dashboard* Analisis"**	
Nomor	UC-07
Nama	Melihat *Dashboard* Analisis
Deskripsi	*Use case* ini memungkinkan aktor untuk mengakses dan melihat *Dashboard* analisis yang menampilkan visualisasi data abnormalitas mesin secara agregat, termasuk grafik batang untuk *root cause (human weakness)*, konsekuensi jika dibiarkan (*what will happen if left as is*), *non-compliance* per operator dan A*ssigned Action*, serta *monitoring* bulanan per *facility/site* (seperti total *OPLs, % OPL per site,* dan tren kumulatif abnormalitas *closed*). *Dashboard* mendukung filter berdasarkan tanggal, *facility, line/cell no, station ID*, dan *time range* untuk analisis yang lebih spesifik, guna mendukung pengambilan keputusan manajemen perawatan di PT. XYZ.
Primary Actor	Semua *User*
Preconditions	• Semua *User* telah *login* ke sistem. • Terdapat data laporan abnormalitas yang telah divalidasi dan *closed* di *database* (minimal satu laporan untuk menampilkan analisis). • Sistem memiliki akses ke data agregat dari entitas terkait (seperti laporan abnormalitas*, root cause, assigned action, facility,* dll.) untuk *generate chart.*

	• *Dashboard* telah diintegrasikan dengan basis data untuk *query real-time* atau *cached*.
Postconditions	• *Dashboard* ditampilkan dengan *chart* dan data yang relevan sesuai filter yang dipilih. • Data analisis tetap tersimpan di *database* tanpa perubahan (*read-only*). • Riwayat akses *dashboard* tercatat untuk audit jika diperlukan. • Semua *User* dapat menggunakan *insights* dari *dashboard* untuk mendukung manajemen perawatan (misalnya, identifikasi tren *root cause* atau *facility* dengan abnormalitas tinggi).
Main Succes Scenario	1. Semua *User* membuka menu atau halaman *dashboard* analisis dari navigasi utama sistem. 2. Sistem menampilkan *dashboard* default dengan visualisasi utama: grafik batang untuk *root cause* (misalnya, *Inadequate Basic Conditions* sebagai tertinggi), konsekuensi (misalnya, *Breakdown* sebagai risiko utama), *non-compliance* per kategori (misalnya, *Fall/Slip/Trip*), serta *monitoring* bulanan (*bar chart* per bulan dengan warna merah/hijau untuk *open/closed, line chart* untuk total OPLs kumulatif, dan *pie chart* untuk % OPL per *site/facility* seperti Cav2 FA, dll.). 3. Semua *User* memeriksa data agregat dan *insights* (misalnya, tren per bulan dari *June'25* hingga *Nov'25*, atau distribusi per *facility* seperti Cav2, Cav3, dll.). *4.* Jika diperlukan, Semua *User* menerapkan *filter* (*Date From/To, Time Range seperti Daily, Facility, Line/Cell no, Station ID).* 5. Sistem memproses *filter* dan memperbarui *dashboard*

	secara *real-time* dengan data yang difilter (misalnya, hanya data Cav2 PCBA). 6. Semua *User* selesai melihat dan kembali ke halaman utama atau menu lain.
Alternative Flow (Extension)	1. Semua *User* memilih filter (misalnya*, Date From:* 01/01/2025, *Date To:* 26/11/2025, *Facility:* Cav2 PCBA Equipment). Sistem *query* database dan *refresh chart* sesuai *filter*. 2. Jika tidak ada data laporan, sistem menampilkan pesan *"No data available"* atau *dashboard* kosong dengan instruksi untuk membuat laporan terlebih dahulu. 3. Semua *User* mengklik elemen *chart* (misalnya, bar untuk Cav2 FA), sistem navigasi ke daftar laporan detail terkait untuk analisis lebih dalam.

h) *Use case Scenario* Mengubah Kata Sandi

Tabel 3.10 *Use case Scenario* Mengubah Kata Sandi

***Use case Scenario* 8: "Mengubah Kata Sandi"**	
Nomor	UC-08
Nama	Mengubah Kata Sandi
Deskripsi	*Use case* ini memungkinkan *user* yang telah *login* untuk mengubah kata sandi akunnya sendiri demi menjaga keamanan akses ke sistem *Machine Abnormality*. *User* harus memasukkan kata sandi lama yang benar, serta kata sandi baru yang kemudian dikonfirmasi ulang sebelum perubahan disimpan.
Primary Actor	*User* (semua peran yang telah *login*, termasuk *Requestor, Action Owner, Assigned Action, Validator*) dan Admin.
Preconditions	• *User* telah berhasil *login* ke sistem. • *User* mengetahui kata sandi lama (*current password*) yang

	sedang digunakan. • Sistem memiliki mekanisme *hashing*/enkripsi kata sandi yang aman di *database*. • Menu atau tombol "*Change Password*" tersedia di navigasi utama atau *sidebar* (misalnya, di bawah profil *user* atau menu *Settings*).
Postconditions	• Kata sandi user berhasil diperbarui di *database*. • Semua *User* tetap berada dalam sesi *Login* yang aktif (tidak perlu *Login* ulang). • Notifikasi sukses ditampilkan bahwa kata sandi telah berhasil diubah.
Main Succes Scenario	1. Semua *User* mengakses menu *"Change Password"* dari *sidebar*/navigasi utama. *2.* Sistem menampilkan form pengubahan kata sandi yang berisi field: *Old Password, New Password, Confirm Password*, serta tombol *"Change Password".* *3.* Semua *User* memasukkan kata sandi lama pada *field Old Password.* *4.* Semua *User* memasukkan kata sandi baru pada *field New Password.* 5. Semua *User* mengulang kata sandi baru pada *field Confirm Password.* 6. Semua *User* menekan tombol *"Change Password".* 7. Sistem memvalidasi: ● *Old Password* sesuai dengan yang tersimpan di database. ● *New Password* memenuhi kebijakan (misalnya, minimal 8 karakter, kombinasi huruf besar-kecil-angka). ● *Confirm Password* sama persis dengan *New Password*.

	8. Jika semua validasi lolos, sistem memperbarui kata sandi di *database* (dengan *hashing*). 9. Sistem menampilkan notifikasi sukses "*Password successfully changed*" dan mengarahkan *User* kembali ke halaman utama.
Alternative Flow (Extension)	1. Sistem mendeteksi *Old Password* tidak cocok. Sistem menampilkan pesan error *"Old password is incorrect". User* memperbaiki dan submit ulang. 2. *Confirm password* berbeda dengan *New Password.* Sistem menampilkan pesan error "*New password and confirmation do not match*". *User* memperbaiki dan submit ulang.

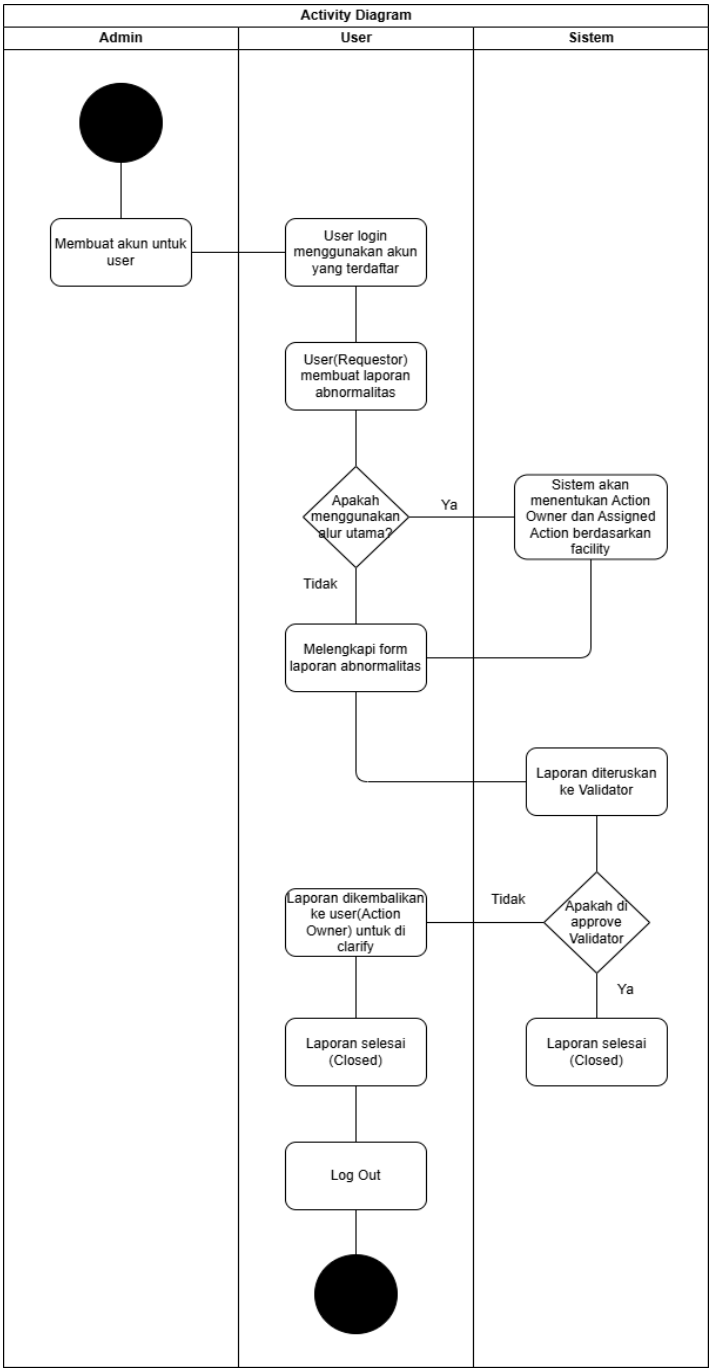
i) *Use case Scenario Logout*

Tabel 3.11 *Use case Scenario Logout*

Use Case Scenario 9: "Logout"	
Nomor	UC-09
Nama	*Logout*
Deskripsi	*Use case* ini memungkinkan *User* yang sedang *Login* untuk mengakhiri sesi secara aman dan keluar dari *aplikasi Machine Abnormality.* Setelah *Logout*, *User* akan diarahkan kembali ke halaman *login* dan sesi aktif dihapus untuk menjaga keamanan.
Primary Actor	Semua *User*
Preconditions	● *User* telah berhasil *login* ke sistem. ● Sesi *User* sedang aktif. ● Menu atau tombol "*Log Out*" tersedia di *sidebar*/navigasi utama.
Postconditions	1. Sesi pengguna berhasil dihapus dari *server*. 2. User dialihkan ke halaman *login* web. 3. Semua data sesi sementara dibersihkan. 4. Notifikasi sukses "*You have successfully logged out*" atau

	sejenisnya dapat ditampilkan sebelum *redirect*. 5. *User* harus *login* ulang untuk mengakses sistem kembali.
Main Succes Scenario	1. *User* mengakses menu "*Log Out*" dari *sidebar*/navigasi utama (misalnya, mengklik ikon atau tombol "*Log Out*" di bawah "*Change Password*"). 2. Sistem menghapus sesi aktif pengguna di *server*. 3. Sistem mengakhiri sesi dan mengarahkan *User* ke halaman *login*. 4. Sistem menampilkan halaman *login* dengan form *username* dan *password.*

3.2.5 *Activity Diagram*

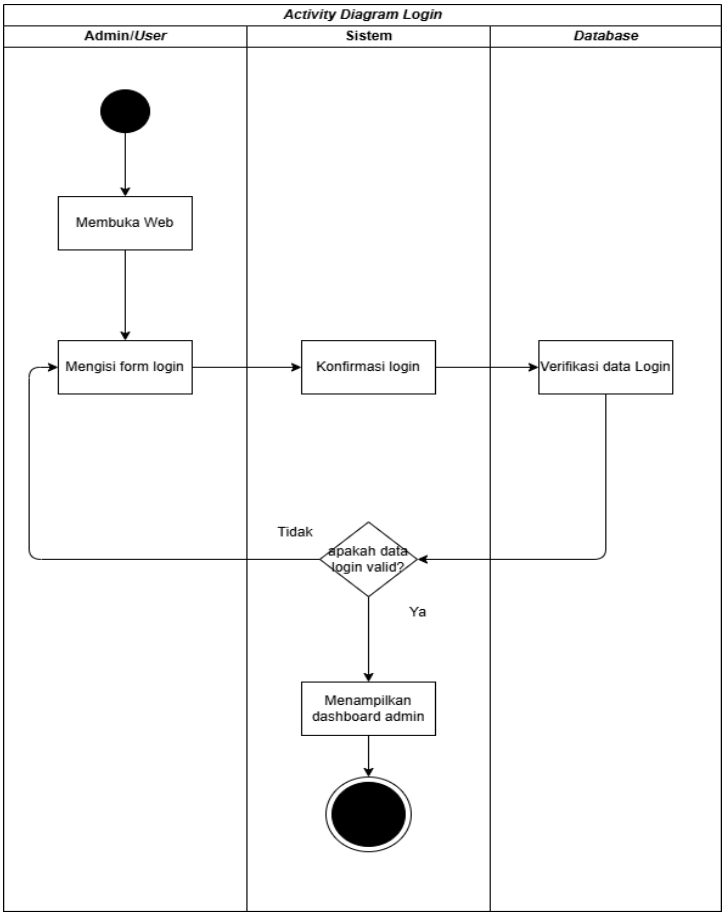


Gambar 3.4 *Activity Diagram Machine Abnormality*

Activity Diagram ini menggambarkan alur aktivitas pengguna dalam sistem pelaporan *Machine Abnormality*, yang dimulai dari *Admin* yang membuat akun untuk *User*, diikuti oleh *User* yang melakukan *Login* menggunakan akun tersebut. Setelah berhasil *Login*, *User* berperan sebagai *Requestor* untuk membuat laporan

abnormalitas mesin. Sistem kemudian memeriksa apakah laporan menggunakan alur utama; jika ya, sistem otomatis menentukan *Action Owner* dan *Assigned Action* berdasarkan *facility* terkait. *User* selanjutnya melengkapi form laporan abnormalitas, dan laporan tersebut diteruskan ke *Validator* untuk divalidasi. *Validator* memutuskan apakah laporan di-*approve*; jika ya, laporan ditandai sebagai "*Closed*", sedangkan jika tidak, laporan dikembalikan ke *Action Owner* untuk diklarifikasi lebih lanjut. Setelah proses selesai, baik melalui *approval* maupun klarifikasi, status laporan menjadi *Closed*. Proses diakhiri dengan *User* melakukan *Logout* dari sistem. Diagram ini menunjukkan interaksi terstruktur antara Admin, *User*, dan Sistem dalam mengelola pelaporan serta penyelesaian abnormalitas mesin secara efisien.

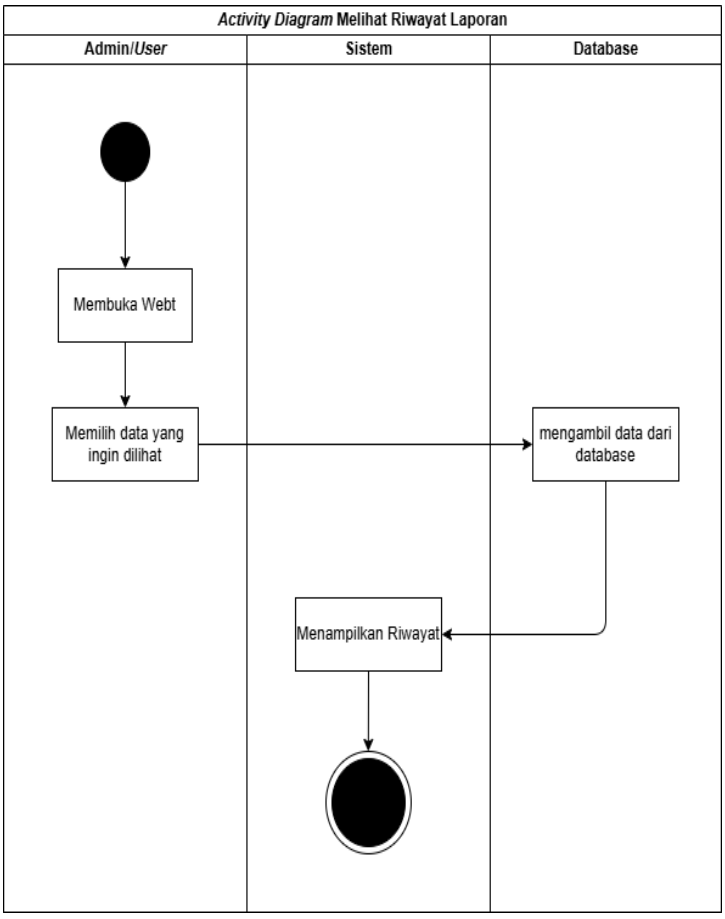
a) *Activity Diagram Login*



Gambar 3.5 *Activity Diagram Login*

Gambar 3.5 menjelaskan alur proses aktivitas *login* ke dalam sistem, di mana Admin atau *User* membuka web, mengisi form *login*, lalu sistem melakukan konfirmasi dan *database* memverifikasi data tersebut. Jika data *login* valid, sistem akan menampilkan *dashboard* admin, sedangkan jika tidak valid, sistem akan mengembalikan pengguna ke form *login* untuk mengisi ulang data.

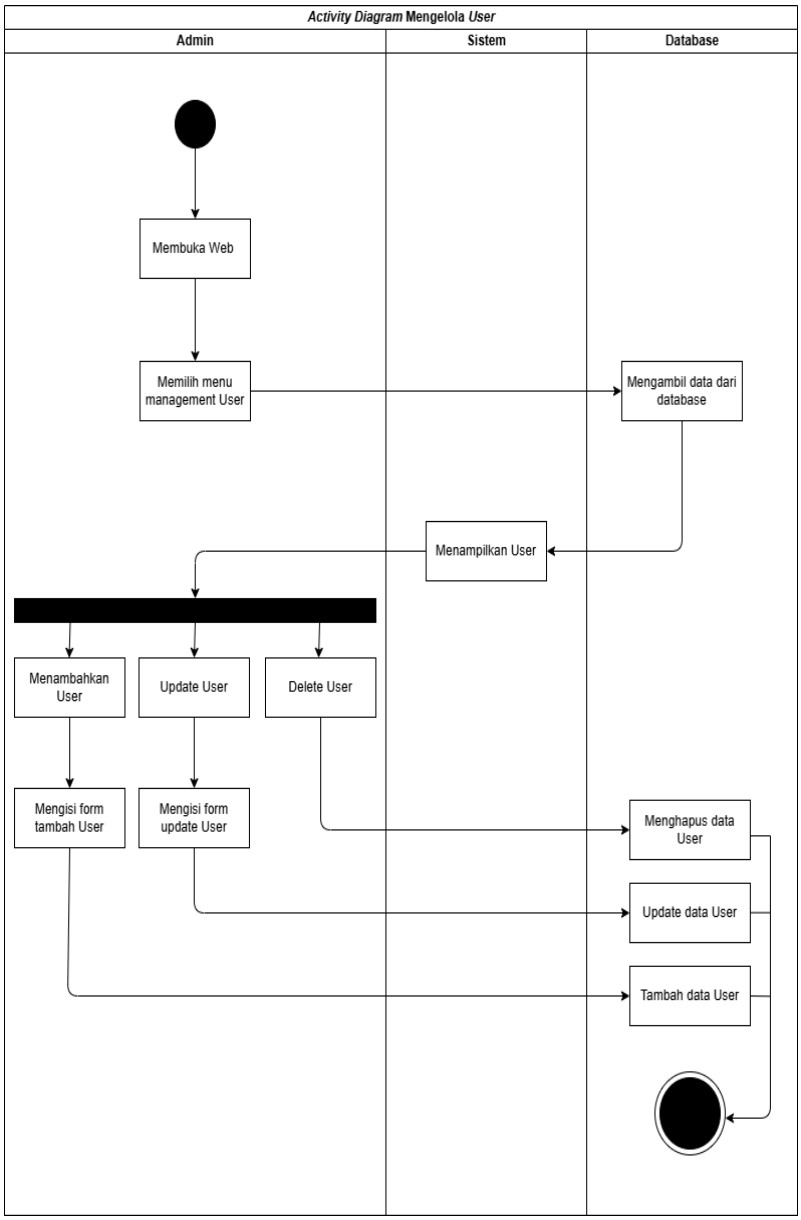
b) *Activity Diagram* Melihat Riwayat Laporan



Gambar 3.6 *Activity Diagram* Melihat Riwayat Laporan

Gambar 3.6 menjelaskan alur aktivitas melihat riwayat laporan. Admin atau *User* membuka web dan memilih data laporan yang ingin dilihat. Sistem kemudian akan meminta *database* untuk mengambil data tersebut, lalu menampilkan riwayat laporan kepada *User*.

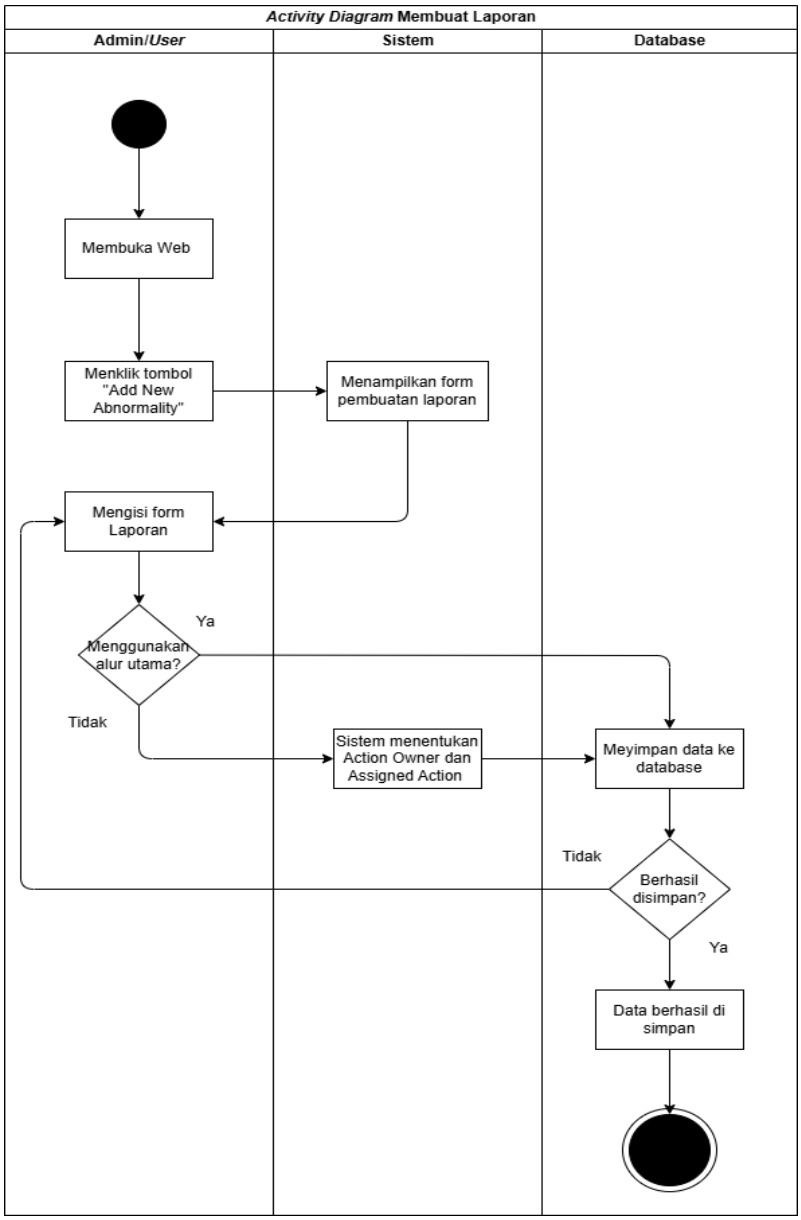
c) *Activity Diagram* Mengelola Data *User*



Gambar 3.7 *Activity Diagram* Mengelola Data *User*

Gambar 3.7 menunjukkan aktivitas manajemen pengguna oleh Admin. Setelah Admin memilih menu *User Management*, sistem menampilkan daftar *user* dari *database*. Admin kemudian dapat memilih untuk menambahkan, memperbarui (*update*), atau menghapus (*delete*) *user*. Setiap aksi tersebut akan diproses dan disimpan perubahannya ke dalam *database*.

d) *Activity Diagram* Membuat Laporan abnormalitas

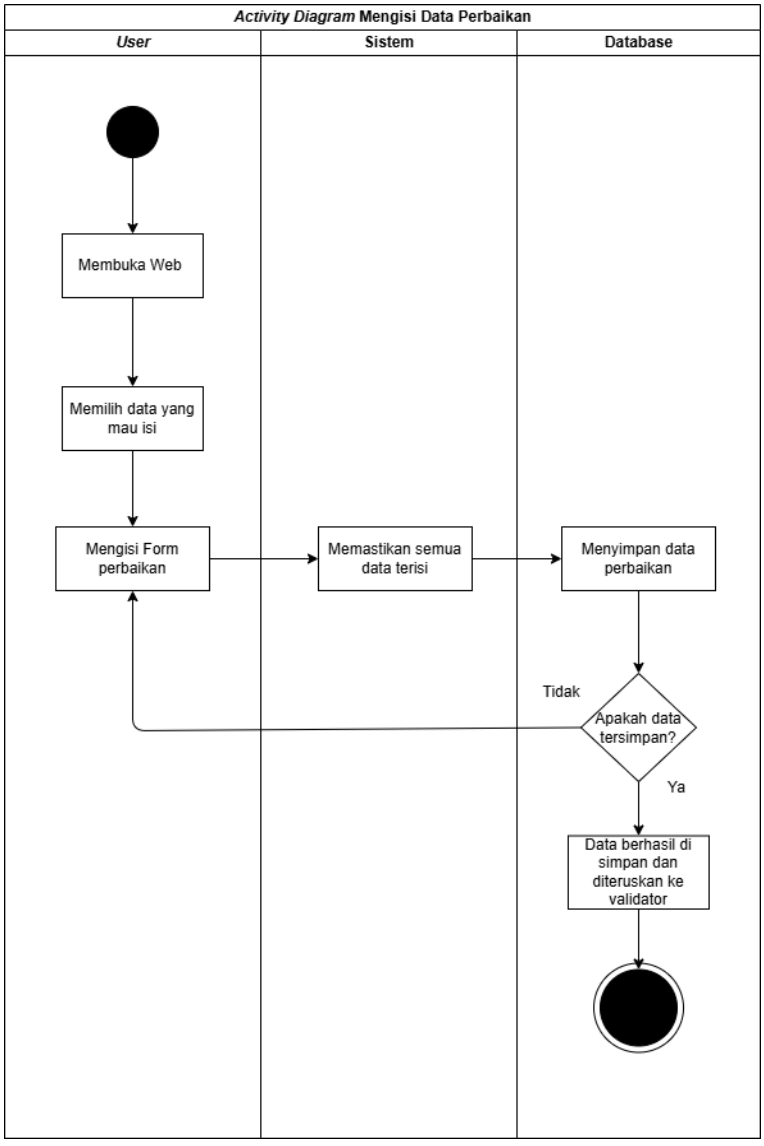


Gambar 3.8 *Activity Diagram* Membuat Laporan abnormalitas

Gambar 3.8 memaparkan alur pembuatan laporan baru. Dimulai dari Admin atau *User* mengklik tombol *"Add New Abnormality"*, sistem menampilkan form yang kemudian diisi oleh *User*. Jika menggunakan alur utama, data langsung disimpan ke *database*; jika tidak, sistem akan menentukan *Action Owner* terlebih dahulu sebelum menyimpan. Jika

penyimpanan berhasil, proses selesai, namun jika gagal, pengguna harus mengisi form kembali.

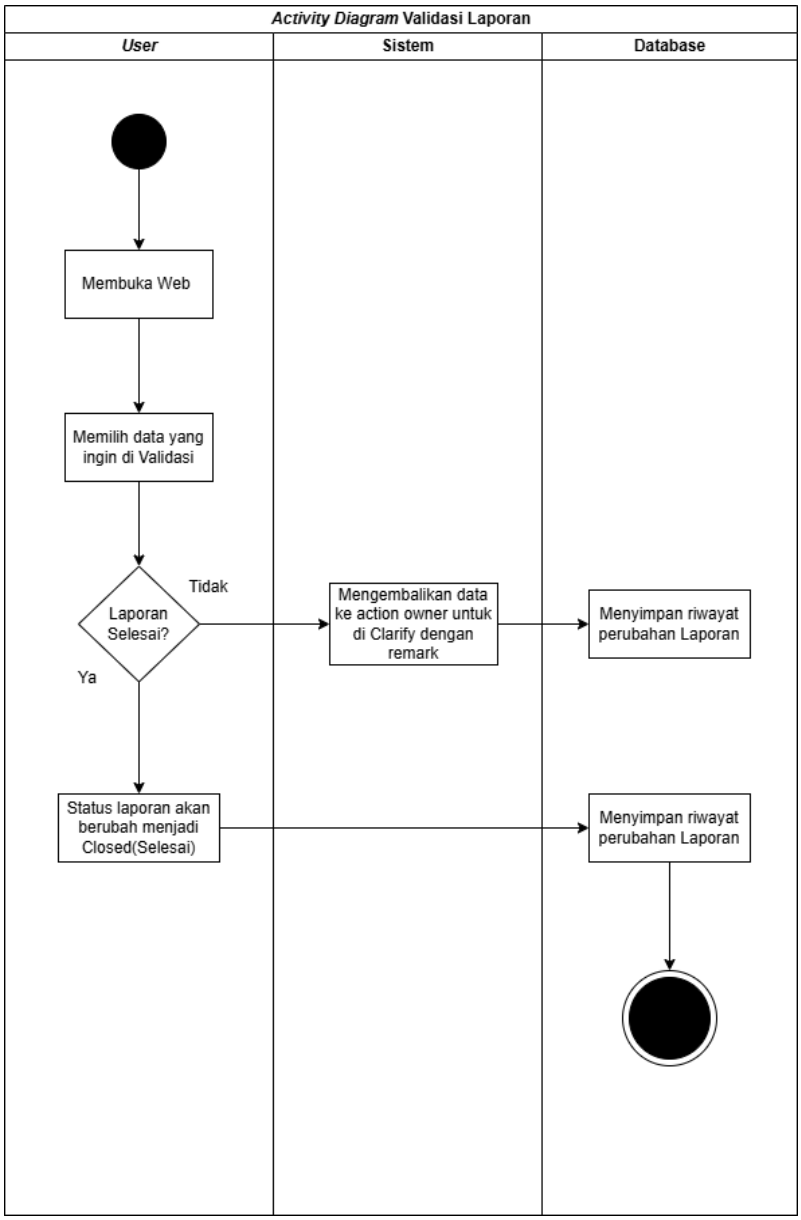
e) *Activity Diagram* Mengisi Data Perbaikan



Gambar 3.9 *Activity Diagram* Mengisi Data Perbaikan

Gambar 3.9 menjelaskan proses pengisian data perbaikan oleh *User*. Setelah membuka *web* dan memilih data, *User* mengisi form perbaikan dan sistem memastikan semua data terisi. Data kemudian disimpan ke *database*; jika penyimpanan berhasil, data akan diteruskan ke *Validator*, sedangkan jika gagal, *User* akan diminta untuk mengisi ulang form.

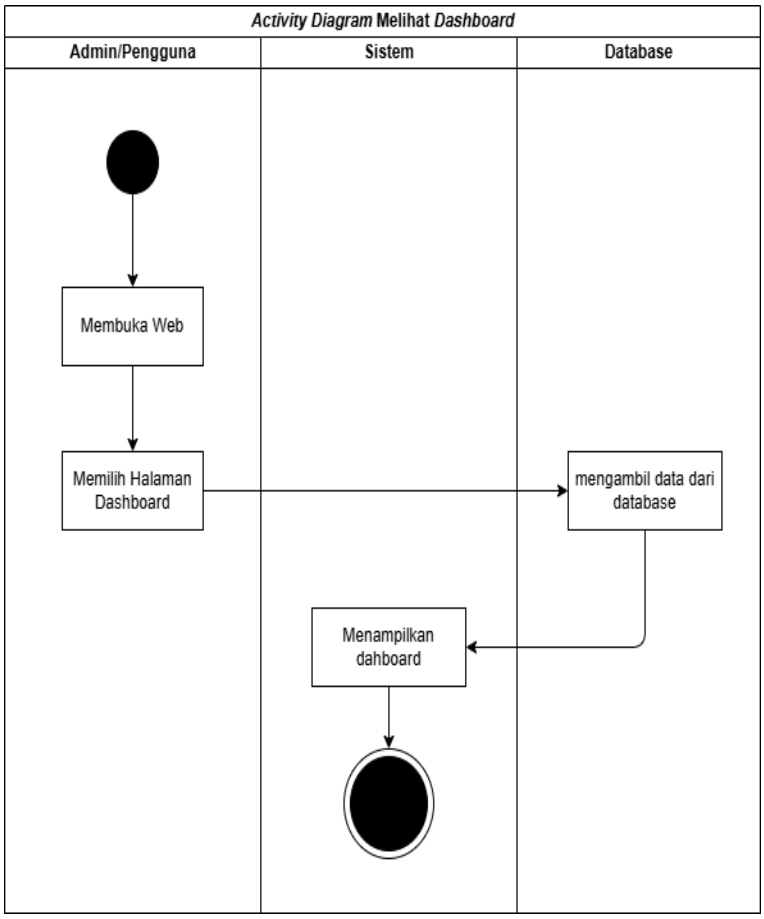
f) Activity Diagram Validasi Laporan



Gambar 3.10 *Activity Diagram* Validasi Laporan

Gambar 3.10 menjelaskan proses validasi laporan oleh *User*. *User* memilih data yang ingin divalidasi dan menentukan apakah laporan sudah selesai. Jika ya, status laporan berubah menjadi *Closed* (Selesai). Jika tidak, data dikembalikan ke *Action Owner* untuk diperjelas (*clarify*) dengan catatan (*remark*). *Database* akan menyimpan riwayat perubahan untuk kedua kondisi tersebut.

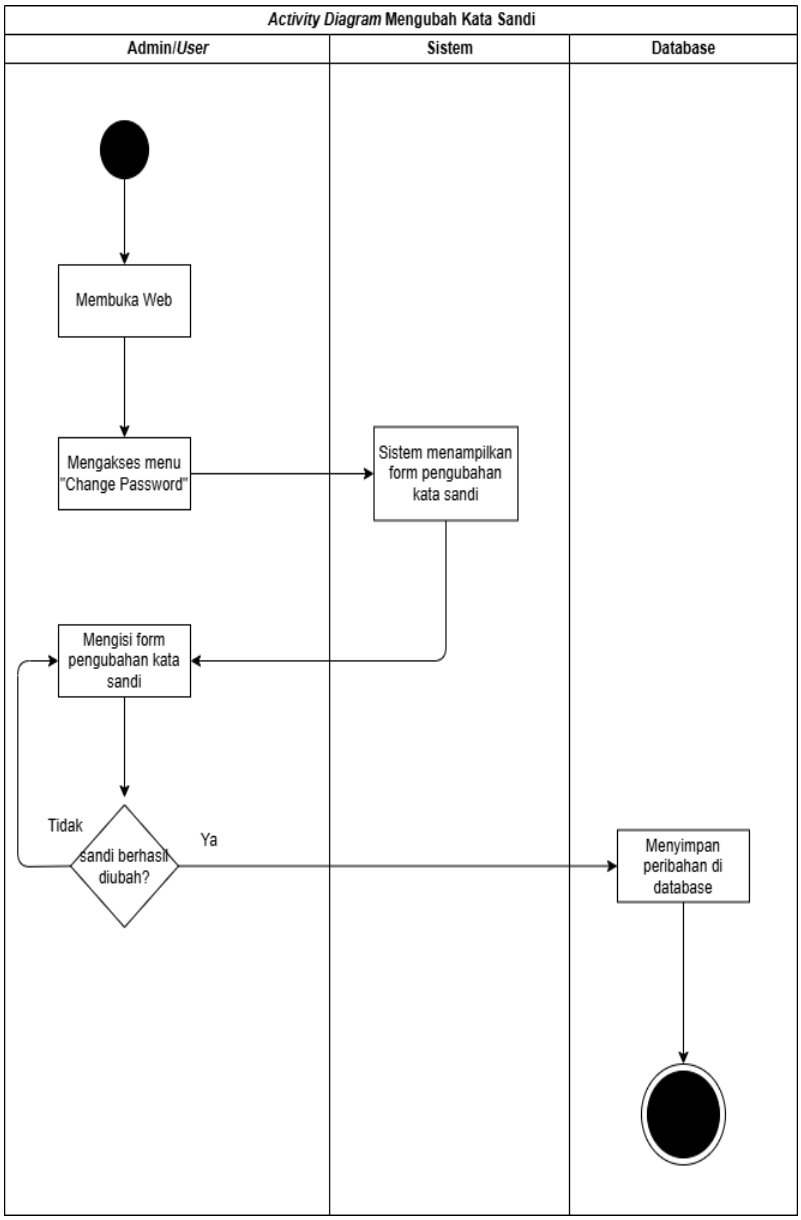
g) *Activity Diagram* Melihat *Dashboard* Analisis



Gambar 3.11 *Activity Diagram* Melihat *Dashboard* Analisis

Gambar 3.11 menjelaskan alur proses untuk melihat halaman *dashboard* analisis, di mana Admin atau *User* membuka *web* lalu memilih halaman *dashboard*. Selanjutnya, sistem akan mengambil data yang diperlukan dari *database* dan menampilkannya pada *dashboard* berupa *chart* agar dapat dilihat oleh *User*.

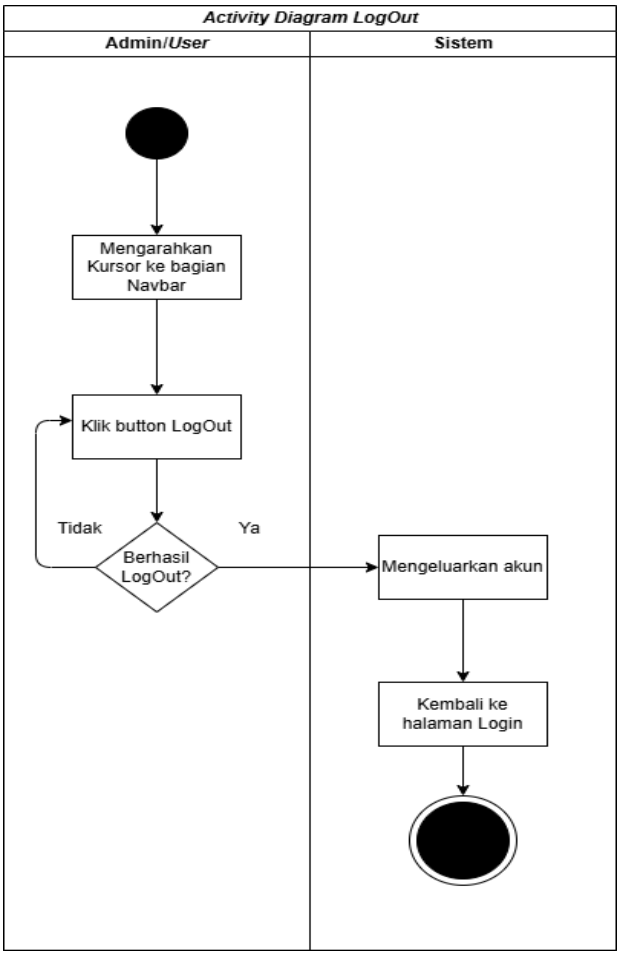
h) *Activity Diagram* Mengubah Kata Sandi



Gambar 3.12 *Activity Diagram* Mengubah Kata Sandi

Gambar 3.12 menjelaskan alur perubahan kata sandi. Admin atau *User* mengakses menu "*Change Password*", lalu sistem menampilkan form perubahan yang harus diisi. Jika sandi berhasil diubah, perubahan tersebut akan disimpan ke *database*. Namun, jika proses gagal, pengguna akan diarahkan kembali untuk mengisi form pengubahan kata sandi.

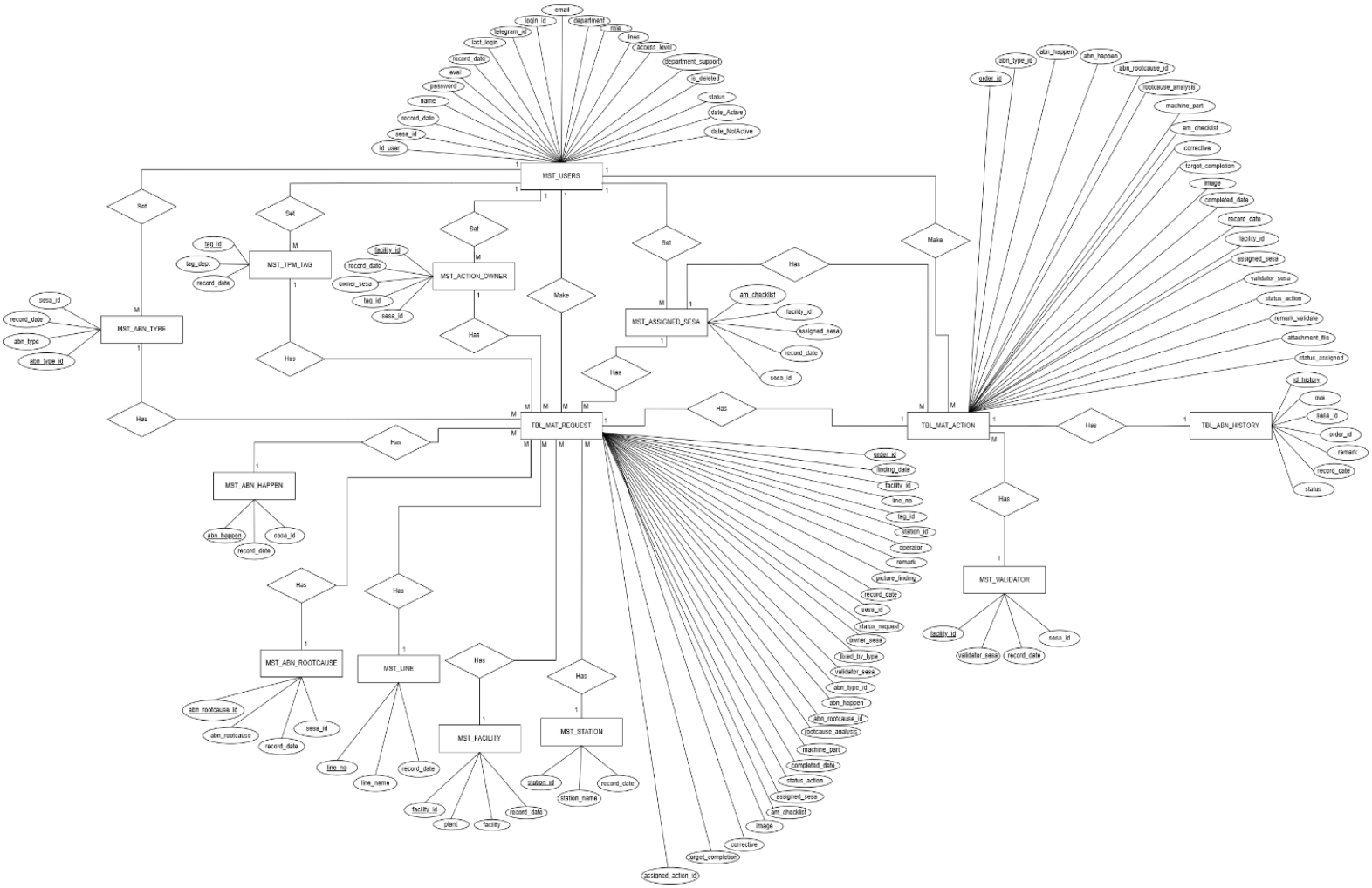
i) *Activity Diagram Logout*



Gambar 3.13 *Activity Diagram Logout*

Gambar 3.13 menjelaskan proses keluar dari sistem (*logout*). Admin atau *User* mengarahkan kursor ke *navbar* dan mengklik tombol *Logout*. Jika proses *logout* berhasil dikonfirmasi, sistem akan mengeluarkan akun dari sesi aktif dan mengembalikan tampilan ke halaman *Login*.

3.2.6 ER Diagram



Gambar 3.14 *Entity Relationship Diagram Machine Abnormality*

ER Diagram di atas merupakan *Entity Relationship Diagram* (ERD) dari sistem *Machine Abnormality*, yang berfungsi sebagai rancangan struktur basis data untuk mendukung proses pelaporan, pengelolaan, dan pelacakan *Machine abnormality.* ERD ini menggambarkan hubungan antar entitas utama yang terlibat dalam sistem, mulai dari pengguna (*users*), *request abnormality*, aksi penanganan, hingga riwayat dan validasi. Setiap entitas dirancang untuk merepresentasikan objek nyata dalam proses bisnis *Machine Abnormality*, sehingga sistem mampu menyimpan data secara terstruktur, konsisten, dan terintegrasi.

Selain itu, ERD ini menyediakan pencatatan histori dan jejak audit untuk memastikan setiap perubahan dan aktivitas dapat ditelusuri sebagai bahan evaluasi dan pelaporan manajemen. Berikut adalah penjelasan dari setiap entitasnya:

1. MST_USERS

   MST_USERS merupakan tabel *master* yang menyimpan seluruh data pengguna sistem P1F WIP AM PM. Entitas ini menjadi pusat dari seluruh aktivitas karena setiap proses selalu melibatkan *user*. Berdasarkan ERD, satu *user* dapat membuat banyak laporan *abnormality* yang tersimpan pada TBL_MAT_REQUEST. Selain itu, *user* yang sama juga dapat ditetapkan sebagai *action owner* pada MST_ACTION_OWNER, sebagai *assigned action* pada MST_ASSIGNED_SESA, maupun sebagai *validator* pada MST_VALIDATOR. Dalam proses transaksi, *user* juga terlibat langsung pada TBL_MAT_ACTION sebagai pelaksana atau penanggung jawab, serta tercatat pada TBL_ABN_HISTORY sebagai bagian dari histori aktivitas. Dengan demikian, satu *user* dapat memiliki peran yang berbeda- beda tergantung pada alur penanganan abnormality yang berjalan.
2. MST_TPM_TAG

   MST_TPM_TAG merupakan tabel *master* yang menyimpan data *tpm tag* sebagai klasifikasi aktivitas atau *area* TPM. Berdasarkan relasi pada ERD, satu TPM tag dapat digunakan oleh banyak data pada MST_ACTION_OWNER untuk menentukan *action owner,* serta direferensikan oleh banyak data pada TBL_MAT_REQUEST dan TBL_MAT_ACTION. *tpm tag* berperan penting karena menjadi salah satu dasar utama sistem dalam menentukan pihak yang bertanggung jawab atas penanganan *abnormality*.
3. MST_ACTION_OWNER

   MST_ACTION_OWNER menyimpan data *user* yang ditetapkan sebagai *action owner* berdasarkan kombinasi *facility* dan *tpm tag*. Setiap data pada tabel ini mengacu pada satu *user* di MST_USERS, satu *facility* di MST_FACILITY, dan satu *tpm tag* di MST_TPM_TAG. Relasi ini memungkinkan sistem untuk secara otomatis menentukan *action owner*

ketika sebuah *request abnormality* dibuat. Pada alur utama, *action owner* berperan sebagai penanggung jawab awal, sedangkan pada alur *fixed by myself, action owner* juga menjalankan peran sebagai *validator* akhir sesuai dengan *mapping* yang ada.

4. MST_ASSIGNED_SESA

   MST_ASSIGNED_SESA merupakan tabel *master* yang menyimpan data *user* yang dapat ditugaskan sebagai pelaksana *action*. Berdasarkan ERD, setiap data *assigned sesa* mengacu pada satu *user* di MST_USERS dan dapat digunakan oleh banyak data pada TBL_MAT_ACTION. Pada alur utama, *assigned sesa* menerima penugasan dari *action owner* untuk melakukan perbaikan, sedangkan pada alur *fixed by myself*, peran *assigned sesa* dapat diisi oleh *user* yang sama dengan *requestor*.

5. MST_VALIDATOR

   MST_VALIDATOR menyimpan data *user* yang memiliki kewenangan sebagai *validator* resmi dalam sistem. Setiap data pada MST_VALIDATOR mengacu pada satu *user* di MST_USERS dan dapat digunakan oleh banyak data pada TBL_MAT_ACTION. *Validator* ini hanya digunakan pada alur utama untuk melakukan validasi akhir penyelesaian *abnormality*, sedangkan pada alur *fixed by myself* peran validasi dialihkan kepada action owner sesuai *mapping facility* dan *tpm tag*.

6. TBL_MAT_REQUEST

   TBL_MAT_REQUEST merupakan tabel transaksi utama yang menyimpan data laporan *abnormality* yang dibuat oleh *user*. Setiap *request* mengacu pada satu *user* sebagai *requestor*, satu *facility*, satu *line*, satu *station*, serta satu *tpm tag*. Berdasarkan ERD, satu *request* dapat memiliki banyak data *action* pada TBL_MAT_ACTION dan banyak histori pada TBL_ABN_HISTORY. Tabel ini menjadi titik awal seluruh proses penanganan *abnormality*, baik pada alur utama maupun alur *fixed by myself.*

7. TBL_MAT_ACTION

   TBL_MAT_ACTION merupakan tabel transaksi yang menyimpan detail tindakan penanganan terhadap *abnormality* yang berasal dari suatu *request*.

Setiap *action* mengacu pada satu *request* di TBL_MAT_REQUEST, satu *action owner,* satu *assigned sesa,* serta satu *validator*. Selain itu, *action* juga dikaitkan dengan klasifikasi *abnormality* melalui MST_ABN_TYPE, kondisi kejadian melalui MST_ABN_HAPPEN, dan penyebab masalah melalui MST_ABN_ROOTCAUSE. Satu *action* dapat memiliki banyak histori pada TBL_ABN_HISTORY, sehingga progres penanganan dapat dipantau secara menyeluruh hingga dinyatakan selesai dan tervalidasi.

8. TBL_ABN_HISTORY

   TBL_ABN_HISTORY digunakan untuk mencatat seluruh riwayat aktivitas dan perubahan status *abnormality*. Setiap histori mengacu pada satu *request*, satu *action*, dan satu *user* yang melakukan aktivitas tersebut. Berdasarkan ERD, satu *request* dan satu *action* dapat memiliki banyak histori, sehingga tabel ini berfungsi sebagai audit *trail* dan pendukung *traceability* dalam sistem.

9. MST_ABN_TYPE

   MST_ABN_TYPE merupakan tabel *master* yang menyimpan data jenis atau kategori *abnormality*. Satu jenis *abnormality* dapat digunakan oleh banyak data pada TBL_MAT_ACTION, sehingga tabel ini berfungsi untuk standarisasi klasifikasi *abnormality* dan mendukung analisis data.

10. MST_ABN_HAPPEN

    MST_ABN_HAPPEN menyimpan data kondisi atau bentuk kejadian *abnormality*. Berdasarkan relasi pada ERD, satu data pada tabel ini dapat digunakan oleh banyak *action* di TBL_MAT_ACTION. Entitas ini membantu menjelaskan bagaimana *abnormality* terjadi di lapangan.

11. MST_ABN_ROOTCAUSE

    MST_ABN_ROOTCAUSE menyimpan data penyebab utama terjadinya *abnormality*. Setiap *root cause* dapat direferensikan oleh banyak data pada TBL_MAT_ACTION, sehingga tabel ini berperan penting dalam analisis penyebab masalah dan perencanaan perbaikan berkelanjutan.

12. MST_LINE

    MST_LINE merupakan tabel *master* yang menyimpan data *line* produksi. Berdasarkan ERD, satu *line* dapat memiliki banyak *station* pada MST_STATION dan banyak request *abnormality* pada TBL_MAT_REQUEST. Entitas ini digunakan untuk mengelompokkan *abnormality* berdasarkan area produksi secara umum.

13. MST_STATION

    MST_STATION menyimpan data *station* atau titik kerja yang berada dalam suatu *line*. Setiap station mengacu pada satu *line* dan dapat digunakan oleh banyak *request* pada TBL_MAT_REQUEST. Dengan adanya tabel ini, lokasi terjadinya *abnormality* dapat diidentifikasi secara lebih spesifik hingga *level station*.

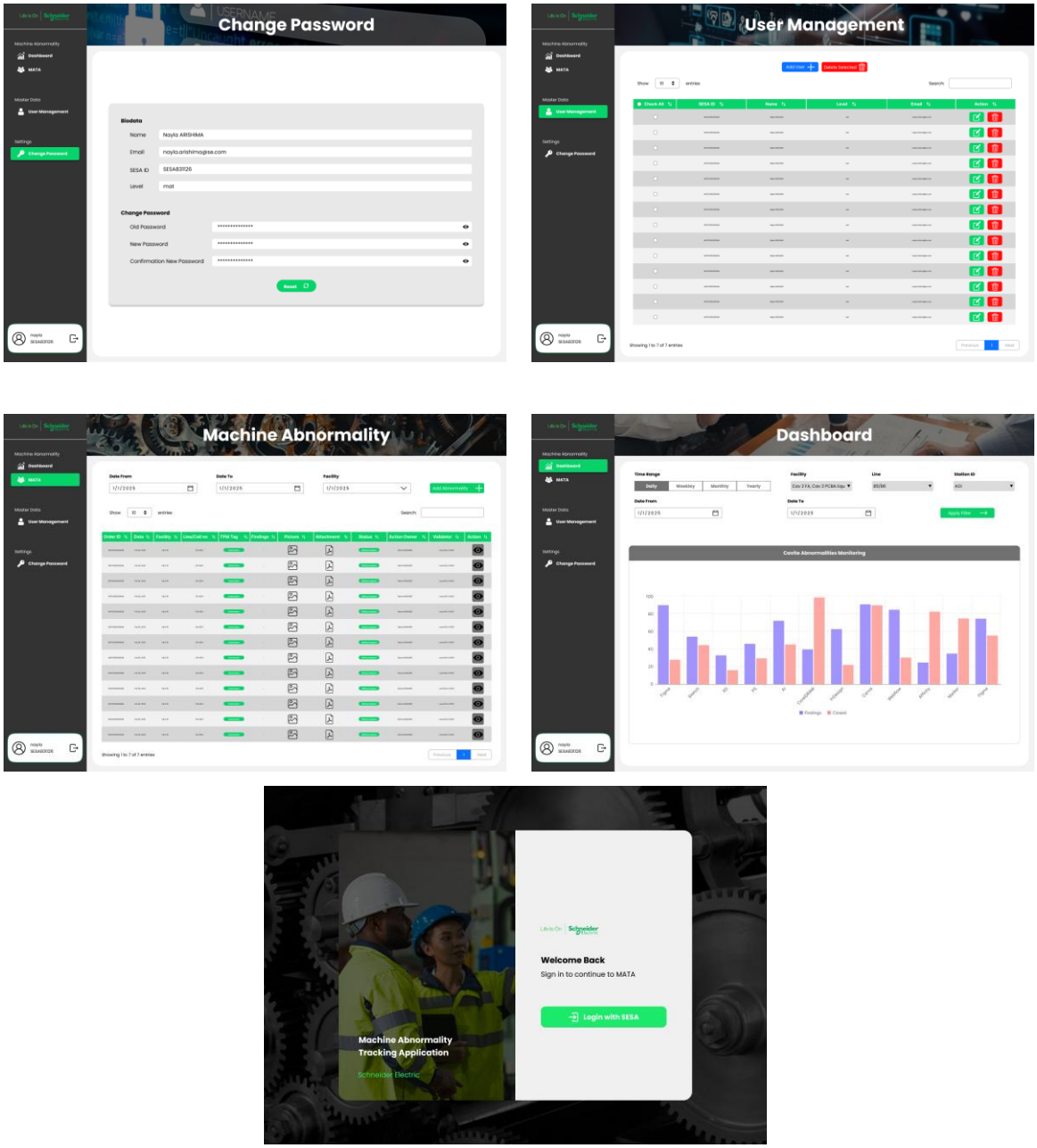
14. MST_FACILITY

    MST_FACILITY merupakan tabel *master* yang menyimpan data *facility* atau area kerja. Satu *facility* dapat memiliki banyak *mapping action owner* pada MST_ACTION_OWNER dan banyak *request abnormality* pada TBL_MAT_REQUEST. *Facility* menjadi salah satu elemen utama dalam penentuan alur penanganan *abnormality* serta penentuan *action owner* yang bertanggung jawab.

3.2.7 *Wireframe*

Gambar 3.15 *Wireframe* Sistem *Machine Abnormality*

3.2.8 *Desain* UI



Gambar 3.16 Desain Antarmuka *Machine Abnormality*

BAB IV IMPLEMENTASI DAN PENGUJIAN

4.1 Hasil Implementasi

Proses implementasi sistem pelaporan abnormalitas mesin dilakukan setelah tahap analisis dan perancangan selesai, dengan tujuan membangun aplikasi berbasis web yang sesuai dengan kebutuhan di lapangan. Sistem dikembangkan menggunakan bahasa pemrograman C# untuk logika backend, sementara SQL *Server* digunakan sebagai basis data utama untuk menyimpan informasi transaksi dan detail perbaikan. Skema *database* mencakup tabel TBL_MAT_REQUEST yang mencatat setiap laporan temuan beserta statusnya, dan tabel TBL_MAT_ACTION yang menyimpan rincian tindakan perbaikan. Hubungan antar tabel diatur menggunakan *primary key* dan *foreign key* untuk menjaga integritas data serta menghubungkan entitas pengguna, lokasi (*facility*), dan klasifikasi abnormalitas.

Sistem mengikuti dua alur kerja utama yang telah ditetapkan sebelumnya. Alur utama melalui *Action Owner* dimulai ketika *Requestor* membuat laporan temuan di sistem. Laporan tersebut diteruskan ke *Action Owner*, yang bertugas menentukan *Assigned Action* untuk menangani masalah tersebut. Setelah *Assigned Action* menyelesaikan perbaikan, laporan dikirim ke *Validator* untuk diverifikasi. Jika *Validator* menemukan ketidaksesuaian atau perbaikan belum lengkap, laporan dikembalikan ke *Action Owner* untuk ditindaklanjuti.

Sedangkan alur *Fixed by Myself* memungkinkan *Requestor* menangani masalah ringan secara mandiri. Dalam alur ini, *Requestor* langsung mengisi laporan perbaikan dan mengunggah bukti tindakan yang telah dilakukan, kemudian laporan dikirim ke *Validator* untuk diverifikasi. Jika *Validator* menolak laporan, tindakan perbaikan tetap diteruskan ke *Action Owner* agar sesuai prosedur. Alur ini mempercepat penyelesaian masalah ringan tanpa mengganggu penanganan masalah yang lebih kompleks.

Di sisi frontend, antarmuka dikembangkan agar sesuai dengan peran pengguna dan alur kerja yang dipilih. Sistem menampilkan form yang memudahkan pengguna dalam mengisi laporan, melampirkan bukti, dan memantau status

temuan. Untuk alur *Fixed by Myself*, kolom perbaikan dan unggah bukti langsung ditampilkan, sedangkan untuk alur utama, informasi mengenai *Action Owner* dan *Assigned Action* ditampilkan secara otomatis berdasarkan konfigurasi *master facility*, memastikan distribusi tugas berjalan sesuai alur baku.

Sebagai contoh kasus nyata, misalkan sebuah mesin produksi di lantai 2 mengalami kerusakan. *Requestor* membuat laporan melalui sistem dan memilih alur utama. Laporan diteruskan ke *Action Owner* B, yang menugaskan *Assigned* C untuk melakukan perbaikan. Setelah perbaikan selesai, laporan dikirim ke *Validator* D untuk diverifikasi. Jika *Validator* D menilai perbaikan sudah sesuai, laporan ditutup (*Closed*). Namun jika *Validator* menemukan data tidak lengkap atau perbaikan belum optimal, laporan dikembalikan ke *Action Owner* B untuk tindakan lebih lanjut. Dengan cara ini, semua laporan diproses secara sistematis, terdokumentasi dengan baik, dan setiap langkah dapat dilacak.

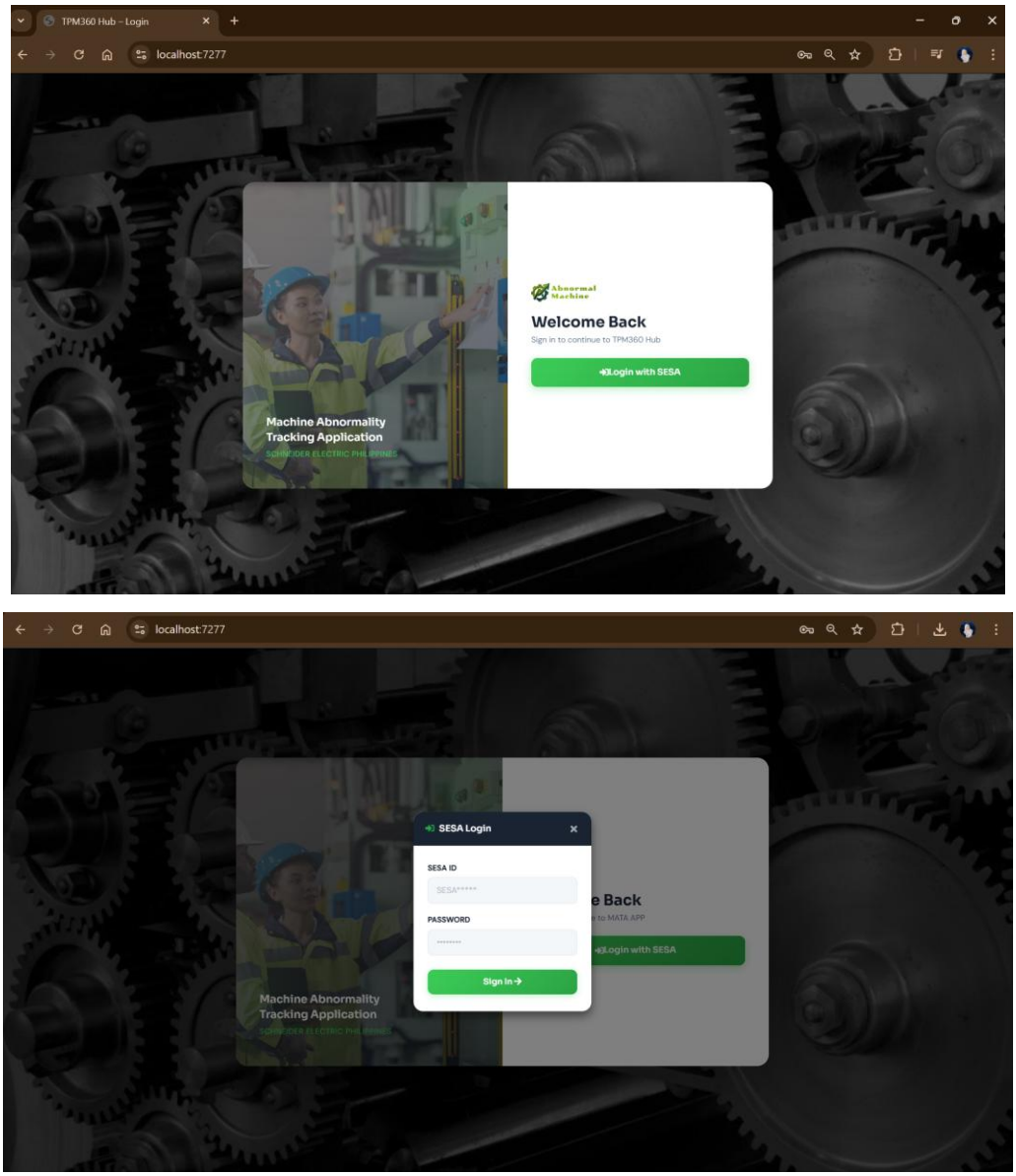
Seluruh sistem di-*deploy* pada *server* lokal perusahaan dan dapat diakses melalui web browser tanpa instalasi tambahan, dengan pengaturan akses yang aman melalui jaringan internal. Aktivitas transaksi dicatat pada *Activity Log*, sehingga memudahkan *monitoring real-time* dan menjamin transparansi proses pelaporan. Dengan implementasi ini, sistem pelaporan abnormalitas mesin berjalan sesuai alur yang telah ditetapkan, memudahkan pemantauan dan validasi, serta mendukung pengambilan keputusan yang lebih efektif dalam manajemen perawatan mesin, sekaligus mengurangi risiko *downtime* yang dapat memengaruhi produktivitas perusahaan.

Selain itu, sistem ini menerapkan *role-based access control* sehingga setiap pengguna hanya mengakses fitur sesuai perannya. Data laporan dikelola secara *real-time* dan terstruktur, dilengkapi validasi *input* untuk meminimalisir kesalahan. Pengujian fungsi CRUD dan alur kerja menunjukkan sistem berjalan dengan baik tanpa kendala kritis. Dengan demikian, sistem ini meningkatkan efisiensi, transparansi, serta mendukung pengambilan keputusan dan mengurangi potensi *downtime*. Sistem ini juga memberikan kemudahan dalam proses *monitoring* dan evaluasi secara berkelanjutan terhadap setiap laporan yang masuk.

4.1.1 Implementasi *User Interface* (UI)

Pada subbab ini menjelaskan hasil implementasi pada *User Interface* (UI) serta keterangan pada setiap halaman.

1. *Home*

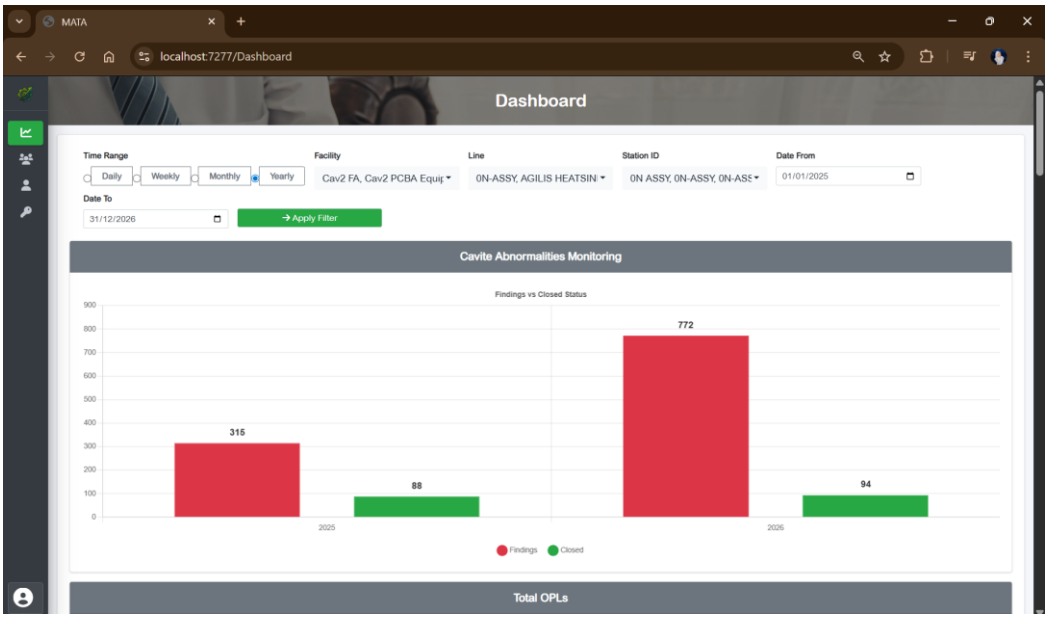


Gambar 4. 1 Implementasi *Home page*

Implementasi antarmuka halaman utama dan proses autentikasi dirancang dengan konsep *split-screen* yang memadukan visual industrial dengan fungsionalitas sistem. Latar belakang halaman menggunakan gambar komponen

mesin dalam nuansa hitam putih untuk menjaga kontras agar elemen interaksi utama tetap menonjol. Di bagian tengah, terdapat kartu interaksi yang memisahkan identitas aplikasi dan akses masuk aktor. Proses *login* dimulai ketika aktor menekan tombol masuk, yang kemudian memicu munculnya jendela *pop-up* (modal) di tengah layar dengan efek latar belakang meredup. Pada tahap ini, aktor wajib memasukkan kredensial berupa SESA ID dan kata sandi pada kolom inputan yang tersedia sebelum menekan tombol konfirmasi untuk memvalidasi identitas ke dalam sistem.

2. *Dashboard*

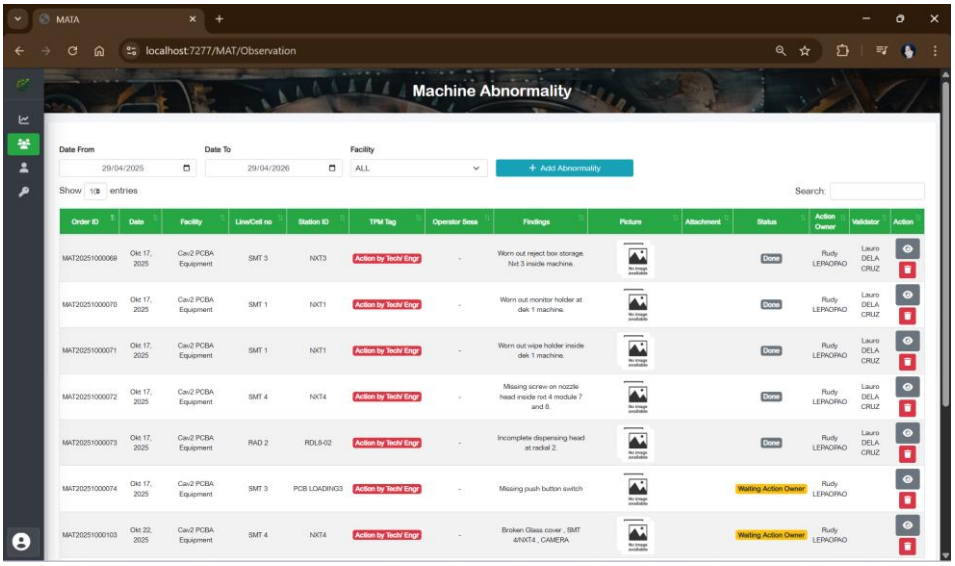


Gambar 4. 2 Implementasi *Dashboard Machine Abnormality*

Halaman Dashboard merupakan antarmuka pusat yang berfungsi untuk menyajikan visualisasi data abnormalitas mesin secara *real-time*. Pada bagian atas, sistem menyediakan fitur *filter* komprehensif yang mencakup rentang waktu, fasilitas, hingga lini produksi untuk memudahkan aktor dalam melakukan analisis data spesifik. Visualisasi data disajikan melalui berbagai diagram interaktif, mulai dari perbandingan temuan (*findings*) dan status penyelesaian, grafik tren *Total OPLs*, hingga diagram lingkaran yang merepresentasikan kontribusi abnormalitas per lokasi kerja. Selain itu, *dashboard* ini mengintegrasikan data kategorisasi TPM, akar penyebab masalah (*root cause*), serta proyeksi dampak risiko untuk

mendukung pengambilan keputusan strategis oleh aktor *administrator* maupun teknisi.

3. MATA



Gambar 4. 3 Implementasi *MATA Page*

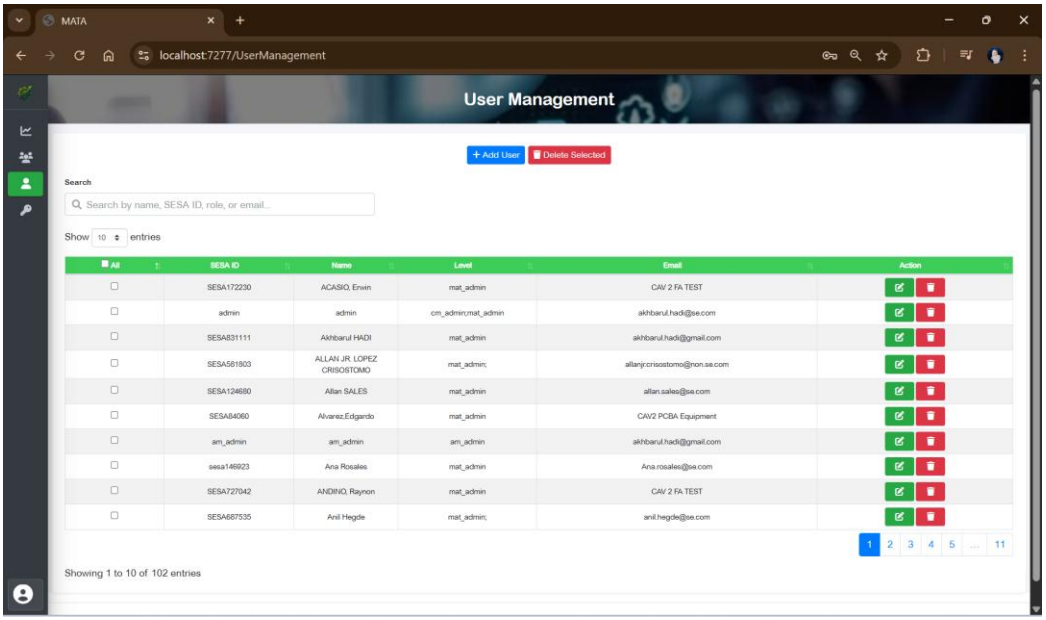
Halaman *Machine Abnormality* dirancang sebagai pusat pengelolaan laporan temuan penyimpangan mesin di area operasional. Antarmuka ini dilengkapi dengan fitur filter tanggal dan fasilitas, serta tombol aksi untuk menambahkan laporan baru ke dalam *database*. Seluruh data laporan ditampilkan secara terstruktur dalam tabel yang mencakup *Order ID*, klasifikasi TPM *tag*, deskripsi temuan, hingga pratinjau lampiran gambar. Setiap baris data memuat informasi mengenai status laporan dan penunjukan pemilik tindakan (*action owner*) serta *validator*, sehingga aktor dapat memantau progres perbaikan secara transparan dan akuntabel melalui fitur aksi yang tersedia.

Selain itu, halaman ini juga menyediakan informasi detail terkait lokasi temuan melalui kolom *Line/Cell* dan *Station ID*, yang membantu pengguna dalam mengidentifikasi titik permasalahan secara lebih spesifik di area produksi. Data *Operator SESA* turut ditampilkan untuk menunjukkan pihak yang melakukan pelaporan, sehingga memudahkan proses koordinasi dan klarifikasi apabila diperlukan. Fitur *search* yang tersedia memungkinkan pengguna untuk melakukan

pencarian data secara cepat berdasarkan kata kunci tertentu, sehingga meningkatkan efisiensi dalam pengelolaan laporan.

Sistem ini juga mendukung pengelolaan status laporan secara bertahap, dimulai dari kondisi awal seperti *Waiting Assigned* hingga tahap penanganan lebih lanjut. Mekanisme penentuan *action owner* dan *validator* dilakukan secara terstruktur berdasarkan fasilitas dan identitas mesin (*tag ID*), sehingga proses distribusi tanggung jawab menjadi lebih tepat dan konsisten. Dengan adanya kolom *Action*, pengguna dapat melakukan interaksi langsung terhadap data, seperti melihat detail laporan atau melakukan tindak lanjut yang diperlukan.

4. *User Management*

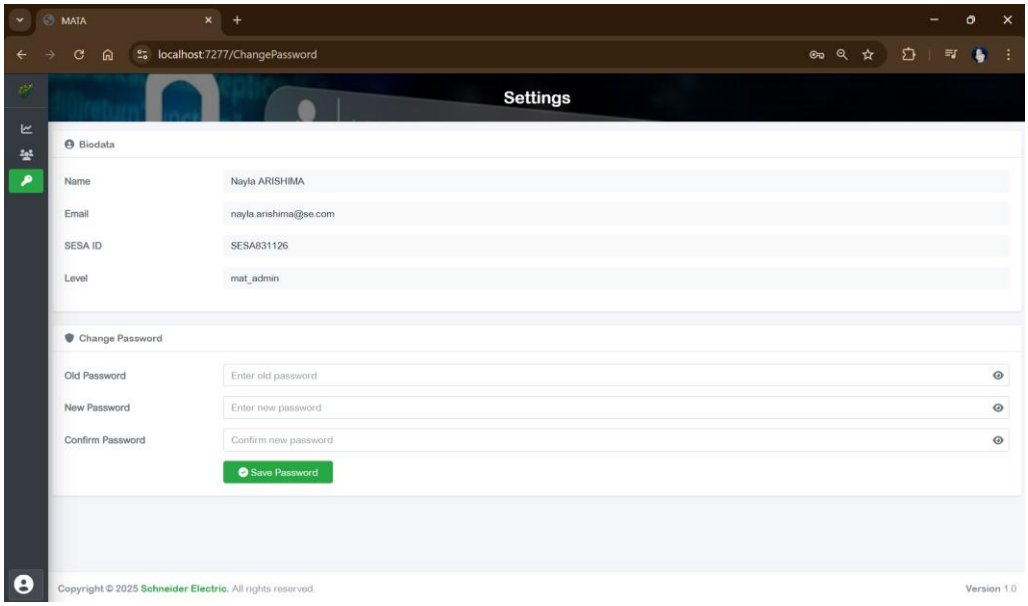


Gambar 4. 4 Implementasi halaman *User Management*

Implementasi halaman *User Management* berfungsi sebagai alat kendali administratif untuk mengelola hak akses pengguna secara terpusat. Aktor administrator dapat mendaftarkan pengguna baru melalui fitur *Add User* atau melakukan penghapusan data secara massal. Sistem menyediakan kolom pencarian yang fleksibel untuk memfilter data berdasarkan nama atau SESA ID guna mempercepat manajemen informasi. Tabel pada halaman ini memuat detail identitas, level akses, dan email pengguna, di mana setiap entri data dilengkapi dengan tombol aksi untuk melakukan pembaruan informasi (edit) atau penghapusan

secara instan, didukung dengan fitur navigasi halaman (*pagination*) untuk efisiensi pengelolaan data berskala besar.

5. *Change Password*



Gambar 4. 5 Implementasi halaman *Change Password*

Halaman pengaturan kata sandi menyediakan fitur keamanan mandiri bagi aktor untuk memperbarui kredensial akses mereka. Antarmuka ini menampilkan informasi biodata pengguna secara *read-only* guna memastikan validitas akun yang sedang aktif. Di bawah bagian biodata, terdapat formulir pembaruan yang mewajibkan aktor memasukkan kata sandi lama sebagai bentuk validasi keamanan, diikuti dengan penginputan kata sandi baru dan konfirmasinya. Setelah validasi inputan sesuai, aktor dapat menyimpan perubahan ke *database* melalui tombol konfirmasi. Seluruh mekanisme ini dibuat untuk menjamin privasi dan integritas akun pengguna di dalam *system*.

4.2 Pengujian *User Acceptance Testing* (UAT)

4.2.1 Fungsional *Testing*

Pada tahap ini dilakukan pengujian terhadap fungsional sistem untuk memastikan setiap fitur pada perangkat lunak dapat berjalan sesuai dengan kebutuhan serta skenario bisnis yang telah ditentukan sebelumnya. Proses

pengujian dilakukan menggunakan metode *black box* testing, yaitu teknik pengujian yang menitik beratkan pada kesesuaian antara input yang diberikan dengan *output* yang dihasilkan tanpa memperhatikan struktur atau kode program di dalam sistem. Dalam penelitian ini terdapat 9 fungsional utama yang diuji. Hasil pengujian fungsional menggunakan metode *black box* dapat dilihat pada tabel-tabel berikut.

1. *Black Box Testing Scenario Use Case* Login

   Black box testing ini berfokus pada pengujian fungsional login menggunakan *black box testing*.

   Tabel 4.1 *Black Box Testing Scenario Use Case Login*

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Succes Scenario	1	Admin dan *User* membuka web.	Halaman *login* berhasil ditampilkan.	Halaman *login* berhasil ditampilkan sesuai desain.	✓	
	2	Admin dan *User* memasukan *Username* dan *Password*.	Input *username* dan *password* dapat diisi tanpa error.	*Field username* dan *password* dapat diisi dengan baik.	✓	
	3	Admin dan *User* menekan tombol "*Log in*".	Sistem menerima *input* dan memproses *login*.	Sistem berhasil memproses data *login*.	✓	
	4	Sistem memvalidasi data yang dimasukkan aktor.	Sistem melakukan validasi kecocokan *username* dan *password* dengan *database*.	Sistem berhasil memvalidasi data sesuai dengan *database*.	✓	
	5	Jika data valid, sistem mengarahkan Admin dan Pengguna ke halaman *Observation*.	User diarahkan ke halaman *Observation*.	*User* berhasil diarahkan ke halaman *Observation*.	✓	

Alternative Flow (Extension)	1	Admin dan *User* memasukan *Username* dan *Password* yang salah.	Sistem menampilkan pesan kesalahan "*User and Password not Registered"* dan meminta pengguna untuk mengisi ulang data yang benar.	Sistem menampilkan pesan error sesuai dan *user* tetap di halaman *login*.	✓	

2. *Black box Testing Scenario Use Case* Melihat Riwayat Laporan

   Black box testing ini berfokus pada pengujian fungsional melihat riwayat laporan menggunakan *black box testing*.

Tabel 4.2 *Black box Testing Scenario Use Case* Melihat Riwayat Laporan

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**
Main Succes Scenario	1	Admin dan *User* memilih satu laporan yang ingin dilihat riwayat dan statusnya.	Sistem menampilkan daftar laporan yang tersedia, dan *user* dapat memilih salah satu laporan tanpa terjadi error atau *loading* yang gagal.	Sistem menampilkan riawayat laporan, dan *user* berhasil memilih satu laporan yang diinginkan.	✓	
	2	Admin dan *User* menekan tombol mata pada laporan.	Sistem menerima aksi klik tombol "*view*" (ikon mata), kemudian memproses permintaan untuk menampilkan riwayat/status laporan yang dipilih.	Sistem berhasil menerima *input* klik dan mulai memproses permintaan untuk mengambil data riwayat/status laporan.	✓	
	3	Sistem menampilkan detail riwayat/status laporan.	Sistem menampilkan informasi riwayat/status laporan secara lengkap	Sistem berhasil menampilkan detail riwayat/status	✓	

			(misalnya: status, tanggal update, PIC, keterangan) dengan tampilan yang sesuai dan mudah dibaca.	laporan secara lengkap dan sesuai dengan data yang tersimpan.		
Alternative Flow (Extension)	1	Internet terputus saat mengambil data riwayat/status.	Sistem menampilkan pesan kesalahan "*An error occurred while loading the report history. Please try again later*."	Sistem menampilkan pesan kesalahan sesuai, dan data riwayat/status tidak ditampilkan.	✓	

3. *Black box Testing Scenario Use Case* Mengelola Data *User*

   Black box testing ini berfokus pada pengujian fungsional mengelola data *user* menggunakan *black box testing*.

Tabel 4.3 *Black box Testing Scenario Use Case* Mengelola Data User

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Succes Scenario	1	Admin memilih menu "*User Management*"	Sistem menampilkan halaman *User Management.*	Halaman *User Management* berhasil ditampilkan.	✓	
	2	Sistem menampilkan daftar data pengguna dalam bentuk tabel	Sistem menampilkan data user dalam bentuk tabel lengkap (misalnya: nama, *username, role,* dll).	Data user berhasil ditampilkan dalam bentuk tabel sesuai data yang tersedia.	✓	
	3	Admin memilih aksi (*Add User / Edit / Delete*)	Sistem merespon aksi yang dipilih admin dan menampilkan form atau konfirmasi sesuai aksi (tambah, ubah, hapus).	Sistem berhasil menampilkan form *input* (*add/Edit*) atau dialog konfirmasi (*Delete*).	✓	

	3a	Admin memilih *"Add User" (Create)*	Sistem menampilkan form untuk menambahkan *user* baru.	Form *add user* berhasil ditampilkan.	✓	
	3b	Admin memilih "*Edit*" (*Update*)	Sistem menampilkan form edit dengan data user yang sudah terisi sebelumnya.	Form edit berhasil ditampilkan dengan data yang sesuai.	✓	
	3c	Admin memilih "*Delete*" (*Delete*)	Sistem menampilkan konfirmasi penghapusan data *user*.	Dialog konfirmasi *delete* berhasil ditampilkan.	✓	
	4	Admin mengisi atau memperbarui informasi data *user*	*Field input* dapat diisi atau diperbarui dengan data yang valid tanpa error.	Data berhasil diinput atau diperbarui oleh admin.	✓	
	5	Admin menyimpan perubahan data	Sistem menerima input dan memproses penyimpanan data.	Sistem berhasil memproses penyimpanan data.	✓	
	6	Sistem memvalidasi data yang diinput	Sistem melakukan validasi (misalnya *field* wajib, format email, dll).	Sistem berhasil melakukan validasi data.	✓	
	7	Sistem menyimpan data ke *database* dan memperbarui tampilan	Data tersimpan ke *database* dan tabel *User Management* otomatis ter-*update* sesuai perubahan (tambah/edit/delete).	Data berhasil disimpan dan tabel *User Management* ter-*update* sesuai perubahan.	✓	
Alternative Flow (Extension)	1	Data tidak lengkap atau tidak valid	Sistem menampilkan pesan validasi (misalnya: "*Field* tidak boleh kosong" atau "Format	Sistem menampilkan pesan validasi sesuai dan admin diminta memperbaiki data.	✓	

			tidak valid") dan tidak menyimpan data.			

4. *Black box Testing Scenario Use Case* Membuat Laporan Abnormalitas

   Black box testing ini berfokus pada pengujian fungsional membuat laporan abnormalitas menggunakan *black box testing*.

Tabel 4.4 *Black box Testing Scenario Use Case* Membuat Laporan Abnormalitas

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Succes Scenario	1	Admin atau *User* memilih menu "*Add New Abnormality*"	Sistem menampilkan halaman/form pembuatan laporan abnormalitas.	Halaman form pembuatan laporan abnormalitas berhasil ditampilkan.	✓	
	2	Sistem menampilkan form pembuatan laporan abnormalitas	Form berisi *field* yang lengkap (waktu kejadian, lokasi, kategori, deskripsi, dll) ditampilkan dengan baik.	Form berhasil ditampilkan lengkap sesuai kebutuhan input laporan.	✓	
	3	Admin atau *User* mengisi data laporan abnormalitas	Seluruh *field* dapat diisi dengan data yang sesuai tanpa error.	Data laporan berhasil diinput oleh *user*.	✓	
	4	Admin atau *User* mengunggah foto temuan (*finding picture*)	Sistem menerima file *upload* (gambar) dan menampilkan *preview* atau status upload berhasil.	File gambar berhasil diunggah dan tersimpan sementara di sistem.	✓	
	5	Admin atau *User* menekan tombol *Submit*	Sistem menerima *input* dan memproses data laporan abnormalitas.	Sistem berhasil menerima dan memproses data yang dikirim.	✓	

	6	Sistem melakukan validasi terhadap data yang diinput	Sistem memvalidasi seluruh *field* (tidak kosong, format sesuai, file valid, dll).	Sistem berhasil melakukan validasi data tanpa error.	✓	
	7	Sistem menyimpan laporan abnormalitas ke dalam *database*	Data laporan tersimpan ke *database* dengan status awal sesuai alur (Utama / *Fixed by Myself*).	Data laporan berhasil disimpan ke *database* dengan status awal yang sesuai.	✓	
	8	Sistem menampilkan notifikasi bahwa laporan berhasil dibuat	Sistem menampilkan notifikasi sukses (misalnya: "Data berhasil ditambahkan").	Notifikasi berhasil ditampilkan kepada *user*.	✓	
Alternative Flow (Extension)	1	Data tidak lengkap atau tidak valid.	Sistem menampilkan pesan validasi (misalnya: "Data wajib diisi" atau "Format tidak valid") dan tidak menyimpan laporan.	Sistem menampilkan pesan validasi sesuai dan *user* diminta melengkapi atau memperbaiki data.	✓	

5. *Black box Testing Scenario Use Case* Mengisi Data Perbaikan

   Black box testing ini berfokus pada pengujian fungsional mengisi data perbaikan menggunakan *black box testing*.

Tabel 4.5 *Black box Testing Scenario Use Case* Mengisi Data Perbaikan

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Succes Scenario	1	*User* membuka laporan yang ditugaskan kepadanya dari list tabel laporan abnormalitas	Sistem menampilkan daftar laporan abnormalitas dan *user* dapat memilih laporan	Daftar laporan berhasil ditampilkan dan *user* berhasil	✓	

			yang ditugaskan kepadanya.	membuka laporan yang ditugaskan.		
	2	Sistem menampilkan form sesuai peran *User* (*Action Owner / Assigned Action*)	Sistem menampilkan form yang sesuai dengan *role user*: • *Action Owner* → form analisis akar masalah & penugasan • *Assigned Action* → form tindakan perbaikan, bukti, dan target	Form berhasil ditampilkan sesuai dengan *role user*.	✓	
	3	User mengisi semua *field* wajib dan mengupload file/foto	Semua *field* wajib (*) dapat diisi dan file/foto dapat diupload sesuai ketentuan.	Data berhasil diinput dan file/foto berhasil diunggah.	✓	
	4	User menekan tombol "*Submit*" / "*Assigned*"	Sistem menerima *input* dan memproses data perbaikan sesuai aksi tombol.	Sistem berhasil menerima dan memproses data yang dikirim.	✓	
	5	Sistem memvalidasi kelengkapan dan format data	Sistem melakukan validasi terhadap seluruh *field* wajib dan file (format & ukuran sesuai).	Sistem berhasil melakukan validasi tanpa error.	✓	
	6	Sistem menyimpan data, memperbarui riwayat, dan memindahkan tahap laporan	• Data tersimpan ke database • Riwayat (*history*) diperbarui.	Data berhasil disimpan, riwayat diperbarui, dan tahap	✓	

			• Status tetap "*Open*" tetapi tahap berpindah sesuai *workflow*	laporan berpindah sesuai alur.		
	7	Sistem menampilkan notifikasi sukses dan mengarahkan kembali ke list laporan	Sistem menampilkan notifikasi sukses dan mengarahkan *user* ke halaman list laporan abnormalitas.	Notifikasi sukses ditampilkan dan *user* diarahkan kembali ke list laporan.	✓	
Alternative Flow (Extension)	1	*Field* wajib tidak diisi / data tidak valid	Sistem menampilkan pesan validasi (misalnya: "*Field* wajib harus diisi") dan tidak melanjutkan proses submit.	Sistem menampilkan pesan validasi dan *user* diminta melengkapi data.	✓	
	2	File/foto tidak sesuai ketentuan (format/ukuran)	Sistem menampilkan pesan error upload (misalnya: "Format file tidak didukung / ukuran terlalu besar").	Sistem menampilkan pesan error dan *user* diminta mengunggah ulang file yang sesuai.	✓	
	3	Proses penyimpanan gagal (gangguan sistem/koneksi)	Sistem menampilkan notifikasi kegagalan dan data tidak tersimpan, status laporan tetap di tahap sebelumnya.	Sistem menampilkan notifikasi gagal dan data tidak berubah.	✓	
	4	User membatalkan proses sebelum *submit*.	Sistem tidak menyimpan perubahan data dan *user* kembali ke form atau daftar laporan.	Sistem tidak menyimpan data dan *user* kembali ke halaman sebelumnya.	✓	

6. *Black box Testing Scenario Use Case* Validasi Laporan

   Black box testing ini berfokus pada pengujian fungsional validasi laporan menggunakan *black box testing*.

Tabel 4.6 *Black box Testing Scenario Use Case* Validasi Laporan

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Succes Scenario	1	*Validator* membuka laporan yang siap divalidasi dari daftar laporan abnormalitas	Sistem menampilkan daftar laporan dengan status siap divalidasi, dan *Validator* dapat membuka laporan tersebut.	Daftar laporan berhasil ditampilkan dan laporan berhasil dibuka oleh *Validator*.	✓	
	2	Sistem menampilkan form validasi (*read-only + field editable*)	Sistem menampilkan detail laporan dalam kondisi *read-only* (data sebelumnya) serta *field* yang dapat diisi: Status* dan *Remark**.	Form validasi berhasil ditampilkan dengan data lengkap dan *field* yang sesuai.	✓	
	3	*Validator* meninjau seluruh data dan bukti perbaikan	Semua data laporan dan file/bukti dapat ditampilkan dan diakses tanpa error.	Data dan bukti perbaikan berhasil ditampilkan dan dapat ditinjau.	✓	
	4	*Validator* mengisi Remark	*Field remark* dapat diisi (opsional jika *Closed*, wajib jika *Open*).	*Remark* berhasil diinput sesuai kebutuhan.	✓	
	5	*Validator* memilih status validasi (*Closed / Open*)	*Validator* dapat memilih status validasi sesuai keputusan.	Status validasi berhasil dipilih.	✓	

	6	*Validator* menekan tombol "*Validate*"	Sistem menerima *input* dan memproses validasi laporan.	Sistem berhasil menerima dan memproses data validasi.	✓	
	7	Sistem melakukan validasi *input*	Sistem memastikan: • Jika status = *Open* → remark wajib diisi • Jika status = *Closed* → *remark opsional*	Sistem berhasil memvalidasi input sesuai aturan.	✓	
	8	Sistem menyimpan hasil validasi dan memperbarui data	• Status laporan diperbarui (*Closed/Open*) • *Remark* tersimpan • *History* diperbarui	Data validasi berhasil disimpan, status dan *history* berhasil diperbarui.	✓	
	9	Sistem menampilkan notifikasi dan *redirect* ke list laporan	Sistem menampilkan notifikasi sukses dan mengarahkan kembali ke daftar laporan.	Notifikasi berhasil ditampilkan dan *Validator* diarahkan ke halaman list laporan.	✓	
Alternative Flow (Extension)	1	Validator meninjau laporan dari *Requestor* (*Fixed by Myself*)	Sistem tetap menampilkan form validasi dan proses berjalan sama seperti alur utama.	Proses validasi berjalan normal sesuai alur.		
	2	*Validator* memilih status "*Closed*"	Sistem menyimpan validasi, menandai laporan sebagai selesai	Status laporan berhasil menjadi	✓	

			(*Closed*), dan laporan tidak dapat diedit kembali.	*Closed* dan tidak dapat diedit.		
	3	*Validator* memilih status "*Open*" dengan *Remark*	Sistem menyimpan validasi dan mengembalikan laporan ke tahap sebelumnya (*clarify*).	Status tetap *Open* dan laporan kembali ke aktor sebelumnya (*action owner*) untuk melakukan *clarify* data tersebut.	✓	

7. *Black box Testing Scenario Use Case* Melihat Dashboard Analisis

   Black box testing ini berfokus pada pengujian fungsional melihat dashboard analisis menggunakan *black box testing*.

Tabel 4.7 *Black box Testing Scenario Use Case* Melihat *Dashboard* Analisis

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**
Main Succes Scenario	1	*User* membuka menu/halaman *Dashboard* Analisis	Sistem menampilkan halaman *dashboard* analisis tanpa error.	Halaman *dashboard* berhasil ditampilkan.	✓	
	2	Sistem menampilkan *dashboard default* dengan visualisasi utama	Sistem menampilkan chart utama, seperti: • Grafik batang *root cause* • Grafik konsekuensi abnormalitas • Grafik *non-compliance* • *Monitoring* bulanan (*bar*	Semua visualisasi berhasil ditampilkan sesuai data yang tersedia.	✓	

			chart, line chart, pie chart)			
	3	*User* melihat dan memahami data agregat (*insight*)	Data agregat (tren, distribusi, dll) dapat ditampilkan dengan jelas dan mudah dipahami.	Data agregat berhasil ditampilkan dan dapat dibaca oleh *user*.	✓	
	4	*User* menerapkan filter (tanggal, *facility*, *line*, *station*, dll)	Sistem menyediakan fitur filter dan *user* dapat memilih parameter *filter* tanpa error.	*Filter* berhasil dipilih oleh *user*.	✓	
	5	Sistem memproses *filter* dan memperbarui *dashboard*	Sistem melakukan *query* data berdasarkan *filter* dan memperbarui tampilan *chart* secara *real-time.*	*Dashboard* berhasil diperbarui sesuai *filter* yang dipilih.	✓	
	6	*User* selesai melihat *dashboard* dan berpindah halaman.	*User* dapat kembali ke menu lain tanpa error.	Navigasi ke halaman lain berhasil dilakukan.	✓	
Alternative Flow (Extension)	1	*User* memilih *filter* tertentu (misalnya tanggal/*facility* spesifik)	Sistem menampilkan data *dashboard* sesuai *filter* yang dipilih.	Data *dashboard* berhasil diperbarui sesuai *filter*.	✓	
	2	Tidak ada data yang tersedia	Sistem menampilkan pesan: "*No data available*" atau *dashboard* kosong dengan informasi yang jelas.	Sistem menampilkan pesan *"No data available"*.	✓	
	3	*User* mengklik elemen *chart* (*drill-down data*)	Sistem mengarahkan *user* ke halaman detail laporan	Sistem berhasil mengarahkan ke	✓	

			sesuai data yang dipilih pada *chart*.	halaman detail laporan.		

8. *Black box Testing Scenario Use Case* Mengubah Kata Sandi

   Black box testing ini berfokus pada pengujian fungsional mengubah kata sandi menggunakan *black box testing*.

Tabel 4.8 *Black box Testing Scenario Use Case* Mengubah Kata Sandi

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**
Main Succes Scenario	1	*User* mengakses menu *"Change Password"*	Sistem menampilkan halaman/form *Change Password.*	Halaman *Change Password* berhasil ditampilkan.	✓	
	2	Sistem menampilkan form pengubahan kata sandi	Form berisi *field: Old Password, New Password, Confirm Password*, dan tombol "*Change Password*" ditampilkan dengan lengkap.	Form berhasil ditampilkan sesuai dengan *field* yang dibutuhkan.	✓	
	3	*User* memasukkan *Old Password*	*Field Old Password* dapat diisi tanpa error.	*Old Password* berhasil diinput oleh *user*.	✓	
	4	*User* memasukkan *New Password*	*Field New Password* dapat diisi sesuai ketentuan sistem.	*New Password* berhasil diinput oleh *user*.	✓	
	5	*User* memasukkan *Confirm Password*	*Field Confirm Password* dapat diisi dan sesuai dengan *New Password*.	*Confirm Password* berhasil diinput oleh *user*.	✓	

	6	*User* menekan tombol *"Change Password"*	Sistem menerima *input* dan memproses perubahan *password*.	Sistem berhasil menerima dan memproses data perubahan *password*.	✓	
	7	Sistem melakukan validasi data	Sistem memvalidasi: • *Old Password* sesuai dengan *database* • *New Password* memenuhi kebijakan (minimal karakter, kombinasi, dll) • *Confirm Password* sama dengan *New Password*	Sistem berhasil melakukan validasi sesuai aturan.	✓	
	8	Sistem memperbarui kata sandi di *database*	*Password* diperbarui di *database* dengan mekanisme *hashing*/enkripsi.	*Password* berhasil diperbarui di *database*.	✓	
	9	Sistem menampilkan notifikasi sukses dan *redirect*	Sistem menampilkan pesan "*Password successfully changed*" dan mengarahkan user ke halaman utama tanpa *logout*.	Notifikasi berhasil ditampilkan dan *user* diarahkan ke halaman utama.	✓	

Alternative Flow (Extension)	1	*Old Password* tidak sesuai	Sistem menampilkan pesan error: *"Old password is incorrect"* dan tidak memproses perubahan *password*.	Sistem menampilkan pesan error dan *user* diminta menginput ulang.	✓	
	2	*Confirm Password* tidak sama dengan *New Password*	Sistem menampilkan pesan error: *"New password and confirmation do not match"*.	Sistem menampilkan pesan error dan *user* diminta memperbaiki *input*.	✓	

9. *Black box Testing Scenario Use Case Logout*

   Black box testing ini berfokus pada pengujian fungsional *logout* menggunakan *black box testing*.

   Tabel 4.9 *Black box Testing Scenario Use Case Logout*

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Succes Scenario	1	User mengakses menu "*Log Out*" dari *sidebar*/navigasi utama	Sistem menerima aksi klik tombol "*Log Out*".	Sistem berhasil menerima aksi *logout* dari *user*.	✓	
	2	Sistem menghapus sesi aktif pengguna di *server*	Sesi *login user* dihapus dari sistem/*server* untuk mencegah akses lanjutan tanpa login ulang.	Sesi *user* berhasil dihapus dari *server*.	✓	
	3	Sistem mengakhiri sesi dan mengarahkan ke halaman *login*	Sistem melakukan *redirect* ke halaman *login* setelah sesi dihapus.	*User* berhasil diarahkan ke halaman *login*.	✓	
	4	Sistem menampilkan halaman *login*	Halaman *login* ditampilkan dengan *form username* dan *password*.	Halaman *login* berhasil ditampilkan.	✓	

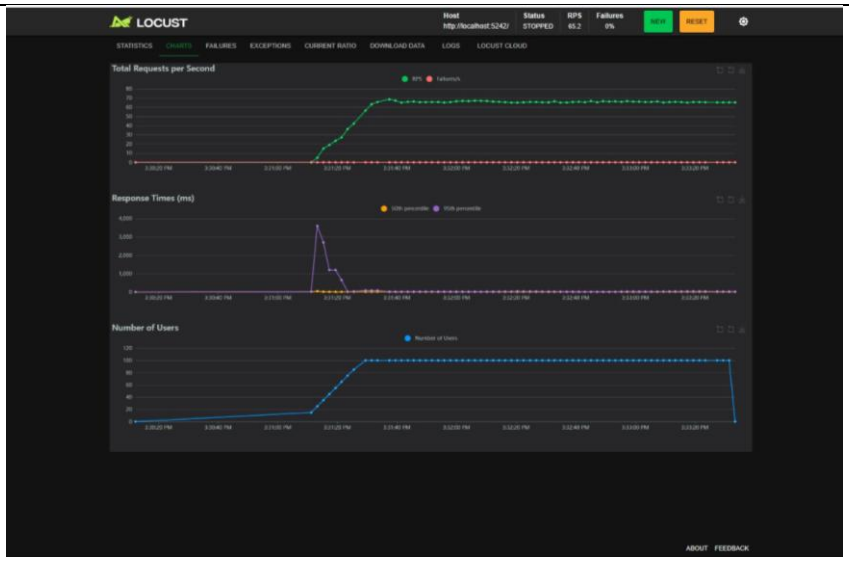
4.2.2 Non-Fungsional Testing

Pada tahap ini dilakukan pengujian terhadap fungsional sistem untuk memastikan setiap fitur pada perangkat lunak dapat berjalan sesuai dengan kebutuhan. Proses pengujian dilakukan menggunakan Locust dan OWASP ZAP. Hasil pengujian fungsional menggunakan metode *black box* dapat dilihat pada tabel-tabel berikut.

1. *Performance Testing*

Tabel 4.10 *Performance Testing*

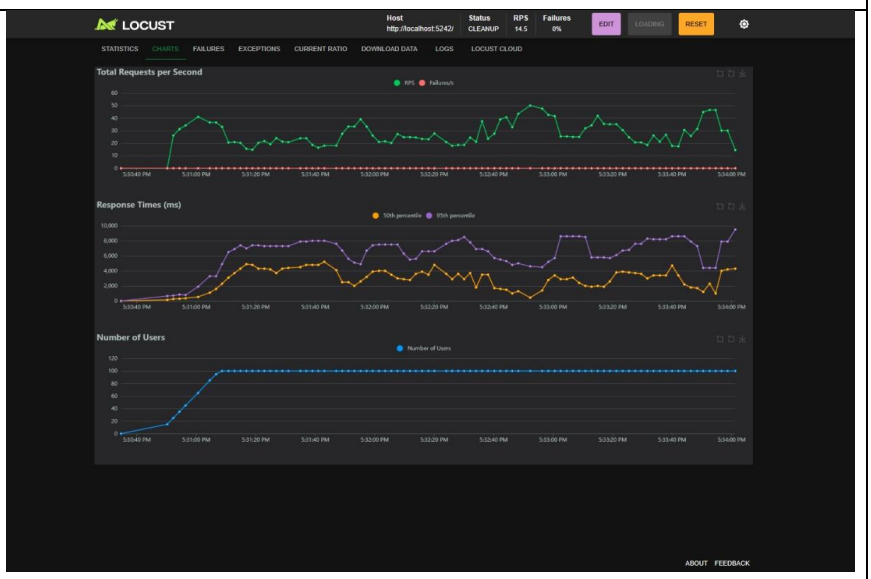
Komponen	**Rincian**	
Tes *Case* ID	TC01-NF01	
Type	*Performance Testing*	
Tujuan	Memastikan sistem dapat beroperasi dengan baik di bawah berbagai beban kerja tanpa penurunan performa.	
Lingkungan	*Localhost*	
Alat	Locust	
Metode	*Load Testing*	
Ekspektasi	Sistem dapat merespons permintaan dalam waktu yang konsisten, dengan konsumsi sumber daya yang efisien dan tanpa crash saat mencapai beban puncak.	
Hasil Uji		
ID	LOT01	
Nama Pengujian	*Load Test - Endpoint Login*	
Deskripsi Pengujian	Menguji performa *endpoint login* dengan autentikasi *user*	
Method	POST	
	Total *Users*	100

Skenario Uji	*Ramp Up*	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	4
	Average (ms)	419.07
	Max (ms)	3627
	95%ile (ms)	3600
Requests	Total *Requests*	100
	Success Requests	100
	Failed Requests	0
Error rate (%)	0%	
Chart Pengujian		
Hasil Uji		
ID	LOT02	
Nama Pengujian	*Load Test - Endpoint Create User*	
Deskripsi Pengujian	Menguji performa endpoint penambahan *user* ke *database*	
Method	POST	
Skenario Uji	Total *Users*	100
	Ramp Up	5

	Time (durasi)	3 menit
Response Time (ms)	Min (ms)	2
	Average (ms)	15.36
	Max (ms)	1025
	95%ile (ms)	40
Requests	Total *Requests*	12716
	Success Requests	12716
	Failed Requests	0
Error rate (%)	0%	
Chart Pengujian		
Hasil Uji		
ID	LOT03	
Nama Pengujian	*Load Test - Endpoint Change Password*	
Deskripsi Pengujian	Menguji performa *endpoint change password user*. (*noted*: Pengujian dilakukan dengan menggunakan data password lama yang tidak valid untuk menghindari perubahan data secara terus-menerus selama proses *load testing*. Meskipun menghasilkan response gagal, request tetap diproses oleh server sehingga hasil performa tetap relevan.)	

Method	POST	
Skenario Uji	Total *Users*	100
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	6
	Average (ms)	83.04
	Max (ms)	2410
	95%ile (ms)	300
Requests	Total *Requests*	10,484
	Success Requests	10,484
	Failed Requests	10,484
Error rate (%)	100%	
Chart Pengujian		
Hasil Uji		
ID	LOT04	
Nama Pengujian	*Load Testing - Endpoint Add Abnormality*	
Deskripsi Pengujian	Pengujian performa saat *user* melakukan proses *input* data abnormality (*Observation*)	

Method	POST	
Skenario Uji	Total *Users*	100
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	6
	Average (ms)	7957.69
	Max (ms)	64639
	95%ile (ms)	23000
Requests	Total *Requests*	1821
	Success Requests	1821
	Failed Requests	0
Error rate (%)	0%	
Chart Pengujian		
Hasil Uji		
ID	LOT05	
Nama Pengujian	*Load Testing - Endpoint Dashboard*	

Deskripsi Pengujian	Pengujian performa sistem saat *user* mengakses halaman *dashboard* dan memuat seluruh data *chart* secara bersamaan melalui *request* AJAX	
Method	GET	
Skenario Uji	Total *Users*	100
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	3
	Average (ms)	3071.68
	Max (ms)	12344
	95%ile (ms)	7300
Requests	Total *Requests*	5377
	Success Requests	5377
	Failed Requests	0
Error rate (%)	0%	
Chart Pengujian		

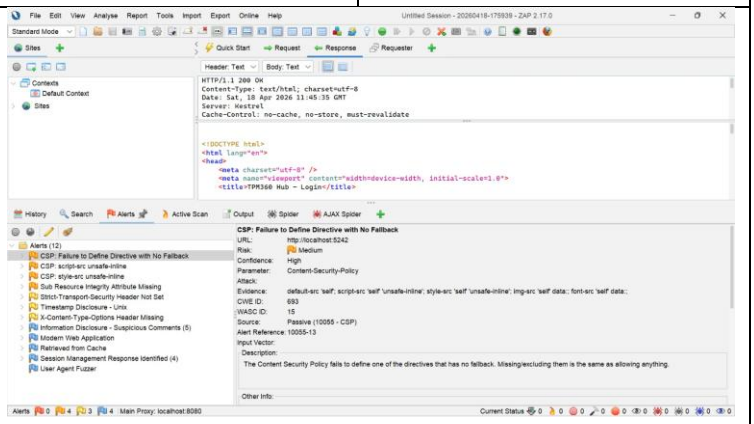
2. *Security Testing*

Tabel 4.11 *Security Testing*

Komponen	**Rincian**
Tes *Case* ID	TC02-NF02
Type	*Security Testing*
Tujuan	Memastikan sistem terlindungi dari ancaman keamanan seperti injeksi SQL, *cross-site scripting*, dan akses tidak sah.
Lingkungan	Pengujian pada lcoal pengembangan dengan akses terbatas untuk mencegah kebocoran data.
Alat	OWASP ZAP
Metode	1. Melakukan pengujian penetrasi (*penetration testing*) terhadap fitur aplikasi. 2. Mengidentifikasi kerentanan dan melakukan eksploitasi terkontrol. 3. Menganalisis keamanan data saat transit dan penyimpanan.
Ekspetasi	Sistem bebas dari kerentanan utama, data terenkripsi dengan baik, dan akses hanya dapat dilakukan oleh pihak yang berwenang.
Hasil Uji	
Jumlah *Alert*	12

No	**Temuan**	**Tingkat Risiko**	**Penjelasan**
1	CSP: Failure to Define Directive with No Fallback	Low	Beberapa *directive* CSP tidak memiliki fallback sehingga kebijakan keamanan tidak optimal.
2	CSP: script-src unsafe-inline	Medium	Penggunaan *'unsafe-inline'* pada script dapat meningkatkan risiko XSS.

3	CSP: style-src unsafe-inline	Medium	Penggunaan *'unsafe-inline'* pada style berpotensi membuka celah injeksi CSS.
4	Subresource Integrity Attribute Missing	Low	*Resource* eksternal belum menggunakan atribut *integrity*.
5	Strict-Transport-Security Header Not Set	Medium	HSTS belum aktif karena *environment* masih menggunakan HTTP.
6	Timestamp Disclosure - Unix	Informational	Informasi waktu server terdeteksi namun tidak sensitif.
7	X-Content-Type-Options Header Missing	Low	Header keamanan belum diterapkan secara menyeluruh.
8	Information Disclosure - Suspicious Comments	Low	Ditemukan komentar pada HTML yang dapat memberikan informasi tambahan.
9	Modern Web Application	Informational	Aplikasi menggunakan teknologi modern.
10	Retrieved from Cache	Low	*Response* masih dapat di-*cache* oleh browser.
11	Session Management	Informational	Sistem menggunakan *session management*.

	Response Identified		
12	User Agent Fuzzer	Informational	Perbedaan *response* berdasarkan *user-agent* terdeteksi.
Screenshot			

BAB V KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian Sistem Informasi Pelaporan Abnormalitas Mesin berbasis web di PT. XYZ, maka dapat disimpulkan bahwa tujuan dari proyek akhir ini telah berhasil tercapai. Adapun kesimpulan yang diperoleh adalah sebagai berikut:

1. Sistem berhasil dikembangkan untuk mendukung pencatatan riwayat abnormalitas mesin serta monitoring progres penanganan secara *real-time*, sehingga meningkatkan transparansi dan kemudahan pengawasan.
2. Mekanisme kategorisasi abnormalitas berbasis kode warna (Merah, Kuning, Hijau, Putih) berhasil diterapkan dan mampu membantu dalam penentuan prioritas serta distribusi penanganan masalah secara lebih efektif.
3. Sistem mampu mengimplementasikan penentuan otomatis *Action Owner, Assigned SESA,* dan *Validator* berdasarkan parameter seperti *AM Checklist, facility,* dan *tag ID*, sehingga meningkatkan efisiensi dalam proses distribusi tugas dan tanggung jawab.
4. Hasil pengujian menggunakan metode *Blackbox Testing* menunjukkan bahwa seluruh fitur utama sistem berjalan sesuai dengan kebutuhan pengguna dan dapat mendukung efektivitas manajemen perawatan mesin.

Dengan demikian, sistem *Machine Abnormality* yang dikembangkan mampu memberikan solusi yang lebih efisien, terstruktur, dan terdigitalisasi dalam proses pelaporan abnormalitas mesin serta berkontribusi dalam mengurangi risiko *downtime* dan meningkatkan produktivitas perusahaan.

5.2 Saran

Berdasarkan hasil pengembangan dan pengujian sistem yang telah dilakukan, terdapat beberapa saran untuk pengembangan lebih lanjut, yaitu:

1. Sistem dapat dikembangkan dengan menambahkan fitur notifikasi otomatis (email atau *push notification*) untuk mengingatkan pengguna terkait laporan yang belum ditindaklanjuti.

2. Perlu dilakukan peningkatan pada aspek keamanan sistem, seperti implementasi enkripsi data yang lebih kuat serta penggunaan autentikasi tambahan (*multi-factor authentication*).
3. Sistem dapat dikembangkan lebih lanjut dengan integrasi ke perangkat *mobile* agar memudahkan akses pengguna di lapangan secara lebih fleksibel.
4. Disarankan adanya integrasi dengan sistem lain di perusahaan, seperti sistem *maintenance* atau *inventory*, untuk meningkatkan keterpaduan data dan proses bisnis.
5. Penelitian selanjutnya dapat mengembangkan sistem dengan menambahkan fitur berbasis analitik atau prediksi (*predictive maintenance*) untuk meningkatkan kemampuan sistem dalam mendukung pengambilan keputusan.

DAFTAR PUSTAKA

Bassil, Y. (2012). *A simulation model for the waterfall software development life cycle. International Journal of Engineering & Technology,* 2(5), 1–7.

Campos, J., Sharma, P., & Jantunen, E. (2020). *Implementation of Computerized maintenance management system* (CMMS) *in Industry. International Journal of Engineering Applied Sciences and Technology,* 5(4), 100–103.

Hidayat, A., Riyanto, D., & Putra, D. Y. (2024). Analisa kualitas CMMS menggunakan kerangka kerja COBIT 5. Jurnal Ekonomi, Manajemen, Sistem Informasi, 5(2), 134–135.

Mohd Rofi, M. H., & Kasim, S. (2022). *Development of Computerized Maintenance Management System (CMMS) for Manufacturing Factory. Advances in Information Technology and Computer Science,* 3(1), 300–309.

Shankar, V., Chandan, G., & Singh, H. (2024). *Evaluating the Effectiveness of CMMS in Two-Wheeler Manufacturing Industries. International Journal of Management & Technology,* 12(3), 88–97.

Sukmana, N., & Rozi, F. (2024). *Efficiency Evaluation of CMMS Project Stage Using K-Means Algorithm.* Jurnal Ilmiah Pendidikan Informatika (JIPI), 9(1), 55–63.

Widyantoro, A., Al Bina, F. F., & Prayoga, T. (2022). *Systematic Literature Review:* Membandingkan pendekatan metode Agile dan Waterfall dalam Pengembangan Perangkat Lunak. *Journal of Comprehensive Science,* 4(1), 25–34.

Microsoft. (2024). ASP.NET Core MVC Overview. Microsoft Learn. https://learn.microsoft.com/en-us/aspnet/core/mvc/overview

Microsoft. (2024). C# documentation. https://learn.microsoft.com/en-us/dotnet/csharp/

Laudon, K. C., & Laudon, J. P. (2020). Management Information Systems: Managing the Digital Firm (16th ed.). Pearson.

Mobley, R. K. (2021). Maintenance Fundamentals (4th ed.). Elsevier.

Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). Database System Concepts (7th ed.). McGraw-Hill.

DAFTAR LAMPIRAN

Lampiran A. Dokumen Proses Pengumpulan Requirement

Link akses Dokumen Work Process:

DOKUMEN WORK PROCESS

Lampiran B. Link Produk

Link akses *Repository* GitHub:

https://github.com/naylabilla/MATA

Lampiran C. Dokumen Pengujian UAT

1. Link Akses UAT:
   DOKUMEN USER ACCEPTANCE TESTING (UAT)
2. Link Akses *Performance* Testing:
   Performance testing MATA
3. Link Akses *Security* Testing:
   Security Testing MATA