Perancangan Website Sistem Prediksi dan Penjadwalan Preventive maintenance Perangkat Komputer Menggunakan Artificial Intelligence di PT XYZ

PROYEK AKHIR

Disusun Oleh:

Marsya Huriyah Ibtisamah 3312301016

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 syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa, yang telah memberikan penulis kemudahan serta kelancaran dalam menyelesaikan laporan proyek akhir dengan judul "Perancangan Website Sistem Prediksi dan Penjadwalan *Preventive maintenance* Perangkat Komputer Menggunakan *Artificial Intelligence* di PT XYZ dapat diselesaikan tepat pada waktunya.

Laporan ini disusun sebagai salah satu syarat untuk menyelesaikan Program Studi Informatika pada jenjang Diploma III. Adapun tujuan utama dari penulisan laporan ini adalah untuk mengembangkan dan mengimplementasikan sistem berbasis *Artificial Intelligence* yang dapat membantu tim *IT Support* dalam melakukan perawatan perangkat komputer secara lebih efektif dan terencana.

Pada kesempatan ini, penulis ingin menyampaikan terima kasih kepada semua pihak yang telah memberikan dukungan moral maupun material sehingga laporan proyek akhir ini dapat selesai. Ucapan terima kasih ini penulis tujukan kepada:

1. Bapak Dr. Ari Wibowo, S.T., M.T. selaku dosen yang telah memberikan bimbingan, saran, dan ide kepada penulis.
2. Seluruh Bapak/Ibu Dosen Penguji, yang telah memberikan masukan, dan pertanyaan yang memotivasi untuk perbaikan laporan ini.
3. Orang tua penulis yang telah memberikan doa, dorongan, dan juga semangat selama penyusunan laporan proyek akhir.
4. Semua pihak yang tidak dapat dituliskan satu per satu yang telah membantu dalam proses penyusunan laporan proyek akhir penelitian ini.

Penulis berharap laporan proyek akhir ini dapat memberikan manfaat bagi pembaca, khususnya bagi mahasiswa yang tertarik dalam bidang *Artificial Intelligence* serta penerapannya pada sistem *preventive maintenance* perangkat komputer.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, penulis dengan tangan terbuka menerima segala bentuk kritik dan saran yang membangun demi perbaikan di masa mendatang.

Batam, 16 Juni 2026

Marsya Huriyah I

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

ABSTRAK

Pengelolaan dan pemeliharaan perangkat komputer di lingkungan kerja, khususnya pada tim IT *Support*, masih dilakukan secara manual dan reaktif berdasarkan laporan kerusakan dari pengguna. Kondisi ini menyebabkan keterlambatan penanganan serta meningkatkan risiko *downtime* perangkat yang dapat menghambat produktivitas kerja. Untuk mengatasi permasalahan tersebut, penelitian ini merancang dan membangun sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis *Artificial Intelligence* (AI) yang dapat membantu tim *IT Support* dalam mengelola kondisi perangkat secara lebih proaktif dan terstruktur. Sistem ini menganalisis data historis perangkat seperti penggunaan CPU, RAM, *disk*, dan suhu perangkat untuk memprediksi potensi kerusakan serta memberikan rekomendasi jadwal pemeriksaan secara otomatis. Pengembangan sistem dilakukan menggunakan metode *Waterfall* dengan memanfaatkan *Framework* Laravel sebagai *backend* utama, bahasa pemrograman Python dengan *Library Scikit-learn* menggunakan algoritma *Random Forest Classifier* yang berbasis *ensemble Decision Tree* untuk proses prediksi, serta *Software Agent* berbasis Python untuk pengambilan data *monitoring* perangkat secara *Real-Time*. Data monitoring dikirimkan ke server melalui API dan disimpan dalam basis data MySQL. Hasil pengujian menggunakan metode *Black Box Testing* dan *User Acceptance Testing* (UAT) menunjukkan bahwa seluruh fungsi utama sistem berjalan sesuai kebutuhan pengguna tanpa ditemukan kesalahan fungsional yang signifikan. Dengan pendekatan ini, sistem diharapkan dapat meningkatkan efisiensi kerja tim IT *Support*, mengurangi risiko *downtime* perangkat, serta mendukung pengelolaan aset IT yang lebih cerdas, akurat, dan terstruktur.

Kata Kunci: *Preventive maintenance*, *Artificial Intelligence*, Prediksi Kerusakan, *Random Forest*, Laravel, IT Support.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi digital pada era saat ini menjadikan sistem komputer sebagai komponen penting dalam operasional perusahaan. Hampir seluruh aktivitas bisnis modern menggunakan perangkat komputer, mulai dari pengolahan data, komunikasi internal, hingga sistem manajemen berbasis digital. Ketergantungan ini secara tidak langsung menuntut perangkat komputer untuk selalu berada dalam kondisi optimal guna menghindari gangguan operasional.

Perawatan perangkat komputer merupakan hal penting dalam menjaga kestabilan operasional perusahaan. Namun, banyak perusahaan yang masih menerapkan pendekatan *corrective maintenance*, yaitu melakukan perbaikan setelah terjadi kerusakan. Model perawatan yang bersifat reaktif ini terbukti menyebabkan *downtime*, mengganggu proses kerja, dan menambah beban tim *IT Support* (Carvalho et al., 2019).

Sebagai solusi, pendekatan *preventive maintenance* dikembangkan untuk mengurangi risiko kegagalan sistem sebelum kerusakan terjadi. *Preventive maintenance* dilakukan secara terjadwal dan berbasis kondisi perangkat. Namun, metode yang hanya mengandalkan jadwal berkala sering kali kurang efektif karena tidak mempertimbangkan kondisi aktual perangkat secara nyata.

Seiring berkembangnya teknologi Artificial Intelligence (AI), pendekatan maintenance ini dapat ditingkatkan melalui analisis data berbasis *machine learning*. Sistem mampu menganalisis pola penggunaan perangkat seperti *CPU usage, RAM usage, disk usage*, dan suhu perangkat untuk mendeteksi indikasi awal potensi gangguan. Menurut Breiman (2001), algoritma *Random Forest* merupakan metode klasifikasi dengan tingkat akurasi tinggi yang mampu mengurangi risiko *overfitting* melalui pendekatan *ensemble learning*, sehingga cocok digunakan untuk mengklasifikasikan kondisi perangkat berdasarkan data performa yang bersifat numerik dan dinamis.

Berdasarkan permasalahan tersebut, penelitian ini mengembangkan sebuah sistem berbasis website yang mampu melakukan monitoring kondisi perangkat secara *Real-Time*, memprediksi potensi kerusakan menggunakan AI, serta menghasilkan jadwal *preventive maintenance* secara otomatis. Dengan adanya sistem ini, diharapkan kegiatan perawatan perangkat dapat dilakukan secara lebih proaktif, efisien, dan berbasis data aktual.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merancang sistem prediksi kerusakan perangkat komputer menggunakan *Artificial Intelligence*?
2. Bagaimana sistem dapat melakukan penjadwalan *preventive maintenance* secara otomatis berdasarkan hasil prediksi AI?
3. Bagaimana sistem dapat menampilkan hasil prediksi, rekomendasi tindakan, dan jadwal *maintenance* kepada tim *IT Support* melalui *dashboard* website?
4. Bagaimana merancang Software agent berbasis python yang mampu mengumpulkan data monitoring perangkat (CPU, RAM, Disk, Suhu) secara real-time dan mengirimkannya ke server secara otomatis menggunakan algoritma Random Forest Classifier?
5. Bagaimana sistem dapat menyediakan manajemen akun untuk lebih dari satu admin IT Support agar dapat mengakses sistem tanpa mengubah source code?

1.3. Tujuan

Tujuan dari proyek akhir ini adalah merancang dan membangun sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis Artificial Intelligence (AI) yang mampu membantu tim *IT Support* dalam mengelola kondisi perangkat secara lebih efektif dan terstruktur. Secara lebih spesifik, tujuan proyek ini meliputi:

1. Merancang dan mengembangkan sistem prediksi kerusakan perangkat komputer berbasis AI menggunakan algoritma *machine learning* untuk menganalisis data monitoring perangkat seperti penggunaan CPU, RAM, disk, dan suhu perangkat, sehingga sistem mampu memberikan hasil prediksi risiko kerusakan secara akurat.
2. Membangun sistem penjadwalan *preventive maintenance* otomatis yang terintegrasi dengan hasil prediksi AI, sehingga sistem dapat menentukan waktu perawatan perangkat berdasarkan tingkat risiko kerusakan tanpa memerlukan input manual dari admin.
3. Mengimplementasikan *dashboard* website berbasis Laravel yang menampilkan hasil prediksi, rekomendasi tindakan perawatan, serta jadwal *preventive maintenance* kepada tim *IT Support* secara interaktif dan mudah dipahami, guna meningkatkan efisiensi pengelolaan aset perangkat komputer.
4. Mengimplementasikan Software Agent berbasis Python menggunakan library psutil untuk monitoring real-time dan algoritma Random Forest Classifier untuk prediksi kondisi perangkat.
5. Membangun fitur manajemen akun multi-admin sehingga lebih dari satu staf IT Support dapat mengakses sistem tanpa perlu mengubah source code.
6. Menyediakan laporan monitoring yang dapat difilter per perangkat, per bulan, dan per tahun dengan fitur preview sebelum diunduh.

1.4. Batasan Masalah

Agar penelitian ini lebih terarah dan fokus pada sasaran utama, maka ditetapkan beberapa batasan sebagai berikut:

1. Data yang digunakan dalam proses prediksi merupakan data *Real-Time* yang dikumpulkan langsung dari perangkat melalui *Software Agent* berbasis Python menggunakan *Library psutil*.
2. Model AI melakukan klasifikasi tingkat risiko kerusakan perangkat berdasarkan agregasi beberapa data monitoring terakhir, bukan berdasarkan satu titik data tunggal.

3. Sistem AI hanya digunakan untuk memprediksi potensi kerusakan dan memberikan rekomendasi jadwal maintenance, bukan melakukan perbaikan perangkat secara otomatis.
4. Website hanya mencakup fitur utama seperti monitoring perangkat, hasil prediksi AI, penjadwalan maintenance otomatis, notifikasi email, dan laporan ringkas maintenance.
5. Integrasi dilakukan antara Laravel (PHP) sebagai *backend* utama dan Python (Scikit-learn) sebagai modul AI melalui pertukaran data berbasis API (*JSON*).
6. Pengujian sistem dilakukan pada lingkungan lokal menggunakan perangkat terbatas, sehingga hasil pengujian bersifat representatif namun belum mencerminkan skala penggunaan secara penuh di lingkungan produksi.
7. Fitur chatbot API yang direncanakan pada proposal awal tidak diimplementasikan karena pertimbangan relevansi dan kompleksitas. Sebagai gantinya, dikembangkan Software Agent berbasis Python yang lebih relevan dengan kebutuhan monitoring real-time sistem preventive maintenance.
8. Sistem menyediakan fitur manajemen akun dengan dua role yaitu Superadmin dan Admin. Superadmin memiliki hak akses tambahan untuk menambah dan menghapus akun admin lain secara langsung melalui antarmuka website.
9. Penjadwalan preventive maintenance berlaku untuk semua kondisi perangkat: HIGH (risiko kritis, +2 hari), MEDIUM (risiko sedang, +5 hari), dan LOW (kondisi normal, +30 hari) untuk memastikan perangkat mendapatkan perawatan rutin sebelum mencapai kondisi risiko.

1.5. Manfaat

Proyek ini diharapkan dapat memberikan manfaat secara teoritis dan praktis dalam pengembangan Sistem Prediksi dan Penjadwalan *Preventive Maintenance* Perangkat Komputer Berbasis *Artificial Intelligence* sebagai berikut:

1. **Manfaat Teoritis**

- Memberikan kontribusi dalam pengembangan ilmu pengetahuan di bidang sistem informasi, khususnya dalam penerapan teknologi *Artificial Intelligence* berbasis *machine learning (Random Forest Classifier*) untuk keperluan monitoring kondisi perangkat komputer dan prediksi kebutuhan maintenance secara otomatis.
- Menjadi referensi bagi penelitian atau pengembangan sistem serupa, terutama dalam penerapan *Software Agent* berbasis Python yang terintegrasi dengan aplikasi web Laravel untuk membentuk sistem monitoring *Real-Time* pada lingkungan jaringan lokal.
- Memberikan gambaran nyata tentang implementasi *pipeline machine learning* secara *end-to-end*, mulai dari pengumpulan data, pelatihan model, penerapan prediksi di sisi *client*, hingga integrasi dengan sistem informasi berbasis web dalam konteks *preventive maintenance.*

2. Manfaat Praktis

- Memberikan solusi terstruktur dan terdigitalisasi bagi tim *IT Support* dalam proses monitoring kondisi perangkat komputer secara *Real-Time*, sehingga kondisi setiap perangkat dapat dipantau secara terpusat melalui satu *dashboard* tanpa perlu pemeriksaan fisik secara langsung.
- Membantu tim *IT Support* dalam mengidentifikasi perangkat yang berpotensi mengalami kerusakan lebih awal melalui prediksi berbasis *Artificial Intelligence*, sehingga tindakan pencegahan dapat dilakukan sebelum terjadi kegagalan sistem yang menyebabkan *downtime* perangkat.
- Mengotomatiskan proses pembuatan jadwal *preventive maintenance* berdasarkan hasil deteksi kondisi perangkat oleh model AI, sehingga mengurangi beban kerja administratif tim *IT Support* dalam merencanakan dan menjadwalkan kegiatan *maintenance*.
- Meningkatkan efektivitas komunikasi antara tim *IT Support* dan pengguna perangkat melalui fitur pengiriman notifikasi email

otomatis yang memuat informasi jadwal *maintenance* secara lengkap dan terstruktur, sehingga pengguna dapat mempersiapkan diri sebelum pelaksanaan *maintenance*.

- Meminimalkan risiko kesalahan pencatatan dan meningkatkan keterlacakan proses *maintenance* melalui sistem pencatatan riwayat yang terintegrasi, mencakup riwayat perubahan status jadwal, riwayat pengiriman email, dan riwayat prediksi AI yang tersimpan secara sistematis di *database*.
- Mendukung pengambilan keputusan berbasis data *(data-driven decision making)* oleh manajemen IT melalui fitur laporan monitoring dalam format PDF yang memuat ringkasan kondisi perangkat, hasil prediksi AI, dan statistik pelaksanaan *maintenance* secara menyeluruh.

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 pengelolaan infrastruktur teknologi informasi melalui penerapan kecerdasan buatan (*Artificial Intelligence*) dalam proses monitoring dan perawatan perangkat komputer. Digitalisasi proses maintenance menggunakan *machine learning* dan *Software Agent* menjadikan pengelolaan infrastruktur IT lebih efisien, terstruktur, dan berbasis data.
- SDG 12: *Responsible Consumption and Production*

  Dengan mendeteksi kondisi perangkat secara dini dan melakukan tindakan *preventive maintenance* sebelum terjadi kerusakan, sistem ini membantu memperpanjang umur perangkat komputer, mengoptimalkan penggunaan sumber daya teknologi, serta mengurangi pemborosan akibat kerusakan mendadak yang memerlukan penggantian komponen atau perangkat secara menyeluruh.

- SDG 8: *Decent Work and Economic Growth*

  Sistem membantu menciptakan lingkungan kerja IT yang lebih produktif, efisien, dan terstruktur melalui otomasi proses monitoring dan penjadwalan *maintenance*. Dengan berkurangnya downtime perangkat, aktivitas operasional organisasi dapat berjalan lebih lancar, mendukung peningkatan produktivitas dan pertumbuhan kinerja secara keseluruhan.

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 monitoring perangkat berbasis *Artificial Intelligence* yang dikembangkan. Bentuk kolaborasi yang dilakukan antara lain:

- Diskusi dengan pembimbing lapangan

  Dilakukan secara berkala untuk memperoleh arahan terkait kebutuhan sistem monitoring di lingkungan PT. XYZ, pemahaman terhadap proses pengelolaan perangkat komputer yang berjalan saat ini, serta validasi terhadap solusi berbasis AI yang dirancang agar sesuai dengan kondisi dan kebutuhan nyata di lapangan.
- Koordinasi dengan tim *IT Support* dan pengguna perangkat di Perusahaan

  Bertujuan untuk memahami kondisi operasional perangkat komputer yang digunakan, mengidentifikasi kebutuhan fitur monitoring yang diperlukan oleh tim *IT Support*, serta memastikan sistem yang dibangun—termasuk *dashboard monitoring*, fitur penjadwalan *maintenance*, dan notifikasi email—sesuai dengan kebutuhan dan alur kerja yang sesungguhnya dijalankan oleh pengguna di lapangan.
- Konsultasi dengan dosen pembimbing

  Dilakukan secara rutin untuk memastikan kesesuaian metode pengembangan sistem, ketepatan implementasi algoritma *machine learning* yang digunakan (*Random Forest Classifier*), kualitas

akademik penulisan laporan, serta struktur penyajian hasil penelitian sesuai dengan standar Proyek Akhir yang berlaku.

- Pengumpulan umpan balik pengguna (*user feedback*)
  Dilakukan melalui uji coba sistem secara langsung oleh calon pengguna, yaitu tim *IT Support* dan perwakilan pengguna perangkat di PT. XYZ. Proses ini mencakup pengujian kemudahan penggunaan (*usability*) antarmuka *dashboard* monitoring, ketepatan informasi jadwal *maintenance* yang ditampilkan, serta kesesuaian fitur notifikasi email dengan kebutuhan operasional. Hasil umpan balik digunakan sebagai dasar perbaikan dan penyempurnaan sistem sebelum implementasi final.
- Pengembangan dan pengujian *Software Agent* secara kolaboratif
  Proses pengembangan *Software Agent* berbasis *Python* yang berjalan di sisi *client* dilakukan dengan melibatkan koordinasi bersama tim *IT Support* untuk menentukan parameter monitoring yang relevan, menetapkan *threshold* kondisi perangkat, serta menguji kompatibilitas agent dengan berbagai jenis perangkat komputer yang digunakan di lingkungan PT. XYZ.

Melalui kolaborasi tersebut, proyek ini tidak hanya bersifat individual, tetapi juga melibatkan sinergi berbagai pihak yang berkontribusi dalam menghasilkan sistem monitoring perangkat berbasis *Artificial Intelligence* yang relevan, aplikatif, dan sesuai dengan kebutuhan nyata pengelolaan infrastruktur teknologi informasi di lingkungan industri.

BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan Solusi dan Sistem Terkait

Pada bagian ini dibahas beberapa penelitian dan sistem yang memiliki keterkaitan dengan pengembangan Sistem Prediksi dan Penjadwalan *Preventive Maintenance* Perangkat Komputer Berbasis *Artificial Intelligence*. Tinjauan ini bertujuan untuk memberikan gambaran mengenai pendekatan yang telah dilakukan oleh penelitian-penelitian sebelumnya, teknologi yang digunakan, serta kelebihan dan keterbatasan masing-masing sehingga dapat dijadikan bahan pertimbangan dan dasar pengembangan sistem pada proyek ini.

1. Carvalho et al. (2019) — *A Systematic Literature Review of Machine learning Methods Applied to Predictive Maintenance*

   Penelitian ini merupakan tinjauan literatur sistematis yang membahas berbagai metode *machine learning* (ML) yang telah diterapkan untuk *predictive maintenance* (PdM) di berbagai industri. Penelitian ini mengidentifikasi algoritma populer seperti *Support Vector Machines* (SVM) dan *Random Forest* sebagai metode yang paling banyak digunakan untuk memprediksi kegagalan peralatan, serta memberikan gambaran komprehensif mengenai kelebihan dan kekurangan setiap pendekatan. Keterbatasan penelitian ini adalah hanya berfokus pada tinjauan metode dan tidak membahas implementasi praktis ke dalam sistem informasi berbasis web yang terintegrasi. Celah inilah yang menjadi salah satu dasar pengembangan sistem pada proyek ini, yaitu mewujudkan implementasi nyata dari algoritma *Random Forest* dalam sebuah sistem monitoring berbasis web untuk kebutuhan tim *IT Support* di lingkungan perkantoran.

2. Zhu et al. (2024) — *A Survey of Predictive Maintenance: Systems, Purposes and Approaches*

   Penelitian ini merupakan survei komprehensif yang membahas evolusi strategi pemeliharaan dari reaktif (*corrective maintenance*) dan preventif (*preventive maintenance*) menuju prediktif (*predictive maintenance*). Survei ini meninjau berbagai arsitektur sistem, tujuan, dan pendekatan

PdM, serta menyoroti peran teknologi modern seperti *Artificial Intelligence* dan IoT sebagai pendorong transisi tersebut. Keterbatasan penelitian ini adalah bersifat survei umum dan tidak menyediakan rancangan sistem spesifik untuk kasus perangkat komputer perkantoran, serta tidak membahas integrasi *backend* berbasis Laravel dan mekanisme penjadwalan *maintenance* otomatis. Hal ini menjadi celah yang coba diisi oleh sistem yang dikembangkan pada proyek ini.

3. Almazaideh & Levendovszky (2022) — *A Predictive Maintenance System for Wireless Sensor Networks: A Machine learning Approach*

   Penelitian ini membahas pengembangan sistem *predictive maintenance* yang menggunakan algoritma *Feedforward Neural Network* (FFNN) untuk memprediksi kualitas transfer data pada *Wireless Sensor Networks* (WSNs). Hasilnya menunjukkan bahwa model *machine learning* dapat memprediksi status operasional sistem dan membantu menjadwalkan tindakan pemeliharaan secara proaktif. Keterbatasan penelitian ini adalah fokusnya yang sangat spesifik pada kualitas jaringan WSN, bukan pada kondisi perangkat keras komputer seperti penggunaan CPU, RAM, Disk, dan suhu yang menjadi fokus utama sistem yang dikembangkan dalam proyek ini untuk keperluan monitoring perangkat komputer di lingkungan tim *IT Support*.

4. Kane et al. (2022) — *Predictive Maintenance Using Machine Learning: Techniques and Applications*

   Penelitian ini membahas *predictive maintenance* sebagai pendekatan proaktif untuk memprediksi kegagalan peralatan sebelum terjadi. Metode yang ditinjau mencakup algoritma *supervised learning* seperti *Decision Trees* dan *Support Vector Machines* (SVM) yang diterapkan untuk menganalisis data sensor dan operasional guna mengoptimalkan jadwal pemeliharaan. Keterbatasan penelitian ini adalah bersifat tinjauan konseptual dan tidak membahas implementasi sistem secara *end-to-end* yang mencakup pengumpulan data *Real-Time* oleh *Software Agent*, prediksi AI, dan penjadwalan *maintenance* otomatis melalui antarmuka

web. Celah inilah yang menjadi landasan pengembangan sistem terintegrasi pada proyek ini.

5. Kale (n.d.) — *Predictive Maintenance of Equipment Using Machine learning Algorithms*

   Penelitian ini membahas penerapan *machine learning* untuk memprediksi kegagalan peralatan di lingkungan industri menggunakan dataset AI4I 2020. Metode yang digunakan meliputi algoritma XGBoost, *Random Forest,* dan *Decision Tree* dengan penerapan *feature engineering* dan *hyperparameter tuning* menggunakan *Grid Search* untuk meningkatkan performa prediksi. Hasil penelitian menunjukkan bahwa *Random Forest* mencapai akurasi tertinggi sebesar 97,91%. Keterbatasan penelitian ini adalah penggunaan dataset industrial yang belum tentu dapat langsung diterapkan pada data perangkat komputer perkantoran, serta tidak membahas aspek integrasi dengan sistem web berbasis Laravel untuk penjadwalan *maintenance* otomatis oleh tim *IT Support*. Proyek ini mengadopsi algoritma *Random Forest* dari penelitian ini sebagai inti model AI yang dilatih menggunakan data monitoring perangkat komputer secara langsung.

Berdasarkan tinjauan terhadap penelitian-penelitian di atas, terdapat celah yang belum ditangani secara menyeluruh oleh penelitian yang ada, yaitu belum adanya sistem yang mengintegrasikan prediksi kondisi perangkat komputer berbasis *machine learning* dengan mekanisme penjadwalan *preventive maintenance* otomatis dalam sebuah aplikasi web yang dapat digunakan secara langsung oleh tim *IT Support*. Sebagian besar penelitian berfokus pada tinjauan metode, peralatan industri berat, atau infrastruktur jaringan, sehingga belum mengakomodasi kebutuhan monitoring perangkat komputer perkantoran secara *Real-Time* melalui *Software Agent* berbasis Python. Oleh karena itu, pada proyek ini dikembangkan sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis *Artificial Intelligence* yang mengintegrasikan *Software Agent Python*, model *Random Forest Classifier*, dan aplikasi web Laravel dalam satu sistem yang berjalan secara otonom dan terintegrasi..

Tabel 2.1 Tinjauan Aplikasi Sejenis

Peneliti/ Tahun	Judul/Topik Penelitian	Metode/ Teknologi	Kelebihan	Kekurangan/ Celah
(Carvalho et al., 2019)	A systematic literature review of *machine learning* methods applied to *predictive maintenance*	Tinjauan Literatur Sistematis (SVM, *Random Forest*, dll.)	Memberikan gambaran komprehensif tentang kelebihan dan kekurangan metode ML untuk PdM	- Hanya berfokus pada tinjauan metode - Tidak membahas implementasi praktis ke sistem web terintegrasi (Laravel) untuk IT Support
Breiman (2001)	*Random Forests*	Random Forest (Ensemble Learning, Decision Tree, Bagging, Random Feature Selection)	- Memiliki akurasi klasifikasi yang tinggi. - Lebih tahan terhadap noise dibanding metode seperti AdaBoost. - Dapat menangani data berdimensi tinggi. - Menghasilkan estimasi error (Out-of-Bag Error) tanpa memerlukan data uji terpisah.	- Penelitian berfokus pada pengembangan algoritma Random Forest secara teoritis. - Tidak membahas implementasi algoritma pada sistem monitoring berbasis web. - Tidak mengintegrasikan Random Forest dengan framework seperti Laravel maupun sistem predictive maintenance perangkat IT.
(Almazaide Reitermanová (2010)	Data Splitting	*Hold-Out Cross Validation, K-Fold Cross*	- Menjelaskan berbagai metode	- Penelitian hanya

		Validation, Simple Random Sampling (SRS), Stratified Sampling, DUPLEX, CADEX	pembagian data untuk meningkatkan kemampuan generalisasi model machine learning. - Membandingkan kelebihan dan kekurangan berbagai teknik data splitting. - Memberikan rekomendasi metode berdasarkan karakteristik dataset.	membahas teknik pembagian data sebelum proses pelatihan model. - Tidak membahas algoritma klasifikasi seperti Random Forest secara mendalam. - Tidak mengimplementasikan metode tersebut pada sistem monitoring atau aplikasi berbasis web untuk predictive maintenance.
(Kane et al., 2022)	*Predictive maintenance* using *machine learning* techniques	*Machine learning* (*Decision Tree*s, SVM)	- Membahas PdM sebagai pendekatan proaktif. - Menunjukkan ML menganalisis data sensor & operasional untuk optimasi jadwal. - Hasilnya mengurangi *downtime* &	- Tinjauan konseptual. - Tidak membahas implementasi sistem end-to-end secara spesifik. - Menyoroti tantangan lama (kualitas data, *Real-Time*) yang akan coba diatasi. Membantu menjadwalkan

			biaya operasionalMembantu menjadwalkan tindakan pemeliharaan	tindakan pemeliharaan
(Kale, n.d.)	*Predictive maintenance* of Equipment Using *Machine learning* Algorithms	*Machine learning* (XGBoost, *Random Forest*, *Decision Tree*), Feature Engineering (Power Transformer), Hyperparameter Tuning (Grid Search	- Menunjukkan akurasi tinggi (RF & XGBoost) dalam prediksi kegagalan industri. - Fokus pada feature engineering & tuning untuk kasus kegagalan jarang terjadiMenyoroti tantangan lama (kualitas data, *Real-Time*) yang akan coba diatasi. Membantu menjadwalkan tindakan pemeliharaan	- Fokus pada dataset industrial (AI4I 2020). - Belum tentu berlaku langsung untuk data perangkat komputer kantor. - Tidak membahas integrasi web (Laravel) atau penjadwalan otomatis untuk IT Support.
Marsya Huriyah Ibtisamah (2026)	Perancangan Website Sistem Prediksi dan Penjadwalan *Preventive maintenance* Perangkat Komputer Menggunakan	*Waterfall*, Laravel, Python (*Scikit-learn*), *Random Forest*, *Decision Tree*, MySQL	- Mengembangkan sistem PdM spesifik untuk perangkat komputer kantor di lingkungan IT *Support*.	- Sistem hanya mencakup prediksi dan penjadwalan *maintenance*, belum mencakup perbaikan perangkat secara

	Artificial Intelligence		- Integrasi *backend* web (Laravel) dengan modul AI (Python *Scikit-learn*) melalui API. - Menggunakan data *monitoring* perangkat secara *Real-Time* melalui *Software Agent* berbasis Python (*psutil*). - Penjadwalan *preventive maintenance* otomatis berdasarkan hasil klasifikasi AI. - Dilengkapi fitur notifikasi email, laporan PDF, dan *dashboard* monitoring terpusat.	otomatis. - Pengujian dilakukan pada lingkungan lokal dengan jumlah perangkat terbatas, sehingga belum mencerminkan skala produksi secara penuh.

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 Prediksi dan Penjadwalan *Preventive Maintenance* Perangkat Komputer Berbasis *Artificial Intelligence* di lingkungan *IT Support* PT. XYZ.

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 pengelolaan infrastruktur IT, sistem informasi berperan penting dalam memusatkan data monitoring perangkat, mempercepat identifikasi masalah, serta memastikan informasi kondisi perangkat dapat diakses secara cepat dan akurat oleh tim *IT Support*.

2. Preventive Maintenance

Menurut Mobley (2002), *preventive maintenance* adalah suatu pendekatan pemeliharaan yang dilakukan secara terjadwal dan proaktif untuk mencegah terjadinya kerusakan atau kegagalan perangkat sebelum gangguan benar-benar terjadi. Tujuan utamanya adalah menjaga keandalan perangkat, mengurangi *downtime* yang tidak terencana, serta memperpanjang umur pakai perangkat. Dalam proyek ini, konsep *preventive maintenance* menjadi landasan utama perancangan mekanisme penjadwalan otomatis yang menggunakan prediksi AI sebagai dasar penentuan waktu dan prioritas *maintenance*.

3. *Artificial Intelligence* dan *Machine Learning*

Menurut Russell & Norvig (2020), *Artificial Intelligence* (AI) adalah bidang ilmu komputer yang berfokus pada pengembangan sistem yang mampu melakukan tugas-tugas yang biasanya memerlukan kecerdasan manusia, termasuk klasifikasi dan pengambilan keputusan. *Machine learning* sebagai cabang AI memungkinkan sistem untuk belajar dari

data dan meningkatkan kemampuan prediksinya tanpa diprogram secara eksplisit. Dalam proyek ini, *machine learning* diterapkan menggunakan algoritma *Random Forest Classifier* yang dilatih menggunakan data monitoring perangkat komputer untuk mengklasifikasikan kondisi perangkat sebagai Normal atau Risiko secara otomatis.

4. Random Forest Classifier

Menurut Breiman (2001), *Random Forest* adalah metode *ensemble learning* yang bekerja dengan membangun sejumlah pohon keputusan (*decision tree*) secara paralel selama proses pelatihan, kemudian menggabungkan hasil prediksi dari seluruh pohon melalui mekanisme voting mayoritas untuk menghasilkan prediksi akhir. Pendekatan *ensemble* ini membuat *Random Forest* lebih robust terhadap *overfitting* dan umumnya menghasilkan akurasi yang lebih tinggi dibandingkan pohon keputusan tunggal. Dalam proyek ini, *Random Forest* digunakan sebagai model prediksi *AI* yang dimuat oleh *Software Agent* untuk melakukan klasifikasi kondisi perangkat secara lokal di sisi *client* setiap 5 detik. Dalam bidang monitoring dan prediksi kondisi perangkat, Random Forest telah terbukti efektif digunakan. Penelitian oleh Kale et al. (n.d.) menunjukkan bahwa Random Forest mencapai akurasi 97,91% dalam memprediksi kegagalan peralatan berdasarkan data performa sistem. Selain itu, Breiman (2001) yang merupakan pencetus algoritma Random Forest membuktikan bahwa metode ensemble ini secara konsisten menghasilkan akurasi lebih tinggi dibandingkan single decision tree dengan tetap mempertahankan generalisasi yang baik,

5. *Software Agent*

Software Agent adalah program komputer yang berjalan secara otonom untuk melaksanakan tugas tertentu tanpa memerlukan intervensi manusia secara terus-menerus (Russell & Norvig,

2020). Dalam proyek ini, *Software Agent* berbasis Python berjalan secara mandiri di sisi *client* (perangkat yang dimonitor) untuk mengumpulkan data metrik sistem secara *Real-Time* menggunakan *Library psutil*, melakukan prediksi menggunakan model AI yang telah dilatih, dan mengirimkan data beserta hasil prediksi ke server Laravel melalui *REST API* setiap 5 detik secara berkelanjutan.

6. **Monitoring Perangkat Komputer Berbasis *Real-Time***

   Monitoring perangkat komputer secara *Real-Time* adalah proses pengumpulan dan pemantauan data kondisi perangkat secara berkelanjutan tanpa jeda yang signifikan. Parameter yang umum dimonitor meliputi persentase penggunaan CPU, RAM, kapasitas penyimpanan disk, dan suhu CPU. Pemantauan secara *Real-Time* memungkinkan tim *IT Support* untuk mendeteksi anomali kondisi perangkat secara dini sebelum berkembang menjadi kegagalan yang lebih serius, sehingga tindakan pencegahan dapat segera dilakukan.

7. ***REST API* dan Arsitektur *Client*-Server**

   REST API (*Representational State Transfer Application Programming Interface*) adalah gaya arsitektur untuk membangun layanan web yang memungkinkan komunikasi antara dua sistem melalui protokol HTTP dengan menggunakan format data JSON. Menurut Richardson & Ruby (2007), arsitektur *client*-server berbasis *REST* memisahkan logika *client* dari server, sehingga masing-masing dapat dikembangkan secara independen. Dalam proyek ini, *Software Agent* Python berperan sebagai *client* yang mengirimkan data monitoring melalui *HTTP POST request* ke endpoint *REST API* yang disediakan oleh server Laravel.

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. *Python* dan *Psutil*

   Python adalah bahasa pemrograman tingkat tinggi yang banyak digunakan dalam pengembangan *machine learning* dan otomasi sistem. *Library psutil (process* and *system utilities)* adalah *Library Python* yang menyediakan antarmuka lintas platform untuk mengambil informasi tentang proses sistem dan penggunaan sumber daya, meliputi CPU, memori, disk, dan jaringan. Dalam proyek ini, psutil digunakan oleh *Software Agent* untuk mengumpulkan data monitoring perangkat secara *Real-Time*.

3. *Scikit-learn* dan Joblib

   Scikit-learn adalah *Library machine learning* Python yang menyediakan implementasi berbagai algoritma pembelajaran mesin, termasuk *Random Forest Classifier* (Pedregosa et al., 2011). Dalam proyek ini, *Scikit-learn* digunakan untuk melatih model *Random Forest* menggunakan data monitoring perangkat. *Library* joblib digunakan untuk menyimpan dan memuat model yang telah dilatih dalam format .pkl, sehingga model dapat digunakan ulang oleh *Software Agent* tanpa perlu melatih ulang setiap kali agent dijalankan.

4. Laravel (PHP *Framework*)

   Laravel adalah *Framework* pengembangan aplikasi web berbasis PHP yang menggunakan pola arsitektur *Model-View-Controller* (MVC). Laravel menyediakan fitur-fitur lengkap seperti *routing* yang ekspresif, *Eloquent* ORM untuk interaksi database, sistem autentikasi bawaan, dan fitur *Mailable* untuk pengiriman email. Dalam proyek ini, Laravel digunakan sebagai *backend server* yang menerima data dari *Software Agent* melalui *REST API*, memproses data, menyimpan ke database MySQL, dan menyajikan informasi melalui antarmuka *dashboard* monitoring.

5. MySQL

   MySQL adalah sistem manajemen basis data relasional (*Relational Database Management System*/RDBMS) yang bersifat open source dan banyak digunakan dalam pengembangan aplikasi web. Dalam proyek ini, MySQL digunakan untuk menyimpan seluruh data sistem, mencakup data perangkat, log monitoring, hasil prediksi AI, jadwal *maintenance*, riwayat perubahan status, log pengiriman email, dan riwayat laporan yang dieksport.

6. Tailwind CSS dan *Bootstrap*

   Tailwind CSS adalah *Framework* CSS yang bersifat *utility-first* dan memungkinkan pengembang untuk membangun antarmuka pengguna secara langsung menggunakan kelas-kelas utilitas. Bootstrap adalah *Framework* frontend yang menyediakan komponen UI siap pakai. Kedua *Framework* ini digunakan dalam proyek ini untuk membangun antarmuka *dashboard* monitoring yang responsif, modern, dan mudah digunakan oleh administrator.

c) Profil Tempat Penerapan

PT. XYZ adalah perusahaan yang bergerak di bidang jasa konstruksi dan rekayasa lepas pantai (*offshore engineering and*

construction) untuk industri energi, khususnya minyak dan gas bumi. Perusahaan ini merupakan anak perusahaan dari salah satu konglomerat konstruksi energi bertaraf internasional dan telah beroperasi di Indonesia sejak awal tahun 1970-an, dengan fasilitas fabrikasi utama yang berlokasi di kawasan industri Batam, Kepulauan Riau. Dalam menjalankan operasionalnya, PT. XYZ mengoperasikan berbagai fasilitas produksi yang didukung oleh infrastruktur teknologi informasi yang cukup besar, mencakup ratusan perangkat komputer yang digunakan oleh karyawan di berbagai departemen, mulai dari *engineering*, *procurement*, *construction*, hingga divisi administrasi dan keuangan.

Departemen *IT Support* di PT. XYZ bertanggung jawab terhadap pemeliharaan dan pengelolaan seluruh perangkat komputer yang digunakan dalam operasional perusahaan. Mengingat skala operasional perusahaan yang besar dan lingkup proyek energi yang berlangsung secara berkesinambungan, ketersediaan perangkat komputer yang optimal menjadi faktor kritis dalam menjaga kelancaran seluruh aktivitas kerja. Kondisi yang melatarbelakangi proyek ini adalah bahwa proses monitoring kondisi perangkat komputer di PT. XYZ saat ini masih dilakukan secara manual dan reaktif, yaitu tim *IT Support* baru menangani perangkat ketika pengguna melaporkan adanya masalah. Belum terdapat sistem yang mampu memantau kondisi perangkat secara proaktif berdasarkan data metrik penggunaan sumber daya secara *Real-Time*.

Kondisi ini menimbulkan sejumlah permasalahan operasional, di antaranya:

- Tidak adanya mekanisme deteksi dini terhadap perangkat yang menunjukkan tanda-tanda beban tinggi atau kondisi tidak normal sebelum terjadi kegagalan sistem.
- Risiko *downtime* perangkat yang tidak terencana akibat kerusakan yang baru diketahui setelah terjadi, sehingga

dapat mengganggu produktivitas kerja karyawan dan kelangsungan operasional proyek.

- Tidak tersedianya data historis kondisi perangkat yang terstruktur dan dapat diakses secara terpusat oleh tim *IT Support* sebagai bahan evaluasi dan perencanaan maintenance.
- Proses penjadwalan *maintenance* yang masih dilakukan secara manual tanpa didukung analisis berbasis data kondisi aktual perangkat.

Berdasarkan kondisi tersebut, kebutuhan utama yang ingin diselesaikan melalui proyek ini adalah tersedianya sistem monitoring perangkat komputer berbasis *Artificial Intelligence* yang mampu mengumpulkan data kondisi perangkat secara *Real-Time* melalui *Software Agent*, memprediksi risiko kondisi perangkat menggunakan model machine learning, mengklasifikasikan tingkat kondisi perangkat menjadi *Normal*, *Warning*, atau *Critical*, serta secara otomatis membuat rekomendasi jadwal *preventive maintenance* berdasarkan hasil prediksi AI. Dengan demikian, sistem ini diharapkan dapat mengubah pendekatan pengelolaan perangkat komputer di PT. XYZ dari yang bersifat reaktif menjadi proaktif dan berbasis data, sehingga dapat mengurangi risiko *downtime* perangkat serta meningkatkan efektivitas manajemen infrastruktur teknologi informasi perusahaan 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 kebutuhan hingga pemeliharaan sistem.

Model *Waterfall* dipilih karena sistem yang dikembangkan memiliki kebutuhan yang relatif jelas dan stabil berdasarkan hasil observasi langsung

terhadap proses pengelolaan perangkat komputer di lingkungan *IT Support* PT. XYZ. Selain itu, pendekatan yang terstruktur dan berurutan memudahkan proses dokumentasi, pengendalian pengembangan, serta pemastian bahwa setiap tahapan diselesaikan secara tuntas sebelum melanjutkan ke tahap berikutnya. Adapun tahapan dalam metode *Waterfall* yang diterapkan pada penelitian ini adalah sebagai berikut:

1. *Requirement Analysis*: Tahap ini dilakukan untuk mengidentifikasi seluruh kebutuhan sistem berdasarkan kondisi pengelolaan perangkat komputer yang berjalan saat ini di PT. XYZ. Pengumpulan kebutuhan dilakukan melalui observasi langsung terhadap proses monitoring perangkat yang sedang berjalan, identifikasi permasalahan yang dihadapi tim IT Support, serta diskusi berkala dengan pembimbing lapangan dan pengguna sistem. Hasil dari tahap ini berupa :
    - Daftar kebutuhan fungsional sistem, mencakup monitoring kondisi perangkat secara *Real-Time*, prediksi kondisi berbasis AI, penjadwalan maintenance otomatis, pengiriman notifikasi email, dan pembuatan laporan PDF.
    - Penetapan parameter monitoring yang digunakan, yaitu persentase penggunaan CPU, RAM, Disk, dan suhu CPU perangkat.
    - Penentuan klasifikasi kondisi perangkat*: Normal*, *Warning*, dan *Critical.*
    - Penentuan *threshold labeling* dataset untuk pelatihan model AI.
    - Kebutuhan *non-fungsional* sistem, meliputi waktu respons, keamanan autentikasi, dan ketersediaan sistem.
2. *System Design* : Pada tahap ini dilakukan perancangan sistem secara menyeluruh berdasarkan kebutuhan yang telah dianalisis pada tahap sebelumnya. Perancangan mencakup seluruh komponen sistem, mulai dari arsitektur perangkat lunak hingga antarmuka pengguna. Perancangan meliputi :
    - Desain arsitektur sistem, mencakup komponen *Software Agent* Python (sisi *client*), server Laravel (backend), dan dashboard monitoring (*frontend*).

- Perancangan basis data (ERD) dengan 8 entitas: devices, device_logs, ai_predictions, maintenance_schedules, maintenance_histories, email_logs, reports, dan users.
- Pembuatan *Use Case Diagram*, *Activity Diagram*, dan *Sequence Diagram* untuk menggambarkan alur interaksi sistem.
- Perancangan antarmuka pengguna (*User Interface*) meliputi halaman *dashboard* monitoring, detail perangkat, jadwal maintenance, profil admin, dan laporan PDF.
- Perancangan alur kerja prediksi AI: pengambilan data oleh agent → prediksi lokal menggunakan model.pkl → pengiriman ke server → klasifikasi lanjutan → pembuatan jadwal otomatis.
- Perancangan dataset dan aturan pelabelan otomatis untuk pelatihan model *Random Forest* Classifier.

3. *Implementation* : Tahap implementasi dilakukan dengan mengembangkan sistem berdasarkan desain yang telah dibuat. Sistem dibangun menggunakan bahasa pemrograman PHP dengan *Framework* Laravel untuk sisi server, Python untuk pengembangan *Software Agent* dan model AI, serta MySQL sebagai sistem manajemen basis data. Pada tahap ini dilakukan implementasi seluruh komponen sistem, meliputi:
    - Pengembangan *Software Agent* Python (agent.py) yang berjalan secara otonom di sisi *client* untuk mengumpulkan data monitoring perangkat menggunakan *Library psutil* dan melakukan prediksi AI menggunakan model.pkl.
    - Pelatihan model *Random Forest Classifier* menggunakan *Scikit-learn* berdasarkan dataset monitoring perangkat yang diekspor dari sistem (*train_model*.py).
    - Pengembangan *REST API* pada server Laravel untuk menerima data dari *Software Agent* melalui endpoint POST /api/device/store.
    - Implementasi logika klasifikasi kondisi perangkat di server: *Normal, Warning,* dan *Critical* berdasarkan nilai *ml_prediction* dan rata-rata 50 log terakhir.

- Implementasi fitur penjadwalan *maintenance* otomatis berdasarkan deteksi kondisi berulang pada 10 log terakhir *(HIGH risk*: ≥5 log *Critical, MEDIUM risk*: ≥7 log *Warning*).
- Pengembangan dashboard *monitoring* berbasis web dengan fitur pencarian, detail perangkat, jadwal *maintenance*, pengiriman email notifikasi, dan *export* laporan PDF.
- Pencatatan seluruh aktivitas sistem ke tabel terkait: riwayat prediksi AI, riwayat perubahan status *maintenance*, dan riwayat pengiriman email.

4. *Testing* : Tahap pengujian dilakukan untuk memastikan sistem berjalan sesuai dengan kebutuhan yang telah ditentukan. Dua metode pengujian digunakan secara bersamaan untuk memastikan kualitas sistem secara fungsional maupun dari perspektif pengguna akhir. Pengujian dilakukan pada seluruh fungsi utama sistem, meliputi:
    - Proses autentikasi: *login, logout*, dan hak akses halaman.
    - Pengiriman dan penerimaan data monitoring dari *Software Agent* melalui API.
    - Mekanisme prediksi AI dan klasifikasi kondisi perangkat (*Normal/Warning/Critical*).
    - Pembuatan jadwal *maintenance* otomatis berdasarkan hasil deteksi kondisi berulang.
    - Pengelolaan jadwal: konfirmasi tanggal, ubah status, dan *reschedule* otomatis saat status *missed*.
    - Pengiriman email notifikasi *maintenance* kepada pemilik perangkat dan pencatatan ke email_logs.
    - Export laporan monitoring dalam format PDF dan pencatatan ke tabel reports.
    - Pengelolaan profil admin: *update* data, ganti *password*, dan *upload* foto profil.
5. *Deployment* : Tahap ini merupakan proses penerapan sistem ke lingkungan operasional agar dapat digunakan oleh pengguna secara nyata. Sistem dijalankan pada server lokal yang terhubung dalam jaringan internal PT.

XYZ, sehingga *dashboard* monitoring dapat diakses oleh administrator melalui browser pada alamat IP server yang telah dikonfigurasi. *Software Agent* Python diinstal dan dijalankan pada setiap perangkat *client* yang ingin dimonitor menggunakan file *start_agent.bat*, sehingga proses monitoring perangkat dapat berjalan secara otomatis dan berkelanjutan tanpa memerlukan intervensi manual dari pengguna.

6. *Maintenance* **:** Melakukan pemeliharaan dan pembaruan sistem secara berkala untuk memperbaiki kesalahan yang ditemukan pasca-implementasi serta menyesuaikan sistem dengan kebutuhan baru pengguna. Tahap pemeliharaan yang dilakukan meliputi:
    - Perbaikan bug dan kesalahan yang teridentifikasi selama penggunaan sistem.
    - Penyesuaian *threshold* klasifikasi kondisi perangkat berdasarkan evaluasi akurasi deteksi di lapangan.
    - Pelatihan ulang model AI (*retraining*) menggunakan data monitoring terbaru apabila distribusi data kondisi perangkat berubah secara signifikan.
    - Penambahan fitur atau penyesuaian antarmuka berdasarkan umpan balik pengguna yang dikumpulkan selama penggunaan sistem.

2.4. Pengujian *Fungsional*

Pengujian *fungsional* dilakukan untuk memastikan bahwa setiap fitur pada Sistem Prediksi dan Penjadwalan *Preventive Maintenance* Perangkat Komputer Berbasis *Artificial Intelligence* bekerja sesuai dengan kebutuhan pengguna yang telah ditetapkan pada tahap analisis. Pengujian ini berfokus pada keluaran sistem berdasarkan masukan yang diberikan tanpa memperhatikan proses internal program. Dua metode pengujian digunakan secara bersamaan dalam proyek ini untuk memastikan kualitas sistem dari dua perspektif yang berbeda, yaitu pengujian teknis fungsional dan penerimaan pengguna akhir.

1. *Black Box Testing*

    Black Box Testing adalah metode pengujian perangkat lunak yang berfokus pada fungsi eksternal sistem tanpa memperhatikan struktur kode

program internal (Myers et al., 2011). Pengujian dilakukan dengan memberikan data masukan (input) ke dalam sistem dan membandingkan hasil keluaran *(output)* yang dihasilkan dengan hasil yang diharapkan. Apabila keluaran sesuai dengan yang diharapkan maka fungsi dinyatakan berhasil (*valid*), sedangkan jika tidak sesuai maka fungsi dinyatakan gagal (*invalid*). Aspek-aspek *fungsional* yang diuji menggunakan metode *Black Box Testing* pada sistem ini meliputi:

- Fungsi autentikasi: proses *login* dengan kredensial *valid* dan *invalid*, proses *logout*, serta pembatasan akses halaman bagi pengguna yang belum *login*.
- Fungsi monitoring *Real-Time*: penerimaan data dari *Software Agent* melalui *REST API*, penyimpanan data ke tabel *device_logs*, dan pembaruan status perangkat secara otomatis berdasarkan hasil prediksi AI.
- Fungsi prediksi AI dan klasifikasi kondisi: pengiriman nilai *ml_prediction* dari *agent*, klasifikasi lanjutan server menjadi *Normal/Warning/Critical*, dan pembaruan status perangkat di database.
- Fungsi penjadwalan *maintenance* otomatis: pembuatan jadwal LOW *risk maintenance* rutin dengan logika hari ini +30 hari, , jadwal *MEDIUM risk* saat terdeteksi ≥7 log *Warning* pada 10 log terakhir dengan logika hari ini +5 hari, dan *HIGH risk* saat terdeteksi ≥5 log *Critical* dengan logika hari ini 2 hari.
- Fungsi pengelolaan jadwal *maintenance*: konfirmasi tanggal pelaksanaan, perubahan status (pending → *completed/missed*), dan pembuatan jadwal *reschedule* otomatis saat status *missed*.
- Fungsi pengiriman email notifikasi: pengiriman email ke pemilik perangkat, validasi ketersediaan email dan tanggal konfirmasi, serta pencatatan ke tabel *email_logs* dengan status *Sent* atau *Failed*.

- Fungsi *export* laporan PDF: pembuatan laporan yang memuat ringkasan monitoring, prediksi AI, dan statistik *maintenance*, serta pencatatan ke tabel *reports*.
- Fungsi pengelolaan profil admin: pembaruan data identitas, penggantian *password* dengan validasi *password* lama, dan *upload* foto profil.
- Fungsi pencarian perangkat berdasarkan *hostname* pada halaman dashboard.

2. *User Acceptance Testing* (UAT)

   User Acceptance Testing (UAT) adalah metode pengujian yang dilakukan oleh pengguna akhir untuk memverifikasi bahwa sistem yang dikembangkan telah memenuhi kebutuhan dan ekspektasi mereka dalam kondisi penggunaan nyata. UAT bertujuan untuk memastikan bahwa sistem tidak hanya berfungsi secara teknis, tetapi juga mudah digunakan dan sesuai dengan alur kerja pengguna sesungguhnya. Pengujian UAT pada sistem ini dilakukan bersama pengguna yang mewakili tim *IT Support* di PT. XYZ dengan menggunakan skenario penggunaan nyata. Aspek yang dievaluasi meliputi:

   - Kemudahan penggunaan (*usability*) antarmuka *dashboard* monitoring dalam membaca kondisi perangkat secara *Real-Time*.
   - Kejelasan informasi status perangkat (*Normal/Warning/Critical*) dan hasil prediksi AI yang ditampilkan di *dashboard*.
   - Kemudahan proses konfirmasi jadwal *maintenance*, perubahan status, dan pengiriman email notifikasi kepada pemilik perangkat.
   - Kesesuaian tampilan dan isi laporan PDF yang dihasilkan sebagai dokumentasi kondisi perangkat.
   - Kepuasan pengguna secara keseluruhan terhadap fitur-fitur yang tersedia dan kemudahan navigasi sistem.

BAB III
ANALISIS DAN PERANCANGAN

Analisis dan perancangan pada penelitian ini difokuskan untuk menghasilkan sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis Artificial Intelligence (AI) yang sesuai dengan kebutuhan operasional tim IT Support. Perancangan dilakukan agar sistem yang dibagun mampu membantu proses monitoring kondisi perangkat secara *Real-Time*, mendeteksi potensi kerusakan lebih awal, serta memberikan rekomendasi jadwal maintenance secara otomatis berdasarkan hasil analisis *machine learning*.

Setiap tahapan analisis diarahkan untuk mengindentifikasi kebutuhan fungsional maupun non-fungsional sistem, sehingga website yang dihasilkan tidak hanya memenuhi aspek teknis, tetapi juga mampu meningkatkan efisiensi kerja, mengurangi Risiko *downtime*, serta mendukung pengelolaan aset perangkat komputer secara lebih terstruktur dan berbasis data. Sistem ini juga dirancang dengan mempertimbangkan integrasi antara website berbasis laravel, modul Artificial Intelligence menggunakan Python dan Scikit-Learn, serta *Software Agent* monitoring yang berjalan pada perangkat pengguna untuk mengambil data kondisi perangkat secara *Real-Time* seperti penggunaan CPU dan RAM.

Selain itu, perancangan sistem juga mempertimbangkan aspek keamanan data, kemudahan penggunaan antarmuka, efisiensi pengiriman data monitoring, serta kemampuan sistem dalam melakukan pengolahan data secara otomatis. Dengan demikian, sistem yang dibangun diharapkan dapat berfungsi secara optimal sebagai Solusi *preventive maintenance* modern yang mendukung kegiatan operasional *IT Support* secara lebih proaktif, efektif, dan berkelanjutan.

3.1. Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk memahami kondisi sistem yang sedang berjalan serta menentukan kebutuhan sistem yang akan dikembangkan. Proses analisis ini bertujuan agar sistem yang dibangun mampu menjadi solusi terhadap permasalahan maintenance perangkat komputer pada departemen *IT Support* secara lebih efektif, efisien, dan terstruktur. Pada kondisi saat ini, proses maintenance perangkat komputer masih dilakukan secara manual dan bersifat

reaktif. Tim *IT Support* umumnya melakukan pemeriksaan atau perbaikan setelah pengguna melaporkan adanya kerusakan pada perangkat yang digunakan. Proses monitoring kondisi perangkat juga belum dilakukan secara *Real-Time* dan terpusat, sehingga tim *IT Support* kesulitan mengetahui kondisi perangkat secara langsung tanpa melakukan pengecekan manual pada masing-masing perangkat.

Selain itu, pencatatan Riwayat maintenance perangkat masih dilakukan secara sederhana dan belum terintegrasi dalam satu sistem. Hal tersebut menyebabkan proses pelacakan histori kerusakan, penjadwalan maintenance, serta pembuatan laporan perangkat menjadi kurang efektif. Kondisi tersebut sering menyebabkan terjadinya kerusakan mendadak yang berdampak pada terganggunya aktivitas operasional pengguna perangkat komputer.

Berdasarkan permasalahan tersebut, diperlukan sebuah sistem yang mampu melakukan monitoring perangkat secara *Real-Time*, memprediksi potensi kerusakan perangkat menggunakan Artificial Intelligence (AI), serta menghasilkan jadwal *preventive maintenance* secara otomatis. Sistem yang dikembangkan berbasis website menggunakan *Framework* Laravel dan terintegrasi dengan modul *machine learning* berbasis python menggunakan *Library* Scikit-learn. Selain itu, sistem juga menggunakan *Software Agent* berbasis python yang berjalan pada masing-masing perangkat komputer untuk mengambil data monitoring perangkat secara *Real-Time* menggunakan *Library psutil*.

Dengan adanya sistem ini, proses maintenance diharapkan menjadi lebih proaktif, efisien, dan berbasis data *Real-Time* sehingga dapat membantu mengurangi Risiko *downtime* perangkat serta meningkatkan efektivitas kerja tim IT Support.

3.1.1. Proses Bisnis Berjalan

Sebelum sistem ini dikembangkan, proses maintenance perangkat komputer pada department *IT Support* masih dilakukan secara manual dan reaktif. Ketika terjadi kerusakan pada perangkat komputer, pengguna harus melaporkan kendala yang dialami kepada tim IT Support. Setelah menerima laporan, tim *IT Support* melakukan pengecekan perangkat secara langsung untuk mengetahui penyebab kerusakan dan melakukan Tindakan perbaikan.

Proses monitoring kondisi perangkat seperti pengunaan CPU, RAM, kapasitas disk, dan temperatur perangkat belum dilakukan secara otomatis maupun terpusat. Tim *IT Support* harus melakukan pengecekan manual pada masing-masing perangkat apabila ingin mengetahui kondisi perangkat secara detail. Hal ini menyebabkan proses monitoring menjadi kurang efisien, terutama ketika jumlah perangkat yang digunakan cukup banyak.

Selain itu, proses penjadwalan maintenance masih dilakukan berdasarkan perkiraan atau kondisi perangkat yang terlihat bermasalah. Tidak adanya sistem prediksi dini menyebabkan maintenance sering terlambat dilakukan, sehingga beberapa perangkat mengalami kerusakan mendadak yang berdampak pada aktivitas operasional pengguna.

Permasalahan lain terdapat pada proses dokumentasi dan pelaporan maintenance yang masih dilakukan secara sederhana. Riwayat perawatan perangkat belum tersimpan secara terpusat sehingga menyulitkan proses pelacakan histori kerusakan maupun evaluasi kondisi perangkat dalam jangka Panjang.

3.1.2. Gambaran Umum Sistem

Sistem yang dikembangkan merupakan website monitoring dan *preventive maintenance* perangkat komputer berbasis Artificial Intelligence (AI). Sistem ini dirancang untuk membantu tim *IT Support* dalam melakukan monitoring kondisi perangkat secara *Real-Time*, memprediksi potensi kerusakan perangkat, serta menghasilkan jadwal maintenance otomatis berdasarkan hasil analisis *machine learning*.

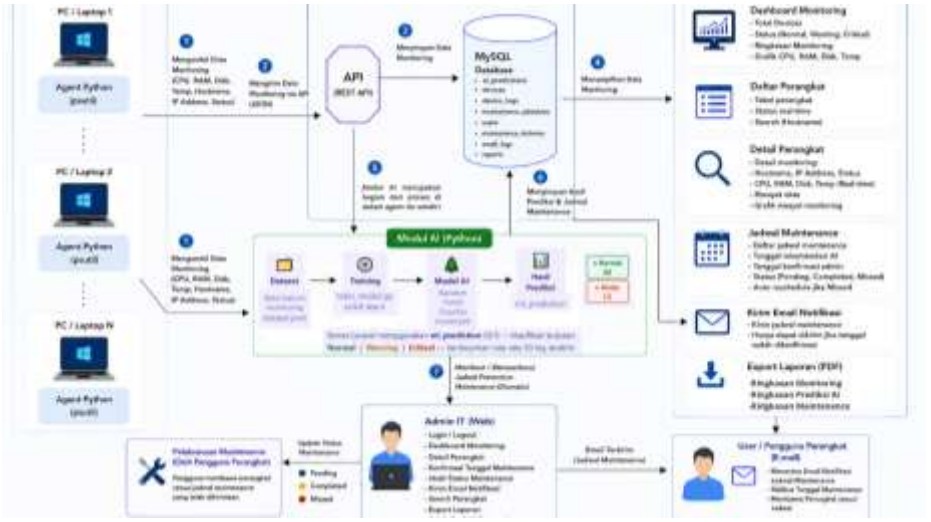
Sistem dibangun menggunakan *Framework* Laravel sebagai *backend* utama dan menggunakan Laravel blade serta Tailwind CSS untuk membangun tampilan antarmuka website. Sedangkan proses Artificial Intelligence dan *machine learning* dikembangkan menggunakan bahasa pemograman Python dengan *Library Scikit-learn* menggunakan algoritma *Decision Tree* dan *Random Forest* Classifier. Pada sistem ini, setiap perangkat komputer akan dipasangkan *Software Agent* berbasis Python yang berjalan secara otomatis di background perangkat. *Software Agent* bertugas mengambil data kondisi perangkat secara

Real-Time menggunakan *Library psutil*, seperti penggunaan CPU, RAM, Kapasitas Disk, dan temperatur perangkat.

Data monitoring yang diperoleh kemudian dikirimkan ke server website melalui API berbasis *JSON* dan disimpan ke dalam database MySQL. Selanjutnya, data tersebut diproses menggunakan model *machine learning* untuk menghasilkan prediksi Tingkat Risiko kerusakan perangkat. Hasil prediksi ditampilkan pada *dashboard* monitoring dalam bentuk status perangkat seperti normal, warning, dan critical. Sistem juga secara otomatis menghasilkan jadwal *preventive maintenance* berdasarkan hasil prediksi AI.

Website menyediakan beberapa fitur utama, seperti *dashboard* monitoring perangkat, halaman detail perangkat, jadwal maintenance otomatis, pengiriman email notifikasi maintenance kepada pengguna perangkat, serta fitur export laporan monitoring dan maintenance perangkat. Dengan adanya sistem ini, proses maintenance perangkat dapat dilakukan secara lebih proaktif, terstruktur, dan berbasis data *Real-Time* sehingga membantu mengurangi Risiko kerusakan mendadak dan meningkatkan efisiensi kerja tim IT Support.

Contoh gambaran umum sistem dapat dilihat pada Gambar 3.1



Gambar 3.1 Contoh Gambaran Umum Sistem

3.2. Perancangan

Perancangan sistem pada penelitian ini menggunakan metode pengembangan *Waterfall*. Metode *Waterfall* dipilih karena proses pengembangan sistem dilakukan secara terstruktur dan beruntun mulai dari tahap analisis

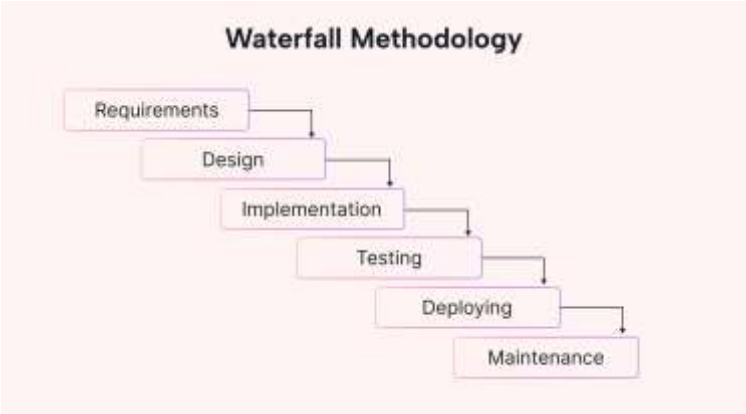
kebutuhan, perancangan sistem, implementasi, pengujian, hingga pemeliharaan sistem. Selain itu, kebutuhan sistem pada penelitian ini telah ditentukan sejak awal sehingga metode *Waterfall* dinilai sesuai untuk mendukung proses Pembangunan sistem secara sistematis.

Sistem yang dikembangkan merupakan website prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis Artificial Intelligence (AI). Sistem ini dibangun menggunakan *Framework* Laravel sebagai *backend* dan *frontend* website, Python dengan *Library Scikit-learn* untuk proses *machine learning*, serta *Software Agent* berbasis Python untuk melakukan monitoring perangkat secara *Real-Time*.

Pada sistem ini, *Software Agent* akan mengambil data perangkat secara otomatis seperti penggunaan CPU, RAM, Disk, dan temperatur perangkat menggunakan *Library psutil*. Data tersebut kemudian dikirimkan ke server Laravel melalui API untuk disimpan ke database MySQL. Selanjutnya, model *machine learning* akan menganalisis data perangkat dan menghasilkan prediksi tingkat risiko kerusakan perangkat. Berdasarkan hasil prediksi tersebut, sistem akan membuat rekomendasi jadwal *preventive maintenance* secara otomatis.

Dengan adanya sistem ini, proses maintenance tidak lagi dilakukan secara manual dan reaktif, melainkan menjadi lebih terstruktur, proaktif, dan berbasis data *Real-Time*. Sistem juga menyediakan *dashboard* monitoring perangkat, detail penggunaan *resource* perangkat, notifikasi status perangkat, serta fitur pengiriman email jadwal maintenance kepada pengguna perangkat.

Pada sub bab berikut akan dibahas mengenai kebutuhan fungsional dan non-fungsional yang harus dipenuhi oleh sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis Artificial Intelligence.



Gambar 3.2 Metode yang digunakan

3.2.1. Kebutuhan Fungsional

Kebutuhan Fungsional merupakan fungsi utama yang harus dimiliki sistem agar dapat berjalan sesuai tujuan penelitian. Berikut adalah kebutuhan fungsional pada sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis Artificial Intelligence.

Tabel 3.1 Kebutuhan Fungsional

Kode	Functional Requirement
FR-01	Superadmin, Admin, dan Staff IT dapat melakukan login ke dalam sistem monitoring perangkat menggunakan email dan password yang telah terdaftar.
FR-02	Superadmin, Admin, dan Staff IT dapat melakukan logout dari sistem.
FR-03	Superadmin dan Admin dapat melihat dashboard monitoring seluruh perangkat yang menampilkan kondisi perangkat secara real-time.
FR-04	Staff IT dapat melihat dashboard monitoring perangkat miliknya sendiri secara real-time.
FR-05	Superadmin dan Admin dapat melihat detail monitoring setiap perangkat yang meliputi CPU, RAM, Disk, Temperatur, IP Address, Hostname, hasil prediksi AI, dan riwayat monitoring.
FR-06	Staff IT dapat melihat informasi perangkat miliknya sendiri beserta riwayat monitoring perangkat tersebut.
FR-07	Superadmin dan Admin dapat menambahkan serta mengubah email pengguna perangkat sebagai tujuan pengiriman notifikasi maintenance.
FR-08	Superadmin dan Admin dapat melihat seluruh jadwal preventive maintenance perangkat.
FR-09	Staff IT dapat melihat jadwal maintenance perangkat miliknya sendiri.
FR-10	Superadmin dan Admin dapat melakukan konfirmasi jadwal

	preventive maintenance.
FR-11	Superadmin dan Admin dapat mengubah status maintenance menjadi Pending, Completed, atau Missed.
FR-12	Superadmin dan Admin dapat mengirim email notifikasi maintenance kepada pengguna perangkat setelah jadwal dikonfirmasi.
FR-13	Superadmin dan Admin dapat menghapus data jadwal maintenance perangkat.
FR-14	Superadmin dan Admin dapat menghasilkan laporan monitoring perangkat dalam format PDF berdasarkan filter bulan dan tahun serta laporan per perangkat.
FR-15	Superadmin dan Admin dapat melakukan pencarian perangkat berdasarkan hostname.
FR-16	Superadmin, Admin, dan Staff IT dapat mengelola profil akun masing-masing, meliputi perubahan nama, email, foto profil, dan password.
FR-17	Superadmin dapat mengelola akun pengguna sistem dengan menambahkan maupun menghapus akun Admin dan Staff IT melalui halaman Manajemen User.
FR-18	Software Agent secara otomatis mengambil data CPU, RAM, Disk, Temperatur, Hostname, dan IP Address dari perangkat client menggunakan library psutil, kemudian mengirimkan data monitoring ke server melalui REST API.
FR-19	Sistem secara otomatis melakukan klasifikasi kondisi perangkat menggunakan algoritma Random Forest Classifier sehingga menghasilkan status Normal, Warning, atau Critical berdasarkan data monitoring yang diterima.
FR-20	Sistem secara otomatis membuat jadwal preventive maintenance berdasarkan hasil prediksi AI dan menyimpan seluruh data monitoring, hasil prediksi, serta riwayat maintenance ke dalam database.

3.2.2. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan kebutuhan pendukung yang berkaitan dengan kualitas sistem, keamanan, performa, dan keandalan sistem yang dikembangkan.

Tabel 3.2 Kebutuhan Non-Fungsional

Kode	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 sensitive dengan menerapkan validasi input.

3.2.3. Diagram *Use Case*

Diagram *Use Case* merupakan representasi visual yang menggambarkan interaksi antara aktor dengan sistem, serta fungsi-fungsi utama yang dapat dijalankan oleh masing-masing aktor. Diagram ini membantu memberikan gambaran mengenai bagaimana pengguna berinteraksi dengan sistem, hak akses yang dimiliki, serta layanan yang tersedia pada sistem. Dengan demikian, *Use Case* diagram berfungsi sebagai penghubung antara kebutuhan pengguna dengan rancangan teknis sistem yang akan dibangun. Berikut merupakan *Use Case* diagram untuk sistem prediksi dan penjadwalan maintenance *preventive maintenance* perangkat komputer berbasis Artificial Intelligence pada studi kasus lingkungan IT Support.

Gambar 3.3 Gambar *Use Case*

Sistem monitoring dan penjadwalan *preventive maintenance* perangkat komputer berbasis *Artificial Intelligence* yang digambarkan pada *Use Case Diagram* di atas memiliki **tiga aktor utama**, yaitu **Superadmin**, **Admin**, dan **Staff IT**. Ketiga aktor tersebut memiliki hak akses yang berbeda sesuai dengan tugas dan tanggung jawab masing-masing. Selain itu, sistem juga terintegrasi dengan **Software Agent** berbasis Python yang bertugas melakukan pengambilan data monitoring perangkat secara otomatis. Sistem dirancang untuk melakukan monitoring perangkat secara *real-time*, memprediksi kondisi perangkat menggunakan algoritma *Random Forest Classifier*, membuat jadwal *preventive maintenance* secara otomatis, mengirimkan notifikasi melalui email, serta menghasilkan laporan monitoring dalam format PDF.

Superadmin merupakan aktor dengan hak akses tertinggi pada sistem. Superadmin dapat melakukan login ke dalam sistem untuk mengakses seluruh fitur yang tersedia. Setelah berhasil login, Superadmin dapat melihat dashboard monitoring yang menampilkan seluruh perangkat yang sedang dimonitor beserta informasi hostname, IP Address, penggunaan CPU, RAM, Disk, temperatur perangkat, serta status hasil prediksi Artificial Intelligence yang diklasifikasikan menjadi **Normal**, **Warning**, atau **Critical**. Data tersebut diperoleh secara *real-time* dari Software Agent yang terpasang pada setiap perangkat.

Selain melakukan monitoring perangkat, Superadmin dapat melihat detail monitoring setiap perangkat sehingga memperoleh informasi yang lebih lengkap mengenai kondisi perangkat beserta riwayat monitoringnya. Superadmin juga dapat menambahkan maupun mengubah alamat email pemilik perangkat sebagai tujuan pengiriman notifikasi maintenance.

Pada proses pengelolaan maintenance, Superadmin memiliki hak untuk melihat daftar jadwal maintenance yang dihasilkan secara otomatis berdasarkan hasil prediksi Artificial Intelligence. Jadwal maintenance diberikan berdasarkan tingkat risiko perangkat, yaitu perangkat dengan status **Critical** memperoleh jadwal maintenance dalam waktu **2 hari**, status **Warning** memperoleh jadwal dalam waktu **5 hari**, sedangkan perangkat dengan status **Normal** tetap memperoleh jadwal maintenance rutin setiap **30 hari** sebagai tindakan preventif. Superadmin dapat melakukan konfirmasi jadwal maintenance, mengubah status maintenance menjadi **Pending**, **Completed**, atau **Missed**, serta menghapus data jadwal maintenance apabila sudah tidak diperlukan. Apabila status maintenance diubah menjadi **Missed**, sistem akan secara otomatis membuat jadwal maintenance baru sesuai aturan penjadwalan yang berlaku.

Superadmin juga dapat mengirimkan email notifikasi kepada pengguna perangkat setelah jadwal maintenance dikonfirmasi. Email tersebut berisi informasi perangkat, tanggal pelaksanaan maintenance, tingkat risiko perangkat, serta rekomendasi tindakan maintenance yang harus dilakukan. Selain itu,

Superadmin dapat melakukan pencarian perangkat berdasarkan hostname, mengunduh laporan monitoring perangkat dalam format PDF berdasarkan filter bulan maupun tahun, mengelola profil akun, serta melakukan logout dari sistem.
Selain seluruh hak akses yang dimiliki Admin, **Superadmin** juga memiliki hak khusus untuk mengelola akun pengguna sistem melalui menu **Manajemen User**. Pada menu ini Superadmin dapat menambahkan akun **Admin** maupun **Staff IT**, menentukan role pengguna, mengubah data akun apabila diperlukan, serta menghapus akun yang sudah tidak digunakan. Fitur ini bertujuan untuk memudahkan pengelolaan pengguna sistem tanpa perlu melakukan perubahan pada source code aplikasi.

Admin merupakan aktor yang memiliki hak akses terhadap seluruh fitur monitoring dan maintenance perangkat, namun tidak memiliki hak untuk mengelola akun pengguna. Setelah melakukan login, Admin dapat melihat dashboard monitoring, melihat detail monitoring perangkat, mengelola email pengguna perangkat, melihat jadwal maintenance, melakukan konfirmasi jadwal maintenance, mengubah status maintenance, mengirimkan email notifikasi maintenance kepada pengguna perangkat, menghapus data jadwal maintenance, melakukan pencarian perangkat berdasarkan hostname, mengunduh laporan monitoring dalam format PDF, mengelola profil akun, serta melakukan logout. Dengan demikian, Admin bertanggung jawab terhadap seluruh aktivitas operasional monitoring dan maintenance perangkat tanpa memiliki hak untuk mengelola data pengguna sistem.

Staff IT merupakan pengguna yang menggunakan perangkat komputer yang dimonitor oleh sistem. Setelah melakukan login, Staff IT hanya dapat mengakses halaman **Perangkat Saya** yang berisi informasi kondisi perangkat miliknya sendiri. Pada halaman tersebut sistem menampilkan penggunaan CPU, RAM, Disk, temperatur perangkat, status hasil prediksi Artificial Intelligence, identitas perangkat berupa hostname dan IP Address, riwayat monitoring perangkat secara *real-time*, serta informasi jadwal maintenance apabila tersedia. Staff IT tidak memiliki hak untuk mengubah data monitoring maupun melakukan pengelolaan maintenance. Hak akses yang diberikan hanya sebatas melihat kondisi perangkat dan mengetahui jadwal maintenance yang telah ditentukan oleh Admin atau Superadmin.

Seluruh proses monitoring perangkat dilakukan oleh **Software Agent** yang berjalan pada masing-masing komputer pengguna. Software Agent dikembangkan menggunakan bahasa pemrograman **Python** dengan memanfaatkan library **psutil** untuk mengambil informasi penggunaan CPU, RAM, Disk, temperatur perangkat, hostname, serta alamat IP secara *real-time*. Data monitoring tersebut kemudian diproses menggunakan model **Machine Learning Random Forest Classifier** yang telah dilatih sebelumnya untuk menghasilkan klasifikasi kondisi perangkat menjadi **Normal**, **Warning**, atau **Critical**. Selanjutnya hasil monitoring beserta hasil prediksi dikirimkan secara otomatis ke server Laravel melalui REST API

dan disimpan ke dalam database MySQL sebagai data monitoring serta riwayat perangkat.

Sistem juga menyediakan fitur pelaporan monitoring perangkat yang dapat diunduh dalam format PDF. Pengguna dengan hak akses Superadmin maupun Admin dapat memilih periode laporan berdasarkan bulan dan tahun sehingga laporan yang dihasilkan sesuai dengan kebutuhan. Laporan tersebut memuat ringkasan jumlah perangkat, jumlah perangkat berdasarkan status Artificial Intelligence, data monitoring perangkat, riwayat maintenance, serta catatan rekomendasi tindakan maintenance untuk setiap perangkat yang dimonitor.

Secara keseluruhan, sistem monitoring dan penjadwalan *preventive maintenance* perangkat komputer berbasis *Artificial Intelligence* ini dirancang untuk membantu proses monitoring kondisi perangkat secara otomatis, *real-time*, dan terintegrasi. Integrasi antara website Laravel, Software Agent berbasis Python, REST API, database MySQL, serta algoritma **Random Forest Classifier** memungkinkan sistem melakukan deteksi dini terhadap potensi kerusakan perangkat sehingga jadwal *preventive maintenance* dapat dibuat secara otomatis sesuai tingkat risiko perangkat. Dengan adanya fitur manajemen pengguna, monitoring *real-time*, pengiriman notifikasi email, serta pelaporan monitoring, sistem ini mampu meningkatkan efektivitas proses pemeliharaan perangkat komputer sehingga risiko *downtime* dapat diminimalkan dan kinerja tim IT menjadi lebih optimal.

3.2.4. Skenario *Use Case*

Skenario *Use Case* merupakan penjelasan mengenai alur interaksi antara aktor dengan sistem dalam menjalankan fungsi atau fitur tertentu. Skenario ini digunakan untuk menjelaskan atau menggambarkan proses sistem bekerja secara rinci mulai dari kondisi awal, aksi yang dilakukan aktor, hingga respon yang diberikan oleh sistem. Pada sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer berbasis Artificial Intelligence ini terdapat dua aktor utama yaitu Admin / *IT Support* dan Pengguna Device.

a. Skenario *Use Case* Login

Tabel 3.3 Use Case Skenario Login

Use Case Scenario 1 : "Login"	
Nomor	UC-01
Nama	Login
Deskripsi	Superadmin, Admin, atau Staff IT memasukkan email dan password untuk masuk ke dalam sistem monitoring perangkat.
Primary Actor	Superadmin, Admin, Staff IT
Preconditions	Pengguna telah memiliki akun yang terdaftar pada sistem.
Postconditions	Pengguna berhasil masuk ke halaman utama sesuai dengan hak aksesnya.
Main Success Scenario	1. Pengguna membuka halaman login. 2. Pengguna memasukkan email dan password. 3. Pengguna menekan tombol **Login.** 4. Sistem melakukan validasi email dan password. 5. Sistem memverifikasi role pengguna. 6. Sistem mengarahkan pengguna ke dashboard sesuai hak aksesnya.
Alternative Flow (Extension)	1. Pengguna memasukkan email atau password yang salah. 2. Sistem menampilkan pesan **"Email atau Password salah."** 3. Pengguna diminta mengisi kembali data login yang benar.dengan benar.

b. Skenario *Use Case* Melihat *Dashboard* Monitoring Perangkat

Tabel 3.4 *Use Case* Skenario Melihat *Dashboard* Monitoring Perangkat

Use Case Scenario 2 : "Melihat *dashboard* monitoring perangkat"	
Nomor	UC-02
Nama	Melihat *Dashboard* Monitoring Perangkat
Deskripsi	Superadmin dan Admin melihat kondisi seluruh perangkat yang dimonitor secara real-time.

Primary Actor	Superadmin, Admin
Preconditions	Pengguna telah berhasil login.
Postconditions	Informasi monitoring seluruh perangkat berhasil ditampilkan.
Main Success Scenarios	1. Pengguna membuka menu Dashboard. 2. Sistem mengambil data monitoring terbaru dari database. 3. Sistem menampilkan daftar perangkat beserta status CPU, RAM, Disk, Temperatur, dan hasil prediksi AI. 4. Pengguna dapat memilih perangkat untuk melihat detail monitoring.
Alternative Flow (Extension)	1. Data monitoring belum tersedia. 2. Sistem menampilkan pesan **"Belum ada data monitoring."**

c. Skenario *Use Case* Dashboard Perangkat Staff

Tabel 3.5 *Use Case* Skenario Dashboard Perangkat Staff

Use Case Scenario 3 : "Dashboard Perangkat Staff"	
Nomor	UC-03
Nama	Melihat Dashboard Perangkat Saya
Deskripsi	Staff IT melihat kondisi perangkat miliknya secara real-time.
Primary Actor	Staff IT
Preconditions	Staff IT telah berhasil login dan memiliki perangkat yang terdaftar.
Postconditions	Informasi monitoring perangkat berhasil ditampilkan.
Main Success Scenario	1. Staff IT membuka halaman Dashboard. 2. Sistem mengambil data perangkat yang dimiliki Staff IT.

	3. Sistem menampilkan informasi CPU, RAM, Disk, Temperatur, status AI, hostname, IP Address, email pemilik, serta riwayat monitoring perangkat. 4. Sistem menampilkan jadwal maintenance perangkat apabila tersedia.
Alternative Flow (Extension)	1. Perangkat belum terdaftar pada sistem. 2. Sistem menampilkan pesan **"Data perangkat tidak ditemukan."**

d. Skenario *Use Case* Detail Monitoring Perangkat

Tabel 3.6 *Use Case* Skenario Detail Monitoring Perangkat

Use Case Scenario 4 : "Detail Monitoring Perangkat"	
Nomor	UC-04
Nama	Melihat Detail Monitoring Perangkat
Deskripsi	Superadmin dan Admin melihat informasi detail monitoring suatu perangkat.
Primary Actor	Superadmin, Admin
Preconditions	Pengguna telah login dan memilih salah satu perangkat.
Postconditions	Detail monitoring perangkat berhasil ditampilkan.
Main Success Scenario	1. Pengguna memilih salah satu perangkat pada dashboard. 2. Sistem mengambil data monitoring perangkat dari database. 3. Sistem menampilkan informasi hostname, IP Address, CPU, RAM, Disk, Temperatur, hasil prediksi AI, riwayat monitoring, serta informasi maintenance perangkat.
Alternative Flow (Extension)	1. Data perangkat tidak ditemukan. 2. Sistem menampilkan pesan **"Perangkat tidak ditemukan."**

e. Skenario *Use Case* Informasi dan Riwayat Monitoring Perangkat Staff

Tabel 3.7 *Use Case* Skenario Informasi dan Riwayat Monitoring Perangkat Staff

Use Case Scenario 5 : "Informasi dan Riwayat Monitoring Perangkat Staff"	
Nomor	UC-05
Nama	Melihat Informasi dan Riwayat Monitoring Perangkat Saya
Deskripsi	Staff IT melihat informasi detail perangkat beserta riwayat monitoring perangkat yang dimilikinya.
Primary Actor	Staff IT
Preconditions	Staff IT telah berhasil login dan perangkat telah terdaftar pada sistem.
Postconditions	Informasi perangkat beserta riwayat monitoring berhasil ditampilkan.
Main Success Scenario	1. Staff IT membuka menu Perangkat Saya. 2. Sistem mengambil data perangkat berdasarkan akun Staff IT. 3. Sistem menampilkan informasi hostname, IP Address, email pemilik, CPU, RAM, Disk, Temperatur, status AI, dan riwayat monitoring perangkat secara kronologis. 4. Staff IT dapat melihat perubahan kondisi perangkat berdasarkan data monitoring yang tersimpan.
Alternative Flow (Extension)	1. Belum terdapat data monitoring pada perangkat. 2. Sistem menampilkan pesan **"Belum ada data monitoring."**

f. Skenario *Use Case* Mengelola Email Pengguna Perangkat

Tabel 3.8 *Use Case* Skenario Mengelola Email Pengguna Perangkat

Use Case Scenario 6 : "Mengelola Email Pengguna Perangkat"	
Nomor	UC-06

Nama	Mengelola Email Pengguna Perangkat
Deskripsi	Superadmin dan Admin menambahkan atau mengubah email pengguna perangkat yang akan digunakan sebagai tujuan pengiriman notifikasi maintenance.
Primary Actor	Superadmin, Admin
Preconditions	Pengguna telah login dan memilih perangkat yang akan diubah emailnya.
Postconditions	Email pengguna perangkat berhasil disimpan ke dalam database.
Main Success Scenario	1. Pengguna membuka halaman Detail Perangkat. 2. Pengguna memilih menu **Edit Email Pengguna.** 3. Pengguna memasukkan alamat email baru. 4. Pengguna menekan tombol **Simpan**. 5. Sistem melakukan validasi format email. 6. Sistem menyimpan email ke database. 7. Sistem menampilkan notifikasi bahwa email berhasil diperbarui.
Alternative Flow (Extension)	1. Format email tidak valid. 2. Sistem menampilkan pesan **"Format email tidak valid."** 3. Pengguna diminta memasukkan email yang benar.

g. Skenario *Use Case* Melihat Jadwal Maintenance

Tabel 3.9 *Use Case* Skenario Melihat Jadwal Maintenance

Use Case Scenario 7: "Melihat Jadwal Maintenance"	
Nomor	UC-07
Nama	Melihat Jadwal Maintenance
Deskripsi	Superadmin dan Admin melihat seluruh jadwal preventive maintenance perangkat yang telah dibuat oleh sistem.

Primary Actor	Superadmin, Admin
Preconditions	Pengguna telah berhasil login.
Postconditions	Daftar jadwal maintenance berhasil ditampilkan.
Main Success Scenario	1. Pengguna membuka menu **Maintenance Schedule**. 2. Sistem mengambil seluruh data jadwal maintenance dari database. 3. Sistem menampilkan hostname perangkat, risk level, tanggal rekomendasi maintenance, tanggal konfirmasi, serta status maintenance. 4. Pengguna dapat memilih salah satu jadwal untuk dilakukan konfirmasi atau perubahan status.
Alternative Flow (Extension)	1. Belum terdapat jadwal maintenance. 2. Sistem menampilkan pesan **"Belum ada jadwal maintenance."** 3.

h. Skenario *Use Case* Melihat Jadwal Maintenance Perangkat Staff

Tabel 3.10 *Use Case* Skenario Melihat Jadwal Maintenance Perangkat Staff

Use Case Scenario 8: "Melihat Jadwal Maintenance Perangkat Staff"	
Nomor	UC-08
Nama	Melihat Jadwal Maintenance Perangkat Staff.
Deskripsi	Staff IT melihat jadwal preventive maintenance yang dimiliki oleh perangkatnya sendiri.
Primary Actor	Staff IT
Preconditions	Staff IT telah berhasil login dan perangkat telah memiliki jadwal maintenance.
Postconditions	Jadwal maintenance perangkat berhasil ditampilkan.
Main Success Scenario	1. Staff IT membuka halaman **Perangkat Saya**. 2. Sistem mengambil jadwal maintenance perangkat berdasarkan akun Staff IT. 3. Sistem menampilkan tanggal rekomendasi maintenance,

	tanggal konfirmasi, risk level, dan status maintenance perangkat. 4. Staff IT dapat mengetahui jadwal maintenance perangkat yang akan dilakukan.
Alternative Flow (Extension)	1. Belum terdapat jadwal maintenance untuk perangkat tersebut. 2. Sistem menampilkan pesan **"Tidak ada jadwal maintenance."**

i. Skenario *Use Case* Konfirmasi Jadwal Maintenance

Tabel 3.11 *Use Case* Skenario Konfirmasi Jadwal Maintenance

Use Case Scenario 9: "Konfirmasi Jadwal Maintenance"	
Nomor	UC-09
Nama	Konfirmasi Jadwal Maintenance.
Deskripsi	Superadmin dan Admin melakukan konfirmasi terhadap jadwal preventive maintenance yang telah direkomendasikan oleh sistem.
Primary Actor	Superadmin, Admin
Preconditions	Pengguna telah login dan terdapat jadwal maintenance yang belum dikonfirmasi.
Postconditions	Jadwal maintenance berhasil dikonfirmasi dan tanggal konfirmasi tersimpan pada database.
Main Success Scenario	1. Pengguna membuka menu **Maintenance Schedule**. 2. Pengguna memilih salah satu jadwal maintenance. 3. Pengguna mengisi tanggal konfirmasi maintenance. 4. Pengguna menekan tombol **Simpan**. 5. Sistem menyimpan tanggal konfirmasi ke database. 6. Sistem menampilkan notifikasi bahwa jadwal maintenance berhasil dikonfirmasi.

Alternative Flow (Extension)	1. Pengguna tidak mengisi tanggal konfirmasi. 2. Sistem menampilkan pesan **"Tanggal konfirmasi wajib diisi."** 3. Data tidak disimpan sebelum tanggal konfirmasi diisi.

j. Skenario *Use Case* Mengubah Status Maintenance

Tabel 3.12 *Use Case* Skenario Mengubah Status Maintenance

Use Case Scenario 10: "Melakukan pencarian perangkat "	
Nomor	UC-10
Nama	Mengubah Status Maintenance
Deskripsi	Superadmin atau Admin mengubah status pelaksanaan maintenance menjadi **Pending**, **Completed**, atau **Missed**.
Primary Actor	Superadmin, Admin
Preconditions	Pengguna telah login dan jadwal maintenance telah dikonfirmasi.
Postconditions	Status maintenance berhasil diperbarui pada database.
Main Success Scenario	1. Pengguna membuka menu **Maintenance Schedule**. 2. Pengguna memilih salah satu jadwal maintenance. 3. Pengguna memilih status maintenance (Pending, Completed, atau Missed). 4. Pengguna menekan tombol **Simpan**. 5. Sistem memperbarui status maintenance pada database. 6. Sistem menampilkan notifikasi bahwa status maintenance berhasil diperbarui.
Alternative Flow (Extension)	Apabila pengguna belum melakukan konfirmasi jadwal maintenance, sistem tidak mengizinkan perubahan status dan menampilkan pesan **"Jadwal maintenance harus dikonfirmasi terlebih dahulu."**

k. Skenario *Use Case* Mengirim Email Notifikasi Maintenance

Tabel 3.13 *Use Case* Skenario Mengirim Email Notifikasi Maintenance

Use Case Scenario 11: "Mengirim Email Notifikasi Maintenance"	
Nomor	UC-11
Nama	Mengirim Email Notifikasi Maintenance
Deskripsi	Superadmin atau Admin mengirimkan email pemberitahuan jadwal maintenance kepada pengguna perangkat.
Primary Actor	Superadmin, Admin
Preconditions	Email pengguna telah tersimpan dan jadwal maintenance telah dikonfirmasi.
Postconditions	Email notifikasi berhasil dikirim kepada pengguna perangkat.
Main Succes Scenario	1. Pengguna membuka menu **Maintenance Schedule**. 2. Pengguna memilih jadwal maintenance yang telah dikonfirmasi. 3. Pengguna menekan tombol **Send Email**. 4. Sistem mengambil data perangkat, email pengguna, dan informasi jadwal maintenance. 5. Sistem mengirim email notifikasi menggunakan layanan email. 6. Sistem menampilkan notifikasi bahwa email berhasil dikirim.
Alternative Flow (Extension)	Apabila email pengguna belum tersedia atau jadwal maintenance belum dikonfirmasi, sistem membatalkan proses pengiriman email dan menampilkan pesan **"Email pengguna belum tersedia atau jadwal belum dikonfirmasi."**

l. Skenario *Use Case* Mengunduh Laporan Monitoring

Tabel 3.14 *Use Case* Skenario Mengunduh Laporan Monitoring

Use Case Scenario 12: "Mengunduh Laporan Monitoring"	
Nomor	UC-12
Nama	Mengunduh Laporan Monitoring
Deskripsi	Superadmin atau Admin menghasilkan laporan monitoring perangkat dalam format PDF berdasarkan filter yang dipilih
Primary Actor	Superadmin, Admin
Preconditions	Pengguna telah berhasil login dan data monitoring tersedia.
Postconditions	Laporan monitoring berhasil diunduh dalam format PDF.
Main Success Scenario	1. Pengguna membuka menu **Report**. 2. Pengguna memilih jenis laporan (Bulanan atau Tahunan). 3. Pengguna memilih bulan dan/atau tahun. 4. Sistem menampilkan preview laporan. 5. Pengguna menekan tombol **Download PDF**. 6. Sistem menghasilkan file PDF sesuai filter yang dipilih. 7. File PDF berhasil diunduh oleh pengguna.
Alternative Flow (Extension)	Apabila tidak terdapat data monitoring sesuai filter yang dipilih, sistem menampilkan pesan **"Tidak ada data untuk periode tersebut."** dan file PDF tidak dibuat.

m. Skenario Use Case Mengelola Profil

Use Case Scenario 13: "Mengelola Profil"	
Nomor	UC-13
Nama	Mengelola Profil
Deskripsi	Seluruh pengguna mengubah informasi profil pribadi seperti nama, foto profil, email, dan password.
Primary Actor	Superadmin, Admin, Staff IT

Preconditions	Pengguna telah login ke dalam sistem.
Postconditions	Data profil berhasil diperbarui.
Main Success Scenario	1. Pengguna membuka menu **Profile**. 2. Sistem menampilkan data profil pengguna. 3. Pengguna mengubah informasi yang diinginkan. 4. Pengguna menekan tombol **Simpan**. 5. Sistem memvalidasi data. 6. Sistem memperbarui data profil pada database. 7. Sistem menampilkan notifikasi bahwa profil berhasil diperbarui.
Alternative Flow (Extension)	Apabila terdapat data yang tidak valid, sistem menampilkan pesan kesalahan dan meminta pengguna memperbaiki data sebelum disimpan.

n. Skenario Use Case Mengelola User

Use Case Scenario 14: "Mengelola User"	
Nomor	UC-14
Nama	Mengelola User
Deskripsi	Superadmin menambahkan dan menghapus akun Admin maupun Staff IT melalui halaman Manajemen User.
Primary Actor	Superadmin
Preconditions	Superadmin telah berhasil login.
Postconditions	Data pengguna berhasil ditambahkan atau dihapus dari sistem.
Main Success Scenario	1. Superadmin membuka menu **Manajemen User**. 2. Sistem menampilkan daftar seluruh akun pengguna.

	3. Superadmin memilih **Tambah User** atau **Hapus User**. 4. Jika menambah user, Superadmin mengisi data pengguna kemudian menekan tombol **Simpan**. 5. Sistem melakukan validasi data. 6. Sistem menyimpan data pengguna baru ke database. 7. Jika menghapus user, sistem meminta konfirmasi sebelum menghapus data. 8. Sistem menampilkan notifikasi bahwa proses berhasil dilakukan.
Alternative Flow (Extension)	Apabila username atau email sudah digunakan, sistem menampilkan pesan **"Username atau Email sudah terdaftar."** Jika Superadmin membatalkan proses penghapusan, data pengguna tetap tersimpan pada sistem.

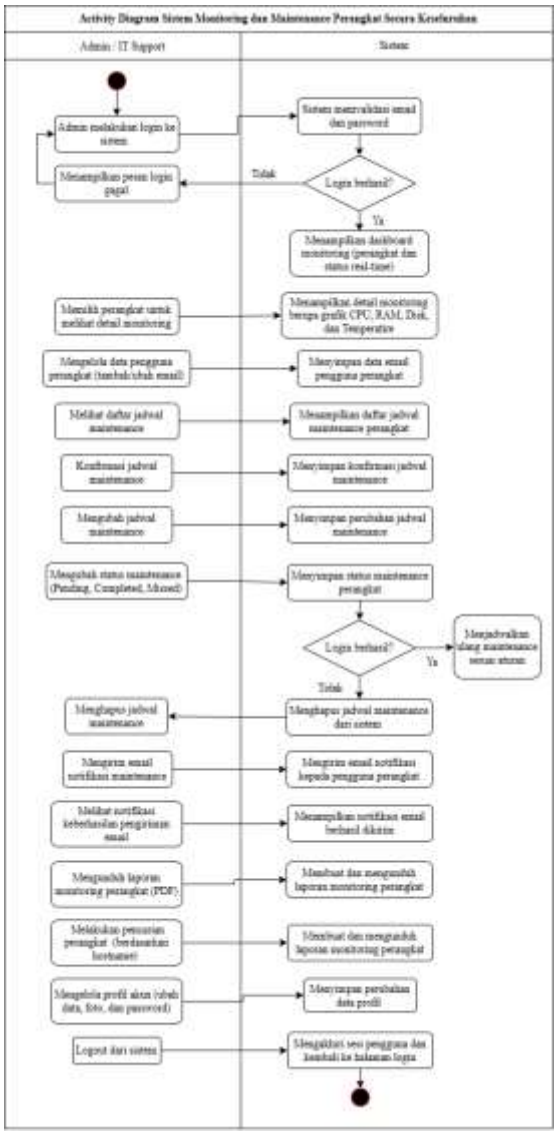
o. Skenario Use Case Logout

Use Case Scenario 15: "Logout"	
Nomor	UC-15
Nama	Logout
Deskripsi	Pengguna keluar dari sistem monitoring perangkat.
Primary Actor	Superadmin, Admin, Staff IT
Preconditions	Pengguna telah berhasil login ke sistem.
Postconditions	Session pengguna berakhir dan sistem kembali ke halaman login.
Main Success Scenario	1. Pengguna menekan menu **Logout**. 2. Sistem menghapus session pengguna. 3. Sistem mengarahkan pengguna ke halaman login.
Alternative Flow	1. Session pengguna telah berakhir. 2. Sistem langsung mengarahkan pengguna ke halaman

(Extension)	login.

3.2.5. Activity diagram

Bagian ini menjelaskan alur proses dalam sistem menggunakan Activity Diagram untuk menunjukkan aliran aktivitas atau pekerjaan dalam suatu proses. Diagram ini memvisualisasikan langkah-langkah kerja yang dilakukan oleh Aktor dan Sistem untuk menyelesaikan *Use Case* utama, mulai dari proses login hingga logout. Berikut adalah activity diagram untuk alur kerja utama sistem:



Gambar 3.4 Activity Diagram Sistem Monitoring dan Maintenance Perangkat Secara Keseluruhan

Gambar 3.4 menggambarkan keseluruhan alur aktivitas yang terjadi pada sistem monitoring dan predictive maintenance perangkat berbasis Artificial Intelligence. Proses dimulai ketika pengguna melakukan login ke dalam sistem menggunakan akun yang telah terdaftar. Sistem kemudian melakukan validasi terhadap data login dan menentukan hak akses pengguna berdasarkan role yang dimiliki, yaitu Superadmin, Admin, atau Staff IT.

Apabila pengguna berhasil melakukan login, sistem akan menampilkan halaman dashboard sesuai dengan hak akses masing-masing pengguna. Superadmin dan Admin dapat melihat dashboard monitoring seluruh perangkat yang terhubung ke dalam sistem, sedangkan Staff IT hanya dapat melihat dashboard perangkat miliknya sendiri beserta informasi monitoring yang berkaitan dengan perangkat tersebut.

Pada sisi sistem, Software Agent yang berjalan pada setiap perangkat akan secara otomatis mengumpulkan data penggunaan CPU, RAM, Disk, dan Temperatur secara real-time menggunakan library **psutil**. Data monitoring tersebut kemudian dikirimkan ke server melalui REST API untuk diproses oleh sistem.

Setelah data diterima, sistem melakukan prediksi kondisi perangkat menggunakan model **Random Forest Classifier** sehingga menghasilkan status perangkat berupa **Normal**, **Warning**, atau **Critical**. Berdasarkan hasil prediksi tersebut, sistem secara otomatis membuat jadwal preventive maintenance sesuai tingkat risiko perangkat. Perangkat dengan kondisi Critical memperoleh jadwal maintenance dalam waktu dua hari, kondisi Warning lima hari, sedangkan perangkat dengan kondisi Normal tetap memperoleh jadwal maintenance rutin setiap tiga puluh hari sebagai bentuk preventive maintenance.

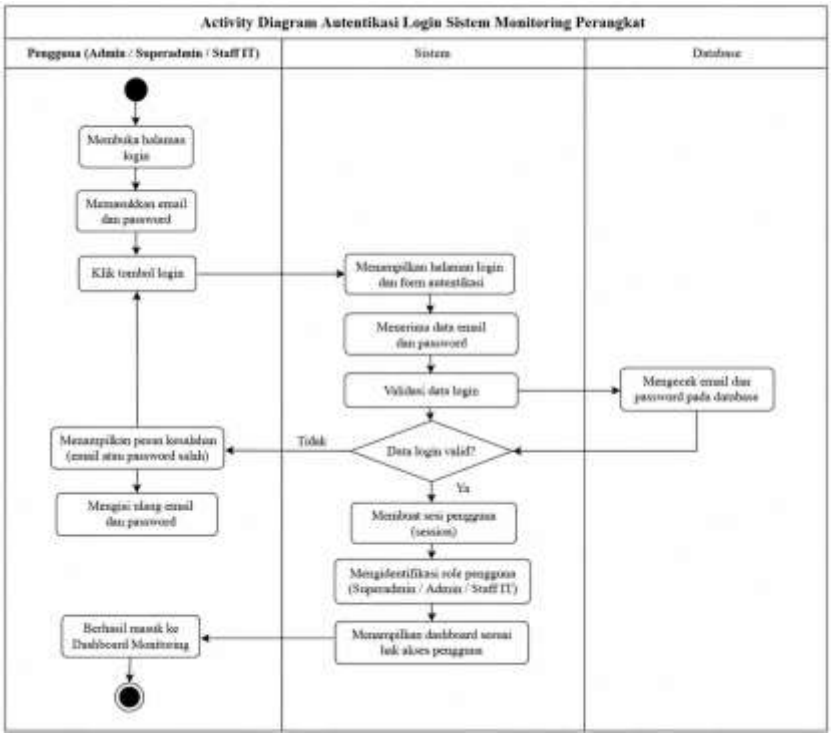
Superadmin maupun Admin selanjutnya dapat melihat detail monitoring perangkat, mengelola email pengguna perangkat, melihat daftar jadwal maintenance, melakukan konfirmasi jadwal, memperbarui status maintenance menjadi Pending, Completed, atau Missed, serta mengirimkan email notifikasi kepada pengguna perangkat setelah jadwal maintenance dikonfirmasi. Selain itu, pengguna juga dapat menghasilkan laporan monitoring perangkat dalam format

PDF berdasarkan periode tertentu serta mengelola informasi profil akun masing-masing.

Khusus Superadmin, sistem menyediakan fitur Manajemen User yang digunakan untuk menambahkan maupun menghapus akun Admin dan Staff IT sehingga pengelolaan pengguna dapat dilakukan langsung melalui aplikasi tanpa perlu mengubah source code.

Setelah seluruh aktivitas selesai dilakukan, pengguna dapat keluar dari sistem melalui proses logout sehingga sesi pengguna berakhir dengan aman. Activity Diagram ini menggambarkan hubungan antara aktivitas pengguna, proses monitoring otomatis oleh Software Agent, prediksi menggunakan Random Forest, penjadwalan maintenance otomatis, hingga proses pelaporan dalam satu alur sistem yang terintegrasi.

1. Activity Diagram Login



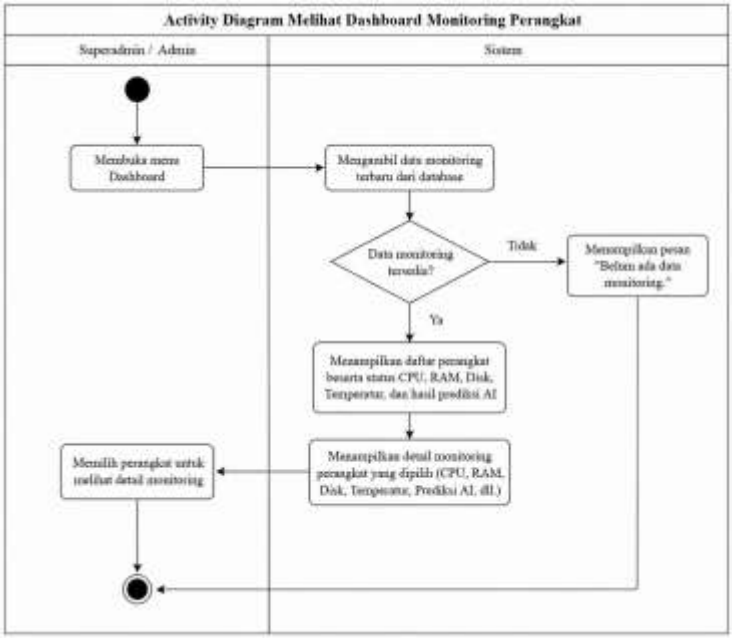
Gambar 3.5 Activity Diagram Login

Gambar 3.5 menggambarkan proses autentikasi pengguna ke dalam sistem monitoring perangkat. Proses dimulai ketika pengguna membuka halaman login kemudian memasukkan alamat email dan password yang telah terdaftar. Sistem

melakukan validasi terhadap data yang dimasukkan dengan membandingkan informasi akun pada database.

Apabila data login valid, sistem membuat sesi pengguna kemudian mengidentifikasi role pengguna, yaitu Superadmin, Admin, atau Staff IT. Selanjutnya sistem mengarahkan pengguna menuju dashboard sesuai hak akses masing-masing. Sebaliknya, apabila email atau password yang dimasukkan tidak sesuai, sistem akan menampilkan pesan kesalahan dan meminta pengguna untuk mengulangi proses login hingga data yang dimasukkan benar. Proses berakhir ketika pengguna berhasil masuk ke dalam sistem atau membatalkan proses login.

2. Activity Diagram Melihat *Dashboard* Monitoring Perangkat

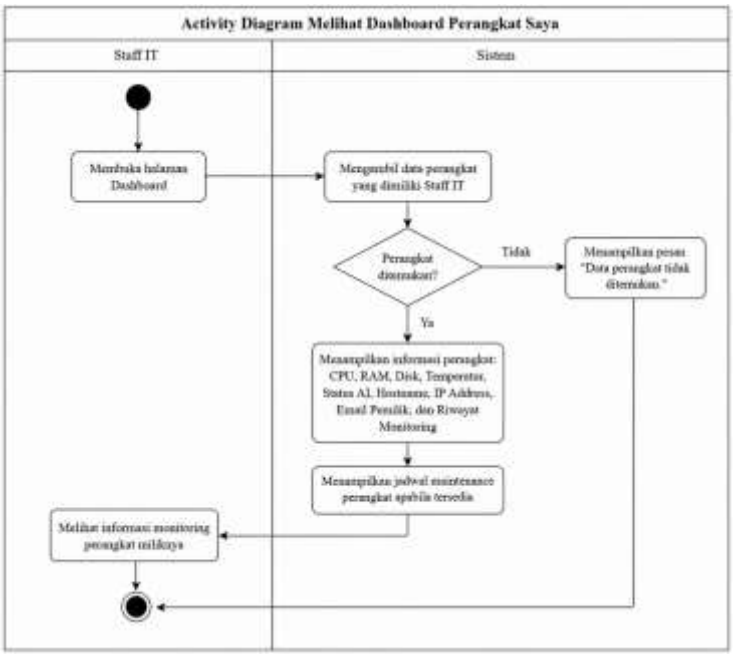


Gambar 3.6 Activity Diagram Melihat *Dashboard* Monitoring Perangkat

Gambar 3.6 menggambarkan proses autentikasi pengguna ke dalam sistem monitoring perangkat. Proses dimulai ketika pengguna membuka halaman login kemudian memasukkan alamat email dan password yang telah terdaftar. Sistem melakukan validasi terhadap data yang dimasukkan dengan membandingkan informasi akun pada database. Apabila data login valid, sistem membuat sesi pengguna kemudian mengidentifikasi role pengguna, yaitu Superadmin, Admin, atau Staff IT. Selanjutnya sistem mengarahkan pengguna menuju dashboard

sesuai hak akses masing-masing. Sebaliknya, apabila email atau password yang dimasukkan tidak sesuai, sistem akan menampilkan pesan kesalahan dan meminta pengguna untuk mengulangi proses login hingga data yang dimasukkan benar. Proses berakhir ketika pengguna berhasil masuk ke dalam sistem atau membatalkan proses login.

3. Activity Diagram Dashboard Perangkat Staff

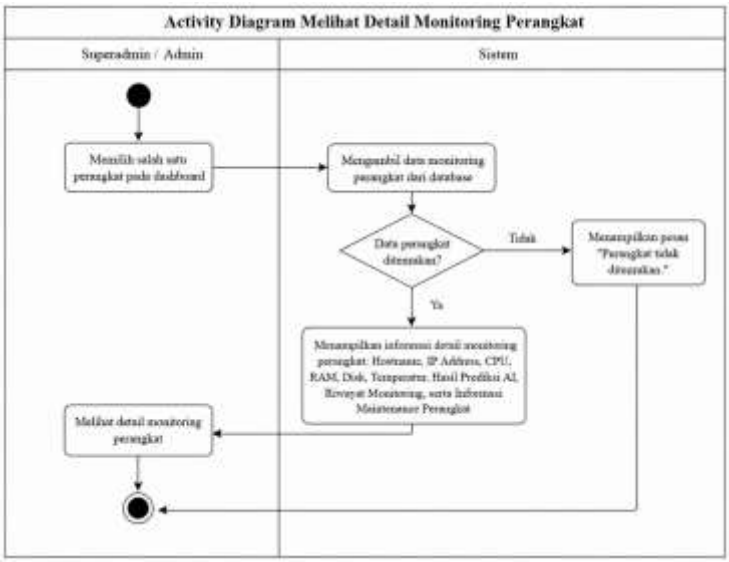


Gambar 3.7 Activity Diagram Dashboard Perangkat Staff

Gambar 3.7 menggambarkan proses Staff IT dalam melihat dashboard perangkat miliknya sendiri. Proses dimulai ketika Staff IT membuka halaman Dashboard. Sistem kemudian mengambil data perangkat berdasarkan akun pengguna yang sedang login. Setelah data berhasil diperoleh, sistem menampilkan informasi hostname, IP Address, CPU, RAM, Disk, Temperatur, status hasil prediksi AI, email pemilik perangkat, serta riwayat monitoring perangkat. Selain itu, apabila perangkat telah memiliki jadwal maintenance, sistem juga menampilkan informasi jadwal tersebut. Sebaliknya, apabila perangkat belum terdaftar pada sistem, sistem menampilkan pesan bahwa data perangkat tidak ditemukan

.

4. Activity Diagram Melihat Detail Monitoring Perangkat

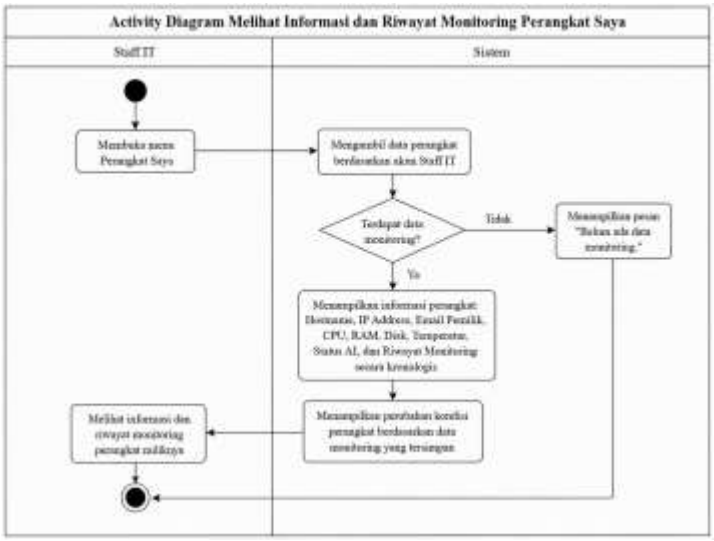


Gambar 3.8 Activity Diagram Melihat Detail Monitoring Perangkat

Gambar 3.8 menggambarkan proses Superadmin atau Admin dalam melihat detail monitoring suatu perangkat. Proses dimulai ketika pengguna memilih salah satu perangkat pada halaman dashboard monitoring. Sistem kemudian mengambil data monitoring perangkat dari database. Selanjutnya sistem menampilkan informasi detail perangkat yang meliputi hostname, IP Address, penggunaan CPU, RAM, Disk, Temperatur, hasil prediksi AI, riwayat monitoring, serta informasi maintenance perangkat. Apabila data perangkat tidak ditemukan, sistem menampilkan pesan bahwa perangkat tidak ditemukan sehingga proses tidak dapat dilanjutkan.

.

5. Activity Diagram Informasi dan Riwayat Monitoring Perangkat Staff

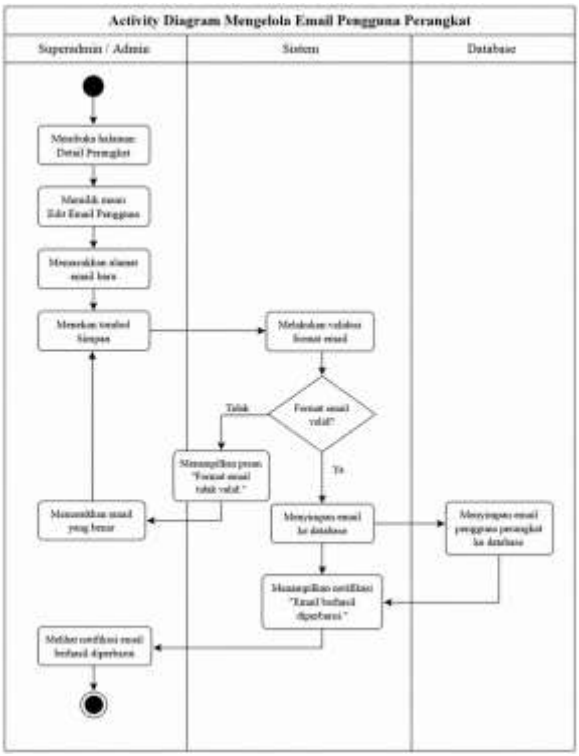


Gambar 3.9 Activity Diagram Informasi dan Riwayat Monitoring Perangkat Staff

Gambar 3.9 menggambarkan proses Staff IT dalam melihat informasi detail perangkat beserta riwayat monitoring perangkat yang dimilikinya. Proses dimulai ketika pengguna membuka menu Perangkat Saya. Selanjutnya sistem mengambil data perangkat berdasarkan akun Staff IT yang sedang login. Setelah data berhasil diperoleh, sistem menampilkan hostname, IP Address, email pemilik, CPU, RAM, Disk, Temperatur, status hasil prediksi AI, serta riwayat monitoring perangkat secara kronologis. Melalui informasi tersebut, Staff IT dapat mengetahui perubahan kondisi perangkat dari waktu ke waktu. Apabila belum terdapat data monitoring, sistem akan menampilkan pesan bahwa data monitoring belum tersedia.

.

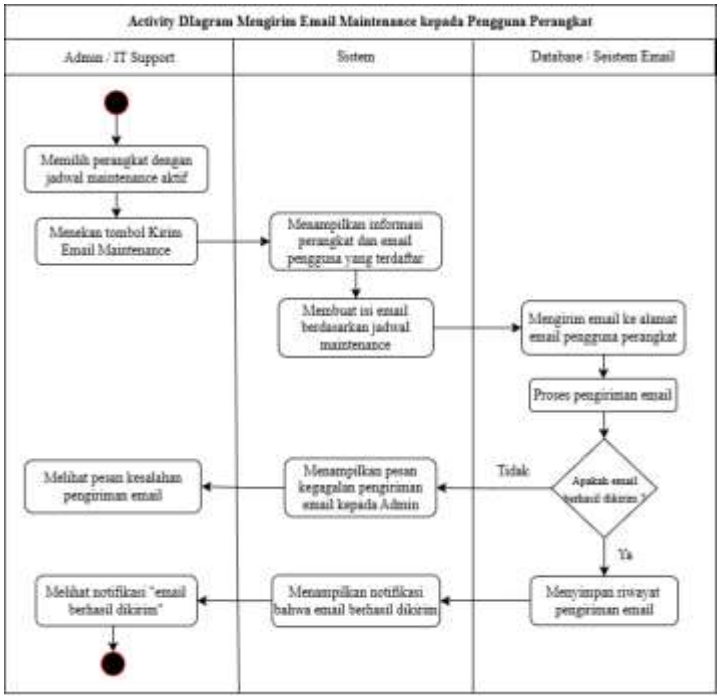
6. Activity Diagram Mengelola Email Pengguna Perangkat



Gambar 3.10 Activity Diagram Mengelola Email Pengguna Perangkat

Gambar 3.10 ini menggambarkan proses Superadmin atau Admin dalam mengelola email pengguna perangkat yang digunakan sebagai tujuan pengiriman notifikasi maintenance. Proses dimulai ketika pengguna membuka halaman Detail Perangkat kemudian memilih menu Edit Email Pengguna. Selanjutnya pengguna memasukkan alamat email baru dan menekan tombol Simpan. Sistem melakukan validasi terhadap format email yang dimasukkan. Apabila format email valid, sistem menyimpan perubahan email ke dalam database dan menampilkan notifikasi bahwa email berhasil diperbarui. Sebaliknya, apabila format email tidak valid, sistem menampilkan pesan kesalahan dan meminta pengguna memasukkan alamat email yang benar sebelum data dapat disimpan.

7. Activity Diagram Melihat Jadwal Maintenance

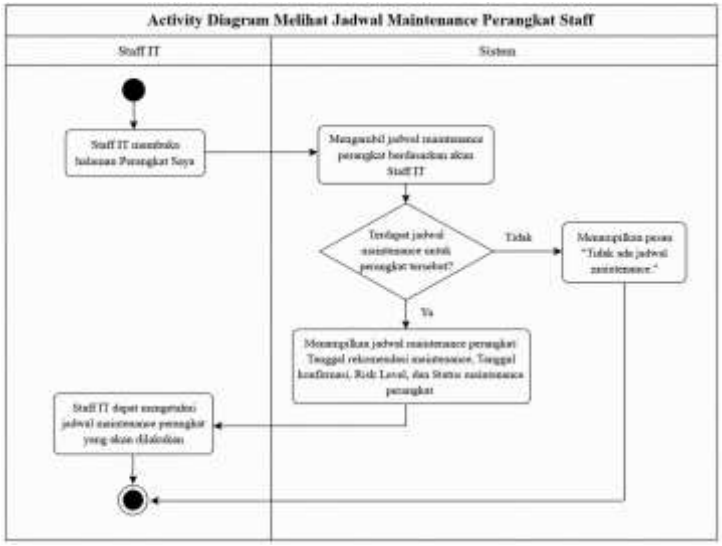


Gambar 3.11 Activity Diagram Melihat Jadwal Maintenance

Gambar 3.11 ini menggambarkan proses Superadmin atau Admin dalam melihat daftar preventive maintenance yang telah dibuat secara otomatis oleh sistem. Proses dimulai ketika pengguna membuka menu *Maintenance Schedule*. Sistem kemudian mengambil seluruh data jadwal maintenance dari database dan menampilkan informasi *hostname* perangkat, tingkat risiko (*risk level*), tanggal rekomendasi *maintenance*, tanggal konfirmasi, serta status maintenance. Pengguna dapat memilih salah satu jadwal untuk melakukan konfirmasi maupun perubahan status. Apabila belum terdapat jadwal maintenance, sistem menampilkan pesan bahwa belum ada jadwal maintenance yang tersedia

.

8. Activity Diagram Jadwal Maintenance Perangkat Staff

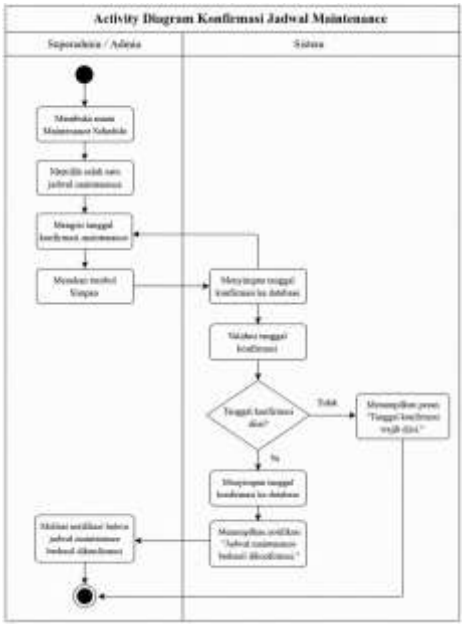


Gambar 3.12 Activity Diagram Jadwal Maintenance Perangkat Staff

Gambar 3.12 ini menggambarkan proses Staff IT dalam melihat jadwal preventive maintenance pada perangkat yang dimilikinya. Proses dimulai ketika pengguna membuka halaman Perangkat Saya. Sistem kemudian mengambil data jadwal maintenance berdasarkan akun pengguna yang sedang login. Selanjutnya sistem menampilkan tanggal rekomendasi maintenance, tanggal konfirmasi, tingkat risiko perangkat, serta status maintenance sehingga Staff IT dapat mengetahui jadwal maintenance yang akan dilakukan. Apabila perangkat belum memiliki jadwal maintenance, sistem menampilkan pesan bahwa tidak terdapat jadwal maintenance.

.

9. Activity Diagram Konfirmasi Jadwal Maintenance

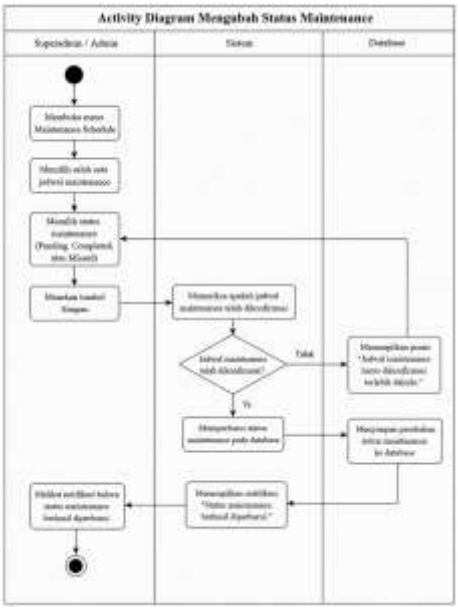


Gambar 3.13 Activity Diagram Konfirmasi Jadwal Maintenance

Gambar 3.13 ini menggambarkan proses Superadmin atau Admin dalam melakukan konfirmasi terhadap jadwal preventive maintenance yang telah direkomendasikan oleh sistem. Proses dimulai ketika pengguna membuka menu *Maintenance Schedule*, memilih salah satu jadwal, kemudian mengisi tanggal konfirmasi maintenance. Selanjutnya pengguna menekan tombol Simpan dan sistem menyimpan tanggal konfirmasi ke dalam database. Setelah proses berhasil, sistem menampilkan notifikasi bahwa jadwal maintenance berhasil dikonfirmasi. Sebaliknya, apabila tanggal konfirmasi belum diisi, sistem menampilkan pesan bahwa tanggal konfirmasi wajib diisi sehingga data tidak dapat disimpan sebelum pengguna melengkapi informasi tersebut

.

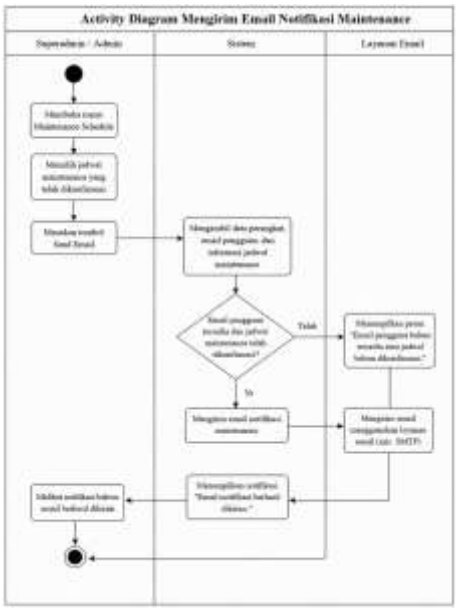
10. Activity Diagram Mengubah Status Maintenance



Gambar 3.14 Activity Diagram Mengubah Status Maintenance

Gambar 3.14 ini menggambarkan proses Superadmin atau Admin dalam mengubah status pelaksanaan maintenance perangkat. Proses dimulai ketika pengguna membuka menu *Maintenance Schedule*, memilih salah satu jadwal maintenance, kemudian memilih status maintenance berupa *Pending*, *Completed*, atau *Missed*. Selanjutnya pengguna menekan tombol Simpan dan sistem memeriksa apakah jadwal maintenance telah dikonfirmasi sebelumnya. Apabila jadwal telah dikonfirmasi, sistem memperbarui status maintenance pada database dan menampilkan notifikasi bahwa status maintenance berhasil diperbarui. Sebaliknya, apabila jadwal belum dikonfirmasi, sistem menampilkan pesan bahwa jadwal maintenance harus dikonfirmasi terlebih dahulu sehingga perubahan status tidak dapat dilakukan.

11. Activity Diagram Mengirim Email Notifikasi Maintenance

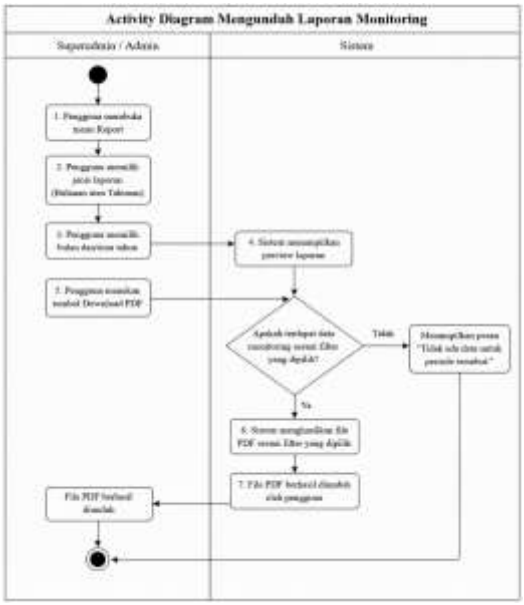


Gambar 3.15 Activity Diagram Mengirim Email Notifikasi Maintenance

Gambar 3.15 menggambarkan proses Superadmin atau Admin dalam mengirimkan email pemberitahuan jadwal maintenance kepada pengguna perangkat. Proses dimulai ketika pengguna membuka menu *Maintenance Schedule*, memilih jadwal maintenance yang telah dikonfirmasi, kemudian menekan tombol *Send Email*. Sistem mengambil data perangkat, alamat email pengguna, serta informasi jadwal maintenance sebelum mengirimkan email menggunakan layanan email. Setelah email berhasil dikirim, sistem menampilkan notifikasi bahwa email berhasil dikirim kepada pengguna. Apabila alamat email pengguna belum tersedia atau jadwal maintenance belum dikonfirmasi, sistem membatalkan proses pengiriman dan menampilkan pesan bahwa email pengguna belum tersedia atau jadwal belum dikonfirmasi.

.

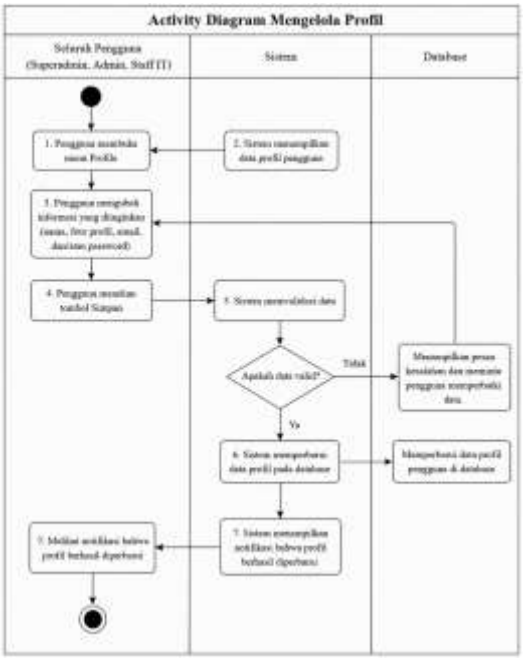
12. Activity Diagram Mengunduh Laporan Monitoring



Gambar 3.16 Activity Diagram Mengunduh Laporan Monitoring

Gambar 3.16 ini menggambarkan proses Superadmin atau Admin dalam menghasilkan laporan monitoring perangkat dalam format PDF. Proses dimulai ketika pengguna membuka menu *Report*, kemudian memilih jenis laporan serta menentukan bulan dan/atau tahun yang akan dijadikan filter. Sistem menampilkan preview laporan sesuai periode yang dipilih. Setelah pengguna menekan tombol *Download PDF*, sistem menghasilkan file laporan dalam format PDF berdasarkan filter yang telah ditentukan. Selanjutnya file PDF diunduh oleh pengguna. Apabila tidak terdapat data monitoring pada periode tersebut, sistem menampilkan pesan bahwa tidak ada data untuk periode yang dipilih sehingga file PDF tidak dibuat.

13. Activity Diagram Mengelola Profile

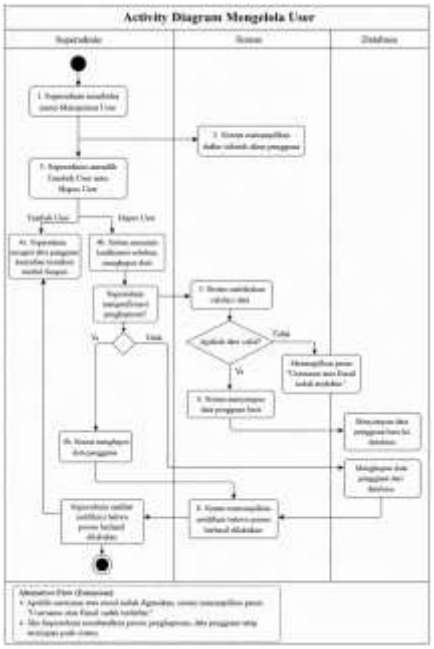


Gambar 3.17 Activity Diagram Mengelola Profile

Gambar 3.17 ini menggambarkan proses seluruh pengguna dalam memperbarui informasi profil akun. Proses dimulai ketika pengguna membuka menu *Profile*. Sistem kemudian menampilkan informasi profil yang tersimpan pada database. Pengguna dapat mengubah data seperti nama, foto profil, email, maupun password, kemudian menekan tombol Simpan. Sistem melakukan validasi terhadap data yang dimasukkan. Apabila data valid, sistem memperbarui informasi profil pada database dan menampilkan notifikasi bahwa profil berhasil diperbarui. Sebaliknya, apabila terdapat data yang tidak valid, sistem menampilkan pesan kesalahan dan meminta pengguna memperbaiki data sebelum proses penyimpanan dilakukan.

14. Activity Diagram Mengelola User

.

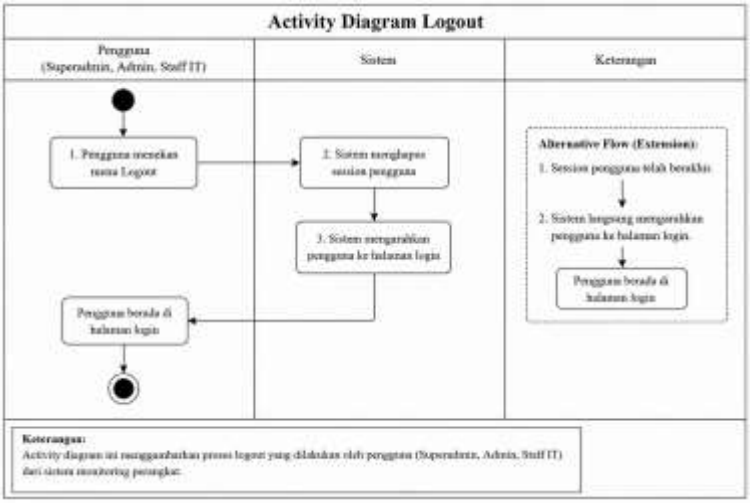


Gambar 3.18 Activity Diagram Mengelola User

Gambar 3.18 ini menggambarkan proses Superadmin dalam mengelola akun pengguna pada sistem monitoring perangkat. Proses dimulai ketika Superadmin membuka menu *Manajemen User* sehingga sistem menampilkan daftar seluruh akun pengguna yang telah terdaftar. Selanjutnya Superadmin dapat memilih untuk menambahkan pengguna baru atau menghapus akun pengguna. Pada proses penambahan pengguna, Superadmin mengisi data pengguna kemudian sistem melakukan validasi sebelum menyimpan data ke dalam database. Pada proses penghapusan, sistem meminta konfirmasi terlebih dahulu sebelum data dihapus. Setelah salah satu proses berhasil dilakukan, sistem menampilkan notifikasi bahwa pengelolaan pengguna berhasil dilakukan. Apabila username atau email telah digunakan, sistem menampilkan pesan bahwa data tersebut sudah terdaftar. Selain itu, apabila proses penghapusan dibatalkan oleh Superadmin, data pengguna tetap tersimpan pada system.

15. Activity Diagram Logout

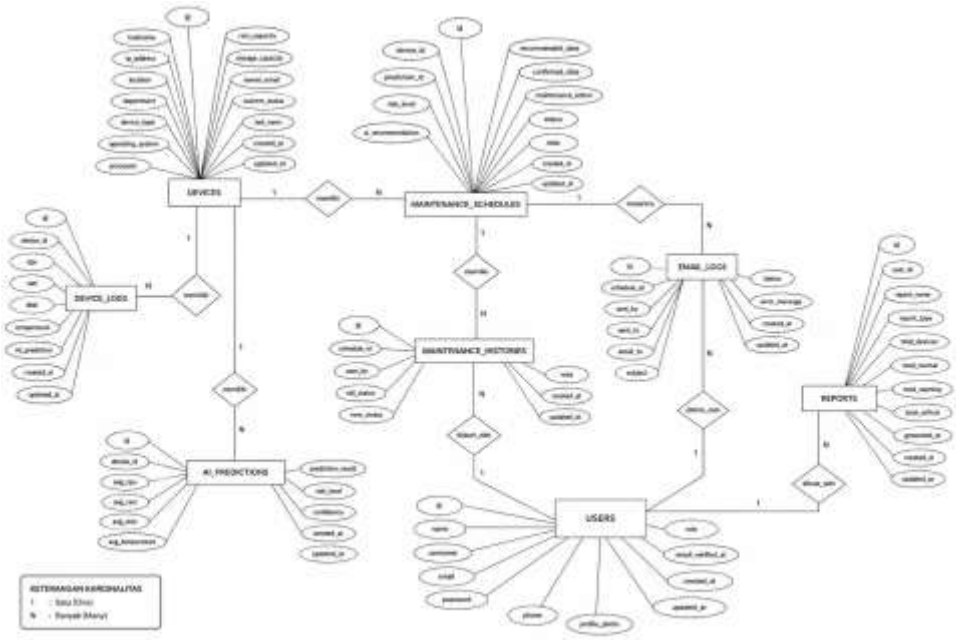
.



Gambar 3.19 Activity Diagram Logout

Gambar 3.19 ini menggambarkan proses pengguna keluar dari sistem monitoring perangkat. Proses dimulai ketika pengguna memilih menu *Logout* pada aplikasi. Selanjutnya sistem menghapus session pengguna sebagai bentuk penghentian hak akses terhadap sistem. Setelah session berhasil dihapus, sistem mengarahkan pengguna kembali ke halaman login sehingga proses logout selesai dilakukan dengan aman. Apabila session pengguna telah berakhir sebelumnya, sistem secara otomatis mengarahkan pengguna ke halaman login tanpa perlu melakukan proses logout kembali.

3.2.6. ER Diagram



Gambar 3.20 Entity Relationship Diagram

Entity Relationship Diagram (ERD**)** pada Sistem Prediksi dan Penjadwalan *Preventive Maintenance* Perangkat Komputer Berbasis *Artificial Intelligence* digunakan untuk menggambarkan struktur basis data yang mendukung proses monitoring perangkat komputer secara *real-time*, prediksi kondisi perangkat menggunakan algoritma *Random Forest*, penjadwalan *preventive maintenance* otomatis, pengelolaan pengguna berdasarkan hak akses (*role*), pengiriman notifikasi email, serta pembuatan laporan monitoring perangkat.

ERD ini menunjukkan hubungan antar entitas sehingga seluruh data dapat tersimpan secara terintegrasi dan mendukung seluruh proses bisnis sistem. Selain digunakan sebagai penyimpanan data monitoring, basis data juga berfungsi sebagai media pencatatan riwayat maintenance, histori prediksi AI, aktivitas pengguna, serta laporan monitoring yang dihasilkan oleh sistem. Dengan demikian, seluruh proses monitoring hingga pelaksanaan maintenance dapat ditelusuri kembali sebagai bahan evaluasi maupun pengambilan keputusan. Berikut merupakan penjelasan dari masing-masing entitas.

1. USERS

Entitas USERS digunakan untuk menyimpan data seluruh pengguna yang memiliki hak akses terhadap sistem. Informasi yang disimpan meliputi id, name, username, email, password, phone, photo, role, email_verified_at, created_at, dan updated_at. Atribut role digunakan untuk membedakan hak akses pengguna sebagai Super Admin, Admin, atau Staff IT. Berdasarkan relasi pada ERD, satu pengguna dapat melakukan banyak aktivitas, seperti mengubah status maintenance yang tercatat pada MAINTENANCE_HISTORIES, mengirim notifikasi email yang tercatat pada EMAIL_LOGS, serta menghasilkan banyak laporan pada tabel REPORTS. Dengan demikian, tabel ini menjadi pusat autentikasi dan pengelolaan hak akses system.

2. DEVICES

   Entitas DEVICES digunakan untuk menyimpan informasi perangkat komputer yang dimonitor oleh sistem. Data yang disimpan meliputi id, hostname, ip_address, location, department, operating_system, processor, ram_capacity, storage_capacity, owner_email, status, created_at, dan updated_at. Setiap perangkat menjadi objek utama dalam proses monitoring. Berdasarkan relasi pada ERD, satu perangkat dapat memiliki banyak data monitoring pada tabel DEVICE_LOGS, banyak hasil prediksi pada AI_PREDICTIONS, dan banyak jadwal maintenance pada MAINTENANCE_SCHEDULES.

3. DEVICE_LOGS

   Entitas DEVICE_LOGS digunakan untuk menyimpan data hasil monitoring perangkat yang dikirim secara otomatis oleh Software Agent berbasis Python. Informasi yang disimpan meliputi id, device_id, cpu, ram, disk, temperature, ml_prediction, created_at, dan updated_at. Setiap data monitoring mengacu pada satu perangkat pada tabel DEVICES, sedangkan satu perangkat dapat memiliki banyak data monitoring. Data pada tabel ini digunakan sebagai sumber utama untuk mengetahui kondisi perangkat secara real-time dan sebagai dasar perhitungan prediksi AI.

4. AI_PREDICTIONS

Entitas AI_PREDICTIONS digunakan untuk menyimpan hasil prediksi yang dihasilkan oleh model Artificial Intelligence menggunakan algoritma Random Forest. Data yang disimpan meliputi id, device_id, avg_cpu, avg_ram, avg_disk, avg_temperature, prediction_result, risk_level, confidence, created_at, dan updated_at. Setiap hasil prediksi mengacu pada satu perangkat pada tabel DEVICES, sedangkan satu perangkat dapat memiliki banyak hasil prediksi. Data ini digunakan sebagai dasar dalam menentukan tingkat risiko perangkat dan rekomendasi penjadwalan preventive maintenance.

5. MAINTENANCE_SCHEDULES

   Entitas MAINTENANCE_SCHEDULES digunakan untuk menyimpan data jadwal preventive maintenance yang dihasilkan secara otomatis maupun dikelola oleh administrator. Informasi yang disimpan meliputi id, device_id, prediction_id, risk_level, recommended_date, confirmed_date, maintenance_action, status, note, created_at, dan updated_at. Setiap jadwal maintenance mengacu pada satu perangkat serta satu hasil prediksi AI. Satu perangkat dapat memiliki banyak jadwal maintenance. Entitas ini menjadi penghubung antara hasil analisis AI dengan pelaksanaan maintenance.

6. MAINTENANCE_HISTORIES

   Entitas MAINTENANCE_HISTORIES digunakan untuk mencatat seluruh perubahan status maintenance sebagai riwayat aktivitas sistem. Data yang disimpan meliputi id, schedule_id, user_by, old_status, new_status, note, created_at, dan updated_at. Setiap histori mengacu pada satu jadwal maintenance dan satu pengguna. Satu jadwal maintenance dapat memiliki banyak riwayat perubahan sehingga seluruh aktivitas maintenance dapat ditelusuri kembali sebagai audit trail.

7. EMAIL_LOGS

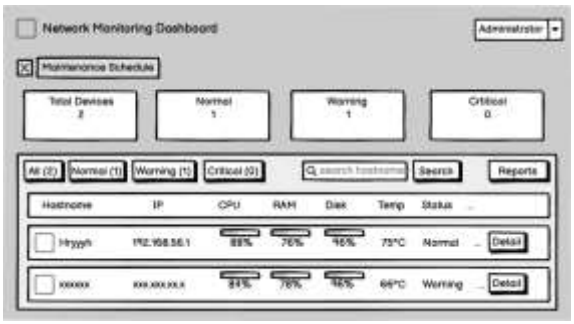
   Entitas EMAIL_LOGS digunakan untuk menyimpan riwayat pengiriman email notifikasi maintenance kepada pemilik perangkat. Informasi yang disimpan meliputi id, schedule_id, sent_by, sent_to, email_to, subject, status, error_message, created_at, dan updated_at. Setiap email mengacu

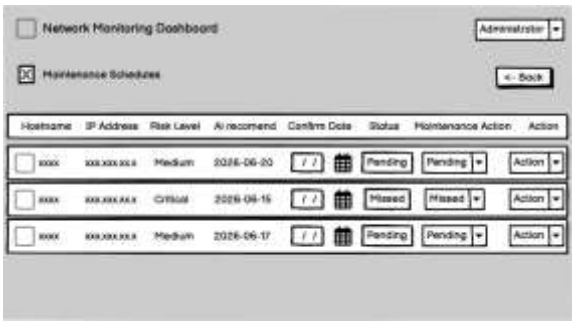
pada satu jadwal maintenance dan dikirim oleh satu pengguna pada tabel USERS. Tabel ini berfungsi untuk mendokumentasikan seluruh aktivitas pengiriman email sehingga administrator dapat mengetahui keberhasilan maupun kegagalan pengiriman notifikasi.

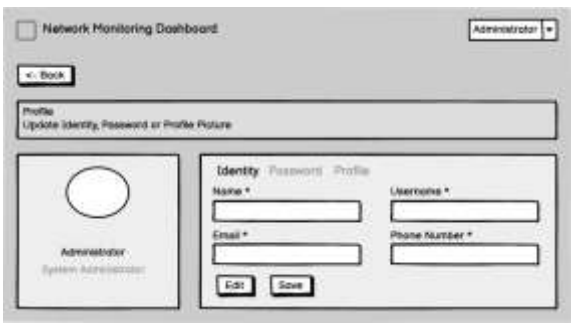
8. REPORTS

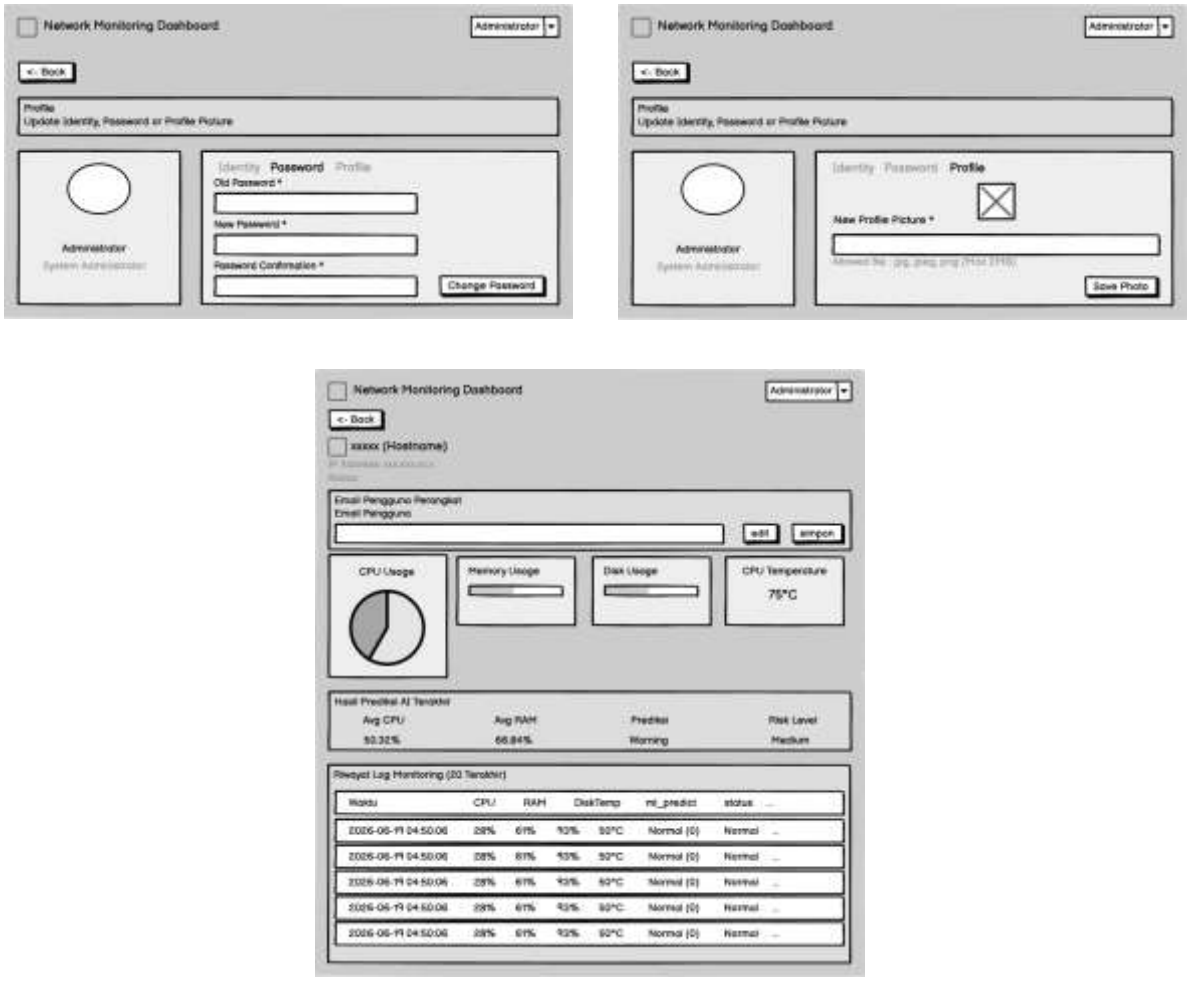
Entitas REPORTS digunakan untuk menyimpan data laporan hasil monitoring perangkat yang dihasilkan oleh sistem dalam format PDF. Informasi yang disimpan meliputi id, user_id, report_name, report_type, total_devices, total_normal, total_warning, total_critical, generated_at, created_at, dan updated_at. Setiap laporan dibuat oleh satu pengguna pada tabel USERS, sedangkan satu pengguna dapat menghasilkan banyak laporan. Entitas ini digunakan sebagai dokumentasi hasil monitoring dan evaluasi kondisi infrastruktur teknologi informasi.

3.2.7. Wireframe



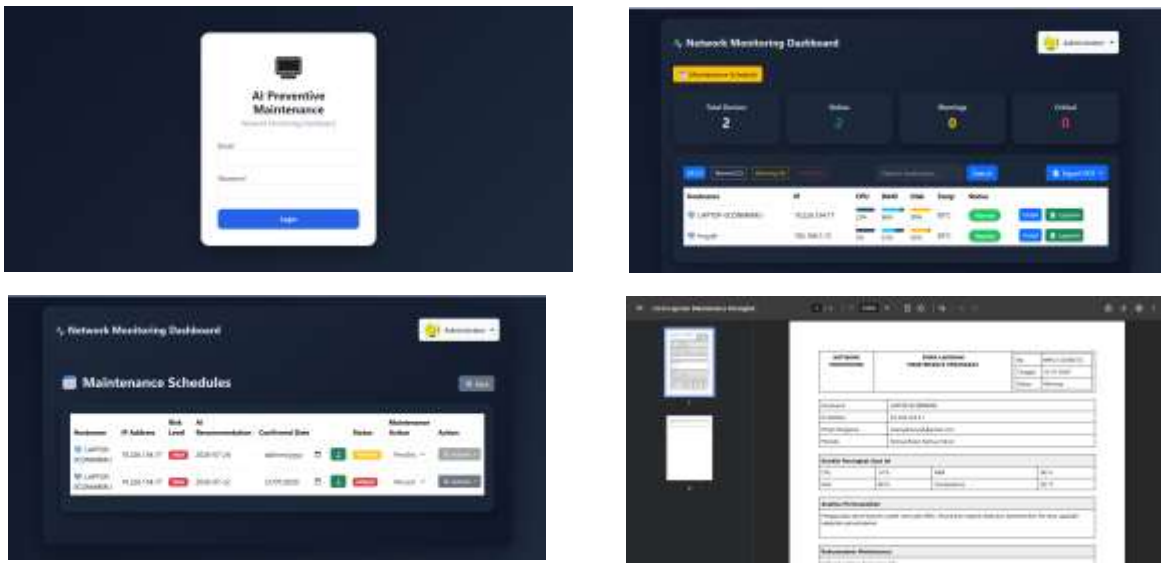


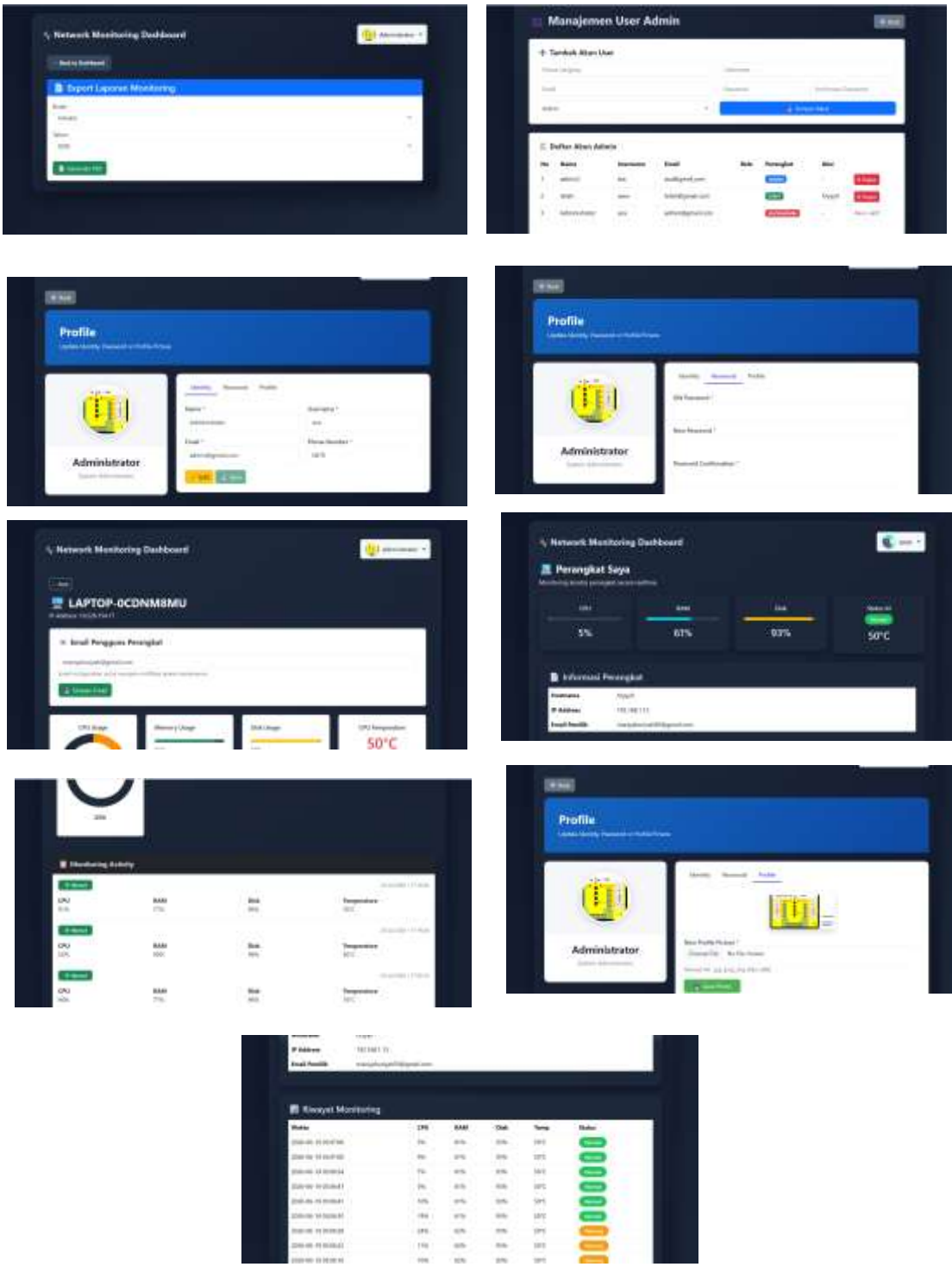




Gambar 3.21 Wireframe Sistem *Preventive maintenance*

3.2.8. Desain UI





Gambar 3.22 Desain Antarmuka Sistem *Preventive maintenance*

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Hasil Implementasi

Proses implementasi sistem monitoring dan penjadwalan preventive maintenance perangkat komputer dilakukan setelah tahap analisis kebutuhan, perancangan sistem, dan perancangan basis data selesai dilaksanakan. Implementasi ini bertujuan untuk membangun sebuah sistem monitoring perangkat komputer yang mampu melakukan pengumpulan data kondisi perangkat secara otomatis, memprediksi tingkat risiko perangkat menggunakan metode Artificial Intelligence, serta menghasilkan jadwal preventive maintenance secara otomatis berdasarkan hasil prediksi tersebut. Sistem dikembangkan menggunakan Framework Laravel sebagai backend dan website monitoring, bahasa Python sebagai Software Agent sekaligus modul Artificial Intelligence, serta MySQL sebagai basis data untuk menyimpan seluruh data pengguna, perangkat, data monitoring, jadwal maintenance, dan informasi sistem lainnya. Selain itu, sistem juga menerapkan mekanisme autentikasi berbasis role sehingga hak akses pengguna dibedakan menjadi Superadmin, Admin IT Support, dan Staff IT sesuai dengan kebutuhan operasional.

Struktur basis data dirancang menggunakan beberapa tabel utama yang saling terintegrasi. Tabel Users digunakan untuk menyimpan data pengguna sistem seperti nama, username, email, password, foto profil, serta role pengguna yang terdiri dari Superadmin, Admin IT Support, dan Staff IT. Tabel Devices digunakan untuk menyimpan informasi perangkat yang dimonitor seperti hostname, alamat IP, email pemilik perangkat, serta status perangkat. Tabel Device_Logs digunakan untuk menyimpan data monitoring yang dikirim secara berkala oleh Software Agent berupa penggunaan CPU, RAM, Disk, Temperature, waktu monitoring, dan hasil prediksi Artificial Intelligence. Selanjutnya, data jadwal preventive maintenance disimpan pada tabel Maintenance_Schedules yang berisi tanggal maintenance, status maintenance, tanggal konfirmasi, catatan maintenance, serta informasi perangkat yang akan dilakukan maintenance.

Seluruh relasi antar tabel dihubungkan menggunakan primary key dan foreign key untuk menjaga konsistensi serta integritas data selama sistem berjalan.

Implementasi sistem dimulai dari sisi perangkat pengguna (client). Pada setiap komputer atau laptop yang akan dimonitor dipasang Software Agent berbasis Python yang memanfaatkan library psutil untuk mengambil informasi penggunaan sumber daya perangkat secara real-time. Software Agent secara otomatis membaca penggunaan CPU, RAM, kapasitas penyimpanan (Disk), dan Temperature perangkat dalam interval waktu tertentu. Data monitoring yang diperoleh kemudian diproses menggunakan model Machine Learning berbasis Random Forest Classifier untuk menentukan kondisi perangkat menjadi Normal, Warning, atau Critical. Selanjutnya data monitoring beserta hasil prediksi dikonversi ke dalam format JSON dan dikirimkan ke server Laravel melalui REST API sehingga proses monitoring dapat berlangsung secara otomatis tanpa memerlukan input manual dari pengguna.

Setelah data monitoring diterima oleh server Laravel, sistem melakukan proses validasi kemudian menyimpan data tersebut ke dalam basis data pada tabel Device_Logs. Data yang tersimpan selanjutnya digunakan sebagai sumber informasi untuk menampilkan kondisi perangkat secara real-time pada dashboard monitoring. Dashboard menyediakan informasi jumlah perangkat yang sedang dimonitor, hostname perangkat, alamat IP, penggunaan CPU, RAM, Disk, Temperature, status Artificial Intelligence, serta waktu monitoring terakhir. Selain itu, sistem juga menyediakan halaman detail monitoring perangkat yang menampilkan riwayat kondisi perangkat sehingga Administrator dapat melakukan analisis terhadap performa perangkat berdasarkan data historis yang tersedia.

Selain menampilkan data monitoring, sistem mengimplementasikan algoritma Random Forest Classifier sebagai metode Artificial Intelligence untuk melakukan klasifikasi kondisi perangkat. Model Machine Learning menganalisis data penggunaan CPU, RAM, Disk, dan Temperature yang dikirimkan oleh Software Agent untuk menentukan tingkat risiko perangkat. Hasil analisis diklasifikasikan menjadi tiga kategori, yaitu Normal, Warning, dan Critical. Status hasil prediksi tersebut kemudian disimpan bersamaan dengan data monitoring

sehingga dapat digunakan sebagai dasar dalam pengambilan keputusan preventive maintenance.

Berdasarkan hasil prediksi Artificial Intelligence tersebut, sistem secara otomatis menghasilkan jadwal preventive maintenance untuk seluruh perangkat yang dimonitor. Perangkat dengan status Critical memperoleh jadwal maintenance dalam waktu dua hari, perangkat dengan status Warning memperoleh jadwal lima hari, sedangkan perangkat dengan status Normal tetap memperoleh jadwal maintenance rutin setiap tiga puluh hari sebagai bentuk preventive maintenance. Informasi jadwal tersebut disimpan pada tabel Maintenance_Schedules dan ditampilkan pada halaman jadwal maintenance yang dapat diakses oleh Superadmin maupun Admin IT Support. Pada halaman tersebut administrator dapat melihat daftar perangkat yang memerlukan maintenance, melakukan konfirmasi tanggal maintenance, mengubah status maintenance menjadi Pending, Completed, atau Missed, serta menambahkan catatan mengenai tindakan maintenance yang dilakukan terhadap perangkat.

Untuk memastikan pengguna perangkat mengetahui jadwal maintenance yang telah ditentukan, sistem menyediakan fitur pengiriman email notifikasi kepada pemilik perangkat. Setelah Admin IT Support mengisi tanggal konfirmasi maintenance, sistem akan mengaktifkan tombol pengiriman email sehingga administrator dapat mengirimkan pemberitahuan secara langsung kepada pengguna perangkat berdasarkan alamat email yang telah tersimpan pada data perangkat. Email tersebut berisi informasi perangkat, tanggal maintenance, status maintenance, serta catatan tindakan maintenance yang akan dilakukan sehingga pengguna dapat mempersiapkan perangkat sesuai jadwal yang telah ditentukan.

Ketika proses maintenance telah selesai dilakukan, Admin IT Support dapat memperbarui status maintenance melalui sistem menjadi Completed, Pending, atau Missed. Apabila status maintenance berubah menjadi Missed, sistem secara otomatis akan membuat jadwal maintenance baru sesuai aturan penjadwalan yang telah ditentukan sehingga proses preventive maintenance tetap dapat berjalan secara berkelanjutan. Selain itu, administrator juga dapat menambahkan catatan maintenance yang menjelaskan tindakan perawatan yang telah dilakukan, seperti pembersihan perangkat, penggantian komponen,

pemeriksaan perangkat keras, maupun tindakan teknis lainnya sebagai dokumentasi proses maintenance.

Sistem juga menyediakan fitur pembuatan laporan monitoring perangkat dalam format PDF. Administrator dapat melihat halaman preview laporan terlebih dahulu dengan memilih filter berdasarkan perangkat, bulan, dan tahun sebelum melakukan proses unduh laporan. Selain laporan monitoring keseluruhan, sistem juga menyediakan laporan khusus setiap perangkat yang dapat diakses langsung melalui halaman dashboard. Laporan tersebut memuat informasi identitas perangkat, riwayat monitoring, statistik kondisi perangkat, serta riwayat jadwal maintenance sehingga dapat digunakan sebagai dokumentasi dan bahan evaluasi kondisi perangkat

Sebagai contoh implementasi, misalkan sebuah komputer pada Departemen Engineering mengalami peningkatan penggunaan CPU hingga mencapai 95%, penggunaan RAM sebesar 88%, penggunaan Disk sebesar 96%, serta Temperature mencapai 79°C. Software Agent akan membaca kondisi tersebut kemudian mengirimkan data ke server Laravel melalui REST API. Selanjutnya model Random Forest Classifier menganalisis data tersebut dan menghasilkan prediksi Critical. Berdasarkan hasil prediksi tersebut, sistem secara otomatis membuat jadwal preventive maintenance dua hari ke depan dan menampilkannya pada halaman dashboard monitoring. Setelah tanggal maintenance dikonfirmasi, Admin IT Support mengirimkan email pemberitahuan kepada pengguna perangkat agar membawa perangkat sesuai jadwal yang telah ditentukan. Setelah proses maintenance selesai dilaksanakan, administrator mengubah status maintenance menjadi Completed serta menambahkan catatan mengenai tindakan maintenance yang telah dilakukan sehingga seluruh aktivitas terdokumentasi di dalam sistem.

Seluruh sistem diimplementasikan pada server lokal menggunakan web server yang mendukung Framework Laravel dan database MySQL. Sistem dapat diakses melalui browser oleh Superadmin, Admin IT Support, maupun Staff IT sesuai dengan hak akses masing-masing tanpa memerlukan instalasi tambahan pada sisi pengguna. Sementara itu, Software Agent berjalan secara otomatis pada perangkat pengguna sehingga proses pengumpulan data monitoring dapat

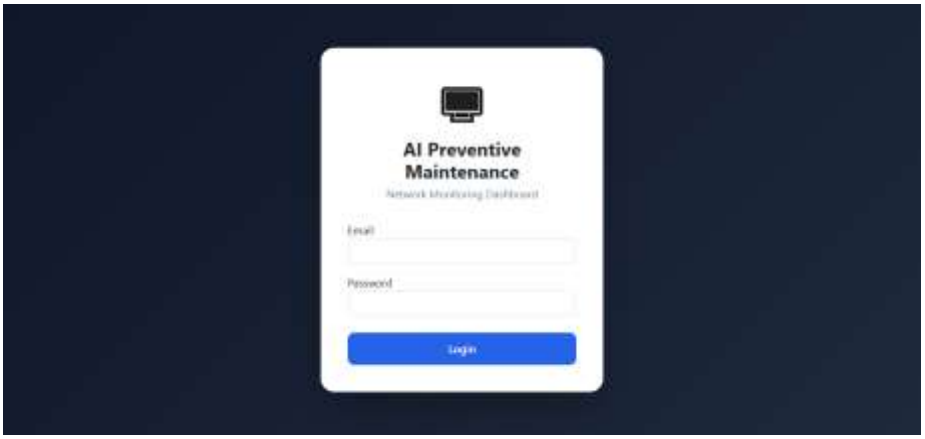
dilakukan secara kontinu. Dengan implementasi tersebut, sistem mampu melakukan monitoring kondisi perangkat secara real-time, melakukan prediksi kondisi menggunakan Artificial Intelligence berbasis Random Forest Classifier, menghasilkan jadwal preventive maintenance secara otomatis, mengirimkan email notifikasi kepada pengguna perangkat, menyediakan laporan monitoring dalam format PDF, serta mendukung proses pengambilan keputusan yang lebih cepat dan akurat dalam pengelolaan perangkat komputer perusahaan.

Selain itu, sistem menerapkan mekanisme autentikasi berbasis Role Based Access Control (RBAC) sehingga setiap pengguna hanya dapat mengakses fitur sesuai dengan hak aksesnya. Superadmin memiliki hak akses penuh termasuk mengelola akun pengguna, sedangkan Admin IT Support bertugas mengelola monitoring, maintenance, laporan, dan pengiriman email. Staff IT hanya dapat melihat kondisi perangkat miliknya, riwayat monitoring, jadwal maintenance, serta mengelola profil akun pribadi. Berdasarkan hasil implementasi dan pengujian yang dilakukan, sistem mampu menjalankan seluruh fungsi utama dengan baik mulai dari proses pengumpulan data monitoring oleh Software Agent, prediksi kondisi perangkat menggunakan Random Forest Classifier, penjadwalan preventive maintenance otomatis, pengiriman email notifikasi, manajemen pengguna berdasarkan role, hingga pembuatan laporan monitoring dalam format PDF. Dengan demikian, sistem ini mampu membantu perusahaan dalam melakukan monitoring dan pemeliharaan perangkat secara lebih proaktif, mengurangi risiko kerusakan mendadak, meminimalkan downtime perangkat, serta meningkatkan efektivitas pengelolaan infrastruktur teknologi informasi.

4.1.1. Implementasi *User Interface* (UI)

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

1. Login



Gambar 4.1 Implementasi Login

Implementasi antarmuka halaman login pada Sistem Monitoring dan Penjadwalan Preventive Maintenance Perangkat Komputer Berbasis Artificial Intelligence dirancang dengan tampilan sederhana, modern, dan mudah digunakan oleh pengguna. Halaman ini berfungsi sebagai gerbang autentikasi bagi Superadmin, Admin IT Support, dan Staff IT sebelum dapat mengakses fitur yang tersedia sesuai dengan hak akses masing-masing. Tampilan halaman menggunakan kombinasi warna biru sebagai identitas aplikasi dengan sebuah kartu (card) di bagian tengah layar yang berisi formulir login. Penempatan komponen pada bagian tengah bertujuan agar pengguna lebih mudah memfokuskan perhatian pada proses autentikasi.

Bagian identitas aplikasi menampilkan logo sistem, nama aplikasi AI Preventive Maintenance, serta deskripsi singkat Network Monitoring Dashboard sebagai informasi mengenai fungsi utama sistem. Formulir login terdiri atas kolom email dan password serta tombol Login untuk melakukan proses autentikasi.

Proses login dimulai ketika pengguna memasukkan alamat email dan password yang telah terdaftar pada sistem. Setelah tombol Login ditekan, sistem melakukan validasi terhadap data yang dimasukkan dengan membandingkannya dengan data pengguna pada basis data. Apabila data valid, sistem akan membuat sesi pengguna (session) dan mengarahkan pengguna menuju dashboard sesuai dengan role yang dimiliki. Sebaliknya, apabila data yang dimasukkan tidak sesuai,

sistem akan menampilkan pesan kesalahan dan meminta pengguna untuk melakukan login kembali menggunakan data yang benar.

2. *Dashboard Super admin dan Admin*

Gambar 4.2 Implementasi *Dashboard* Monitoring *Real-Time*

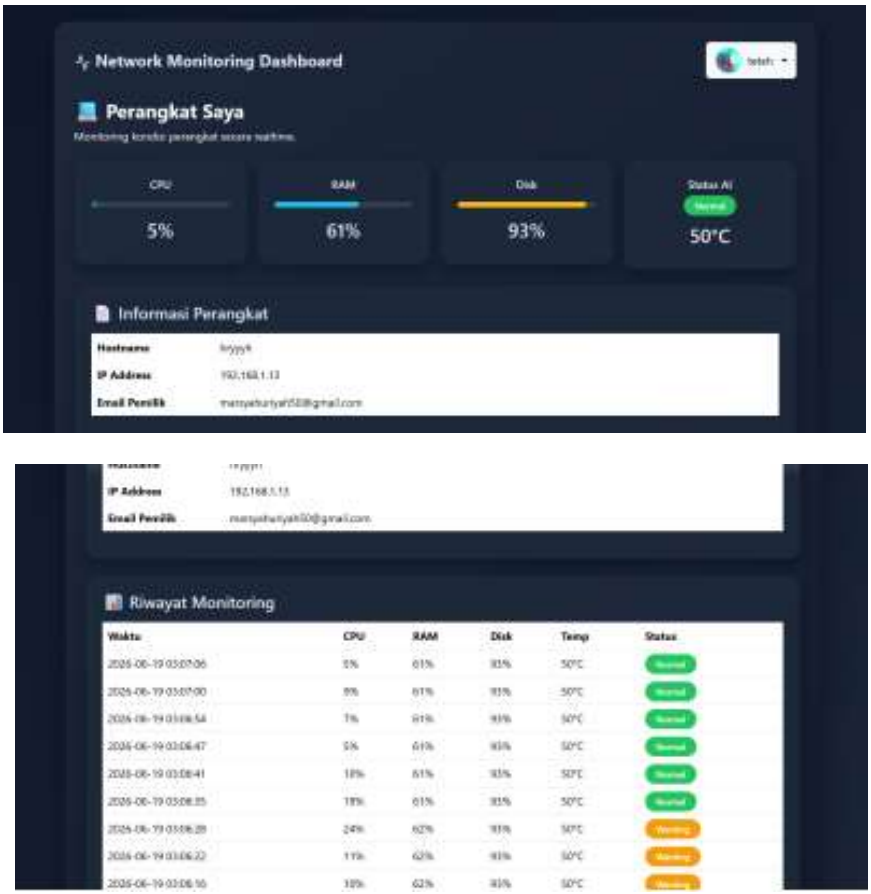
Implementasi halaman dashboard merupakan halaman utama yang ditampilkan setelah proses login berhasil dilakukan oleh Superadmin maupun Admin IT Support. Halaman ini menyajikan informasi monitoring perangkat secara real-time yang diperoleh dari Software Agent berbasis Python. Dashboard dirancang untuk memberikan gambaran menyeluruh mengenai kondisi seluruh perangkat yang sedang dimonitor sehingga administrator dapat mengambil keputusan dengan cepat apabila ditemukan perangkat yang mengalami potensi kerusakan.

Pada bagian atas halaman ditampilkan ringkasan jumlah perangkat yang dimonitor beserta jumlah perangkat dengan status Normal, Warning, dan Critical. Selanjutnya sistem menampilkan tabel daftar perangkat yang berisi informasi hostname, alamat IP, penggunaan CPU, RAM, Disk, Temperature, hasil prediksi Artificial Intelligence, status perangkat, serta waktu monitoring terakhir. Setiap perangkat juga dilengkapi dengan tombol Detail untuk melihat riwayat monitoring secara lebih rinci dan tombol Laporan untuk melihat laporan khusus perangkat tersebut.

Dashboard juga menyediakan fitur pencarian perangkat berdasarkan hostname sehingga administrator dapat menemukan perangkat tertentu dengan

lebih cepat. Selain itu tersedia tombol untuk mengakses halaman jadwal maintenance, halaman laporan monitoring, serta halaman profil pengguna. Khusus untuk Superadmin, dashboard juga menyediakan menu Manajemen User yang digunakan untuk mengelola akun Admin IT Support maupun Staff IT.

3. Dashboard Staff IT



Gambar 4.3 Implementasi dashboard staff IT

Implementasi halaman dashboard Staff IT dirancang lebih sederhana dibandingkan dashboard administrator karena hanya menampilkan informasi yang berkaitan dengan perangkat milik pengguna tersebut. Halaman ini bertujuan agar Staff IT dapat mengetahui kondisi perangkat yang digunakan tanpa memiliki akses terhadap data perangkat lain maupun fitur administrasi sistem.

Dashboard menampilkan informasi penggunaan CPU, RAM, Disk, Temperature, hasil prediksi Artificial Intelligence, hostname, alamat IP perangkat, serta alamat email pemilik perangkat. Selain itu sistem juga menampilkan riwayat

monitoring perangkat secara real-time dalam bentuk tabel sehingga pengguna dapat melihat perubahan kondisi perangkat berdasarkan waktu monitoring.

Pada bagian bawah halaman ditampilkan informasi jadwal preventive maintenance yang diberikan oleh Admin IT Support. Apabila belum terdapat jadwal maintenance, sistem akan menampilkan informasi bahwa perangkat belum memiliki jadwal maintenance. Dengan adanya dashboard ini, Staff IT dapat memantau kondisi perangkatnya sendiri secara mandiri tanpa mengganggu hak akses administrator.

4. Jadwal Maintenance

Gambar 4.4 Implementasi Jadwal Maintenance

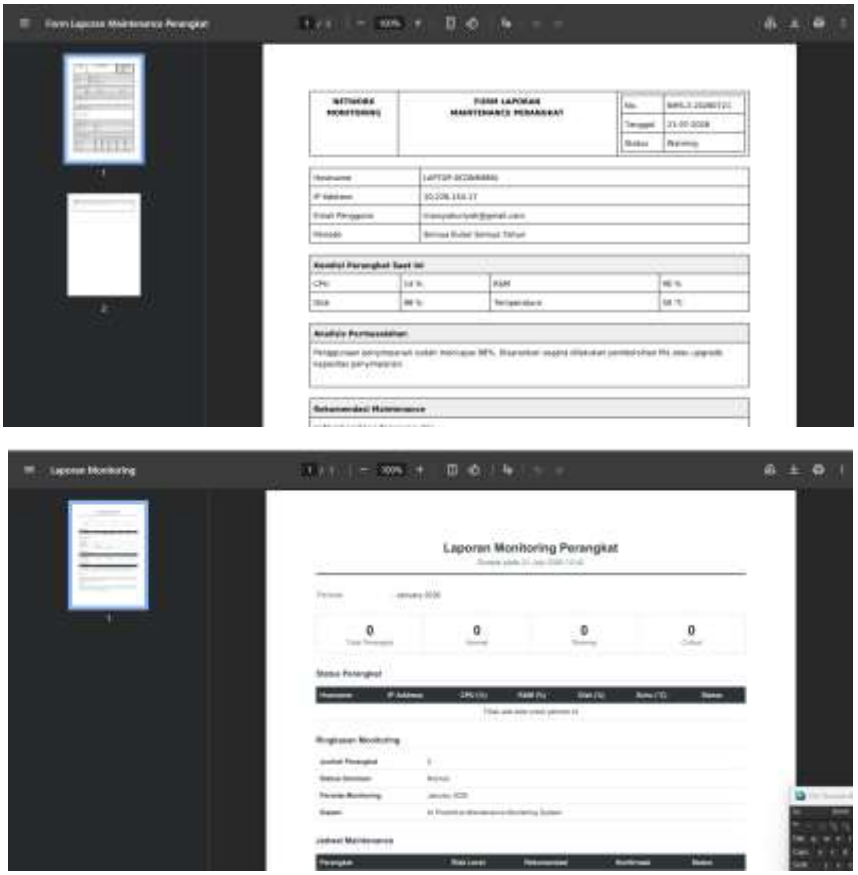
Implementasi halaman jadwal maintenance digunakan oleh Superadmin dan Admin IT Support untuk mengelola seluruh aktivitas preventive maintenance perangkat. Halaman ini menampilkan daftar perangkat yang memiliki jadwal maintenance beserta informasi hostname, alamat IP, tingkat risiko, tanggal maintenance, tanggal konfirmasi, status maintenance, serta catatan maintenance.

Administrator dapat melakukan konfirmasi jadwal maintenance, mengubah status maintenance menjadi Pending, Completed, atau Missed, serta menambahkan catatan mengenai tindakan maintenance yang dilakukan. Catatan maintenance digunakan untuk mendokumentasikan aktivitas perawatan seperti pembersihan perangkat, penggantian komponen, pembaruan sistem, maupun tindakan teknis lainnya.

Apabila status maintenance diubah menjadi Missed, sistem akan secara otomatis membuat jadwal maintenance baru sesuai dengan aturan penjadwalan

yang telah ditentukan sehingga proses preventive maintenance tetap dapat berjalan secara berkelanjutan. Setelah tanggal konfirmasi diisi, administrator juga dapat mengirimkan email notifikasi kepada pengguna perangkat secara langsung melalui halaman ini.

5. Laporan Monitoring



Gambar 4.5 Laporan Monitoring Perangkat

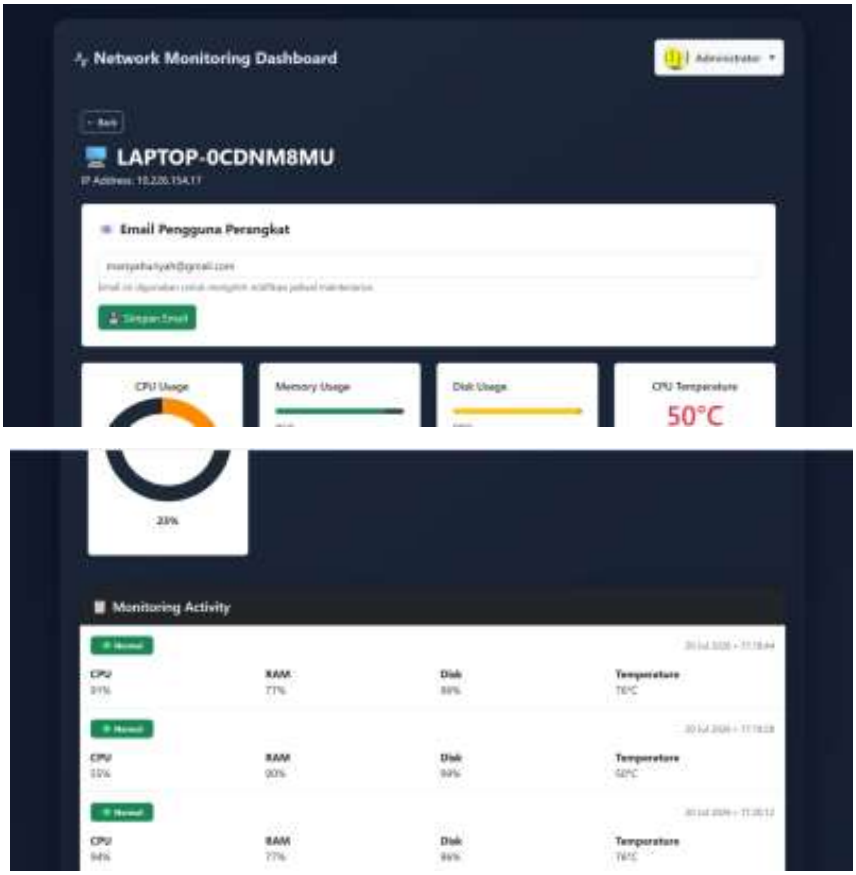
Implementasi halaman laporan monitoring digunakan oleh Superadmin dan Admin IT Support untuk melihat serta mengunduh laporan monitoring perangkat dalam format PDF. Sebelum laporan diunduh, sistem menampilkan halaman preview sehingga administrator dapat memastikan data yang akan dicetak telah sesuai dengan kebutuhan.

Halaman laporan menyediakan filter berdasarkan perangkat, bulan, dan tahun sehingga pengguna dapat menampilkan data monitoring sesuai periode tertentu. Setelah filter diterapkan, sistem akan menampilkan ringkasan jumlah

perangkat berdasarkan status Artificial Intelligence, tabel data monitoring, serta informasi jadwal maintenance yang berkaitan dengan perangkat tersebut.

Selain laporan monitoring keseluruhan, sistem juga menyediakan laporan khusus setiap perangkat yang dapat diakses langsung melalui tombol Laporan pada dashboard monitoring. Seluruh laporan dapat diunduh dalam format PDF sebagai dokumentasi kegiatan monitoring dan maintenance perangkat.

6. Halaman Detail Perangkat



Gambar 4.6 Detail Perangkat

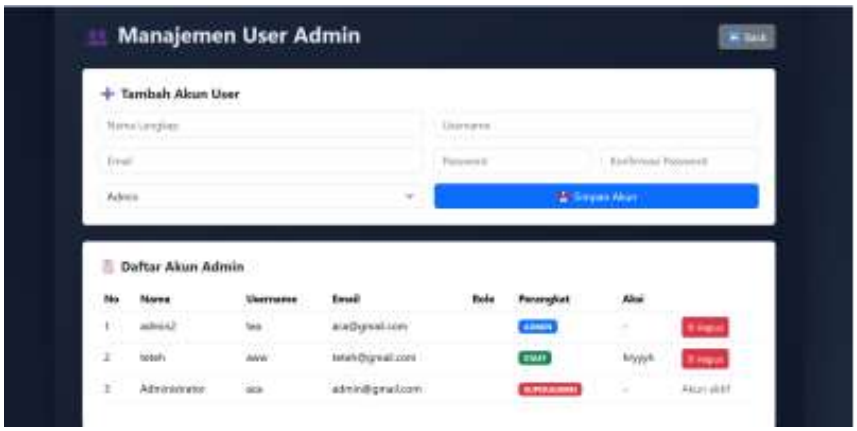
Implementasi halaman detail perangkat digunakan oleh Superadmin dan Admin IT Support untuk melihat kondisi suatu perangkat secara lebih rinci. Halaman ini menampilkan identitas perangkat berupa hostname, alamat IP, alamat email pengguna, serta status hasil prediksi Artificial Intelligence.

Selain informasi identitas perangkat, sistem juga menampilkan riwayat monitoring berupa penggunaan CPU, RAM, Disk, Temperature, dan status perangkat berdasarkan waktu monitoring. Data tersebut diperoleh secara otomatis

dari Software Agent berbasis Python yang mengirimkan informasi monitoring ke server melalui REST API.

Administrator juga dapat memperbarui alamat email pemilik perangkat sehingga email notifikasi maintenance dapat dikirimkan secara langsung kepada pengguna yang bersangkutan. Halaman detail perangkat menjadi salah satu komponen utama dalam proses analisis kondisi perangkat sebelum dilakukan tindakan maintenance.

7. Manajemen User



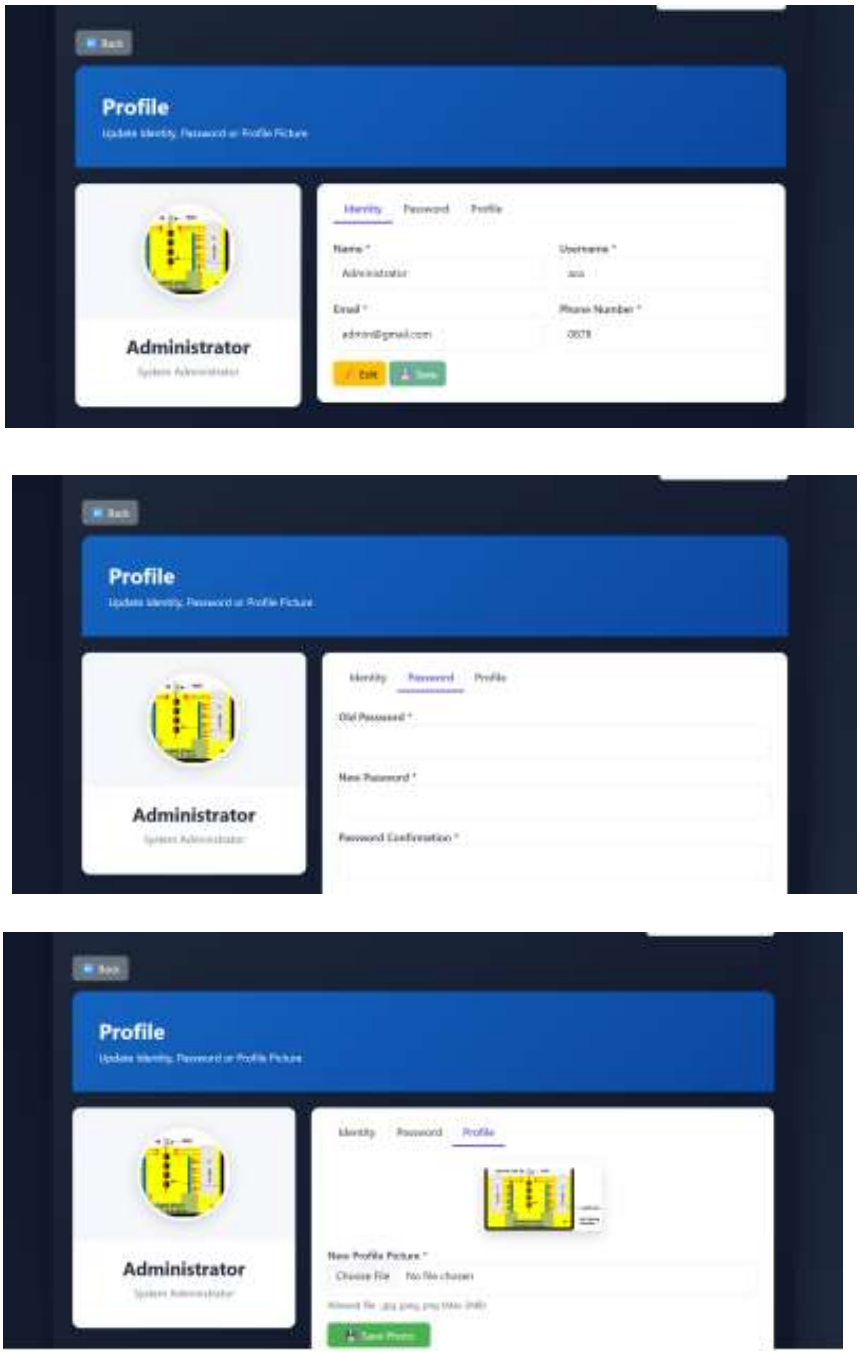
Gambar 4.7 Manajemen User

Implementasi halaman Manajemen User hanya dapat diakses oleh Superadmin sebagai pengguna dengan hak akses tertinggi pada sistem. Halaman ini digunakan untuk mengelola akun pengguna yang dapat mengakses sistem monitoring perangkat.

Pada halaman ini ditampilkan daftar seluruh akun pengguna beserta informasi nama, email, username, dan role pengguna. Superadmin dapat menambahkan akun baru dengan menentukan role sebagai Admin IT Support atau Staff IT, mengubah informasi akun apabila diperlukan, serta menghapus akun yang sudah tidak digunakan.

Fitur ini memungkinkan pengelolaan pengguna dilakukan langsung melalui antarmuka website tanpa perlu melakukan perubahan pada basis data maupun source code aplikasi sehingga proses administrasi sistem menjadi lebih mudah dan efisien.

8. Profil Pengguna



Gambar 4.8 Profil Pengguna

Implementasi halaman profil digunakan oleh seluruh pengguna sistem, yaitu Superadmin, Admin IT Support, maupun Staff IT untuk mengelola informasi akun masing-masing. Halaman ini terdiri atas tiga bagian utama, yaitu informasi profil, foto profil, dan keamanan akun.

Pada bagian Informasi Profil, pengguna dapat mengubah nama, username, alamat email, dan nomor telepon sesuai kebutuhan. Selanjutnya pada bagian Foto Profil, pengguna dapat mengunggah maupun mengganti foto profil sehingga identitas akun menjadi lebih mudah dikenali.

Bagian terakhir yaitu Keamanan Akun digunakan untuk mengubah password akun. Pengguna diwajibkan memasukkan password lama terlebih dahulu sebelum menentukan password baru sebagai bentuk pengamanan terhadap akses akun. Setelah seluruh perubahan disimpan, sistem akan memperbarui data pengguna pada basis data dan menampilkan notifikasi bahwa perubahan profil berhasil dilakukan.

4.2. Pengujian *Blackbox Testing*

4.2.1. Fungsional Testing

Pada tahap ini dilakukan pengujian terhadap fungsional sistem untuk memastikan seluruh fitur pada sistem Prediksi dan Penjadwalan *Preventive maintenance* Perangkat Komputer Berbasis Artificial Intelligence dapat berjalan sesuai kebutuhan pengguna dan tujuan penelitian. Pengujian dilakukan menggunakan metode *Black Box Testing*, yaitu metode pengujian yang berfokus pada kesesuaian keluaran sistem terhadap masukan yang diberikan tanpa memperhatikan struktur kode program yang digunakan. Pengujian dilakukan berdasarkan seluruh *Use Case* yang telah dirancang pada sistem. Hasil pengujian menunjukkan bahwa seluruh fungsi utama sistem dapat berjalan sesuai dengan kebutuhan pengguna. 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 Success Scenario	1	Admin membuka website monitoring perangkat.	Halaman *login* berhasil ditampilkan.	Halaman *login* berhasil ditampilkan sesuai desain sistem.	✓	
	2	Admin memasukkan email dan password.	Input *email* dan *password* dapat diisi tanpa error.	*Field email* dan *password* dapat diisi dengan baik.	✓	
	3	Admin menekan tombol "*Login*".	Sistem menerima *input* dan memproses *login*.	Sistem berhasil memproses data *login*.	✓	
	4	Sistem memvalidasi data yang dimasukkan admin.	Sistem melakukan validasi kecocokan *emal* dan *password* dengan *database*.	Sistem berhasil memvalidasi data sesuai dengan *database*.	✓	
	5	Jika data valid, sistem mengarahkan Admin ke *Dashboard Monitoring* Perangkat.	Admin diarahkan ke halaman *Dashboard Monitoring* Perangkat.	Admin berhasil diarahkan ke *Dashboard* Monitoring Perangkat.	✓	
Alternative Flow (Extension)	1	Admin memasukkan email atau password yang salah..	Sistem menampilkan pesan kesalahan "*Email* atau *Password* Salah" dan meminta admin mengisi ulang data yang benar.	Sistem menampilkan pesan error dan admin tetap berada pada halaman login.	✓	

2. *Black Box Testing* Scenario *Use Case* Melihat *Dashboard* Monitoring Perangkat.

   Black Box Testing ini berfokus pada pengujian fungsional melihat *dashboard* monitoring perangkat menggunakan *Black Box Testing*.

Tabel 4.2 *Black Box Testing* Scenario *Use Case* Melihat *Dashboard* Monitoring Perangkat

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Success Scenario	1	Admin membuka halaman *Dashboard* Monitoring.	Sistem menampilkan halaman *dashboard* monitoring perangkat.	Halaman *dashboard* berhasil ditampilkan.	✓	
	2	Sistem mengambil data perangkat dari database.	Data perangkat berhasil dimuat.	Data perangkat berhasil dimuat dan ditampilkan.	✓	
	3	Sistem menampilkan informasi perangkat.	*Hostname*, *IP Address*, CPU, RAM, Disk, dan Temperature tampil.	Informasi perangkat berhasil ditampilkan secara *Real-Time*.	✓	
	4	Sistem menampilkan hasil prediksi AI.	Status Normal, Warning, atau Critical tampil pada *dashboard*.	Status prediksi berhasil ditampilkan sesuai hasil AI.	✓	
	5	Admin melihat daftar perangkat yang terhubung.	Seluruh perangkat yang aktif ditampilkan.	Daftar perangkat berhasil ditampilkan.	✓	
Alternative Flow (Extension)	1	Tidak ada perangkat yang terhubung ke sistem.	Sistem menampilkan informasi bahwa perangkat belum tersedia.	Pesan perangkat belum tersedia berhasil ditampilkan.	✓	

3. *Black Box Testing* Scenario *Use Case* Melihat Detail Monitoring Perangkat.

   Black Box Testing ini berfokus pada pengujian fungsional Melihat detail monitoring perangkat menggunakan *Black Box Testing*.

Tabel 4.3 *Black Box Testing* Scenario *Use Case* Melihat *Dashboard* Monitoring Perangkat

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Success Scenario	1	Admin memilih salah satu perangkat pada *dashboard*.	Sistem membuka halaman detail perangkat.	Halaman detail perangkat berhasil ditampilkan..	✓	
	2	Sistem mengambil data monitoring perangkat.	Data monitoring berhasil dimuat.	Data monitoring berhasil dimuat dari database.	✓	
	3	Sistem menampilkan grafik monitoring perangkat.	Grafik CPU, RAM, Disk, dan Temperature tampil.	Grafik monitoring berhasil ditampilkan	✓	
	4	Sistem menampilkan hasil analisis AI perangkat.	Status Normal, Warning, atau Critical ditampilkan.	Status AI berhasil ditampilkan sesuai kondisi perangkat.	✓	
Alternative Flow (Extension)	1	Data monitoring perangkat tidak tersedia.	Sistem menampilkan pesan gagal memuat data monitoring.	Pesan gagal memuat data berhasil ditampilkan.	✓	

4. *Black Box Testing* Scenario *Use Case* Menambahkan Email Pengguna Perangkat.

   Black Box Testing ini berfokus pada Menambahkan email pengguna perangkat menggunakan *Black Box Testing*.

Tabel 4.4 *Black Box Testing* Scenario *Use Case* Menambahkan Email Pengguna Perangkat

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G

Main Success Scenario	1	Admin membuka detail perangkat.	Halaman detail perangkat ditampilkan.	Halaman detail perangkat berhasil ditampilkan.	✓	
	2	Admin mengisi email pengguna perangkat.	Field email dapat diisi tanpa error.	Email berhasil diinput.	✓	
	3	Admin menekan tombol Simpan.	Sistem menerima input email.	Sistem berhasil menerima data email.	✓	
	4	Sistem memvalidasi format email.	Format email diperiksa.	Format email berhasil divalidasi.	✓	
	5	Sistem menyimpan email ke database.	Email tersimpan pada data perangkat.	Email tersimpan pada data perangkat.	✓	
Alternative Flow (Extension)	1	Admin memasukkan email tidak valid.	Sistem menampilkan pesan "Email tidak valid".	Pesan error berhasil ditampilkan.	✓	

5. *Black Box Testing* Scenario *Use Case* Mengelola Jadwal Maintenance.

   Black Box Testing ini berfokus pada Mengelola Jadwal Maintenance menggunakan *Black Box Testing*.

Tabel 4.5 *Black Box Testing* Scenario *Use Case* Mengelola Jadwal Maintenance

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**

Main Success Scenario	1	Admin membuka menu Jadwal Maintenance.	Daftar jadwal maintenance ditampilkan.	Daftar jadwal berhasil ditampilkan.	✓	
	2	Admin memilih salah satu jadwal maintenance.	Detail jadwal ditampilkan.	Detail jadwal berhasil ditampilkan.	✓	
	3	Admin mengubah tanggal maintenance.	Data dapat diperbarui.	Data berhasil diperbarui.	✓	
	4	Admin menekan tombol Simpan.	Sistem memproses perubahan jadwal.	Sistem berhasil memproses perubahan.	✓	
	5	Sistem menyimpan perubahan ke database.	Jadwal maintenance diperbarui.	Jadwal berhasil diperbarui.	✓	
Alternative Flow (Extension)	1	Admin memasukkan tanggal tidak valid.	Sistem menampilkan pesan kesalahan.	Pesan error berhasil ditampilkan.	✓	

6. *Black Box Testing* Scenario *Use Case* Mengubah Status Maintenance.

   Black Box Testing ini berfokus pada Mengubah status Maintenance menggunakan *Black Box Testing*.

Tabel 4.6 *Black Box Testing* Scenario *Use Case* Mengubah Status Maintenance

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**

Main Success Scenario	1	Admin membuka daftar maintenance.	Daftar maintenance ditampilkan.	Daftar maintenance berhasil ditampilkan.	✓	
	2	Admin memilih status maintenance.	Status dapat dipilih.	Status berhasil dipilih.	✓	
	3	Admin memilih status Completed.	Admin memilih status Completed.	Status berhasil diperbarui.	✓	
	4	Sistem menyimpan perubahan status.	Status tersimpan pada database.	Status berhasil tersimpan.	✓	
Alternative Flow (Extension)	1	Admin memilih status Missed.	Sistem membuat jadwal maintenance baru otomatis.	Jadwal maintenance baru berhasil dibuat.	✓	

7. *Black Box Testing* Scenario *Use Case* Mengirim Email Maintenance.

   Black Box Testing ini berfokus pada Mengirim email Maintenance menggunakan *Black Box Testing*.

Tabel 4.7 *Black Box Testing* Scenario *Use Case* Mengirim Email Maintenance

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**
Main Success Scenario	1	Admin membuka daftar maintenance.	Daftar maintenance tampil.	Daftar maintenance berhasil ditampilkan.	✓	
	2	Admin memilih perangkat yang akan dikirim email.	Data perangkat ditampilkan.	Data perangkat berhasil ditampilkan.	✓	

	3	Admin menekan tombol Kirim Email.	Sistem membuat isi email maintenance.	Email berhasil dibuat.	✓	
	4	Sistem mengirim email ke pengguna perangkat.	Email terkirim.	Email berhasil dikirim.	✓	
	5	Sistem menampilkan notifikasi sukses.	Pesan sukses ditampilkan.	Notifikasi berhasil ditampilkan.	✓	
Alternative Flow (Extension)	1	Email pengguna belum tersedia.	Sistem menampilkan pesan email tidak ditemukan.	Pesan error berhasil ditampilkan.	✓	

8. *Black Box Testing* Scenario *Use Case* Menghapus Jadwal Maintenance.
   Black Box Testing ini berfokus pada Menghapus jadwal maintenance menggunakan *Black Box Testing*.

Tabel 4.8 *Black Box Testing* Scenario *Use Case* Menghapus Jadwal Maintenance

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**
Main Success Scenario	1	Admin memilih jadwal maintenance.	Data jadwal ditampilkan.	Jadwal berhasil ditampilkan..	✓	
	2	Admin menekan tombol Hapus.	Sistem menampilkan konfirmasi.	Konfirmasi berhasil ditampilkan.	✓	

	3	Admin menyetujui penghapusan.	Data dihapus dari sistem.	Data berhasil dihapus.	✓	
	4	Sistem memperbarui daftar jadwal.	Jadwal tidak lagi muncul.	Daftar berhasil diperbarui.	✓	
Alternative Flow (Extension)	1	Admin membatalkan penghapusan.	Data tetap tersimpan.	Data tetap tersedia.	✓	

9. *Black Box Testing* Scenario *Use Case* Mengunduh Laporan Monitoring.
   Black Box Testing ini berfokus pada Mengunduh laporan maintenance menggunakan *Black Box Testing*.

Tabel 4.9 *Black Box Testing* Scenario *Use Case* Mengunduh Laporan Monitoring.

Test Case	**NO**	**Skenario Pengujian**	**Luaran Yang Diharapkan**	**Luaran Yang Dihasilkan**	**Hasil**	
					B	**G**
Main Success Scenario	1	Admin membuka *dashboard* monitoring.	Data monitoring ditampilkan.	Data monitoring berhasil ditampilkan.	✓	
	2	Admin menekan tombol Download PDF.	Sistem memproses laporan.	Data berhasil diproses.	✓	
	3	Sistem menghasilkan file PDF.	Dokumen PDF dibuat.	PDF berhasil dibuat.	✓	
Alternative Flow (Extension)	4	Sistem mengunduh file PDF.	File berhasil diunduh.	PDF berhasil diunduh.	✓	

	1	Terjadi kesalahan saat generate PDF.	Sistem menampilkan pesan gagal mengunduh.	Pesan error berhasil ditampilkan.	✓	

10. *Black Box Testing* Scenario *Use Case* Melakukan Pencarian Perangkat.

    Black Box Testing ini berfokus pada Melakukan pencarian perangkat menggunakan *Black Box Testing*.

Tabel 4 .10 *Black Box Testing* Scenario *Use Case* Melakukan Pencarian Perangkat

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Success Scenario	1	Admin memasukkan *hostname* perangkat.	Data dapat dicari.	Data berhasil diinput.	✓	
	2	Admin menekan tombol Cari.	Sistem melakukan pencarian.	Pencarian berhasil diproses.	✓	
	3	Sistem menampilkan hasil pencarian.	Perangkat ditemukan.	Hasil pencarian berhasil ditampilkan.	✓	
Alternative Flow (Extension)	1	*Hostname* tidak ditemukan.	Sistem menampilkan pesan "Perangkat tidak ditemukan".	Pesan berhasil ditampilkan.	✓	

11. *Black Box Testing* Scenario *Use Case* Mengelola Profil Admin.

    Black Box Testing ini berfokus pada Mengelola profil admin menggunakan *Black Box Testing*.

Tabel 4.11 *Black Box Testing* Scenario *Use Case* Mengelola Profil Admin

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Success Scenario	1	Admin membuka menu Profil.	Data profil ditampilkan.	Data profil berhasil ditampilkan.	✓	
	2	Admin mengubah data profil..	Data dapat diperbarui.	Data berhasil diperbarui.	✓	
	3	Admin mengunggah foto profil.	Foto tersimpan.	Foto berhasil tersimpan.	✓	
	4	Admin mengubah password akun.	Password diperbarui.	Password berhasil diperbarui.	✓	
	5	Sistem menyimpan perubahan.	Data tersimpan ke database.	Data berhasil tersimpan.	✓	
Alternative Flow (Extension)	1	Ukuran foto melebihi 2 MB.	Sistem menampilkan pesan kesalahan.	Pesan error berhasil ditampilkan.	✓	

12. *Black Box Testing* Scenario *Use Case* Logout.

Black Box Testing ini berfokus pada pengujian fungsional logout menggunakan *Black Box Testing*.

Tabel 4.12 *Black Box Testing* Scenario *Use Case* Logout

Test Case	NO	Skenario Pengujian	Luaran Yang Diharapkan	Luaran Yang Dihasilkan	Hasil	
					B	G
Main Success Scenario	1	Admin menekan menu Logout.	Sistem menerima aksi klik tombol "*Log Out*".	Sistem berhasil menerima aksi *logout.*	✓	
	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.	*Admin* berhasil diarahkan ke halaman *login*.	✓	
	4	Sistem menampilkan halaman *login*	Halaman *login* ditampilkan dengan *form email* 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 4.13 Performance Testing

Komponen	Hasil Uji
Test Case ID	TC01-NF01
Type	*Performance Testing*
Tujuan	Memastikan sistem dapat beroperasi dengan baik di bawah berbagai beban kerja tanpa penurunan performa yang signifikan.

Lingkungan	Localhost (http://192.168.1.13:8000)	
Alat	Locust	
Metode	*Load Testing*	
Ekspetasi	Sistem dapat merespons permintaan dengan waktu respons yang konsisten, dengan tingkat keberhasilan yang tinggi serta tidak mengalami crash saat mencapai beban puncak.	
Hasil Uji		
ID	LOT01	
Nama Pengujian	Load Test - *Endpoint* GET /login	
Deskripsi Pengujian	Menguji performa *endpoint* halaman login (GET /login) yang menampilkan form autentikasi sistem kepada pengguna.	
Method	GET	
Skenario Uji	Total users	50
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	855
	Average (ms)	12.046,65
	Max (ms)	21.827
	95%aile (ms)	22.000
Request	Total Request	100
	Success Request	100
	Failed Request	0
Error Rate (%)	0%	

Chart Pengujian		
Hasil Uji		
ID	LOT02	
Nama Pengujian	Load Test - *Endpoint Dashboard* (GET /)	
Deskripsi Pengujian	Menguji performa sistem saat pengguna mengakses halaman *dashboard* utama yang memuat data seluruh perangkat, status monitoring *Real-Time*, dan hasil prediksi AI secara bersamaan.	
Method	GET	
Skenario Uji	Total users	50
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	23.265
	Average (ms)	27.806,63
	Max (ms)	35.247
	95%aile (ms)	33.000
Request	Total Request	120
	Success Request	120
	Failed Request	0
Error Rate (%)	0%	

Chart Pengujian		
Hasil Uji		
ID	LOT03	
Nama Pengujian	Load Test - *Endpoint* API Devices (GET /api/devices)	
Deskripsi Pengujian	Menguji performa *endpoint* /api/devices yang bertugas mengambil seluruh data perangkat beserta log monitoring terbaru secara *Real-Time* dalam format *JSON*.	
Method	GET	
Skenario Uji	Total users	50
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	21.698
	Average (ms)	27.429,96
	Max (ms)	33.801
	95%aile (ms)	32.000
Request	Total Request	92
	Success Request	92
	Failed Request	0
Error Rate (%)	0%	

Chart Pengujian		
Hasil Uji		
ID	LOT04	
Nama Pengujian	Load Test - *Endpoint* Schedules (GET /schedules)	
Deskripsi Pengujian	Menguji performa *endpoint* /schedules yang menampilkan seluruh jadwal *preventive maintenance* beserta informasi perangkat terkait. Pengujian dilakukan dengan sesi login aktif melalui mekanisme autentikasi Locust.	
Method	GET	
Skenario Uji	Total users	50
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	22.721
	Average (ms)	27.446,32
	Max (ms)	34.967
	95%aile (ms)	33.000
Request	Total Request	42
	Success Request	42
	Failed Request	0
Error Rate (%)	0%	

Chart Pengujian		
Hasil Uji		
ID	LOT05	
Nama Pengujian	Load Test - *Endpoint* POST Login (Autentikasi)	
Deskripsi Pengujian	Menguji performa proses autentikasi pengguna melalui *endpoint* POST /login. Pengujian mencakup proses validasi kredensial, pengecekan CSRF token, dan pembuatan sesi pengguna oleh server Laravel.	
Method	POST	
Skenario Uji	Total users	50
	Ramp Up	5
	Time (durasi)	3 Menit
Response Time (ms)	Min (ms)	23.723
	Average (ms)	53.865,77
	Max (ms)	67.067
	95%aile (ms)	67.000
Request	Total Request	50
	Success Request	50
	Failed Request	0
Error Rate (%)	0%	

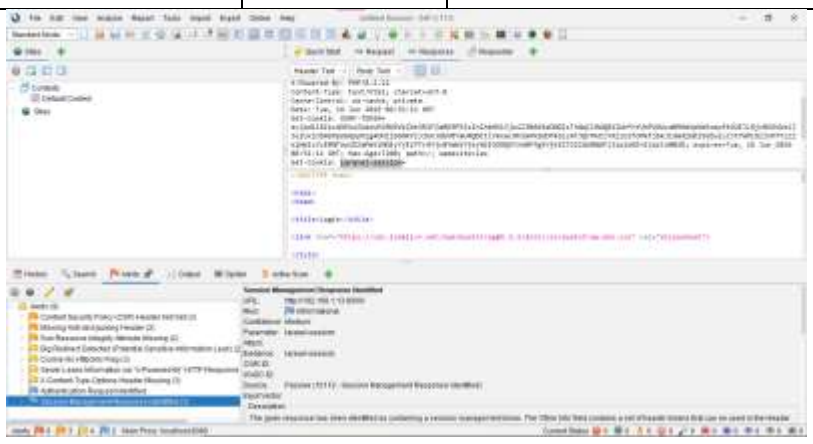
Chart Pengujian	

2. Security Testing

Tabel 4.14 Security Testing

Komponen	Hasil Uji
Test Case ID	TC01-NF02
Type	*Security Testing*
Tujuan	Memastikan sistem terlindungi dari ancaman keamanan seperti injeksi SQL, cross-site scripting, dan akses tidak sah
Lingkungan	Localhost (http://192.168.1.13:8000)
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	9

NO	Temuan	Tingkat Risiko	Penjelasan
1	Content Security Policy (CSP) Header Not Set	Medium	Website tidak menerapkan header **Content Security Policy (CSP)** yang berfungsi untuk mengontrol sumber konten yang diizinkan dijalankan oleh browser, seperti JavaScript, CSS, gambar, dan *resource* lainnya.
2	Missing Anti-clickjacking Header	Medium	Website tidak memiliki mekanisme perlindungan terhadap serangan Clickjacking, seperti header: X-Frame-Options dan juga Content-Security-Policy frame-ancestors.
3	Sub *Resource* Integrity Attribute Missing	Medium	Website menggunakan *resource* eksternal seperti JavaScript atau CSS dari CDN tanpa atribut **integrity**.
4	Big Redirect Detected (Potential Sensitive Information Leak)	Leak	Ditemukan proses redirect yang berpotensi menyebabkan kebocoran informasi sensitif selama perpindahan halaman.
5	Cookie No HttpOnly Flag	Low	Cookie sesi tidak menggunakan atribut **HttpOnly**.
6	Server Leaks	Low	Server mengungkapkan

	Information via "X-Powered-By" HTTP Response Header Field(s)		informasi teknologi yang digunakan melalui header **X-Powered-By**.
7	X-Content-Type-Options Header Missing	Low	Website tidak menggunakan header **X-Content-Type-Options**.
8	Authentication Request Identified	Informational	*Tools* pengujian berhasil mengidentifikasi mekanisme autentikasi pada website, seperti halaman login atau proses verifikasi pengguna.
9	Session Management Response Identified	Informational	*Tools* mendeteksi adanya mekanisme manajemen sesi (session management) yang digunakan untuk mempertahankan status login pengguna.
Screenshoot			

4.3. Implementasi Sistem Artificial Intelligence dan *Software Agent*

Pada subbab ini dijelaskan secara lengkap proses implementasi komponen Artificial Intelligence (AI) dan *Software Agent* yang menjadi inti dari sistem prediksi dan penjadwalan *preventive maintenance* perangkat komputer. Implementasi ini mencakup spesifikasi lingkungan yang digunakan, proses persiapan dataset, pelatihan model *machine learning* menggunakan algoritma *Random Forest*, mekanisme kerja *Software Agent* berbasis Python, integrasi antara agent dengan server Laravel melalui *REST API*, serta visualisasi struktur *Decision Tree* yang terbentuk selama proses pelatihan model.

4.3.1. Spesifikasi Lingkungan Implementasi

Implementasi sistem terbagi menjadi dua lingkungan utama, yaitu lingkungan server dan lingkungan agent (*client*). Kedua lingkungan ini bekerja secara terintegrasi melalui jaringan lokal menggunakan protokol HTTP. Server berfungsi sebagai pusat pengelolaan data, penyimpanan, dan tampilan *dashboard*, sedangkan agent berjalan secara otonom pada setiap perangkat yang ingin dimonitor untuk mengumpulkan data dan melakukan prediksi AI secara local.

Tabel 4.15 Spesifikasi Lingkungan Server

Komponen	Spesifikasi
Sistem Operasi	Windows 10 / Windows 11 (64-bit)
Web *Framework*	Laravel 10.x
Bahasa Pemrograman *Backend*	PHP 8.1
Database	MySQL 8.0
Web Server	Apache (via XAMPP)
***IP Address* Server (Lokal)**	192.168.1.12
Port Aplikasi	8000
Browser Pendukung	Google Chrome

Format Data API	*JSON* (JavaScript Object Notation)
Arsitektur Sistem	*Client*–Server berbasis *REST API*

Spesifikasi pada Tabel 4.15 ini menunjukkan bahwa server dibangun menggunakan *Framework* Laravel 10 dengan bahasa pemrograman PHP 8.1 yang dijalankan di atas web server Apache. Database MySQL digunakan untuk menyimpan seluruh data perangkat, log monitoring, hasil prediksi AI, dan jadwal maintenance. Server dapat diakses melalui jaringan lokal pada alamat IP 192.168.1.12 dengan port 8000, sehingga seluruh agent yang terhubung dalam jaringan yang sama dapat mengirimkan data monitoring ke server secara langsung.

Tabel 4.16 Spesifikasi Lingkungan Agent (*Client*)

Komponen	Spesifikasi
Sistem Operasi	Windows 10 / Windows 11 (64-bit)
Bahasa Pemrograman	Python 3.10+
***Library* Monitoring**	psutil 5.9+ (pengambilan data CPU, RAM, Disk, Suhu)
Library Machine learning	*Scikit-learn* 1.3+ (*Random Forest* Classifier)
***Library* Data Processing**	pandas 2.0+ (manipulasi *DataFrame*)
***Library* Model Persistence**	joblib 1.3+ (load/save model .pkl)
***Library* HTTP Request**	requests 2.31+ (pengiriman data ke server via API)
File Model AI	model.pkl (*Random Forest* terlatih)

Interval Pengiriman Data	Setiap 5 detik (time.sleep(5))
Target *Endpoint* API	http://192.168.1.12:8000/api/device/store
File Konfigurasi & Eksekusi	agent.py, model.pkl, dataset.*JSON*, train_model.py, start_agent.bat
Mode Operasi	Normal (*Real-Time*) dan Simulasi Risiko (acak 30%)

Spesifikasi pada Tabel 4.16 ini menunjukkan bahwa agent dikembangkan menggunakan bahasa pemrograman Python 3.10 dengan sejumlah *Library* pendukung. *Library psutil* digunakan untuk mengambil data monitoring sistem secara *Real-Time*, *Library Scikit-learn* untuk memuat dan menjalankan model *Random Forest*, *Library* pandas untuk memproses data menjadi format *DataFrame* sebelum diprediksi, *Library* joblib untuk memuat file model.pkl, dan *Library* requests untuk mengirimkan data monitoring beserta hasil prediksi ke server Laravel melalui *HTTP POST* request. Agent berjalan secara terus-menerus dengan interval pengiriman data setiap 5 detik.

4.3.2. Dataset

Dataset yang digunakan untuk melatih model *machine learning* pada sistem ini diperoleh dari data monitoring perangkat yang dikumpulkan secara *Real-Time* oleh *Software Agent* dan disimpan pada tabel device_logs dalam database MySQL. Data tersebut kemudian diekspor melalui *endpoint* khusus pada server Laravel yaitu /export-dataset yang menghasilkan file dataset.*JSON*. Proses ekspor dataset dilakukan dengan mengambil seluruh log monitoring yang tersimpan, kemudian menambahkan label kondisi secara otomatis berdasarkan aturan threshold yang telah ditentukan.

Tabel 4.17 Informasi Fitur Dataset

Nama Fitur	Deskripsi	Rentang Nilai	Tipe Data
cpu	Persentase penggunaan CPU perangkat (%)	0 – 100	Numerik
ram	Persentase penggunaan memori RAM (%)	0 – 100	Numerik
disk	Persentase penggunaan kapasitas penyimpanan disk (%)	0 – 100	Numerik
temp	Suhu CPU perangkat (°C)	0 – 100	Numerik
label	Kelas kondisi perangkat (0 = Normal, 1 = Risiko)	0 atau 1	Kategorik (Target)

Dataset memiliki empat fitur input yaitu cpu, ram, disk, dan temp, serta satu kolom target yaitu label. Keempat fitur input tersebut merupakan data numerik yang merepresentasikan kondisi perangkat secara kuantitatif. Fitur cpu merepresentasikan persentase penggunaan prosesor, fitur ram merepresentasikan persentase penggunaan memori, fitur disk merepresentasikan persentase penggunaan kapasitas penyimpanan, dan fitur temp merepresentasikan suhu CPU dalam satuan derajat Celcius.

Proses pemberian label pada dataset dilakukan secara otomatis menggunakan aturan berbasis threshold yang telah didefinisikan pada sistem. Pemberian label otomatis ini memastikan konsistensi dan objektivitas dalam menentukan apakah kondisi suatu perangkat termasuk dalam kategori Normal atau Risiko tanpa memerlukan anotasi manual.

Tabel 4.18 Aturan Pemberian Label (Labeling Rules)

Kondisi Parameter	Label	Penjelasan
CPU > 70%	1 (Risiko)	Prosesor bekerja melebihi batas aman berkelanjutan

RAM > 70%	1 (Risiko)	Penggunaan memori mendekati kapasitas maksimum
Disk > 92%	1 (Risiko)	Penyimpanan hampir penuh, berpotensi menyebabkan kegagalan sistem
Suhu > 65°C	1 (Risiko)	Suhu CPU melebihi batas aman, risiko overheating
Semua parameter di bawah threshold	0 (Normal)	Perangkat beroperasi dalam kondisi normal dan aman

Berdasarkan Tabel 4.18, suatu record data diberikan label 1 (Risiko) apabila memenuhi setidaknya satu dari empat kondisi: penggunaan CPU melebihi 70%, penggunaan RAM melebihi 70%, penggunaan disk melebihi 92%, atau suhu CPU melebihi 65 derajat Celcius. Sebaliknya, apabila seluruh parameter berada di bawah nilai threshold, record data diberikan label 0 (Normal). Aturan labeling ini bersifat deterministik sehingga batas pemisah antara kelas Normal dan Risiko sangat jelas, yang berkontribusi pada tingginya akurasi model yang dihasilkan. Implementasi aturan labeling pada kode *endpoint* /export-dataset dapat dilihat pada potongan kode berikut :

```php
Route::get('/export-dataset', function () {
    $logs = DeviceLog::all();
    $data = [];
    foreach ($logs as $log) {
        // Buat label otomatis
        $label = (
            $log->cpu > 70 ||
            $log->ram > 70 ||
            $log->disk > 92 ||
            $log->temp > 65
        ) ? 1 : 0;
        $data[] = [
            'cpu' => $log->cpu,
            'ram' => $log->ram,
            'disk' => $log->disk,
            'temp' => $log->temp,
            'label' => $label
        ];
    }
    return response()->json($data);
});
```

Gambar 4.9 Code Dataset

Tabel 4.19 Ringkasan Informasi Dataset

Informasi Dataset	Keterangan
Total Data (record)	210 data
Jumlah Fitur Input	4 fitur (cpu, ram, disk, temp)
Jumlah Kelas Output	2 kelas (0 = Normal, 1 = Risiko)
Distribusi Kelas 0 (Normal)	±185 data
Distribusi Kelas 1 (Risiko)	±25 data
Format Dataset	*JSON* (dataset.*JSON*)
Sumber Data	Export dari *endpoint* /export-dataset pada sistem Laravel
Pembagian Data Training	80% (168 data)
Pembagian Data Pengujian	20% (42 data)

Berdasarkan Tabel 4.19, dataset yang digunakan terdiri dari 210 record data yang dibagi menjadi 80% data training (168 data) dan 20% data pengujian (42 data). Pembagian data dilakukan menggunakan fungsi *train_test_split* dari *Library Scikit-learn* dengan parameter random_state=42 untuk memastikan reproduktibilitas hasil pembagian data.

4.3.3. Implementasi *Machine learning* (Pelatihan Model)

Implementasi *machine learning* pada sistem ini menggunakan algoritma *Random Forest Classifier* dari *Library Scikit-learn* Python. *Random Forest* adalah metode *ensemble learning* yang bekerja dengan membangun sejumlah *Decision Tree* secara paralel selama proses pelatihan, kemudian menggabungkan hasil prediksi dari seluruh tree melalui mekanisme *voting* mayoritas untuk menghasilkan prediksi akhir. Pendekatan ensemble ini membuat *Random Forest* lebih robust terhadap

overfitting dibandingkan dengan single *Decision Tree* dan umumnya menghasilkan akurasi yang lebih tinggi.

Proses pelatihan model dilakukan melalui file train_model.py yang berisi serangkaian tahapan mulai dari pemuatan dataset, pemrosesan data, pelatihan model, evaluasi, hingga penyimpanan model. Potongan lengkap kode train_model.py dapat dilihat pada kode berikut:

```python
import json
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import joblib

# Load dataset
with open("dataset.json") as f:
    data = json.load(f)

df = pd.DataFrame(data)

X = df[['cpu','ram','disk','temp']]
y = df['label']

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Train Random Forest
model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

# Test model
y_pred = model.predict(X_test)

print("=== HASIL EVALUASI ===")
print(classification_report(y_test, y_pred))

# Simpan model
joblib.dump(model, "model.pkl")

print("Model berhasil disimpan sebagai model.pkl")
```

Gambar 4.10 Code train_model.py

Penjelasan setiap tahapan pada kode train_model.py adalah sebagai berikut. Pertama, dataset dimuat dari file dataset.*JSON* menggunakan *Library JSON* dan diubah menjadi objek *DataFrame* pandas agar mudah diolah. Kedua, fitur input dipisahkan menjadi variabel X yang terdiri dari kolom cpu, ram, disk, dan temp, sedangkan variabel y berisi kolom label

sebagai target klasifikasi. Ketiga, data dibagi menjadi data training dan data pengujian menggunakan fungsi *train_test_split* dengan proporsi 80% training dan 20% testing, serta parameter random_state=42 untuk memastikan reproduktibilitas pembagian data. Keempat, model *Random Forest* diinisialisasi dengan 100 estimator (pohon keputusan) kemudian dilatih menggunakan data training melalui metode *fit()*. Kelima, model dievaluasi pada data uji menggunakan metode *predict()* dan hasilnya dicetak dalam bentuk classification report yang mencakup metrik *precision, recall, f1-score*, dan accuracy. Keenam, model yang telah dilatih disimpan ke dalam file model.pkl menggunakan *Library* joblib agar dapat dimuat kembali oleh agent.py untuk melakukan prediksi tanpa perlu melatih ulang model.

Tabel 4.20 Parameter Model *Random Forest* Classifier

Parameter	Nilai	Keterangan
n_estimators	100	Jumlah pohon keputusan yang dibangun dalam forest
max_features	sqrt(n_features)	Jumlah fitur yang dipertimbangkan saat mencari split terbaik
max_depth	None (tidak dibatasi)	Kedalaman maksimum setiap pohon
min_samples_split	2	Jumlah minimum sampel untuk membagi sebuah node
min_samples_leaf	1	Jumlah minimum sampel pada leaf node
bootstrap	True	Menggunakan *bootstrap sampling* untuk membangun setiap tree
random_state	42	Nilai seed untuk reproduktibilitas hasil

criterion	gini	Fungsi untuk mengukur kualitas split pada setiap node

Parameter n_estimators=100 yang digunakan pada model menunjukkan bahwa *Random Forest* membangun sebanyak 100 pohon keputusan yang masing-masing dilatih pada subset data yang dipilih secara acak (*bootstrap sampling*). Setiap pohon dalam forest melakukan prediksi secara independen, dan hasil akhir prediksi ditentukan melalui *voting* mayoritas dari seluruh 100 pohon tersebut. Pendekatan ini secara signifikan mengurangi varians model dibandingkan dengan single *Decision Tree*, sehingga menghasilkan prediksi yang lebih stabil dan dapat diandalkan.

4.3.4. Hasil Evaluasi Model

Evaluasi model dilakukan untuk mengukur kemampuan model *Random Forest Classifier* dalam mengklasifikasikan kondisi perangkat secara akurat. Evaluasi menggunakan metrik *precision*, *recall*, *f1-score*, dan accuracy yang dihitung berdasarkan hasil prediksi model pada data pengujian (data yang tidak digunakan selama proses pelatihan). Hasil evaluasi model ditampilkan secara otomatis oleh fungsi *classification_report* dari *Library Scikit-learn* setelah proses pelatihan selesai.

```
C:\Users\LENOVO\OneDrive\Desktop\sempro's file>python train_model.py
=== HASIL EVALUASI ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       1.00      1.00      1.00         4

    accuracy                           1.00        41
   macro avg       1.00      1.00      1.00        41
weighted avg       1.00      1.00      1.00        41

Model berhasil disimpan sebagai model.pkl
```

Gambar 4.11 Hasil Evaluasi Model

Berdasarkan Gambar 4.8 ini, model *Random Forest Classifier* yang digunakan pada sistem ini menghasilkan akurasi sebesar 100% pada data pengujian. Nilai precision, recall, dan f1-score untuk kedua kelas yaitu

kelas 0 (Normal) dan kelas 1 (Risiko) masing-masing mencapai nilai 1.00 atau 100%. Hal ini berarti model berhasil mengklasifikasikan seluruh 41 data uji tanpa satu pun kesalahan klasifikasi (false positive maupun false negative).

Precision sebesar 1.00 pada kelas Normal menunjukkan bahwa dari seluruh perangkat yang diprediksi Normal oleh model, 100% di antaranya memang benar-benar berstatus Normal. Precision sebesar 1.00 pada kelas Risiko menunjukkan bahwa seluruh perangkat yang diprediksi Risiko memang benar-benar berstatus Risiko, artinya tidak ada perangkat yang sebenarnya aman tetapi salah diprediksi sebagai berbahaya (tidak ada false alarm).

Recall sebesar 1.00 pada kelas Normal menunjukkan bahwa model berhasil mendeteksi 100% dari seluruh perangkat yang sebenarnya berstatus Normal. Recall sebesar 1.00 pada kelas Risiko menunjukkan bahwa model berhasil mendeteksi 100% dari seluruh perangkat yang sebenarnya berstatus Risiko, artinya tidak ada perangkat berbahaya yang luput dari deteksi (tidak ada missed detection).

F1-Score sebesar 1.00 merupakan rata-rata harmonik antara precision dan recall. Nilai ini mengkonfirmasi bahwa model memiliki keseimbangan sempurna antara kemampuan menghindari false alarm dan kemampuan mendeteksi seluruh kasus risiko yang sesungguhnya.

Tingginya akurasi model ini dapat dijelaskan oleh karakteristik dataset yang digunakan. Label pada dataset dibuat berdasarkan aturan threshold yang bersifat deterministik dan memiliki batas pemisah yang sangat jelas antara kelas Normal dan kelas Risiko. Perbedaan nilai fitur antara kedua kelas sangat signifikan, misalnya kelas Normal memiliki CPU umumnya di bawah 70% sedangkan kelas Risiko memiliki CPU di atas 75%, sehingga model dapat dengan mudah menemukan pola pemisah yang tepat. Algoritma *Random Forest* dengan 100 *Decision Tree* kemudian mengkonsolidasikan pola-pola tersebut untuk menghasilkan prediksi yang konsisten dan akurat.

4.3.5. Implementasi *Software Agent*

Software Agent merupakan komponen kunci dari sistem ini yang berjalan secara otonom pada sisi *client* (perangkat yang dimonitor). Agent bertugas untuk mengumpulkan data kondisi perangkat secara *Real-Time*, melakukan prediksi risiko menggunakan model AI yang telah dilatih, dan mengirimkan seluruh data beserta hasil prediksi ke server Laravel melalui *REST API*. Konsep *Software Agent* dalam sistem ini mengimplementasikan sifat-sifat agen cerdas yaitu autonomy (berjalan mandiri tanpa intervensi manusia), reactivity (merespons perubahan kondisi sistem secara berkala), dan persistence (berjalan secara terus-menerus dalam *loop* tak terbatas).

Agent terdiri dari beberapa file yang saling berkaitan dan membentuk satu kesatuan sistem monitoring berbasis AI. Penjelasan fungsi masing-masing file ditunjukkan pada tabel berikut:

Tabel 4.21 Komponen File *Software Agent*

Nama File	Fungsi dan Peran
agent.py	File utama *Software Agent*. Bertugas mengambil data sistem secara *Real-Time* menggunakan psutil, memuat model AI dari model.pkl, menjalankan prediksi, dan mengirim data beserta hasil prediksi ke server Laravel melalui *HTTP POST* request setiap 5 detik.
model.pkl	File model *Random Forest Classifier* yang sudah dilatih dan disimpan menggunakan *Library* joblib. File ini dimuat (di-load) oleh agent.py setiap kali agent dijalankan untuk melakukan prediksi kondisi perangkat tanpa perlu melatih ulang model.
dataset.*JSON*	Dataset training dalam format *JSON* yang berisi 210 record data monitoring perangkat beserta label kondisinya. Dataset ini digunakan oleh train_model.py untuk proses pelatihan model.

train_model.py	Script Python untuk melatih ulang model *Random Forest* menggunakan dataset.*JSON*. Script ini dijalankan secara manual oleh developer apabila ingin memperbarui atau melatih ulang model AI.
start_agent.bat	File batch script Windows yang digunakan untuk menjalankan agent.py secara otomatis. Admin cukup klik dua kali file ini untuk memulai proses monitoring tanpa perlu membuka terminal secara manual.

Dari tabel di atas terlihat bahwa agent.py merupakan komponen utama yang mengintegrasikan seluruh fungsi: pengambilan data sistem, prediksi AI, dan pengiriman data ke server. File model.pkl berperan sebagai otak prediktif dari agent yang memungkinkan prediksi dilakukan secara lokal di sisi *client* tanpa perlu mengirimkan data ke server terpisah untuk diproses. Hal ini membuat proses prediksi menjadi sangat cepat dan tidak bergantung pada koneksi internet karena model berjalan langsung di perangkat.

Berikut adalah potongan lengkap kode agent.py yang menggambarkan implementasi seluruh komponen *Software Agent*:

Gambar 4.12 Code agent.py

Gambar 4.9 ini menunjukkan bahwa agent.py memiliki struktur yang sederhana namun efektif. Fungsi get_system_data() bertanggung jawab untuk mengambil data monitoring dari sistem operasi menggunakan *Library psutil* dan menentukan mode operasi (normal atau simulasi). *Loop* utama *while True* memastikan agent berjalan secara berkelanjutan. Di dalam *loop*, data monitoring diambil, diproses menjadi *DataFrame*, diprediksi menggunakan model AI, kemudian dikirimkan ke server. Mekanisme *try-except* memastikan agent tetap berjalan meskipun terjadi kesalahan koneksi atau error lainnya.

Pembagian dataset menggunakan rasio 80% data latih (164 record) dan 20% data uji (41 record) didasarkan pada rekomendasi Reitermanova (2010) yang menyatakan bahwa rasio 80:20 merupakan pembagian yang umum dan optimal untuk dataset berukuran sedang dalam machine learning. Rasio ini memberikan data latih yang cukup bagi model untuk mempelajari pola data, sekaligus data uji yang representatif untuk evaluasi performa yang valid. Pemilihan rasio ini juga konsisten dengan penelitian serupa di domain monitoring sistem (Carvalho et al., 2019) yang menggunakan rasio 80:20 sebagai standar pembagian data.

Rasio (Train:Test)	Data Latih	Data Uji	Kelebihan	Kekurangan
70:30	143 data	62 data	Data uji lebih banyak, evaluasi lebih robust	Data latih lebih sedikit, model kurang belajar
80:20 (dipilih)	164 data	41 data	Keseimbangan optimal antara latih dan uji	Standar yang paling umum digunakan
90:10	184 data	21 data	Model mendapat lebih banyak data latih	Data uji terlalu sedikit, evaluasi kurang representatif

Berdasarkan perbandingan ketiga rasio tersebut, rasio 80:20 dipilih karena memberikan keseimbangan paling optimal antara data latih dan data uji. Dibandingkan rasio 70:30, rasio 80:20 memberikan lebih banyak data untuk proses pembelajaran model. Dibandingkan rasio 90:10, rasio

80:20 menghasilkan data uji yang lebih representatif (41 data) sehingga evaluasi akurasi lebih valid. Hal ini sesuai dengan rekomendasi Reitermanova (2010) bahwa rasio 80:20 merupakan pilihan optimal untuk dataset berukuran sedang.

Tabel 4.22 Mode Operasi *Software Agent*

Mode Operasi	Kondisi Pemicu	Data yang Dikirim	Tujuan
Mode Normal (70%)	Probabilitas 70% dari setiap siklus	Menggunakan data *Real-Time* dari sistem menggunakan psutil: CPU aktual, RAM aktual, Disk aktual, dan suhu perangkat (suhu default 50°C karena keterbatasan sensor).	Data monitoring akurat yang mencerminkan kondisi perangkat saat ini.
Mode Simulasi Risiko (30%)	Probabilitas 30% dari setiap siklus	Mensimulasikan kondisi beban tinggi dengan nilai: CPU = 75–95%, RAM = 70–90%, Disk = 93–98%, Suhu = 65–80°C menggunakan random.randint().	Data simulasi untuk menguji kemampuan sistem dalam mendeteksi dan merespons kondisi perangkat berisiko tinggi.

Adanya dua mode operasi pada agent.py merupakan keputusan desain yang penting. Mode normal memastikan sistem menerima data monitoring yang akurat dan mencerminkan kondisi perangkat yang sesungguhnya. Mode simulasi risiko (30% probabilitas) diterapkan untuk memastikan sistem dapat diuji coba dan divalidasi kemampuannya dalam mendeteksi kondisi berbahaya, terutama pada lingkungan pengembangan

di mana perangkat yang dimonitor jarang mencapai kondisi beban tinggi secara alami.

4.3.6. Alur Kerja *Software Agent*

Alur kerja *Software Agent* menggambarkan urutan proses yang terjadi dari saat agent dijalankan hingga data monitoring berhasil diterima dan diproses oleh server. Setiap siklus monitoring berlangsung setiap 5 detik secara otomatis dan berulang tanpa henti selama agent aktif.

Tabel 4.23 Alur Kerja *Software Agent*

No.	Tahap	Deskripsi
1	**Inisialisasi**	Agent dijalankan melalui start_agent.bat. Python memuat file agent.py dan *Library* yang dibutuhkan (psutil, requests, joblib, pandas). Model *Random Forest* dimuat dari file model.pkl ke dalam memori.
2	**Pengambilan Data Sistem**	Fungsi get_system_data() dipanggil. Agent mengambil data *hostname* dan *IP Address* perangkat menggunakan *Library* socket. Data CPU, RAM, dan Disk diambil secara *Real-Time* menggunakan psutil.
3	**Penentuan Mode**	Sistem membangkitkan bilangan acak antara 1–10. Jika nilai 1–7 (probabilitas 70%): mode normal, data *Real-Time* digunakan. Jika nilai 8–10 (probabilitas 30%): mode simulasi, data risiko tinggi dibangkitkan.
4	**Prediksi AI**	Data CPU, RAM, Disk, dan Suhu diubah menjadi *DataFrame* pandas. Model *Random Forest* melakukan prediksi (*predict()*) menghasilkan label 0 (Normal) atau 1 (Risiko). Hasil prediksi disimpan pada field *ml_prediction*.
5	**Pengiriman Data ke Server**	Data monitoring beserta hasil prediksi AI dikirimkan ke server Laravel melalui *HTTP POST* request ke *endpoint* /api/device/store dalam format *JSON* menggunakan *Library* requests.

6	**Penerimaan oleh Server**	Server Laravel menerima data via API, menyimpannya ke database (tabel device_logs), menganalisis hasil prediksi, dan apabila terdeteksi risiko, membuat jadwal maintenance otomatis.
7	**Jeda dan Pengulangan**	Agent menunggu 5 detik (time.sleep(5)) kemudian kembali ke langkah 2. Proses ini berjalan secara terus-menerus (*loop* tak terbatas) selama agent aktif.

Berdasarkan Tabel 4.23, alur kerja agent dimulai dari proses inisialisasi ketika file start_agent.bat dijalankan oleh pengguna. Setelah model AI berhasil dimuat, agent memasuki *loop* pengumpulan data yang berjalan secara terus-menerus. Setiap siklus mencakup pengambilan data sistem, penentuan mode operasi, prediksi menggunakan model AI, dan pengiriman data ke server. Apabila terjadi kesalahan pada salah satu tahap, mekanisme error handling memastikan agent tidak berhenti melainkan menunggu 5 detik kemudian mencoba kembali dari awal siklus.

4.3.7. Integrasi Agent dengan Server via *REST API*

Komunikasi antara *Software Agent* dan server Laravel dilakukan melalui *REST API* menggunakan protokol HTTP. Agent berperan sebagai API *client* yang mengirimkan data monitoring, sedangkan server Laravel berperan sebagai API server yang menerima, memvalidasi, menyimpan, dan memproses data tersebut. Integrasi ini memungkinkan sistem bekerja secara terdistribusi dimana setiap perangkat dapat menjalankan agent secara mandiri dan mengirimkan data ke server pusat.

Tabel 4.24 Spesifikasi Integrasi *REST API* Agent–Server

Atribut API	Keterangan
Method	POST
***Endpoint* URL**	http://192.168.1.12:8000/api/device/store
Content-Type	application/*JSON*

Accept	application/*JSON*
Data yang Dikirim (*JSON*)	*hostname*, ip_address, cpu, ram, disk, temp, *ml_prediction*
Controller Penerima	DeviceController@store (Laravel)
Response Sukses	HTTP 200 / 201 – Data berhasil disimpan
Response Gagal	HTTP 422 / 500 – Validasi gagal atau server error
Frekuensi Pengiriman	Setiap 5 detik dari setiap perangkat yang menjalankan agent
Autentikasi	Tidak menggunakan token (akses internal jaringan lokal)

Implementasi *endpoint* API pada sisi server Laravel terdapat pada file routes/api.php dan ditangani oleh DeviceController. Berikut adalah kode implementasi *endpoint* penerimaan data dari agent:

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\DeviceController;

Route::post('/device/store', [DeviceController::class, 'store']);
Route::get('/devices', [DeviceController::class, 'index']);
```

Gambar 4.13 Code *endpoint* API

Endpoint POST /api/device/store menerima data *JSON* yang dikirimkan oleh agent, kemudian meneruskannya ke method store() di DeviceController, server menerima data monitoring beserta nilai *ml_prediction* yang dikirimkan oleh agent. Nilai *ml_prediction* yang merupakan hasil prediksi model *Random Forest* (0 = Normal, 1 = Risiko) digunakan sebagai penentu utama status perangkat. Apabila *ml_prediction* bernilai 1 (Risiko), server melakukan klasifikasi lanjutan berdasarkan rata-

rata 50 log terakhir untuk membedakan tingkat keparahan: apabila rata-rata CPU ≥ 85%, RAM ≥ 80%, Disk ≥ 95%, atau Suhu ≥ 75°C maka status ditetapkan sebagai Critical, sedangkan apabila di bawah threshold tersebut status ditetapkan sebagai Warning. Apabila *ml_prediction* bernilai 0 (Normal), status perangkat langsung ditetapkan sebagai Normal tanpa pemeriksaan lebih lanjut. Selain menentukan status, server juga memeriksa 10 log terakhir untuk memutuskan apakah perlu membuat jadwal maintenance otomatis: jika ditemukan 5 atau lebih log Critical maka jadwal HIGH priority dibuat dengan rekomendasi 2 hari ke depan, sedangkan jika ditemukan 7 atau lebih log Warning maka jadwal MEDIUM priority dibuat dengan rekomendasi 5 hari ke depan.

Selain *endpoint* untuk menerima data dari agent, sistem juga menyediakan *endpoint* GET /api/devices yang mengembalikan data monitoring terbaru dari seluruh perangkat yang terhubung. *Endpoint* ini digunakan oleh *dashboard* monitoring di sisi *frontend* untuk menampilkan kondisi perangkat secara *Real-Time* dengan cara melakukan *polling* data secara berkala ke API tersebut.

4.3.8. Visualisasi dan Analisis *Decision Tree*

Model *Random Forest Classifier* yang digunakan terdiri dari 100 *Decision Tree* (pohon keputusan) yang dibangun secara paralel selama proses pelatihan. Setiap tree dalam forest dibangun dari subset data training yang dipilih secara acak menggunakan teknik *bootstrap sampling*, dan setiap node percabangan pada tree hanya mempertimbangkan subset fitur yang dipilih secara acak pula (menggunakan nilai sqrt dari jumlah total fitur). Kombinasi teknik-teknik acak ini menghasilkan 100 tree yang berbeda-beda satu sama lain, sehingga *voting* mayoritas dari 100 prediksi yang beragam menghasilkan prediksi akhir yang lebih akurat dan robust. Meskipun model terdiri dari 100 tree, struktur logika percabangan pada setiap tree mengikuti pola yang serupa karena dataset memiliki batas threshold yang jelas antara kelas Normal dan Risiko. Berikut adalah penjelasan struktur node yang terbentuk pada *Decision Tree* dalam model:

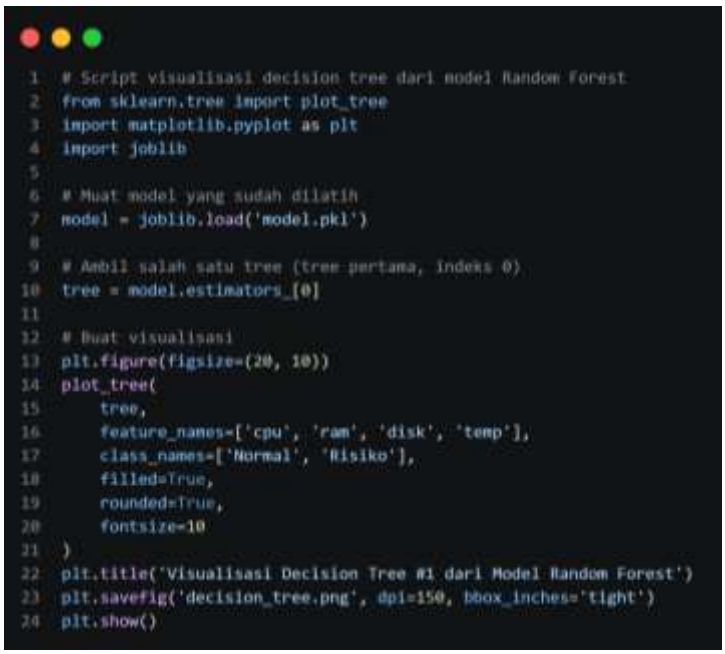
Tabel 4.25 Analisis Struktur Node *Decision Tree*

Jenis Node	Fitur/Kondisi	Fungsi	Cakupan Data
Root Node	Fitur pembagi pertama (umumnya CPU atau Disk)	Node paling atas, membagi seluruh dataset	Seluruh 210 data training
Internal Node	Fitur pembagi berikutnya (CPU, RAM, Disk, Temp)	Node percabangan berdasarkan threshold tertentu	Subset dari data parent node
Leaf Node (Normal)	Tidak ada pembagian lebih lanjut	Mengklasifikasikan perangkat sebagai Normal (label 0)	Mayoritas data Normal
Leaf Node (Risiko)	Tidak ada pembagian lebih lanjut	Mengklasifikasikan perangkat sebagai Risiko (label 1)	Data dengan nilai tinggi pada salah satu atau lebih fitur

Pada root node setiap tree, algoritma *Random Forest* secara otomatis memilih fitur dan nilai threshold terbaik yang memisahkan data dengan tingkat kemurnian (*purity*) tertinggi. Berdasarkan karakteristik dataset yang digunakan, fitur CPU dan Disk umumnya menjadi fitur pemisah utama di root node karena perbedaan nilai antara kelas Normal dan Risiko pada kedua fitur tersebut sangat signifikan dan memiliki nilai *information gain* tertinggi.

Setiap percabangan pada internal node merepresentasikan satu pertanyaan keputusan, misalnya: apakah nilai CPU lebih besar dari 72.5? Apabila ya, data diarahkan ke cabang kanan yang mengarah ke node risiko; apabila tidak, data diarahkan ke cabang kiri yang mengarah ke node normal. Pola percabangan ini terus berlanjut hingga data mencapai leaf node yang memberikan klasifikasi akhir.

Untuk melakukan visualisasi salah satu *Decision Tree* dari model *Random Forest*, dapat digunakan script Python berikut menggunakan *Library* matplotlib dan sklearn.tree:



```python
# Script visualisasi decision tree dari model Random Forest
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
import joblib

# Muat model yang sudah dilatih
model = joblib.load('model.pkl')

# Ambil salah satu tree (tree pertama, indeks 0)
tree = model.estimators_[0]

# Buat visualisasi
plt.figure(figsize=(20, 10))
plot_tree(
    tree,
    feature_names=['cpu', 'ram', 'disk', 'temp'],
    class_names=['Normal', 'Risiko'],
    filled=True,
    rounded=True,
    fontsize=10
)
plt.title('Visualisasi Decision Tree #1 dari Model Random Forest')
plt.savefig('decision_tree.png', dpi=150, bbox_inches='tight')
plt.show()
```

Gambar 4.14 model.pkl

Script di atas memuat model *Random Forest* dari file model.pkl, kemudian mengambil pohon pertama (estimators_[0]) dari 100 pohon yang ada untuk divisualisasikan. Parameter feature_names diisi dengan nama keempat fitur input, dan class_names diisi dengan nama kedua kelas output yaitu Normal dan Risiko. Parameter filled=True menyebabkan setiap node diwarnai berdasarkan kelas mayoritas yang dikandungnya, sehingga visualisasi lebih mudah dipahami. Gambar 4.12 merupakan hasil visualisasi salah satu *Decision Tree* dari model *Random Forest* yang digunakan pada sistem.

Gambar 4.15 Visualisasi *Decision Tree*

Pada visualisasi *Decision Tree*, setiap node internal ditampilkan bersama informasi fitur yang digunakan sebagai kriteria percabangan, nilai threshold, nilai *gini impurity*, jumlah sampel yang mencapai node tersebut, dan distribusi kelas pada node. Node yang berwarna oranye mewakili kelas Normal, sedangkan node yang berwarna biru mewakili kelas Risiko. Semakin pekat warna sebuah node, semakin tinggi tingkat kemurnian (*purity*) kelas pada node tersebut, artinya semakin banyak proporsi data pada node tersebut berasal dari satu kelas yang sama.

Dari visualisasi tersebut terlihat bahwa tree berhasil memisahkan kelas Normal dan Risiko dengan sangat baik hanya dalam beberapa level percabangan. Hal ini konsisten dengan hasil evaluasi model yang menunjukkan akurasi 100%, karena batas keputusan antara kedua kelas dalam dataset ini sangat jelas dan dapat direpresentasikan dengan struktur tree yang relatif sederhana. Dengan 100 tree yang dibangun melalui proses *Random Forest*, *voting* mayoritas dari seluruh tree menghasilkan prediksi yang sangat konsisten dan akurat untuk setiap record data yang diberikan.

Secara keseluruhan, implementasi Artificial Intelligence dan *Software Agent* pada sistem ini berhasil mengintegrasikan proses pengumpulan data *Real-Time*, prediksi risiko berbasis *machine learning*, dan pengiriman data ke server dalam satu rantai proses yang berjalan secara otonom dan berkelanjutan. Model *Random Forest Classifier* yang digunakan terbukti mampu mengklasifikasikan kondisi perangkat dengan akurasi tinggi, sementara *Software Agent* memastikan data monitoring

dikumpulkan dan diprediksi secara konsisten setiap 5 detik dari setiap perangkat yang terhubung ke sistem.

BAB V
KESIMPULAN

5.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian Sistem Prediksi dan Penjadwalan *Preventive maintenance* Perangkat Komputer Berbasis Artificial Intelligence, maka dapat disimpulkan bahwa tujuan dari penelitian ini telah berhasil dicapai. Adapun kesimpulan yang diperoleh adalah sebagai berikut:

1. Sistem berhasil dikembangkan untuk melakukan monitoring kondisi perangkat komputer secara *Real-Time* melalui *Software Agent* berbasis Python yang berjalan secara otonom pada sisi *client*. Agent mampu mengumpulkan data penggunaan CPU, RAM, Disk, dan temperatur perangkat secara otomatis menggunakan *Library psutil*, kemudian mengirimkan data tersebut ke server Laravel melalui *REST API* dengan interval pengiriman setiap 5 detik, sehingga kondisi setiap perangkat dapat dipantau secara berkelanjutan tanpa intervensi manual.
2. Sistem berhasil mengimplementasikan metode Artificial Intelligence menggunakan algoritma *Random Forest Classifier* yang dilatih menggunakan dataset berlabel hasil export dari sistem monitoring. Model mengklasifikasikan kondisi perangkat ke dalam dua kelas yaitu Normal (label 0) dan Risiko (label 1), dan berhasil mencapai akurasi 100% pada data pengujian dengan nilai precision, recall, dan F1-Score masing-masing 1.00. Hasil prediksi AI (*ml_prediction*) yang dikirimkan oleh agent kemudian digunakan oleh server sebagai penentu utama status perangkat, di mana server melakukan klasifikasi lanjutan menjadi tiga tingkat kondisi yaitu Normal, Warning, dan Critical berdasarkan rata-rata metrik dari 50 log terakhir, sehingga administrator dapat mengidentifikasi perangkat berisiko secara lebih spesifik dan terukur.

3. Sistem mampu menghasilkan rekomendasi jadwal *preventive maintenance* secara otomatis berdasarkan hasil prediksi Artificial Intelligence. Ketika agent mendeteksi kondisi Risiko secara berulang, server secara otomatis membuat jadwal maintenance dengan tingkat prioritas HIGH (rekomendasi 2 hari ke depan) apabila terdeteksi 5 atau lebih log Critical, atau prioritas MEDIUM (rekomendasi 5 hari ke depan) apabila terdeteksi 7 atau lebih log Warning, tanpa memerlukan intervensi manual dari administrator.
4. Sistem berhasil menyediakan fitur pengelolaan jadwal maintenance yang lengkap, memungkinkan administrator melakukan konfirmasi tanggal pelaksanaan, mengubah status maintenance menjadi completed atau missed, serta melakukan penjadwalan ulang secara otomatis apabila jadwal maintenance terlewat, sehingga seluruh proses maintenance dapat terdokumentasi dan dikelola dalam satu sistem yang terintegrasi.
5. Sistem berhasil mengimplementasikan fitur pengiriman notifikasi email maintenance kepada pemilik perangkat menggunakan Laravel Mailable, sehingga informasi jadwal maintenance dapat disampaikan secara langsung, terstruktur, dan terdokumentasi melalui tabel email_logs yang mencatat seluruh riwayat pengiriman email beserta statusnya.
6. Sistem mampu menghasilkan laporan monitoring perangkat dalam format PDF menggunakan *Library* Barryvdh DomPDF yang mencakup ringkasan kondisi seluruh perangkat, daftar jadwal maintenance, dan statistik status perangkat. Laporan ini dapat digunakan sebagai dokumentasi resmi kondisi infrastruktur, bahan evaluasi berkala, serta pendukung pengambilan keputusan oleh tim IT Support.
7. Hasil pengujian menggunakan metode *Black Box Testing* dan *User Acceptance Testing* (UAT) menunjukkan bahwa seluruh fungsi utama sistem berjalan sesuai dengan kebutuhan pengguna, mencakup fungsi monitoring *Real-Time*, prediksi AI, penentuan status tiga kelas, penjadwalan maintenance otomatis, pengiriman email notifikasi,

pengelolaan jadwal, dan pembuatan laporan PDF. Seluruh skenario pengujian berhasil dijalankan tanpa ditemukan kesalahan fungsional yang signifikan.

Dengan demikian, Sistem Prediksi dan Penjadwalan *Preventive maintenance* Perangkat Komputer Berbasis Artificial Intelligence yang dikembangkan terbukti mampu mengintegrasikan teknologi *machine learning*, *Software Agent*, dan aplikasi web berbasis Laravel dalam satu sistem yang bekerja secara otonom, *Real-Time*, dan proaktif. Sistem ini dapat mengurangi risiko *downtime* perangkat serta meningkatkan efektivitas pengelolaan infrastruktur teknologi informasi melalui pendekatan *preventive maintenance* berbasis data.

5.2. Saran

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

1. Sistem dapat dikembangkan dengan menambahkan fitur manajemen pengguna (*User Management*) yang memungkinkan administrator mengelola akun, hak akses, dan pembagian tugas maintenance secara lebih terstruktur, sehingga sistem dapat digunakan oleh tim *IT Support* dengan peran dan tanggung jawab yang berbeda-beda.
2. Model Artificial Intelligence yang digunakan dapat ditingkatkan lebih lanjut dengan mengumpulkan dataset yang lebih besar, lebih beragam, dan mencakup berbagai jenis perangkat serta kondisi lingkungan yang berbeda-beda. Hal ini bertujuan agar model yang dihasilkan memiliki kemampuan generalisasi yang lebih baik dan lebih representatif terhadap kondisi nyata di lapangan, mengingat dataset yang digunakan pada penelitian ini bersumber dari lingkungan jaringan lokal yang terbatas. Selain itu, tabel ai_predictions yang telah tersedia di database dapat dimanfaatkan untuk menyimpan history prediksi AI secara terpisah sebagai bahan analisis dan evaluasi model di masa mendatang.

3. Sistem dapat diintegrasikan dengan platform komunikasi lain seperti *Telegram Bot, WhatsApp Business API*, atau Microsoft Teams untuk memperluas saluran notifikasi maintenance, sehingga informasi dapat dijangkau melalui berbagai media komunikasi yang sudah digunakan sehari-hari oleh tim *IT Support* dan pengguna perangkat.
4. Penelitian selanjutnya dapat mengembangkan sistem ke arah *Predictive maintenance* yang lebih kompleks dengan memanfaatkan teknik *Deep Learning* seperti LSTM (*Long Short-Term Memory*) untuk analisis data deret waktu (*time series*), sehingga sistem tidak hanya mampu mengklasifikasikan kondisi perangkat saat ini, tetapi juga dapat memperkirakan kapan potensi kerusakan akan terjadi di masa mendatang secara lebih akurat dan spesifik.

DAFTAR PUSTAKA

Almazaideh, M., & Levendovszky, J. (2022). A predictive maintenance system for wireless sensor networks: A *machine learning* approach. *Indonesian Journal of Electrical Engineering and Computer Science*, *25*(2), 1047–1058. https://doi.org/10.11591/ijeecs.v25.i2.pp1047-1058

Carvalho, T. P., Soares, F. A. A. M. N., Vita, R., Francisco, R. da P., Basto, J. P., & Alcalá, S. G. S. (2019). A systematic literature review of *machine learning* methods applied to predictive maintenance. *Computers and Industrial Engineering*, *137*(August), 106024. https://doi.org/10.1016/j.cie.2019.106024

Hunt, J. (2023). *Machine learning in Python*. *12*, 633–642. https://doi.org/10.1007/978-3-031-40336-1_56

Jin, Z., Shang, J., Zhu, Q., Ling, C., Xie, W., & Qiang, B. (2020). RFRSF: Employee Turnover Prediction Based on Random Forests and Survival Analysis. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12343 LNCS*, 503–515. https://doi.org/10.1007/978-3-030-62008-0_35

Kale, S. S. (n.d.). *Predictive Maintenance of Equipment Using Machine learning Algorithms Sakshi Sanjay Kale National College of Ireland Supervisor : Naushad Alam*.

Kane, A., Khandale, A., Nigade, P., & Joshi, P. (2022). Predictive maintenance using *machine learning* techniques. *Proceedings from the International Congress on Project Management and Engineering*, *May*. https://arxiv.org/pdf/2205.09402

Machine learning in Python®. (2015). In *Machine learning in Python®*. https://doi.org/10.1002/9781119183600

Nur Hidayati. (2019). Penggunaan Metode Waterfall Dalam Rancang Bangun Sistem Informasi Penjualan. *Generation Journal*, *3*(1), 1–10.

Quinlan, J. R. (1996). Learning decision tree classifiers. *ACM Computing Surveys*, *28*(1), 71–72. https://doi.org/10.1145/234313.234346

Saravanos, A., & Curinga, M. X. (2023). Simulating the Software Development

Lifecycle: The Waterfall Model. *Applied System Innovation*, *6*(6). https://doi.org/10.3390/asi6060108

Song, Y. Y., & Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, *27*(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044

Surwase, V. (2016). *REST API* Modeling Languages -A Developer ' s Perspective Related papers *REST API* Modeling Languages - A Developer ' s Perspective. *IJSTE - International Journal of Science Technology and Engineering*, *2*(10), 634–637. https://www.academia.edu/27064725/REST_API_Modeling_Languages_A_Developers_Perspective?bulkDownload=thisPaper-topRelated-sameAuthor-citingThis-citedByThis-secondOrderCitations&from=cover_page

Telefonica. (2019). *済無No Title No Title No Title*. *1*, 105–112.

World, N. (1990). *1 . WHAT IS ARTIFICIAL INTELLIGENCE ? 1988*, 3–4.

Zhu, T., Ran, Y., Zhou, X., & Wen, Y. (2024). *A Survey of Predictive Maintenance: Systems, Purposes and Approaches*. *Dl*, 1–38. https://doi.org/10.1109/COMST.2025.3567802

DAFTAR LAMPIRAN

Lampiran 1. Dokumen Proses Pengumpulan Requirement

Link akses Dokumen Work Process:

Work Process

Lampiran 2. Link Produk

Link akses Repository GitHub:

https://github.com/hryyyh/Sistem-Monitoring-Perangkat.git

Lampiran 3. Dokumen Pengujian UAT

1. Link Akses UAT :
   DOKUMEN USER ACCEPTANCE TESTING (UAT)
2. Link Akses Performance Testing
   PERFROMANCE TESTING
3. Link Akses Security Testing:
   SECURITY TESTING